SH. NAZIROV, G. DIVALD

DASTURLASH ASOSLARI

O'ZBEKISTON RESPUBLIKASI OLIY
VA O'RTA MAXSUS TA'LIM VAZIRLIGI
O'RTA MAXSUS, KASB-HUNAR TA'LIMI MARKAZI

Sh. Nazirov, G. Divald

DASTURLASH ASOSLARI

Axborot-kommunikatsiya texnologiyalari sohasidagi kasb-hunar kollejlarining "Axborot-kommunikatsiya tizimlari (3521916)" mutaxassisligi talabalari uchun o'quv qo'llanma

«SHARQ» NASHRIYOT-MATBAA
AKSIYADORLIK KOMPANIYASI
BOSH TAHRIRIYATI
TOSHKENT – 2007

Mazkur o'quv qo'llanma Germaniya texnikaviy hamkorlik tashkiloti (GTZ) hamda Germaniya taraqqiyot banki (KfW) ishtirokidagi "Axborot-kommunikatsiya texnologiyalari sohasida kasb-hunar ta'limini rivojlantirishga ko'maklashish" – loyihasi doirasida ishlab chiqilgan.

O'zbekiston Respublikasi Oliy va o'rta maxsus ta'lim vazirligi O'rta maxsus, kasb-hunar ta'limi markazi tomonidan axborot-kommunikatsiya texnologiyalari sohasidagi kasb-hunar kollejlari uchun tavsiya etilgan.

Mualliflar:

Sh. A. Nazirov – fizika-matematika fanlari doktori, professor;
Gyunter Divald – Germaniya rivojlantirish xizmati (DED) mutaxassisi, "Axborot-kommunikatsiya texnologiyalari (AKT) sohasida kasb-hunar ta'limini rivojlantirishga ko'maklashish" – loyihasi eksperti.

Taqrizchilar:

R.D.Aloyev – O'zMU professori, fizika-matematika fanlari doktori;
M.O.Rahimov – Hamza nomli kompyuter texnologiyalari kasb-hunar kolleji maxsus fan o'qituvchisi;
H.N.Naxangov – Hamza nomli kompyuter texnologiyalari kasb-hunar kolleji maxsus fan o'qituvchisi.

Nozirov Sh.A.

Dasturlash asoslari: Axborot-kommunikatsiya texnologiyalari sohasidagi kasb-hunar kollejlarining "Axborot-kommunikatsiya tizimlari (3521916)" mutaxassisligi talabalari uchun o'quv qo'llanma. / Sh. A. Nazirov, Gyunter Divald; O'zbekiston Respublikasi Oliy va o'rta maxsus ta'lim vazirligi O'rta maxsus, kasb-hunar ta'limi markazi.– T.: Sharq, 2007. – 200 b.

1. Gyunter Divald.

BBK 32.973-018ya722

ISBN 978-9943-00-212-8

9725

MUNDARIJA

2-Qism. Visual basic tili

SO‘Z BOSHI

Mazkur o‘quv qo‘llanma "Dasturlash asoslari" maxsus texnologiyasiga oid bo‘lib, u AKT sohasining O‘zbekistondagi quyidagi yangi yo‘nalishlariga mo‘ljallangan:

- Biznes yo‘nalishidagi AKT tizimlari bo‘yicha mutaxassis.
- Dasturlash yo‘nalishidagi AKT tizimlari bo‘yicha texnik.
- Elektronika yo‘nalishidagi AKT tizimlari bo‘yicha texnik.

Ushbu o‘quv qo‘llanma dasturlashning muayyan tillaridan qat‘iy nazar dasturiy ta’minot ishlab chiqish va ma’lumotlar bazalarini yaratish bo‘yicha bilimlar asoslarini mujassamlashtirgan. Ishlab chiqish loyihasi va loyihani rejalashtirishdan kelib chiqib, o‘quvchilarga tizimli, tarkiblangan dasturlash usullari va texnologiyalari hamda uning hujjatlari beriladi.

Bundan tashqari unda modulli dasturlash usullari, dasturiy modulning asosiy tavsiflari, dastur tuzulmasini ishlab chiqish usullari, dastur tizimi nazorati hamda dasturiy modulni ishlab chiqish tarkibi, tarkibiy dasturlash va bu yerda ishlatiladigan qadam-baqadam detallashtirish va dasturiy modul ustidan nazorat, shuningdek, dasturiy modullarni yaratishda ob’ektli (bu so‘z keyingi o‘rinlarda ob’ekt tarzida o‘qilsin) yondashuv to‘g‘risida qisqacha ma’lumotlar berilgan.

Dasturiy ta’minot (DT) sifati bugungi kunning muhim mavzusi bo‘lib, ushbu masalaga o‘quv qo‘llanmada alohida bob bag‘ishlangan. Hujjatlarni yuritishning test usullari va texnikalari DT ning sifati bilan bog‘liqki, ular ham mufassal bayon qilingan. Dasturlash tillarining axboroti esa darslikni bilimlar asoslari bo‘yicha tugallaydi.

Yuqorida bayon etilgan fikrlar o‘quv qo‘llanmaning birinchi qismiga tegishli bo‘lib, bu tushunchalar dasturlash texnologiyalarining asosini tashkil etadi.

O‘quv qo‘llanmsining ikkinchi qismi esa Visual Basic tilini bayon etishga bag‘ishlangan bo‘lib, bunda Visual Basic tilining Visual muhiti, tildagi o‘zgaruvchilarni nomlash, ularning ma’lumotlari tarkibi va e’lon qilish, vazifalar, o‘zgarmaslar hamda bularga oid misollar keltirilgan. Mazkur qismning keyingi boblarida esa tilning asosiy matematik operatorlari hamda dasturni boshqarish va nazorat qilish tuzilmalari bayon etilgan.

Keyingi darsliklarda o‘quvchi ma’lumotlar bazalarini tarkiblangan dasturlar ishlab chiqish bo‘yicha muayyan bilimlar oladi va bu bilimlarni mashqqa oid topshiriqlar yordamida to‘ldiradi. Bu darslik shu bilan parallel holda va hatto undan ham ko‘proq hollarda o‘quvchiga uning bitiruv imtihoniga qadar doimiy hamroh bo‘lishi kerak deb o‘ylaymiz.

O‘zbekistonning AKT yo‘nalishidagi kollejlari o‘qituvchilar kengashiga hamda Texnik Hamkorlik bo‘yicha Germaniya Jamiyati (GTZ) ekspertlar komandasiga mamlakatning o‘ziga xos sharoitlariga moslashtirilgan "Dasturlash/Ma’lumotlar bazasi" maxsus texnologiya kontseptsiyasining yaratilishiga olib kelgan ko‘p sonli bahs-munozaralar, talab va istaklar uchun samimiy minnatdorchilik bildiraman.

Toshkent, 2006 yil, noyabr
Nazirov Sh.A.
Gyunter Divald

O‘QUV QO‘LLANMADAN FOYDALANISH BO‘YICHA KO‘RSATMALAR

Ushbu o‘quv qo‘llanma standart dasturlash, ma’lumotlar bazalarini dasturlash va WEB dasturlash sohasining bundan keyingi barcha bosqichlari uchun asosdir, chunki bu o‘quv qo‘llanmada dasturlash texnologiyalari bayon etilgan. Dasturlash texnologiyalari esa barcha dasturlash tillarida ishlatiladi.

O‘qituvchi o‘quvchilar tomonidan dasturlashning tavsiya qilingan usul va texnikalarini yaxshi o‘zlashtirishiga va ayniqsa, yagona va aniq atamalardan foydalanishga e’tibor qaratishi kerak. Bu bilan turli tillardan foydalanishda yuzaga kelishi mumkin bo‘lgan anglashilmovchiliklardan qutilishga muvaffaq bo‘linadi. AKT mutaxassislari dunyosi o‘zining alohida tiliga ega va faqat atamalarni to‘g‘ri qo‘llash dasturlash jarayonida ishtirok etuvchi odamlar o‘rtasidagi o‘zaro aloqaning bekamu ko‘st bo‘lishini ta’minlaydi. Alohida muhim atamalar kitobda qo‘shimcha ravishda inglizcha nomlari bilan ko‘rsatilgan. Bu xalqaro doirada (mas. Tenderlarda, ingliz adabiyotida yoki ingliz tilidagi maqolalarni Internet qidiruvida) o‘zaro aloqa uchun katta ahamiyatga ega va zarurat bo‘lganda bu atamalar o‘quvchilar tomonidan qo‘shimcha maxsus ingliz tili (kasblar uchun ingliz tili) darslarida o‘rganilishi kerak.

Ushbu darslikning mazmuni o‘quvchilar tomonidan boblar bo‘yicha o‘rganilishi kerak. Keyingi boblarda oldingi boblardagi ma’lumotlar va atamalardan foydalanilishi sababli, boblar tartibi ongli ravishda tanlandi. Har bir bobning o‘quv materialini amaliy topshiriqlar va mashqlar yordamida chuqurlashtirish lozim.

Har bir mavzu oldidan qora rang matn bilan qisqacha umumlashtirilgan shaklda o‘quv maqsadi taqdim qilingan:

> ***O‘quv maqsadi***
>
> O‘quvchilar uchun alohida muhim ma’lumotlar va asosiy atamalar bobda kulrang matn maydonlarida ko‘rsatilgan:
>
> ***Alohida muhim ma’lumotlar, qoidalar va asosiy atamalar eslab qolish uchun qora matn yordamida ko‘rsatilgan.***

Nihoyat, bu o‘quv qo‘llanma dasturlash sohasidagi asosiy mashg‘ulotlarni qo‘lga kiritish uchungina xizmat qilmasdan, balki o‘quvchiga barcha amaliy mashg‘ulotlarda hamda bundan keyingi dasturlashda hamroh bo‘lishi kerak.

1. DASTURLAR ISHLAB CHIQISHNING LOYIHAVIY USULLARI

O'quv maqsadi

O'quvchi dasturlar ishlab chiqishning eng muhim loyiha usullarini biladi va ularni misollardan biri orqali tushuntirib berishi mumkin.

Tarkibi

- Modul texnikasi.
- Faza chizmasi.
- Top Down - loyiha (pasayuvchi loyiha).
- Boshqa loyiha usullari.
- Ob'ektga mo'ljallangan tahlil.

1.1. DASTURIY TA'MINOT ISHLAB CHIQISH MASALASINING QO'YILISHI

Apparat vositalari sohasida ma'lumotlarga ishlov berish va ularni qayta ishlah so'nggi yillarda konstruktsiyada va texnologiyada jiddiy muvaffaqiyat qozondi. Har ikki-besh yilda yangilanadigan kompyuterlarning yanada samaraliroq avlodlari buning natijasidir. **Dasturiy ta'minot sohasi** o'tmishda mahsulotning hayotiy sikli bilan 10 yil atrofida ishlardi.

- Amaliy dasturiy ta'minot sohasidagi mahsulotning ancha uzoq hayotiy sikli shunga asoslanadiki, professional talablar apparat vositalarining texnik imkoniyatlari singari tez o'zgarmaydi.

Apparat vositalarining o'zgarishi, shuningdek, yangi kompilyatorlar (Compiler) va operatsion tizimlar paydo bo'lishi sababli dasturiy ta'minot xizmat muddati davomida o'zgartirishlarning katta miqdori zarur bo'lib qoladi. Ma'lum darajada "yuqoridan pastga" qoidasi ishlamaydi, chunki yangi muhit eski dasturlarga xizmat ko'rsata olmaydi va eski turdagi kompilyatorlar komandasini uzoqroq tutib turmaydi. Bundan tashqari mazkur yangi vositalar eski dasturlashni osonlashtiradigan yangi vositalarning katta miqdoriga ega.

Keyingi sabab shuki, kompilyatorlarning dolzarb versiyasi apparat vositalari yangi platformasidagi ortiq ishlashga qodir emas (ya'ni funktsional bo'lmay qoladi) va ularni tegishlisi bilan almashtirishga tayyorgarlik ko'rish lozim. Bu asosda ko'p hollarda iloji boricha ko'proq dasturlarni tez o'zgartira olish uchun ularni notartib deb atalgan o'zgartirishlar natijasiga qaratish lozim. Yetarli darajada hujjatlashtirishni o'zgartirishni amalga oshirmaslik dasturlarni tushunib bo'lmaydigan qiladi.

Bundan tashqari, ko'pincha shunday ham bo'ladiki, xodim (mutaxasis) tashkilotni tark etadi, dasturlarda xatolar bo'lib, hujjatlar to'liq bo'lmaydi va natijada bu dasturdagi Nou-Hau yo'qotiladi. Shuning uchun ayrim tashkilotlar "So'nggi chora"ni yangi tizimni qo'lga kiritishda ko'radi.

Shu bilan birga tan olishmaydiki, yangi standart dasturiy ta'minotni rejalashtirish va kiritishning o'zi ko'pincha juda uzoq davom etadi va o'z kuchlari bilan ishlab chiqilgan va yana kutish kerak bo'ladigan bir qator to'ldirishlar bilan tugaydi.

Mana shu sabablarga ko'ra sanoati industrallashgan mamlakatlarda eski odatlarni yengib o'tish yoki texnik xizmat ko'rsatish tarmog'i bo'yicha dasturiy ta'minot sohasida mashg'ul bo'lgan barcha xodimlarning 50 dan 80 foizigina mehnat qiladi. Qolgan 20-50 foiz xodim tizimli va amaliy dasturiy ta'minotni ishlab chiqish va uni kelgusida rivojlantirish bilan shug'ullanadi.

O'zbekistonda sobiq sovet davridan qolgan zamonaviy dasturiy ta'minot bilan baravar qo'llaniladigan va albatta, dasturiy ta'minot bilan soddalashtirilmagan bir qator o'ziga xos dasturiy ta'minot mavjudki, mamlakatda umuman olganda bu o'ziga xos xususiyat bilan qiyoslasa bo'ladigan vaziyat hukmronligini ifoda etadi. Bu yerda dasturchilar uchun dasturiy ta'minot xizmatini ko'rsatish, uni bundan keyin rivojlantirish (takomillashtirish) bo'yicha, zarurat tug'ilganda esa mamlakatning iqtisodiy va ma'muriy jarayonlari uchun zarur bo'lgan yangi dasturiy ta'minotni ishlab chiqish bo'yicha faoliyatning keng maydoni mavjud. Bu jihatdan mazkur o'quv qo'llanmaning "Xizmat ko'rsatish va bundan keyingi rivojlantirish" masalalari bo'yicha kitobning alohida bobida ma'lumot beriladi.

Keyingi mulohozalar eng avvalo, dasturiy ta'minotning turlari va ularning vazifalari haqidagi tasavvurlar dasturlar yoki butun bir amaliy tizimlarni loyihalash, amalga oshirish va xizmat ko'rsatishdagi harakat usullari va chizmalari bilan birga beriladi.

1.2. DASTURIY TA'MINOT VA ULARNING TURLARI

Inson va ma'lumotlarga ishlov berish o'rtasidagi aloqa kompyuter dasturlari orqali boshqariladi. Markaziy protsessor (CRI) belgilari to'plami ikkita belgigacha "0" va "1" yoki "yoqilgan" va "o'chirilgan" bilan chegaralangan. Inson harflardan, raqamlardan,tasvirlardan foydalanadi. Inson va mashina o'rtasidagi interfeysni (muloqotni), kodlashtirishni dasturiy ta'minot tartibga solib turadi. Bu vazifadan tashqari kompyuter dasturi **operatsion tizimi** ko'rinishida kompyuterning butun apparat qismining boshqarilishini tartibga soladi. Shunday qilib dasturiy ta'minot ikki qismga bo'linadi:

- tizimli dastur ko'rinishidagi apparat qismining boshqarilishi va
- muammolarning yechilishiga qaratilgan amaliy dasturlar

Bu bilan bir qatorda tizimli va amaliy dasturlar ishlab chiqish va kompyuter xizmatini yengillashtirish uchun bir qator **dasturlash tillari** va qo'llash servisi mavjud.

Dasturni aniqlash:

Dastur - bu kompyuter uchun tushunarli bo'lgan buyruqlar ko'rinishidagi ishchi konstruktsiyalarning takrorlanib turuvchi izchillik (ketma-ketlik) bo'lib, bu buyruqlar ma'lumotlarga foydalanuvchi istagan shaklda ishlov berish uchun barcha apparat vositalarini faollashtirish, boshqarish va nazorat qilishga xizmat qilishdir.

Tizimli dastur shu bilan birga, kompyuter va unga tegishli periferiyalardan foydalanishga imkon beradi. Tizimli dasturlarga foydalanuvchining ma'lumotlarga ishlov berish tizimining ishlash tamoyillari bo'yicha keyingi texnik belgilarga ega bo'lishini talab qilmasdan, masalani kiritish va chiqarish boshqaruvini tayyorlaydigan operatsion tizimlarini o'z ichiga oladi.

Xizmat uchun belgilangan va yordamchi dasturlar ham tizimli dasturlarga oid bo'lib, masalan formatlash va nusxalashda axborot tashuvchilar bilan muomala qilish singari operatsion tizimlar bilan munosabatni yengillashtiradi. Windows XP, Windows 2003 mijoz-server arxitekturali Novell, UNIX va LINUXning har xil variantlari mashhur operatsiya tizmilaridandir.

Professional va kundalik masalalar qo'yilishini hal qilish uchun iqtisodiy, texnik va ilmiy sohadan **amaliy dasturlar** foydalaniladi

yoki ishlab chiqiladi. Foydalanuvchilar va qo'llanish miqdoriga ko'ra soni va ushbu amaliy dasturlarni yana bo'laklarga bo'lib chiqish mumkin.

Foydalanuvchilar miqdori	**Qo'llanish miqdori**	**Amaliy dasturiy ta'minot**
Ko'p	Ko'p	Standart dasturiy ta'minot
Ko'p	Bitta / Oz paketi	Amaliy dasturlarning tarmoq
Bitta / Oz	Bitta / Oz	Alohida dasturiy ta'minot

Standart dasturiy ta'minot quyidagicha ko'rinishda deyarli har qanday masala qo'yilishi uchun amal qiladi, masalan:

- Matnga ishlov berish (masalan, MS Word).
- Elektron jadvallar bilan ishlash (masalan, MS Excel).
- Ma'lumotlar bazalari (masalan, MS Access, Oracle).
- Grafiklar (masalan, Visio, CorelDraw).
- Nashriy tizimlar (Desktop Publishing) (masalan, Adobe Pagemaker, QuarkXPress).
- Loyihalash (masalan, MS Project).

Standart dasturiy ta'minot foydalanuvchilarning keng doirasi talablarini hisobga olgan holda ishlab chiqilgan va bu bilan eng ko'p umumiy vazifalarni namoyon qilgan bir paytda, soliq idorasi, arxitektorlar, mebel tayyorlovchilar kabi muayyan kasbiy guruhlar tegishli talablarga muvofiqlashtirilgan tayyor xizmatlar ko'lamiga ega bo'lgan **amaliy dasturlarning tarmoq paketlari** deb ataluvchi dasturlardan foydalanadilar, ya'ni bunday dasturlarga talabgor bo'ladilar.

Agar xizmatlarning bunday o'ziga xos tarmoq ko'lami alohida tashkilot muammolarini hal qilish uchun baribir haddan tashqari umumiy bo'lib chiqsa, unda bir yoki bir nechta tashkilotlar talablariga moslashuvchi **alohida dasturiy ta'minot** ishlab chiqish imkoniyatigina qoladi, xolos. Bu yerda misol sifatida bir yoki bir nechta tashkilotlar tomonidan foydalaniladigan mutlaqo o'zgacha ishlab chiqarish mashinalarini boshqarish uchun qo'llanuvchi ishlab chiqarishni rejalashtirish tizimini tasavvur qilish mumkin.

Yuqorida bayon qilingan fikr-mulohazalarga muvofiq texnik yoki iqtisodiy sohalardagi muammolarni ma'lumotlarga ishlov berish

texnikasi va texnalogiyalari yordamida yechish uchun turlicha alternativalar (yondoshuvlar) mavjud:

• Mavjud dasturiy ta'minotdan foydalanish yoki uni qo'lga kiritish.

• Yangi standart, tarmoq yoki alohida dasturiy ta'minotni ishlab chiqish.

Bu orqali **o'z kuchlari** bilan yoki ishlab chiqish haqidagi yoki **chetdan taklif qilingan mutaxassislar** tomonidan ishlab chiqish haqidagi masalaga bevosita bog'liq bo'lib, u har doim ham tashkilotning ichki "Nou-Xau"si tomonidan aniqlanavermaydi. AT tashkilotiga dasturiy mahsulot ishlab chiqishni topshirish foydalimi yoki bunday dasturiy loyihani o'z tashkilotida amalga oshirish foydalimi degan savolga javob beradi va mohiyatiga ko'ra ishlab chiqarish iqtisodiy tabiatini namoyon qiladi (xarajatlar, foydalar, tahlil).

Ushbu savolning yechimi, xususan biznesga yo'naltirilgan AKT tizimlari bo'yicha mutaxassislarning vazifalariga kiradi. Shuning uchun ham bu o'rinda ushbu mavzuga bundan keyin qaytilmaydi, faqat bir mulohazani qayd qilamiz, ya'ni dasturiy loyihaning xarajat-foyda, ishlab-chiqarish iqtisodiy tahlili masalaning fanlararo tipik qo'yilishi bo'lib, bunda O'zbekistonda o'qitiladigan barcha yo'nalishga AKT mutaxassislarining hamkorligi talab qilinadi.

1.3. FAZA CHIZMASI VA MODUL TEXNIKASI

Dasturiy ta'minot ishlab chiqish jarayonini rejalashtirish, boshqarish va nazorat qilishning keng tarqalgan usuli - bu dasturiy ta'minot ishlab chiqishda ishtirok etgan barcha shaxslarning qat'iy sur'atda **faza chizmasining** qo'llanilishidir.

Bu bobda 3 ta keng tarqalgan faza chizmalari tavsiflab beriladi:

• Forward Engineering (Pog'onali model).

• Spiral model.

• V model.

Barcha 3 ta sxema (faza) uchun umumiy bo'lgan amalga oshirish fazasida **modulli texnikasi** qo'llaniladi, shu bilan birga **Top down loyiha** va Bottom up loyihalashlardan yondashuvning ikkita usuli ajralib turadi. Dasturiy ta'minot ishlab chiqishni davriy bo'laklarga (= Phasen) bo'lish haqidagi asosiy g'oya uyg'un umumiy jarayonni ishlab chiqishni pog'onama-pog'ona bosqichiga

bo'lish istagidir. Bu bilan dasturiy ta'minotni tayyorlash loyihasi ancha aniq bo'ladi, hatto bunda ishlab chiqishning tegishli darajasi esa bevosita ishtirok etmaganlar uchun ham ochiq-oyding namoyon bo'ladi.

Bu bobdan keyingi bobda hujjatlashtirishni olib borishning turli texnikalariga ko'rsatmalar beriladi.

1.3.1. Forward Engineering (Pog'onali model)

Forward Engineering - model, shuningdek, pog'onali model deb nomlangan model bir-biridan ajratilgan to'rt fazadan iborat: talablar tahlili, ixtisoslashtirilgan reja, dizayn va amalga oshirish. Har bir faza keyingi faza boshlanmasidan oldin qabul qilib olish bilan tugaydi.

Pog'onali kaskad modeli dasturiy ta'minot ishlab chiqish usulining ana'naviy modelini bildiradi. Pog'onali modelda har bir faza yaxshi aniqlangan boshlang'ich va oxirgi nuqtalarda bir qiymatli natijalarga ega bo'ladi. Bu yerda har bir faza faqat bir marta bajariladi va natijasi hujjatlanadi.

"Kaskad", ya'ni pog'ona nomi (ingl.: Waterfall Model) fazaning kaskad ko'rinishidagi tanlab olingan grafik tasavvuridan kelib chiqadi (1.1-rasmga qarang).

Agar qaysidir fazada nimadir noto'g'ri ketsa, unda sodda modelning kengaytirilishi (qaytuvchi kaskad modeli) bitta pog'ona yuqorida xatoni bartaraf qilish mumkin bo'lishi uchun qadamba-qadam kaskadning "yuqorisiga harakat qilish" imkonini beradi.

Kaskad modelidan asosan rejalashtirish fazasida talablar, natijalar va jarayonlar nisbatan aniq bayon qilinadigan joyda foydalaniladi (1.1-jadval).

1.1-jadval

"Kaskad" modeli

Faza	Tarkibi / Natijalar
1	2
Talablar tahlili (Tizimli tahlil)	Tashkiliy (maxsus) tarkibiy qismlar bayoni va realizatsiya tomonidan ajralish (shuningdek, u daliliy tahlil deb ham ataladi).

1	2
Ixtisoslashtirilgan reja	Yaratilishi va dasturlanishi kerak bo'lgan maxsus tarkibiy qismlar va holatlar bayoni, shuningdek, ularning tashqi muhit bilan o'zaro aloqasi (foydalanuvchilar interfeysi va b.q.) (hisob-kitob aloqasi deb ham ataladi)
	→ **Texnik topshiriq**
Dizayn	Dasturiy texnik qismlar va holatlar bayoni (ma'lumotlarning tuzilishi, dasturlar tuzilishi va b.q.)
	→ **Tizimli tasniflash**
Amalga oshirish	Dasturlar, ma'lumot bazalari va h.k. ko'rinishidagi dizayn tavsifini amalga oshirish (faqat bu yerda haqiqatda dasturlash sodir bo'ladi).

1.3.1.1. Faza 1: Talablar tahlili

Talablar tahlili doirasida to'la va muayyan shaklda yechilishi lozim bo'lgan maxsus muammolar va barcha qo'shimcha sharoitlarni to'laligicha to'plash uchun avval **haqiqiy ahvol tahlili** o'tkaziladi. Bu o'rinda texnik yoki ishlab chiqarishning iqtisodiy masalasi qo'yilishiga kiradigan axborot tizimining alohida qismlarini o'rganish kerak. Bundan keyin masalan, quyidagi masalalarni hal qilish kerak:

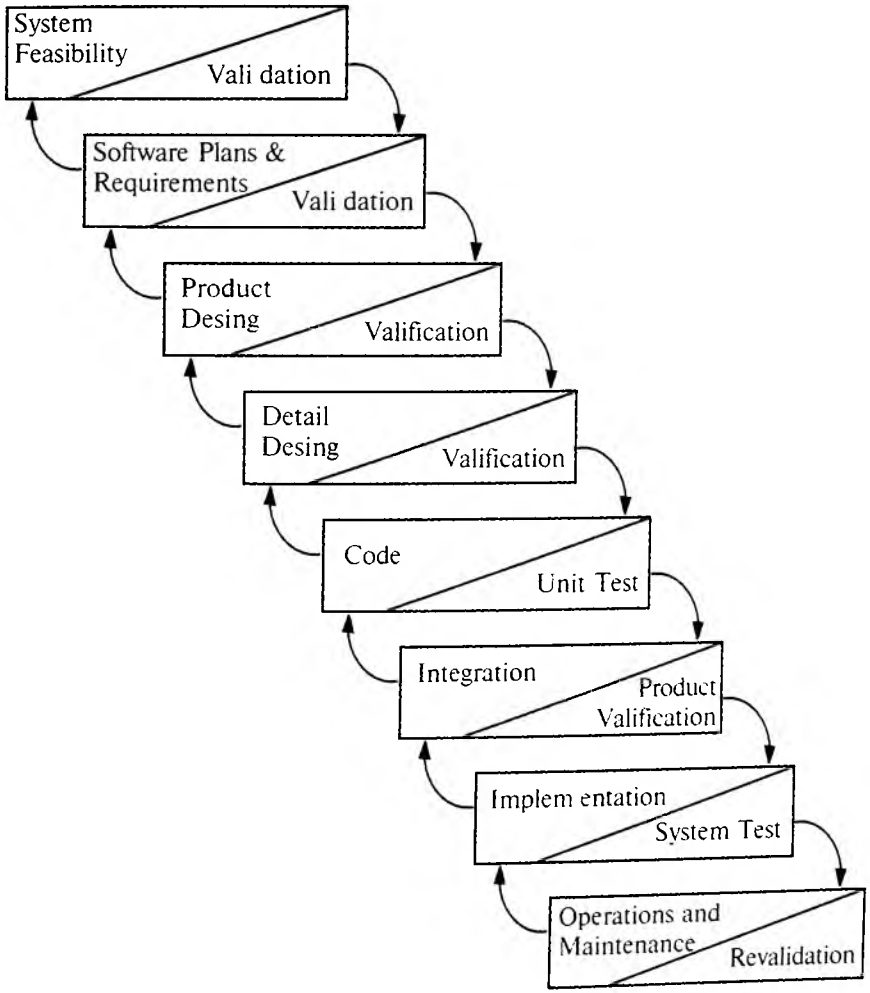
- Tashkilotning qanday joyida (masalan, bo'limlarda yoki o'rinlarda) qanday axborot yoki ishchi jarayonlar paydo bo'ladi?
- Aniqlangan axborot yoki ishchi jarayonlar haqida ularning funktsiyalari va o'zaro aloqalariga nisbatan qanday mulohazalarga yo'l qo'yilishi mumkin?
- Qayerda, qanday natijalar paydo bo'ladi va qayerda yoki kimga ular kerak bo'ladi?

Haqiqiy ahvol tahlili natijalarini rasman ko'rsatish uchun masalan, ma'lumotlar oqimi diagrammasiga murojaat qiladi.

Haqiqiy holat tahlili doirasida yaratilgan dasturiy yechimlar yordamida ma'lumotlarni qayta ishlash nuqtai nazaridan ularning muayyan tadbiqini ko'rsatmasdan turib hal etish mumkin.

Bu esa maxsus rejaga ko'ra keyingi faza bilan farqlanishini ko'rsatadi.

Talablarni tahlil qilish doirasida avvaliga haqiqiy ahvol tahlili o'tqaziladi, bunda dasturiy taqsimot tizimining alohida qismlari o'rganiladi va hujjatlashtiriladi.



1.1-rasm. Kengaytirilgan "kaskad" modeli (oraliq nazoratli model). Alohida fazalar "kaskad" shaklida ko'rsatilgan. Agar fazalardan birida biron narsa noto'g'ri ketayotgan bo'lsa, oldingi fazaga qaytishi mumkin. Fazalar bir-biridan qat'iy ajratilgan va shunga yarasha hujjatlashtirilgan natija bilan tugaydi

1.3.1.2 Faza 2: Ixtisoslashtirilgan reja

Ixtisoslashtirilgan rejada (ko'pincha hisob-kitob rejasi deb ataluvchi) ishlab chiqish zarur bo'lgan dasturiy ta'minot bo'yicha ma'lumotlar bayon qilinadi. Bu esa, jumladan quyidagi tarkibiy qismlarning ko'rib chiqilishini va quyidagi savollarga javoblarni bildiradi:

1.2-jadval

Ixtisoslashtirilgan rejaning tarkibiy qismlari

Maxsus rejaning tarkibiy qismi	Savolning qo'yilishi
Tizimli maqsadlar	Rejalashtirilgan tizim qanday muhim vazifalarni bajarishi kerak?
Foydalanuvchilar modeli	Ishlab chiqarish iqtisodiy yoki texnik vazifalarga nisbatan bo'lg'usi foydalanuvchilar uchun ular bajarishi lozim bo'lgan qanday talablar yuzaga keladi? Informatikaga oid qanday zaruriy maxsus bilimlar foydalanuvchilarda dasturiy ta'minot bilan muomalada bo'lish uchun zarur hisoblanadi? Dasturiy tizimdan ish jarayonida qanday holatlarda tez-tez foydalaniladi?
Baza mashinasi	Ishlab chiqiladigan dasturiy ta'minot uchun apparat vositalari platformasiga nisbatan (bunga kiritish va chiqarish maxsus qurilmasi ham kiradi) qanday eng kichik talablar asosida yuzaga keladi? Va uning qanday tizimli dasturiy ta'minot (operatsiya tizimi) uchun qo'llanishi rejalashtiriladi?
Foydalanuvchi interfeysi	Alohida foydalanuvchilar uchun, masalan, kutishga ruxsat, printer muhiti va dasturiy muhitga nisbatan qanday ixtisoslarni oldindan ko'zda tutish kerak? Dasturiy ta'minot tizimi foydalanuvchi xatosini (mas. ma'lumotlarni nojoiz kiritish) va dasturning o'zini noto'g'ri tutishini qanday qabul qiladi va xato haqida foydalanuvchiga qanday xabar beradi? Foydalanuvchi bilan dialog qanday shaklda o'tkaziladi? Konsolli interfeyslar haqida gap ketayaptimi yoki grafik interfeyslarga intilishmoqdami?

Ixtisoslashtirilgan reja doirasida 2-jadvalga muvofiq ma'lumotlarga ishlov beruvchi qurilmalar tavsifi va ishlab chiqilayotgan dasturiy ta'minot uchun aytib o'tilgan vazifa qo'yilishi javoblarni o'z ichiga

oladi. Hozircha amalga oshirishning alohida dasturiy texnik jihatlari haqida mulohaza bildirishdan tiyilish kerak. Huddi shu tufayli ishlab chiqilayotgan dasturiy ta'minot tizimining dasturiy-spetsifikatsyalini amalga oshirish tafsilotlariga berilmasdan, dasturiy ta'minotga talablarni o'rganish bosh vazifadir. Bu tartib, boshqalar qatorida, dasturiy ta'minotni turli-tuman operatsion tizimlariga keyinchalik ko'chirishga imkon beradi (ta'minot dasturining harakatchanligi).

Tadqiqot natijalari va haqiqiy ahvol hamda ixtisoslashtirilgan reja tahlili, masalalar qo'yilishiga javoblar umumlashtirilgan holda **texnik topshiriqda** to'planadi. Agar dasturiy ta'minotni ishlab chiqish AKT firmasiga topshirilsa, unda ushbu texnik topshiriq pudratning tuziladigan shartnomasi yoki mehnat shartnomasi uchun asosdir. Bu shartnomada dasturiy ta'minotni ishlab chiqish bo'yicha firma tomonidan bajarilishi kerak bo'lgan va pudratchi hamda buyurtmachi uchun majburiy bo'lgan talablar qat'iy belgilanadi.

Texnik topshiriq yuqorida ko'rsatilgan faza bo'yicha natija sifatida maxsus aniq (oxirgi) rejaga ega. Muayyan realizatsiya (dasturlash) shubhasiz avvaliga faqat xomaki holda tavsiflanganligi sababli, mazkur realizatsiya va tadbiq qilish nuqtai nazaridan texnik topshiriq faqat dasturiy-texnik shakilda xomaki rejaga ega bo'ladi.

Ixtisoslashtirilgan reja fazasida ishlab chiqilayotgan dasturiy ta'minot masalalari tavsiflanadi. Haqiqiy ahvol tahlili natijalari bilan birga Ixtisoslashtirilgan reja texnik topshiriq ko'rinishida hujjatlashtiriladi. Texnik topshiriqdagi muayyan dasturiy texnik tafsilotlar faqat taxminan bayon qilinadi.

1.3.1.3 Faza 3: Dizayn

"Dizayn" fazasida (shuningdek dasturiy loyiha deb ham ataladi texnik topshiriqdagi dasturiy-texnik taxminiy reja bundan keyi aniq (oxirgi) rejaga qayta ishlanishi kerak. Ushbu aniq reja keyinro amalga oshirilishi kerak bo'lgan dasturlar yoki dasturiy ta'mino tizimi uchun asosni, platformani yaratadi.

Dizayn - bu dasturiy-texnik aniq reja va bundan keyin dasturlash uchun asosdir. Modul yaratish dasturlashn osonlashtiradi. Dizayn yoki Top down ishlab chiqish yo'nalishid yoki Bottom up yo'nalishida ifodalanadi.

Endi eng kichik talablardan foydalanuvchi interfeysidan kelib chiqib tayanch modelining tarkibiy qismlarini (2-jadvalga qarang) batafsil tasniflash zarur. Bu bilan mufassal ishlangan kompyuter tizimi dasturiga havola qilingan va bu masalani yechish uchun zarur dasturni rejalashtirilishi mumkin.

1.3.1.4 Dizayn fazasida modullashtirish

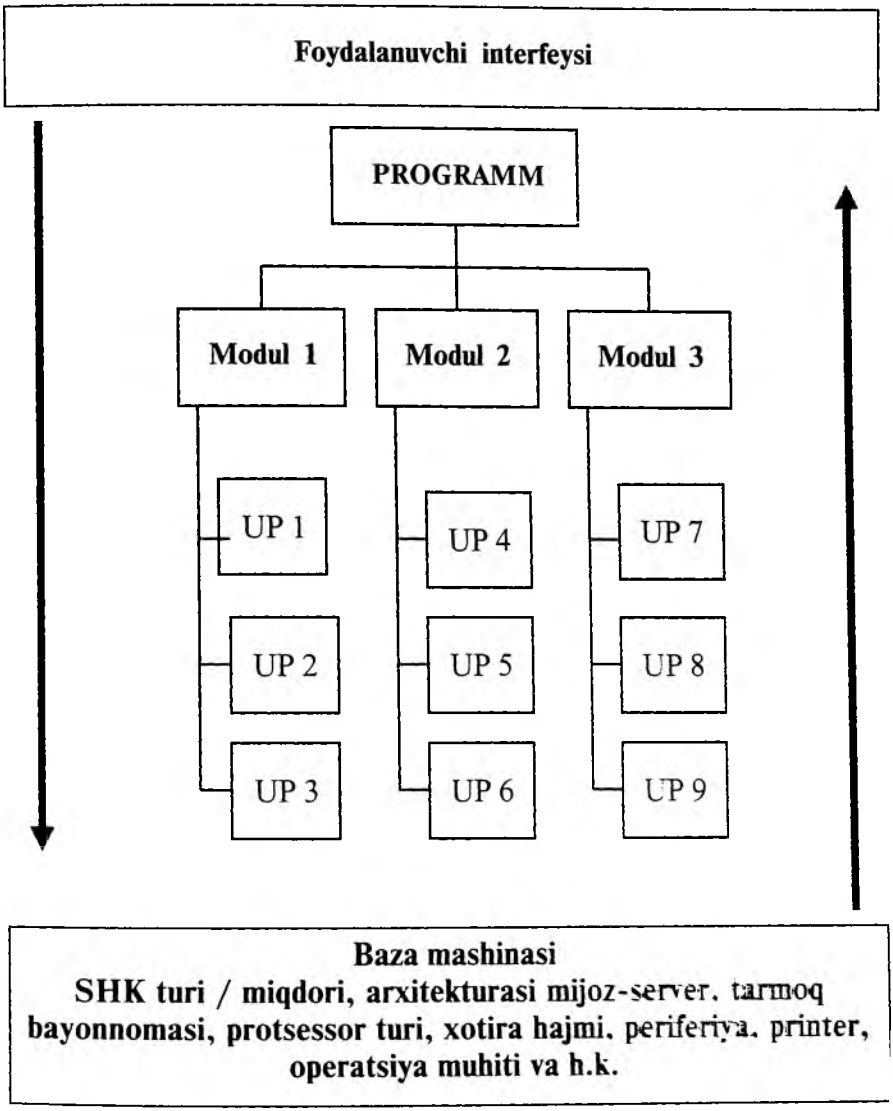
Dasturni bevosita rejalashtirish va bundan keyingi dasturlash uchun asos bo'ladigan dasturiy-texnik aniq rejani tuzish modullashtirishni amalga oshirilgandan so'ng (2.2-rasmga qarang) yaratiladigan dasturning iyerarxik tuzilishi loyihasi orqali sodir bo'ladi. Modullashtirish masalasining keng miqyosli qo'yilishini yaxshi ochiq ko'rinishda ko'rsatilishini va keyingi dasturiy kodlarning (Quellcode) kichkina yaxshi ko'zga ko'rinadigan birliklarga bo'linishni bildiradi. Ushbu bo'lingan masalalar **modullar** va ularning **dasturlaridir**.

Modullarning loyihalanishida ular mantiqiy tugallanganligi va bir-biriga bog'liq bo'lmagan holda amalga oshishi mumkinligiga diqqatni qaratish kerak. Shu sababli modullar o'rtasida tushunarli **tashqi aloqalarni** belgilash zarurki, ular orqali keyinroq alohida komponentalar (qismlar) yagona mahsulotga birlashadi.

Bu usullarning afzalligi shundan iboratki, butun dasturni tayyorlashni bir necha dasturchiga taqsimlab berish mumkin, shu bilan birga ular ish usullari yoki ularning modellarini amalga oshirish bilimlari bilan cheklanishlari mumkin. Qolgan hollarda modullar o'rtasidagi aloqa belgilanuvchi interfeyslar orqali amalga oshiriladi. Bu eslatib o'tilgan butun tizimni testdan o'tkazishning ko'zga ko'rinadigan va soddalashgan imkoniyati uchun zamin yaratadiki, unda tarkibiy unsurlarining ish tamoyillari tegishlicha bittadan testlash uchun jalb qilinadi. Zarurat tug'ilganda dasturiy ta'minot butun tizimining yangi taxminiy xom qolipini tuzishga ehtiyoj sezmasdan, bunday modullarni shuningdek osongina almashtirish, o'zgartirish yoki to'ldirish mumkin.

Dasturning taqdim qilingan iyerarxik tuzulmasini amalga oshirish uchun asosan ikkita har xil yondashuv mavjud:

- Top down ishlab chiqish yo'nalishi.
- Bottom up tayyorlash yo'nalishi.
- Bu yondashuvlar quyida batafsilroq tushuntirilib beriladi.



1.2-rasm. Oraliq va foydalanuvchi interfeysi va tayanch mashinasi o'rtasidagi ishlab chiqish ishlanmasining intervali

1.3.1.5 Top Down ishlab chiqish yo'nalishi

Dasturiy ta'minot asosida qismiy masalalar yotgan bo'lib, ula masalaning umumiy qo'yilishi "Yuqoridan-pastga" qadam-baqadam bo'laklarga bo'lish orqali hosil qilinadi (mos ravishda foydalanuvch

interfeysiga ko'ra, 2.2 rasmga qarang). Bir necha qadam-baqadam qismiy dasturni (nimdastur(protsedura yoki funktsiya)) detallashtirish yo'li bilan taklif qilingan kompyuter tizimiga intilinadi. Boshqacha aytganda: takomillashtirishning (sifatni yaxshilashning) har bir qadamidan boshlab foydalanuvchi interfeysi va tayanch mashinasi o'rtasidagi oraliq qisqaradi.

Mazkur usul shunday hollarda juda yaxshi qo'llanishi mumkinki, unda realizatsiyani amalga oshirish, ya'ni to'la shakllantirilgan dastur foydalanuvchi nuqtai nazaridan to'la tushunarli va tasniflangan bo'lishi mumkin. Biroq bu to'laligicha tushunarli bo'lmasligi, tasniflanmasligi mumkin bo'lmaydigan uyg'un tizimlar talablariga har doim ham mos kelavermaydi.

Amalda dasturlar tez-tez soddalashtirilib turilishi kerak, chunki tashkilotdagi uyg'un texnik ishlab chiqarish iqtisodiyoti yoki huquqiy vaziyatlari o'zgarmas o'zgarishga moyildir. Bundan kelib chiqadiki, tashkilotdagi haqiqiy ahvol o'zgarib turadi va tasniflar uzoq muddatli soddalashtirish jarayonidan zarar ko'radi.

Standart ish haqi yozish yoki unchalik sezilarli bo'lmagan o'zgarishlarga muhtoj materiallarni boshqarish kabi dasturlar ushbu usul yordamida yaxshi tuzuladi.

O'zgarmas texnologik o'zgarishlarga duchor bo'lgan harakat va o'zgarishlarga boy boshqa tizimlar (mas. ishlab chiqarish tizimlari, avtomat boshqaruv tizimi va buxgalteriya hisobi tizimlari kabi) tasniflash darajasida va dasturiy darajada muntazam moslashtirib borilishi kerak. Shu sababdan esa ular uzluksiz ortib boradigan texnik xizmat ko'rsatishning o'zgarmas xarajatlariga duchor bo'ladi, chunki dasturiy tuzilmalar ularning o'zgarmas o'zgarishlari tufayli borgan sari noaniq bo'lib boradi.

Agar ishlab chiqarilayotgan dastur foydalanuvchi nuqtai nazaridan aniq va katta texnik, ishlab chiqarish iqtisodiy yoki huquqiy o'zgarishlarga uchramagan bo'lsa Top Down usulining qo'llanilishi shundagina dizayn fazasida ma'noga ega boladi.

1.3.1.6. Bottom UP ishlab chiqish yo'nalishi

Top down usulining yuzaga kelishi mumkin muammolari tufayli bu yondashuvga alternativlar ko'rib chiqsa arziydi.

Bugun Bottom Up ishlab chiqish yo'nalishidagi boshqa bir usul tez-tez qo'llanilmoqda. Bu avvaldan mavjud bo'lgan dastur

kompyuter tizimidan nima kelib chiqishini bildiradi va vazifaning qo'yilishi nuqtai nazaridan "orqaga qaytib ishlab chiqiladi".

"Yuqoridan pastga" yo'nalishida avval funktsiyalar va nimdasturlar (qismiy dastur) ishlab chiqiladi va tayanch mashinasi yaqinida joylashgan modullarga olib boriladi. Tegishli ravishda eng yuqori darajadagi modullarning keyingi ma'lumotlari orqali endi bu o'rinda qo'yilgan vazifani hal qilish uchun qadam-baqadam dastur hosil bo'ladi. Belgilangan sharoitlarda kompyuter tizimi bilan bir qatorda dasturlashda foydalaniladigan tili shuningdek, ishlab chiqarilayotgan dasturning tuzilishini belgilab beradi.

O'zining mohiyati bilan Bottom Up dasturiy ta'minotdan ko'p marotaba foydalanish nazariyasiga borib taqaladi. Turli xil foydalanuvchilar, ayniqsa AQSH va Yaponiya kabi yirik dasturiy ta'minot ishlab chiqaruvchilari tajribasidan kelib shu chiqadiki, bunda mavjud dasturiy ta'minot bir oz vaqt o'tgach o'z hayotiy siklini davom ettirishi uchun jiddiy ravishda qayta ishlanishi yoki tozalanishi kerak. Bu usul bilan dasturiy ta'minotni ishlab chiqish yoki dasturni kuzatib borish uchun tashabbus ko'pincha avvaldan mavjud tizimlardan kelib chiqadi. Buni esa Bottom Up usuli qo'llab quvvatlaydi.

Dasturiy ta'minot ishlab chiqish amaliyotida pragmatik yondashuv ham mavjud: shunday ifoda topiladiki, dasturchining harakat usullari Bottom Up va Top Down nazariyalarini birlashtirishdan iborat bo'ladi. Bundan kelib chiqadiki, ishlanma yaratishning yagona usuli mavjud emas. Har bir dasturchi vazifaning turiga bog'liq holda turlicha yondashuvlardan foydalanadi. Bundan kelib chiqadiki, dasturiy ta'minot ishlab chiqish uchun instrument vositasi yoki harakat usuli nazariyalari aralashmasi usulidan foydalanishni talab etadi.

Agar dasturchi talablarni hal qilishda mavjud dasturiy ta'minotga murojaat qilsa, dizayn fazasida Bottom Up usulining qo'llanishi doim bir xil bo'ladi. Amalda dizayn fazasidan kelib chiqadigan tizim tasnifi ko'proq Top Down va Bottom Up ishlanmalari yo'nalishining aralashmasi sifatida qo'llanadi.

Modulyarlashdan so'ng va Top Down yoki Bottom Up yondashuvlari qat'iy nazar dastur dizayni nihoyasida natija sifatida tizim tasnifiga ega bo'ladi. Bu yerda yana bir marta tasnifning tarkibiy qismlari umumlashtirilgan bo'ladi:

- Kompyuter tizimi tayanch mashinasini mufassal tavsiflash.
- Dasturiy ta'minotni ishlab chiqarish uchun zarur modullar

haqidagi umumiy tasavvur quyidagilarning biri ko'rinishida bo'ladi:

- - modul-vazifalari
- - modul-interfeyslari va
- - modul-bog'liqliklari

Hozircha dasturlash tilidagi algoritmning jarayonlar yoki ma'lumotlarning muayyan tuzilishi kabi realizatsiya qilishning aniq jihatlaridan qochish kerak. Chunki birinchi navbatda ishlab chiqilayotgan dasturiy ta'minotning xususiyatlarini yaratish kerak. Dizayn fazasi natijalarining rasmiy tavsifi uchun tasnif va hujjatlarni tuzish uchun xizmat qiladi (bu ma'lumotlar 3-bobda bayon qilingan). Hujjatlashtirishni olib borishning bundan keyingi ko'rsatmasi ushbu qo'llanmaning 8-bobida va ma'lumotlar bazalarini yaratish asosida ko'rsatiladi (mas. ob'ekt-munosabat modeli ma'lumotlarning relyatsion bazasida Entity-Relationship-Modell).

1.3.1.7 Faza 4: Amalga oshirish

Dizayn fazasi natijasi sifatida tizimning tasnifi ko'rsatilgandan so'ng, amalga oshishdan ishga yaroqli dastur (yoki ma'lumotlar bazasi) ishlab chiqish kerak bo'ladi, bu ishlab chiqish kiritish va chiqarish usullarida tasnifdagi topshiriqlarga amal qiladi. Bu o'rinda modullar tarkibli dasturlash usuli yoki ob'ektga qaratilgan dasturlash usuli orqali realizatsiya'ni amalga oshirishni hal qilish kerak. Bunga bevosita yana dasturlash uchun foydalanilayotgan til haqidagi savol ham qo'shiladi.

Bu masalani hal qilish uchun VBdagi tarkiblashgan dasturlash bo'yicha ob'ektga yonaltirilgan dasturlash tili C++ dagi va MS Access dagi ma'lumotlar bazalarini dasturlash bo'yicha darslikka murojaat qilish lozim.

Realizatsiya (amalga oshirish) fazasi so'ngida birlashtirish kerak bo'lgan modullar yig'indisi kabi hujjatlashtirilgan dastur (yoki ma'lumotlar bazasi) mavjud bo'ladi. Ishlab chiqilgan dasturiy rasmiy tavsifi boshqalar qatori quyidagilar orqali amalga oshiriladi:

- Dasturning mantiqiy chizmasi.
- Tarkibiy diagrammasi (Struktogramme).
- Modellashning yagonalashgan tili diagrammasi (UML, ob'ektga qaratilgan dasturlash bo'yicha darslik).

Aytib o'tilgan barcha texnikalar amalga oshirish fazasining bosh mahsuloti uchun, dastur matniga tushuntirish bo'lib xizmat qiladi (Quellcode).

Faqat amalga oshish fazasida bevosita dasturlash sodir bo'ladi. U hujjatlar bilan tasdiqlanishi kerak, chunki hujjatlshtirilmagan dasturlar qiymatga ega emas.

Tasnif mazmuni modullarni amaliy taqbiq qilishga muvofiqligini qayta tekshirib ko'rish uchun bevosita amalga oshishdan keyin keladigan **test fazasi** xizmat qiladi. **Modulni testlashda** barcha testdan o'tkazilgan va xatosiz modullar **integratsiya** doirasida ishga yaroqli dasturiy ta'minot dasturiga yoki tizimiga birlashadi.

Agar integratsiyadan so'ng oxirgi dastur taqdim qilingan bo'lsa, uning masalani hal qilish texnik yoki ishlab chiqish iqtisodiy layoqatini tekshiruvchi batafsil integratsiya testiga jalb qilinishi kerak. Shu o'rinda qayd etish kerakki, integratsiyali testlash dasturiy ta'minotni ishlab chiqaruvchilarning o'zlari tomonidan emas, balki yaxshisi professional, malakali foydalanuvchilar tomonidan bajarilishi kerak. Nihoyat dasturiy ta'minot buyurtmachisi va ishlab chiqaruvchisi o'rtasidagi qabul qilish va uzatish doirasida ishlab chiqilgan dasturiy ta'minotga texnik topshiriqda aytilgan va shartnomaga muvofiq majburiy talablar bajarilganligini tekshirib ko'rish kerak.

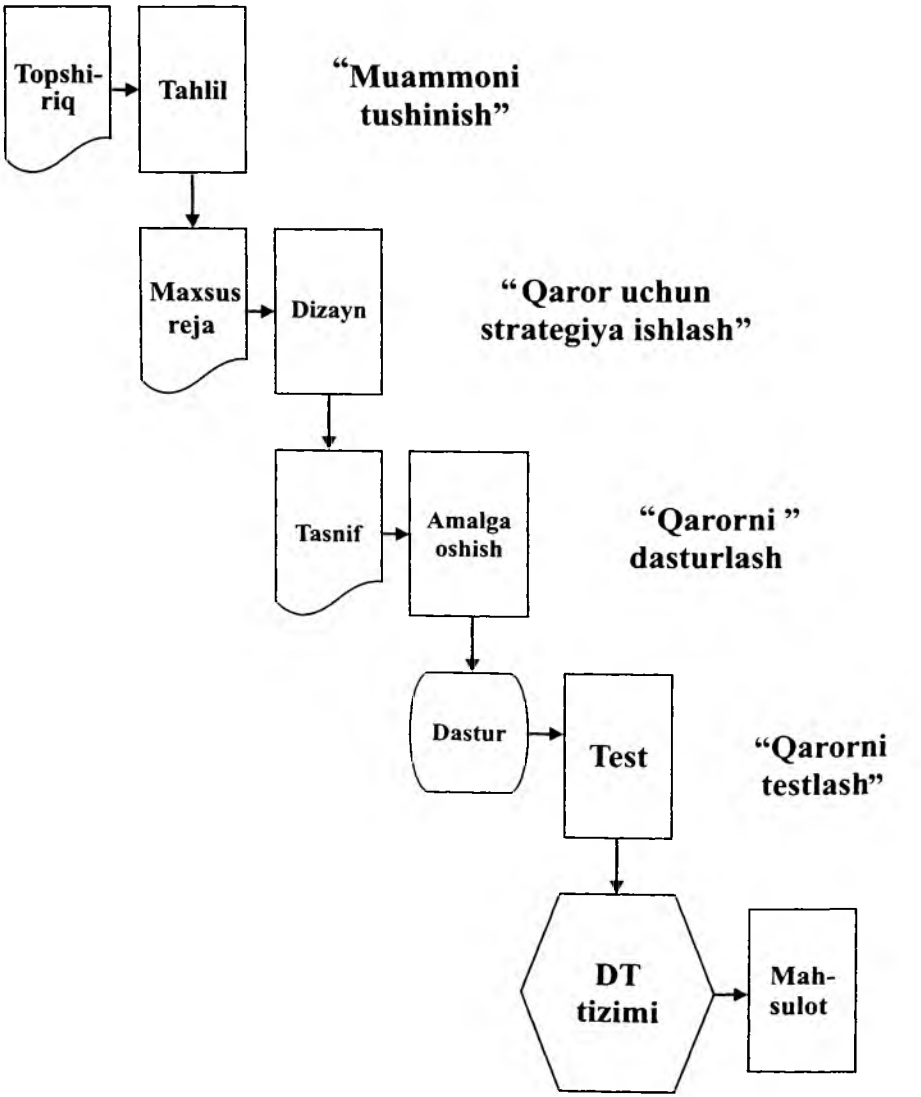
Ma'lum sharoitlarda dasturning keyingi tekshiruvi yaratilgan dasturiy ta'minot buyurtmachida mavjud ma'lumotlarga elektron ishlov berish tizimiga (dasturiy ta'minot va yoki apparat vositalari) ulanganida amalga oshadi va faqat shundagina keng qamrovli dasturiy mahsulot topshiriladi. Dasturlash munosabati bilan bu yerda dasturiy ta'minot tizimining bo'lg'usi foydalanuvchilarini o'qitish haqidagi masala ko'tariladi.

Yuqorida bayon qilingan nazariy mulohazalarni ko'zdan kechirish uchun "kaskad" modelining quyidagi batafsil rasmi orqali tushunib olish mumkin (1.3-rasmiga qarang).

1.3.2 Spiral model (Prototyping)

Forward Engineering (kaskad modeli) konseptsiyasiga qaratilgan spiral modelida alohida fazalar qat'iy ravishda keskin tarzda ketmaket keladi. Shu tarzda, masalan, buyurtmachining yaratilgan dasturiy ta'minotini o'zgartirish istagi ko'pincha bir qancha vaqtdan so'ng ma'lum bo'ladi. Aniqlangan belgilar hatto sinchiklab o'tkazilgan

tahlilda ham 100%ga namoyon bo'lmaydi yoki bo'lmasa ishlab chiqilgan dastur, tadbiq qilinishida dasturiy ta'minot tomonidan hisobga olinishi kerak bo'lgan yangi maxsus bilimlar aniqlanadi. Bunday o'zgartirishlarni belgilab bo'lingan reja fazasidagi qat'iy tamoyil holatida amalga oshirish juda qiyin, ba'zan esa umuman mumkin emas.

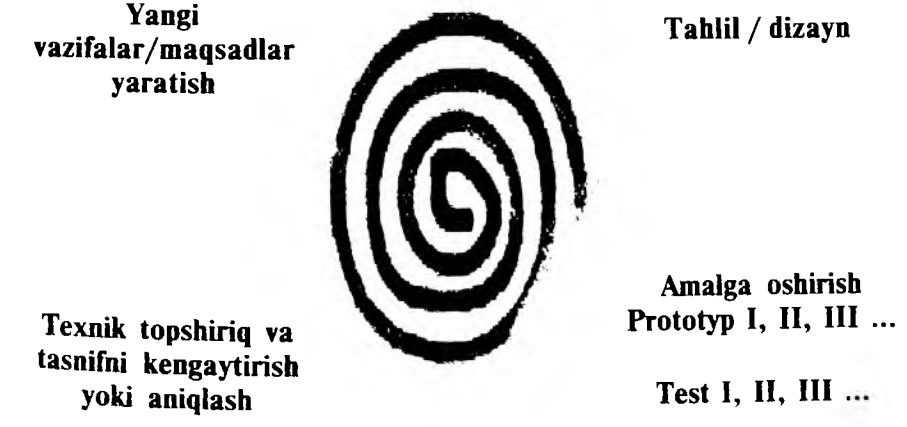


1.3-rasm. Ish modeli. Forward Engineering (kaskad modeliniki). Testlash fazasi

Shuning uchun rejalashni texnik topshiriqlar tuzishdayoq bajarish kerak. Ushbu texnik topshiriq mazmuni aniq ifodalangan hamda buyurtmachiga ham, buyurtmani bajaruvchiga ham tushunarli bo'lishi talab qilinadi. Bu o'rinda informatikaning maxsus bilimlariga nisbatan mijoz tashkilot va dasturiy ta'minot ishlab chiqaruvchi sifatidagi AKT-firmasi o'rtasidagi tafovut qanday bartaraf qilinishi kerak, degan savol tug'iladi.

Prototiplar yaratish (Prototyping) doirasidagi ushbu dilemmani yechish uchun dasturiy ta'minotning bo'lg'usi tizimining qismi singari ishga yaroqli model ishlab chiqiladi. Ekran niqoblarini, dialog oynalarini rasmiylashtirish kabi ushbu birinchi prototip xossalari ishlab chiquvchilar tomonidan ham, buyurtmachilar tomonidan ham sinaladi va bunga tegishli talablar tavsifi texnik topshiriqda aniqlanadi. Birinchi prototip tajribasiga asoslangan dasturning funksionalligi kengayib boradi va ikkinchisi bajariladi va yana uning eski va yangidan qo'shilgan xossalari testdan o'tkaziladi. Undan keyin yana texnik topshiriq doirasidagi talablar yangidan ma'lim bo'ladi yoki aniqlanadi.

Bu bilan dastur inkriment, ya'ni qadam-baqadam foydalanuvchi va ishlab chiqaruvchini jalb qilish orqali amalga oshib boradi. Bayon qilingan usulning har biri bajarilishi Forward Engineering tahlil fazasi, dizayn va testlashdan iborat o'z ishlab chiqarish siklini ifodalaydi. Bu bilan dasturiy ta'minot ishlab chiqish eski qat'iy kontseptsiyaga boshqa rioya qilmaydi, balki evolutsiya yo'li bilan o'tkaziladi. Dastur iterative yaratiladi



1.4-rasm. Spiral modelda iterativ va inkriment harakat usuli

Forward Engineering fazasi prototiplarini (Prototyping) yaratishda tahlil, dizayn, amalga oshirish va testlash spiral ko'rinishda birinchi prototipdan tayyor umumiy tizim hosil bo'lmagunga qadar bajariladi.

1.3.3 V model

V-model Germaniyada Federal ma'muriyat uchun dasturiy mahsulotlarni amalga oshiradigan ishlab chiqaruvchilar uchun majburiy ravishda buyurilgan standartdir. Model, shuningdek, avtomobil sanoatida va bank sohasida keng tarqalgan va xalqaro e'tirof qozongan.

V-model bo'yicha ishlab chiqish jarayoni dasturiy ta'minot ishlab chiqish doirasidan tashqariga chiqadi va butun instrumental muhitni yaxlitlik nuqtai nazaridan ko'rib chiqadi. V-model faoliyatning to'rtta submodel deb ataluvchi sohasini o'z ichiga oladi:

- Dasturiy ta'minotni ishlab chiqish.
- Sifatni ta'minlash.
- Konfiguratsiya'ni boshqarish.
- Loyihani boshqarish.

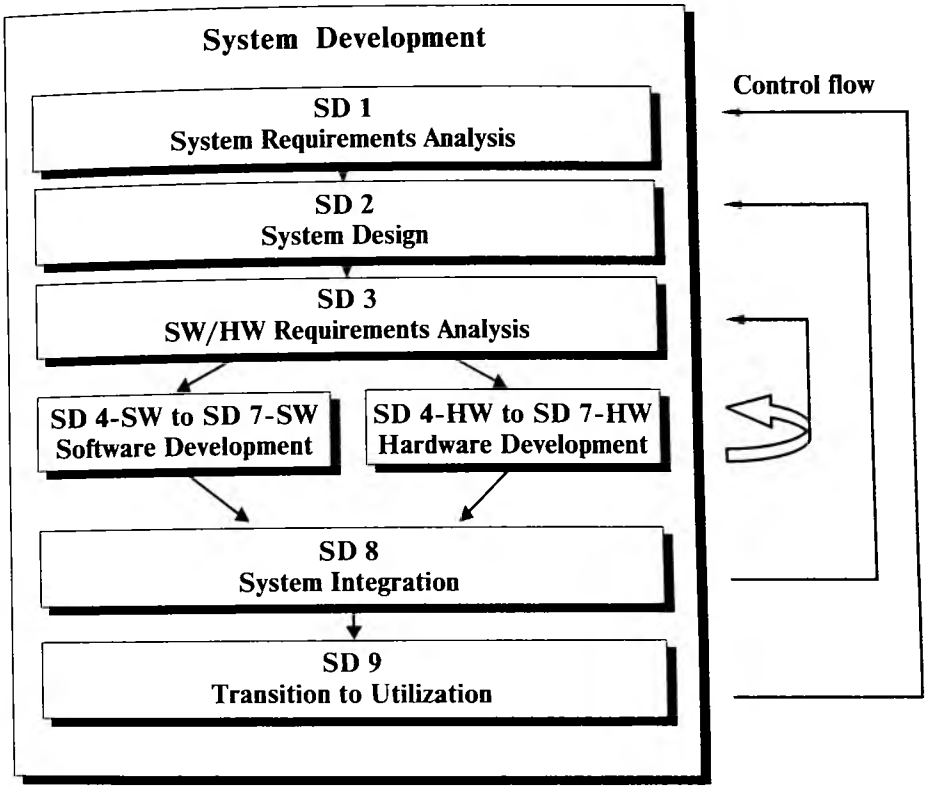
To'rt submodellarning har biri uchun standart va yagona me'yor etib bajarish shart bo'lgan harakatlar belgilangan. Har bir harakatlar uchun hujjatlar va xizmat daftarchasi bor bo'lib, ular o'z ishlaridan tashqari hujjatni ishlab chiquvchi shaxslar va bundan keyingi harakatlar haqidagi ma'lumotlarga ega. Shu bilan birga har bir hujjat uchta bo'lishi mumkin bolgan holatga ega:

- Hujjat ishlovda.
- Hujjat ko'rsatilgan.
- Hujjat qabul qilingan.

Agar hujjat faqat sifat nazorati tomonidan qabul qilingan bo'lsa, ishlab chiqarishning keyingi fazasi uchun reja topshiriqlari boshlanadi.

Dasturiy ta'minotni bevosita ishlab chiqish dasturiy ta'minot ishlab chiqish submodelida (SWE) (inglizcha: System Development (SD)) amalga oshiriladi, uning harakati 2.5-rasmda ifodasini topgan.

Kaskad modelidan farqli ravishda V-modelda nazorat harakatlari (control flow) sababli, loyihadagi barcha qatnashuvchilar o'rtasida aniqlikning yuksak darajasini, talablardagi o'zgarishlarga moslashuvchan reaksiya va dasturiy ta'minotning yuqori sifatini ta'minlaydigan sikllik teskari aloqada berilgan.



1.5-rasm. Nazorat qiluvchi harakatlarga ega bo'lgan dasturiy ta'minot ishlab chiqarish submodeli harakati

V-model o'zining butun uyg'unligida tasvirlanishi kerakki, bu mazkur darslik doirasidan chetga chiqish bo'ladi. Shuning uchun bu yerda faqat asosiy xossalargina ko'rsatilgan. To'liq tasavvurni-shuningdek - ingliz tilida www.**v-modell**.iabg.de saytida topish mumkin.

1.3.4. Modellarni qiyoslash

Klassik kaskad modelining asosiy kamchiligi fazalarning "yuqoridan pastga" qat'iy ketma-ketligidan iboratdir. Bu quyidagilarga sabab bo'ladi:

- Keyingi soddalashtirishlarning imkoni bo'lmaydi.
- Ishlab chiqishning uzoq davri.
- Turli "tillar" mavjud bo'lganda ixtisoslashtirilgan reja va dizayn o'rtasidagi tez-tez uzilish.

Faqat spiral model (Prototyping) va V-model fazalarining keskin izchilligiga (ketma-ketligida) barham beradi. Bundan tashqari, teskari aloqa mexanizmi va nazorat mexanzmlari, shuningdek, foydalanuvchini ishlab chiqish jarayoniga muntazam jalb qilinishi sababli ancha balandroq moslanuvchanlik, sifat va aniqlik taqdim qilinadi.

Shunga qaramay, klassik kaskad model bemalol o'zini oqlagan, chunki uning fazalari boshqa modellarda yana paydo bo'lishi va Forward Engineeringda, ayniqsa, yaxshi "ishlab berishi" mumkin.

Bundan dasturiy ta'minot ishlab chiqishda foydalaniladigan model bo'yicha quyidagi tavsiya kelib chiqadi.

Klassik "kaskad" model qisman dasturiy ta'minot ishlab chiqishda hamda kichkina va sodda loyihalarda boshlovchilar uchun to'g'ri keladi. Spiral model (Prototyping) kaskad modelning keyingi rivojlanishi bo'lib u ko'proq ochiq loyihalarda qo'llanishi kerak. V-model sanoatdan va jamoat huquqiy boshqaruvidan loyihada qatnashuvchilarning katta miqdori bilan yirik loyihalarda qo'llanadi.

1.3.5. Dasturiy ta'minot ishlab chiqishning taxminiy stsenariysi

Bugun AKT sohasida masalalar ko'pincha yaxlitlik nuqtai nazaridan **stsenariylar** ko'rinishida aniqlanadi. Bu amalda qo'llashga juda yaqinlashtiradi, chunki masalalarning qo'yilishi ko'proq, masalan dasturlash bo'yicha, iqtisodiyot bo'yicha va elektronika bo'yicha mutaxassislar hamkorligini talab qiladigan kompleks sohani qamrab oladi. Buning uchun maxsus ifoda "vaziyat topshirig'i" sifatida namoyon bo'ladi.

Vaziyatlarning bundan keyingi bayonidan ushbu darslikda masala va misollarning qo'yilishi uchun berib boriladigan amaliy topshiriqlarda ham har doim foydalanib boriladi.

1.3.5.1. N shahridagi avtomobil to'xtash joyi uchun to'lov

N shahri 2010 yilga borib shahar markazida avtomobil to'xtash joyi uchun to'lov undirishni rejalashtiradi. Buning uchun park avtomatlari o'rnatilgan bo'lishi kerak, unda haydovchi o'zi istagan

turish vaqtini belgilab beradi, to'lovdan keyin esa kvitansiya olib, uni endi to'xtash joyi izmiga olingan avtomobilining old oynasi ichkarisiga, ko'rinarli joyga qistirib qo'yadi.

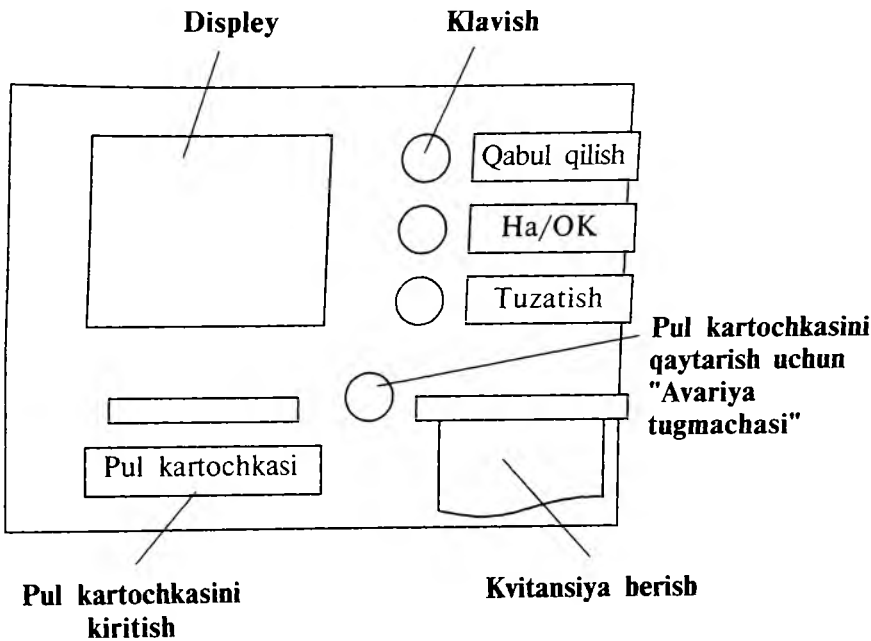
An'anaviy park avtomatlari tanga bilan ishlashlari sababli, bu xil to'g'ri tangalar O'zbekistonda har doim ham yordam beravermaydi, N shahri avtomatlarini ichiga mikrochip o'rnatilgan pul kartochkalari ko'rinishidagi plastik kartochkalar yordamida ishga solishni rejalashtirayapti.

Bu pul kartochkasiga fuqaro to'lov orqali yoki o'z bank hisobidan pul o'tkazish bilan ma'lum pul miqdorini yuklashi mumkin.

Shaharning tegishli tenderiga park avtomatlari ishlab chiqaruvchi "Park easy" davogarlik qiladi.

Siz "Park easy" bilan hamkorlik qiladigan va dasturiy ta'minot ishlab chiqishga javob berganidek, pul kartochkalari ishlab turishi uchun ham javob beradigan apparat vositalari va dasturiy ta'minot bo'yicha tashkilot "Uzbek Systems" xodimisiz.

Park avtomati modeli quyidagicha ko'rinishga ega.



1.6-rasm. Parkovka avtomatining modeli

1-bo'limga doir nazorat savollari

1. Kaskad modelining asosiy xossalari va uning asosiy modefikatsiyalari nimalardan iborat? Kaskad modeli spiral modelidan nimasi bilan farqlanadi?
2. Qaysi fazada dasturlash amalga oshiriladi?
3. Dasturiy ta'minot va uning turlari nimadan iborat?
4. Pog'onali modelni tushuntiring?
5. Talablar tahlili va ixtisoslashtirilgan reja hamda dizayn fazasining maqsadi nimadan iborat?
6. top down va bottom up usullari dizayn fazasida qachon qo'llaniladi?
7. Amalga oshirish fazasini tushuntiring.
8. Spiral model mohiyatini tushuntiring.
9. V model maqsadini tushuntiring.
10. Modellarni qiyoslash deganda nimani tushunasiz?
11. Dasturiy ta'minotni ishlab chiqishning na'munaviy stsenariyalari nimadan iborat?

2. DASTURLAR ISHLAB CHIQISH TAVSIFI

O'quv maqsadi

O'quvchi tavsifning eng muhim elementlarini biladi, tarkibiy diagrammasini o'qiy oladi (Struktogramm) va tarkibiy diagrammasida qo'yilgan vazifani tasavvur qila oladi.

Mazmuni

- Ma'lumotlar oqimi o'tish chizmasi (blok-chizma).
- Yechimlar jadvali.
- Dasturning mantiqiy chizmasi.
- Tarkibiy diagrammasi (Struktogramm).

2-faza bobida dasturiy ta'minotni turli xil yondashuvlar bilan muvofiq ko'rib chiqib, 3-bobga esa texnik, dasturchi qanday qilib qo'yilgan ishlab chiqarish iqtisodiy yoki texnik vazifalarni mantiqiy dastlabki tuzulmasiga kirisha olishi mumkinligi izohlanadi.

Shu bilan birga, bu yerda gap dasturlashtirilgan jarayonlarni va ularning natijalarini qog'ozda ifodalash uchun qabul qilingan va ularga xizmat qiluvchi loyiha usuli haqida ketayotganini eslatib o'tamiz.

Shu bilan bir qatorda tasvirlash (yoritish) elementlari kabi masalan tarkibiy diagramma tarkiblashgan dasturlar loyihasini qo'llab-quvvatlaydi, misol uchun, yechimlar jadvallari qabul qilinga yechimlarning tugallanganligini ta'minlashga yordam beradi.

Bayon etishning standart texnikalari shuningdek, dasturch bo'lmaganlarga ham ma'lumotlar oqimi va dasturlashtirilgan jarayon larni tushunishga yordam beradi va natijada foydalanuvchi va ishla chiqaruvchi o'rtasidagi munosabatning ochiq-oydin bo'lishig ko'maklashadi.

Quyida hujjatlashtirishni olib borishning ba'zi muhim va ken tarqalgan texnikalari bayon qilib berilgan.

2.1. TASIRLASHNING ASOSIY TEXNIKALARI

Quyidagi asosiy bayon etish texnikalari "blok-chizma (sxema (Inglizcha: Flowchart) guruhiga tegishlidir. Bayon etish texnikala internatsional qo'llanuvchi standart belgilardan foydalanadi.

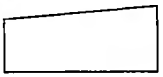
2.1.1. Ma'lumotlar oqimini boshqarish diagrammasi

Ma'lumotlar oqimining o'tish chizmasi (Inglizcha: Data Flow-chart, DFD) ma'lumotlarning mantiqiy oqimi xomaki rejasini va kompyuterdagi kiritish/chiqarish jarayonini bajaradi. Ma'lumotlar


Umumiy jarayon

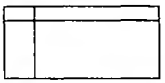

Qo'l jarayoni, qo'lda ishlov berish


Ma'lumot tashuvchilarning belgilarisiz berilgan batafsilroq umumiy ma'lumotlar


Qo'lda kiritish qurilmasi (mas. klaviatura)

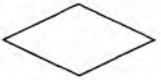

Hujjat, masalan, kvitansiya chop etish


Bir qancha hujjatlar


Tezkor xotira


Ekranga chiqarish, optik yoki akustik ma'lumotlar


Bevosita kiritiladigan ma'lumotlar tashuvchi, masalan qattiq disk


Tarmoqlanish

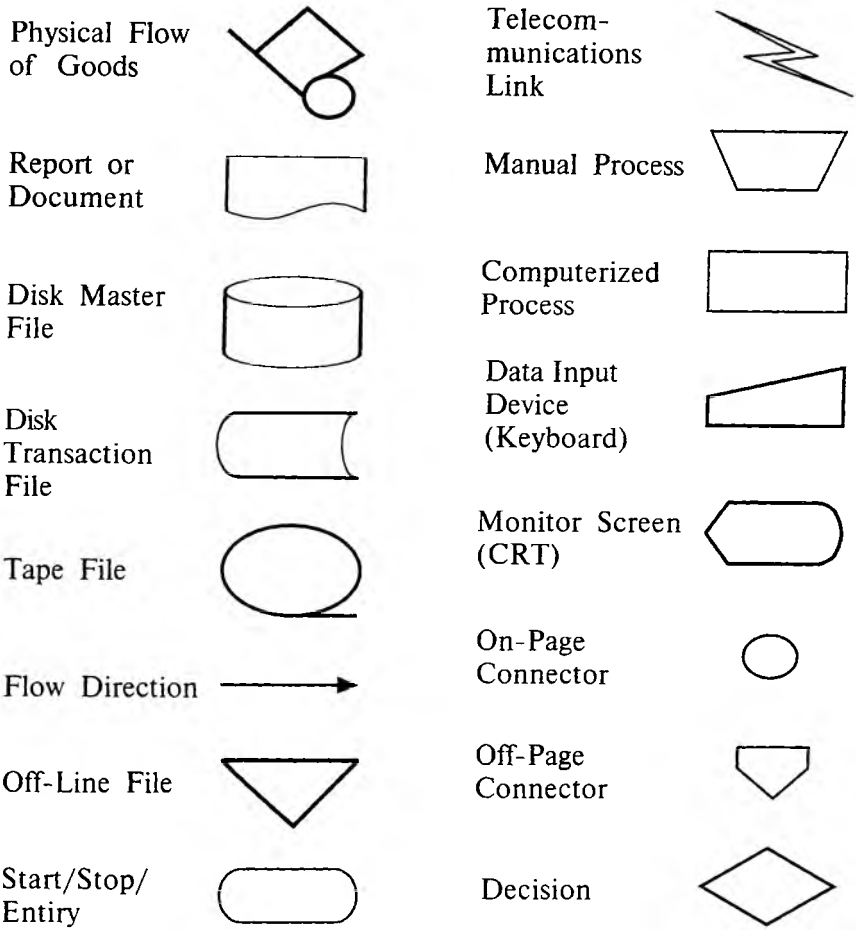

Cheklovchilar, masalan, boshlanishi, oxiri


Ishora qilish bo'yicha jarayon (masalan, nimdastur (podprogramma))

Umumiy ma'lumotlar oqimi


Ma'lumotlarni masofadan uzatish

2.1-rasm. Ma'lumotlar oqimi va boshqa diagrammalardagi ramzlar va sxemalarining o'tishi



2.2-rasm. Ma'lumotlar oqiminig va boshqa diagrammadagi belgilarning (ramzlarning) inglizcha nomlari

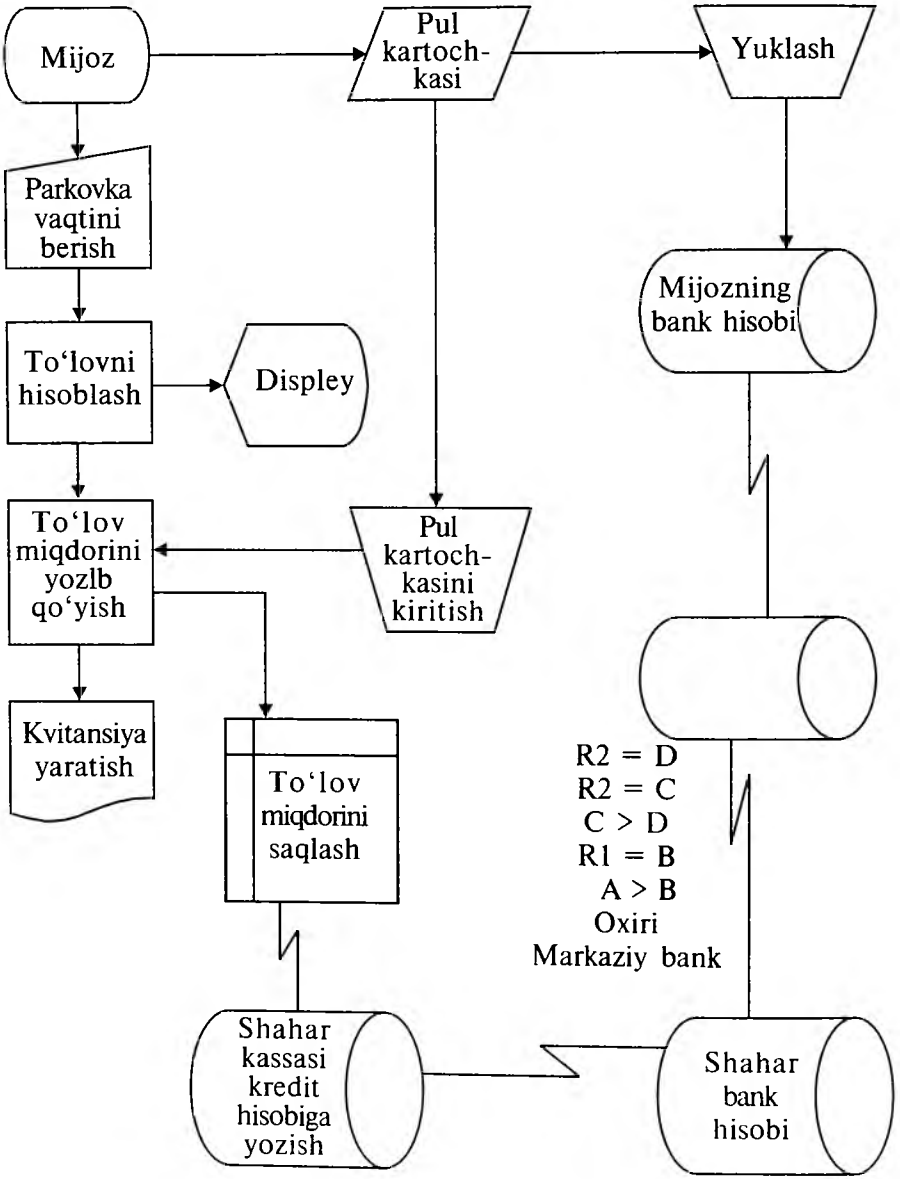
oqimi yo'nalishi uchun standart belgilar va ko'rsatkich(strelka)dar foydalaniladi.

Bu belgilar (rasmlar), shuningdek, bayon etishning boshqa tex-nikalarida ham (masalan, dasturning mantiqiy chizmasida) mavjud

Keyingi rasm inglizcha/amerikacha nomlar bilan belgila (ramzlar) to'plamini ko'rsatadi:

Bayon etish (tushuntirish)ni mumkin qadar osonlashtiris va keragicha murakkablashtirish mumkin

1.3.5.1. na'muna stsenariyisidagi parklovchi avtomat (mashinalarni toxtatish joyini hisobga olish avtomati) uchun ma'lumotlar oqimi o'tishining quyidagi chizmasi hosil bo'ladi:



2.3-rasm. Pul kartochkali parkovkalash avtomati haqida ma'lumot

2.1.2. Dasturning mantiqiy chizmasi

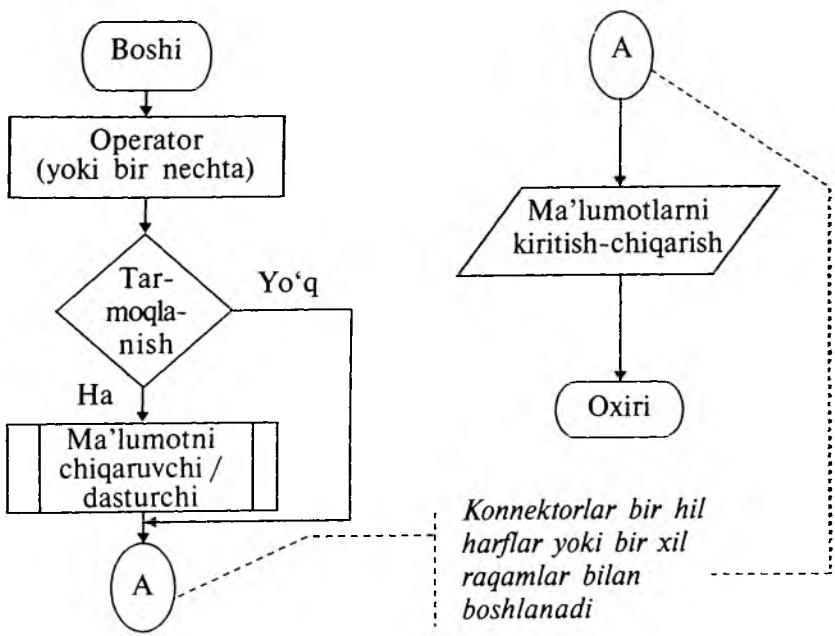
Dasturning mantiqiy chizmasi jarayonlar, kiritish-chiqarish va oqim chiziqlari kabi belgilarni o'z ichiga oladi. Bu loyiha texnikasi yordamida masalani yechish uchun zarur takliflar zanjiri yaqqol ko'rsatilgan qaror jarayonini chizish orqali tasvirlash mumkin.

Dasturning mantiqiy chizmasida har doim "Boshi" va "oxiri" mavjud bo'ladi (inglizcha: Start and Stop).

Quyidagi qoidalarga amal qilish kerak:

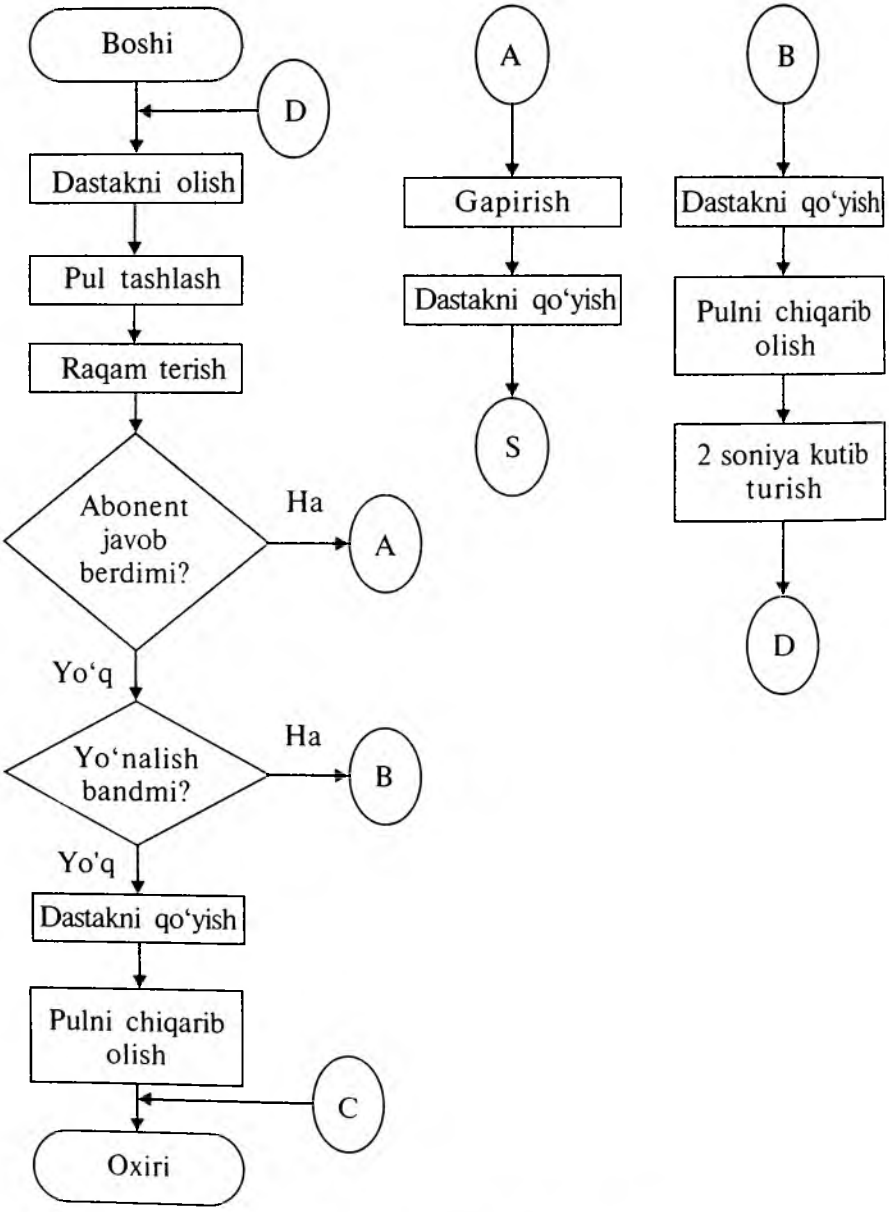
- Dasturning mantiqiy chizmasini shunday qurish kerakki, uni "tepadan pastga" o'qish mumkin bo'lsin.
- Ma'lumotlarga arxivlangan axborot matnlari sig'ishi kerak.
- Agar dasturning mantiqiy chizmasi o'ta kompleksli (murakkab) va aniq bo'lmasa, konnektorlardan foydalaniladi.
- Bu yerda ham, tasvirlashni mumkin qadar osonlashtirish va keragicha murakkablashtirish qoidasiga amal qilinadi.

Dasturlarning mantiqiy chizmalarida tez-tez uchrab turuvchi "Tarmoqlanish" va "konnektor" belgilari quyida chizma tarzida oydinlashtirilgan:



2.4-rasm. Dasturning mantiqiy chizmasining tipik belgilari (ramzlari)

Oddiy misol sifatida quyida jamoa tanga telefon apparatidan gaplashish qadamlarini tahlil qilish uchun dasturning mantiqiy chizmasi havola qilingan:



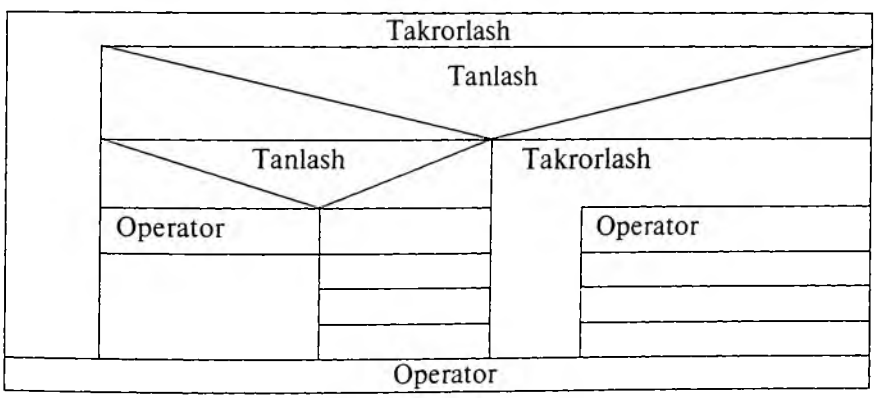
2.5-rasm. Dasturning mantiqiy chizmasi

2.1.3. Tarkibiy diagramma (Struktogramm)

Dasturlash vositasi sifatida tarkibiy diagramma (Struktogramm (inglizcha: Nassi-Shneidermann-Diagram)) dasturning mantiqiy chizmasiga qaraganda ancha qulayroq. Tarkibiy diagramma yordamida dastur bitta tuzilish bloki chegarasida ko'rsatilishi mumkin. Tarkibiy bloki ichida tarkibiy ostki (podstruktura) bloklari ulanadi.

Tarkibiy diagramma dasturlashda foydalanilayotgan dasturlash tiliga va apparat vositalarining istalgan platformasiga bog'liq bo'lmagan dasturiy mahsulotni rejalashtirish va hujjatlashtirish bo'yicha hujjatlarni ifodalaydi. Masalaning mantiqiy yechilishi va buning uchun zarur bajarish tarkiblari oldingi qatorda joylashadi.

Atigi bir nechta asosiy belgilar qo'llanishi sababli tarkibiy diagrammalar yordamida o'qish va modullashni o'rganish juda tez kechadi. Asosiy joylashuv quyidagi rasmda ko'rsatilgan:



*2.6-rasm.*Tarkibiy diagrammasining asosiy joylashuvi

Tarkibiy diagrammadagi barcha jarayonlar uchta asosiy "Takrorlash" (Repetition), "Tanlash" (Selection) va "Operator" (Sequence) tuzilmalari bilan namoyon bo'ladi.

2.1.3.1. Takrorlash

Takrorlash (Repetition) sikli (masalan, while-sikl) dasturlashga muvofiq keladi. Har bir siklning uzilishi mezoniga ega, while-sikl uchun "yuqori" (yuqorigi satr bilan boshqariluvchi sikl) turadi, do-until - sikl uchun "quyi" (quyi satr bilan boshqariluvchi sikl) turadi. Tarkibiy diagrammada uzilishning ushbu mezoni shunga muvofiq "yuqori" yoki "quyi" deb yoziladi.

Mezon noto‘g‘ri ekanmi, takrorlansin
Operator 1
Operator 2
Operator 3
Operator 4

2.7-rasm. Tarkibiy diagrammada yuqorigi satr bilan boshqariladigan tsikl

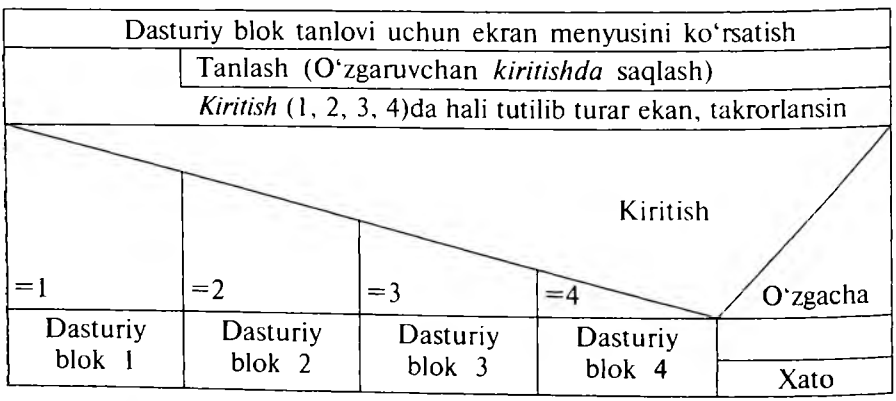
2.1.3.2. Tanlash

Tanlash (Selection) dasturlashda oddiy tarmoqlanish (if-then-nazorat qiluvchi tuzilma)ga ham, hodisalarni aniqlash (case-nazoratchi tuzilma)ga ham mos keladi.

Shart bajarildimi? HA	YO‘Q
Operator 1	Operator 3
	Operator 4
Operator 2	Operator 5
	Operator 6

2.8-rasm. Tuzilma diagrammasidagi tarmoqlanish

Keyingi rasm hodisalarni aniqlashni ekran menyusi misolida (1, 2, 3 va 4) bandlardan birini tanlash, keyin esa tegishli dasturiy blokkacha olib borish mumkin.



2.9-rasm. Tarkibiy diagrammada hodisalarni aniqlash

Yuqorida ko'rsatilgan rasmda shunga e'tibor qilish kerakki, quyi satr bilan boshqariladigan tsikl tomonidan avvaliga 1, 2, 3 va 4 bo'lmagan barcha xato va kiritishlar tutib qolinadi.

2.1.3.3. Operator

Tarkibiy diagrammada oddiy to'g'ri burchakda joylashgan bo'lsa, ular operator yoki jarayon odimi(Inglizcha: Sequence)dir. Operatorni ko'rsatish qisqa, lo'nda, shunga qaramay, tushunarli bo'lishi kerak.

Quyida keltirilgan rasm yana jamoa tangali telefon apparatidan telefon so'zlashuvi misolini, mazkur tarkibiy diagramma ko'rinishida (strukturagrammalar (Struktogramm)) takrorlaydi.

Tarkibiy diagramma (Struktogramm) AKT o'qitishda dasturning mantiqiy o'tmishini ko'rsatish uchun eng afzal ko'riladigan tasvir texnikasidir

<table>
<tr><td colspan="3">Telefon budkasini tark etmaguningcha takrorla</td></tr>
<tr><td></td><td colspan="2">Dastakni olish</td></tr>
<tr><td></td><td colspan="2">Pul tashlash</td></tr>
<tr><td></td><td colspan="2">Raqam terish</td></tr>
<tr><td></td><td colspan="2">Abonent javob berdimi?
Ha — Yo'q</td></tr>
<tr><td></td><td>So'zlash</td><td>Liniya bandmi?
Ha — Yo'q</td></tr>
<tr><td></td><td rowspan="2">Dastakni qo'yish</td><td>Dastakni qo'yish</td><td>Dastakni qo'yish</td></tr>
<tr><td></td><td>Pulni chiqarib olish</td><td>Pulni chiqarib olish</td></tr>
<tr><td></td><td>Telefon budkasini tark etish</td><td>2 soniya kutish</td><td>Telefon budka-sini tark etish</td></tr>
</table>

2.10-rasm. "Telefon so'zlashuvi" misoli. Dasturning 2.5-rasmida tarkibiy diagramma qiyofasida ko'rsatilgan mantiqiy chizma (Struktogramm)

2.2. TAVSIFNING MAXSUS TEXNIKASI

3.1 bo'limda taqdim qilingan tavsif texnikalari (Ma'lumotlar oqimi o'tish chizmasi, dasturning mantiqiy chizmasi va tarkibiy

diagrammasi) dasturning mantiqiy o'tishini hujjatlashtiradi va bu bilan bevosita amalga oshishga yoki dasturlashga juda "yaqin". Bu texnikalar ishlab chiqilgan yoki ishlab chiqilayotgan dasturning sifatini ya'ni uning to'laqonligini va bexatoligini tekshirish uchun unchalik yaroqli emas.

Bu maqsad uchun quyida ko'rsatilgan usullar xizmat qiladi.

2.2.1. Yechimlar jadvali

Yechimlar jadvali (Inglizcha: Decision table) - bu masalani tavsiflash usuli bo'lib, dasturiy ta'minot ishlab chiquvchiga loyihani tavsiflash uchun xizmat qiladi.

Yechimlar jadvalining asosiy g'oyasi shu bilan belgilanadiki, bunda yechimning mantiqiy jarayoni "Agar-Unda" sabab bog'lanishidan kelib chiqqan holda amalga oshiriladi. Bu asosiy g'oya, ya'ni ma'lum shartlarning bajarilishi u bilan bog'liq harakatlarni yurgizib yuboradi, yechimlar jadvali tarkibini belgilaydi.

Kiritish									
Shart qoidalari									
Shartlar	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
O'ziga xos shartlar									
Harakat qoidalari									
Harakatlar	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
O'ziga xos harakatlar									

2.11-rasm. Yechimlar jadvali tuzilishi

Yechimlar jadvali 4 kvadratli dekart koordinatali tizimi singari har doim 4 qismdan iborat bo'ladi. Yuqorigi qismda **shartlar** (inglizcha: Conditions), quyi qismda **harakatlar** (inglizcha:Actions)

joylashadi. Chap tarafda tegishlicha aniq shart va harakatlar ko'rsatilgan, o'ng tarafda esa tegishli yechimlar (to'g'ri yoki noto'g'ri) taqdim qilingan.

Yechimlar jadvali uchun eng muhimi shuki, shartlar va harakatlar to'la va aniq ifodalab beriladi. Buni misolda mufassalroq tushuntirish lozim bo'ladi:

N shahridagi ichimlik omborlaridan biriga chakana savdo olib boruvchi savdogarlardan buyurtma tushadi. Avvalo omborda buyurtirilgan ichimliklar borligini tekshirib ko'rish kerak. Undan keyin esa tashkilot ixtiyorida haydovchi bilan transport vositasi bormi-yo'qligini tekshirib ko'rish kerak. Agar har ikkala shart ham bajarilgan bo'lsa, unda buyurtma bajarilishi va ichimliklar yetkazilishi mumkin bo'ladi.

Shartlar

S1: Omborda buyurtirilgan ichimliklar to'la-to'kismi?

S2: Tashkilot ixiyorida haydovchi va transport vositasi bormi?

Harakatlar

A: Ekranga chiqadigan ma'lumot: Barcha transport vositalari band va keyingi transport vositasining taskilot ixtiyorida bo'lishi haqidagi ma'lumotlar.

A: Ekrahda chiqadigan ma'lumot: buyurtirilgan ichimliklar miqdorining kamligi.

A: Hujjat berish.

Shartlar				
C1	True	True	False	False
C2	True	False	True	False
Harakatlar				
A1	0	1	0	1
A2	0	0	1	1
A3	1	0	0	0

2.12-rasm. Yechimlar jadvali namunasi

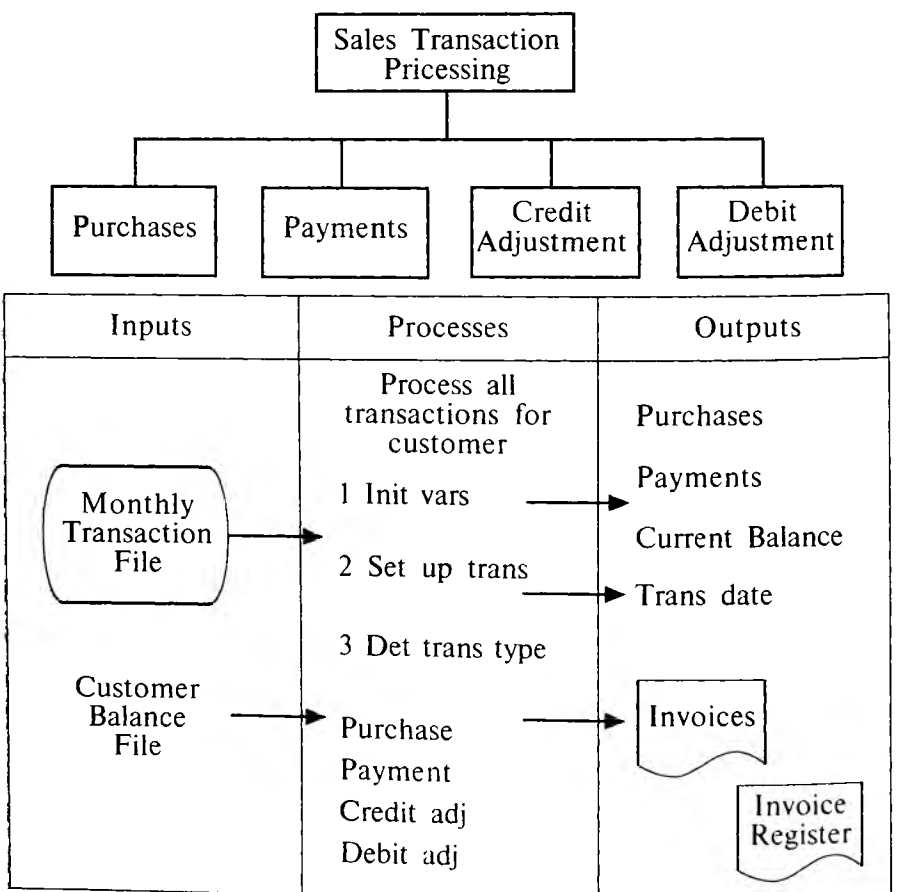
Jadvalning o'ng qismidagi ma'lumotlar uchun istalgan ifodali belgilar (masalan, 0 "noto'g'ri" va 1 "to'g'ri" uchun) qo'llash mumkin.

Ustunlar miqdori shartlar miqdori bilan aniqlanadi va ularning soni bilan bog'liq holda oshib boradi (chunki shartlarning istisnosiz barcha True/False-kombinatsiyalari tekshirilishi kerak).

2.2.2. HIPO-usuli va boshqa texnikalar

HIPO - dan inglizcha "Hierarchy plus Input-Process-Output" belgi uchun foydalaniladi. "Hierarchie" birinchi qismi - bu organigramma kabi alohida modullarni ko'rsatadigan mundarija turidir.

Ikkinchi qism - bu diagrammadir, u barcha kiritishlar, jarayonlar va chiqarishlarni ma'lumotlar oqimi bilan birgalikda ko'rsatkichlar qiyofasida taqdim qiladi.



2.13-rasm. HIPO diagrammasi savdo-sotiq jarayoni misolida

Ob'ekt munosabatidagi (tipidagi) model (inglizcha: Entity Relationship Diagram) ma'lumotlar manbaini qabul qilish uchun juda muhim tavsif usulidir. Ob'ektlarning (inglizcha: Entities) munosabatlarga(englisch: Relationship) aloqasi o'rnatiladi.Ob'ekt munosabat (tipidagi) modeli ma'lumotlar manbai bo'yicha qo'llanmada batafsil ko'rib chiqilgan.

Avtomatlashtirilgan loyihalash va dastur yaratish (CASE Tools) servisi (Computer Aided Software Engineering)) dasturlarga o'z ishlanmalaridagi mavjud vositalar va hujjatlashtirish bo'yicha servis yordamida o'z ishlanmalarini tahlil qilish va rejalashtirishda yordam beradigan dasturiy ta'minotdir. Avtomatlashtirilgan loyiha va dastur yaratish servisi (CASE Tools) dasturiy ta'minot ishlab chiqishni yaxlit ko'rib chiqadi, ya'ni dasturning blok-chizmasi va mantiqiy chizmasi uchun vositalar bilan chegaralanmaydi, balki loyiha menedjmenti va sifat ta'minoti sohasida ham yordam beradi. Avtomatlashtirilgan loyiha va dastur yaratish servisidan (CASE Tools) jumladan, dasturlash bo'yicha katta loyihalarda foydalaniladi.

2-bo'limga oid nazorat savollari

1. Dasturning mantiqiy sxemasi nima?
2. Ma'lumotlar oqimining o'tishi nima?
3. Yechimlar jadvali nima?
4. Tarkibiy diagramma nima?

3. TARKIBLASHGAN VA OB'EKTGA MO'LJALLANGAN DASTURLASH

O'quv maqsadi

O'quvchi modulli, tarkiblashgan va ob'ektga mo'ljallangan dasturlash texnologiyalarini o'rganadi va ularni amaliy masalalarda qo'llay oladi.

3.1. DASTUR TUZILISHINI ISHLAB CHIQISH VA MODULLI DASTURLASH

Tarkibi

Dastur tuzilishi (strukturasi)ni ishlab chiqishning maqsadi. Dasturiy modul tushunchasi. Dasturiy modulning asosiy tavsiflari. Dastur tuzilishini ishlab chiqish usullari. Dasturiy modul spetsifikatsiyasi (tasnifi). Dastur tuzilishining nazorati.

3.1.1. Modulli dasturlash maqsadi

Dasturiy vosita (DV) ning har bir dasturini ishlab chiqishga kirishilar ekan, bu dastur odatda katta tizim ekanini nazarda tutish kerak. Shuning uchun biz uni soddalshtirish choralarini ko'rishimiz lozim. Buning uchun ushbu dastur qismma-qism ishlab chiqiladi va bu qismlar **dasturiy modullar** deb ataladi. Dasturlarni ishlab chiqishdagi bu usulning o'zi esa **modulli dasturlash** deb ataladi. Dasturiy *modul* jarayon tavsifining biron fragmenti (qismi) bo'lib, u jarayon tavsiflarida qo'llanish uchun yaroqli bo'lgan mustaqil dasturiy mahsulot sifatida shakllantiriladi. Bu degani - har bir dasturiy modul dasturning boshqa modullaridan alohida dasturlashtiriladi, kompilyatsiya qilinadi (to'planadi) va sozlanadiki, buning natijasida u dastur boshqa modullaridan jismonan ajratilgan bo'ladi. Inchunin, ishlab chiqilgan har bir dasturiy modul turli dasturlar tarkibiga kiritilishi mumkin. Buning uchun ushbu modul bo'yicha hujjatlarda e'lon qilingan qo'llanish shartlari bajarilgan bo'lishi kerak. Shunday qilib, dasturiy modul dasturlar murakkabligiga qarshi kurash vositasi sifatida ham, dasturlashda dubllashtirishga qarshi kurash vositasi sifatida ham olib qaralishi mumkin.

Modulli dasturlash dasturlarni ishlab chiqish jarayonida murakkabliklar bilan kurashning ikkala umumiy usuli (tizim komponentalari mustaqilligini ta'minlash usuli ham, ierarxik (tabaqaviy) tuzilmalardan foydalanish usuli ham)ni o'zida mujassam etadi. Birinchi usuldan foydalanish uchun dasturiy modul javob berishi lozim bo'lgan ma'lum talablar ta'riflanadi, ya'ni "yaxshi" dasturiy modulning asosiy tavsiflari aniqlanadi. Ikkinchi usuldan foydalanish uchun dasturlarning daraxtsimon (shu jumladan shoxlari qapishib ketgan daraxtlarga o'xshash) modulli tuzilmalar qo'llanadi.

3.1.2. Dasturiy modulning asosiy tavsiflari

Har qanday dasturiy modul ham dasturni soddalshtirishga olib kelavermaydi. Shu nuqtai nazardan yaxshi modulni ajratib olish jiddiy ijodiy masalani tashkil etadi. Ajratib olingan modulning muvofiqligini baholash uchun turli mezonlardan foydalaniladi. Masalan, Xolt quyidagi ikkita mezonni taklif qiladi:

- yaxshi modul tashqi tomondan, ichki tomonga nisbatan, soddaroq bo'ladi;
- yaxshi moduldan foydalanish uni yaratishdan ko'ra osonroq.

Mayers esa dasturiy modulning muvofiqligini baholash uchun uning tuzilishidagi yanada aniqroq quyidagi tavsiflardan foydalanishni taklif qiladi:

- modul o'lchami (razmeri);
- modul mustahkamligi;
- boshqa modullar bilan birikuvi;
- modulning mustaqilligi (ya'ni uning avvalgi murojaatlardan mustaqilligi).

Modul o'lchami uning tarkibidagi operatorlar yoki satrlar soni bilan o'lchanadi. Modul o'ta katta yoki o'ta kichik bo'lmasligi lozim. Kichkina modullar dasturiy modul tuzilmasining qo'pollashib ketishiga olib keladi hamda ularni rasmiylashtirish bilan bog'liq sarf-harajatlarni qoplamasligi mumkin. Katta modullar esa ularni o'rganish va o'zgartirishda noqulayliklar tug'diradi, ular dasturni sozlash paytida uni qayta translyatsiya qilishning jamlama vaqtini ancha oshirib yuborishi mumkin. Odatda o'lchami bir necha o'ndan bir necha yuz operatorgacha bo'lgan dasturiy modullar tavsiya qilinadi

Modul mustahkamligi bu uning ichki aloqalarining me'yorlaridir Modul mustahkamligi qancha yuqori bo'lsa, u dasturning o'zga

nisbatan tashqi qismidan shu darajada ko'proq aloqalarni berkitishi hamda buning natijasi o'laroq, dasturning soddalashuviga shu darajada ko'proq hissa qo'shishi mumkin. Modul mustahkamligi darajasini baholash uchun Mayers mustahkamlik darajasi bo'yicha tartibga solingan modullarning yettita sinfidan iborat to'plamini taklif qiladi. *Moslik bo'yicha mustahkam* modul eng kam darajali mustahkamlikka ega. Bu shunday modulki, uning elementlari o'rtasida ongli aloqa mavjud emas. Bunday modul qanday holatda ajratib olinishi mumkin? Masalan, dasturning turli o'rinlarida bir xil operatorlar ketma-ketligi takrorlansa, mana shu ketma-ketlik alohida modul sifatida shakllantiriladi. Matnning ma'noli qismlaridan biri (kontekst)da ushbu ketma-ketlikni o'zgartirish zarur bo'lib qolsa, bu modulning ham o'zgarishiga olib kelishi mumkin, bu esa ushbu modul matnning boshqa ma'noli qismlari (kontekstlari)da qo'llanganda xatolikka olib kelishi mumkin. Dasturiy modullarning bu sinfidan foydalanmaslik ma'qul. Umuman olganda, Mayers taklif qilgandek modullar sinfining ularning mustahkamlik darajasiga qarab tartibga solinishi anchayin bahsli masaladir. Biroq bu uncha ahamiyatli emas, chunki mustahkamlik bo'yicha modullarning faqat dastlabki ikkita oliy sinfi foydalanish uchun tavsiya qilinadiki, biz mana shularni batafsilroq ko'rib chiqamiz.

Funksional jihatdan mustahkam modul biron-bir bitta muayyan funksiyani bajaruvchi modul hisoblanadi. O'z funksiyasini amalga oshirishda bunday modul boshqa modullardan ham foydalanishi mumkin. Dasturiy modullarning aynan shu sinfidan foydalanish tavsiya etiladi.

Axboriy jihatdan mustahkam modul bitta ma'lumotlar tuzilmasi (axborot ob'ekti) ustida bir nechta operatsiya (funksiya)ni bajaradigan (amalga oshiradigan) modul bo'lib, bunda ma'lumotlar tuzilmasi ushbu moduldan tashqarida noma'lum hisoblanadi. Bunday modulda ushbu operatsiyalarning har biri uchun alohida murojaat shakliga ega bo'lgan maxsus kirish mavjud. Modullarning bunday sinfi oliy darajadagi mustahkamlikka ega bo'lgan modullar sinfi sifatida olib qaralishi lozim. Axboriy jihatdan mustahkam modul, masalan, abstrakt turdagi ma'lumotlarni ishga sola oladi.

Modulli dasturlash tillarida kamida funksional jihatdan mustahkam modullarning berilishi uchun vositalar mavjud (masalan, FORTRAN tilidagi FUNCTION turdagi modul shular jumlasiga kiradi). Dastlabki dasturlash tillarida axboroiy jixatdan mustahkam

modullarni berish uchun vositalar mavjud emas edi. Bunday vositalar ancha keyingi tillarda paydo bo'ldi. Masalan, Ada dasturlash tilida axboriy jihatdan mustahkam modulni berish vositasi paketdir.

Modul birikuvi bu modulning ma'lumotlar bo'yicha boshqa modulga qanchalik tobe'ligini ko'rsatuvchi me'yordir. Modulning boshqa bir modul bilan birikuvi qanchalik zaif bo'lsa, uning boshqa modullardan mustaqilligi shunchalik kuchli bo'ladi. Birikuv darajasini baholash uchun Mayers modullar birikuvining oltita turidan iborat bo'lgan tartibga keltirilgan to'plamni taklif qiladi. Birikuvning eng yomon turi bu *ichki tarkibga ko'ra birikuvdir.* Ikkita moduldan biri ikkinchisining ichki tarkibiga to'g'ridan-to'g'ri murojaat qilish imkoniga ega bo'lsa, bunday modullar birikuvi eng zaif hsioblanadi. Mouddlarning bunday birikuviga yo'l qo'yib bo'lmaydi. *Umumiy soha bo'yicha birikuv*dan ham foydalanish tavsiya etilmaydi. Bu modullarning shunday birikuviki, bunda bir nechta modul bitta xotira sohasidan foydalanadi. Modullar birikuvining bunday turi FORTRAN tilida COMMON bloklaridan foydalanib dasturlashda amalga oshiriladi. Hozirgi zamon dasturlash texnologiyasi tavsiya etadigan modullarning yagona birikuv turi bu *parametrik birikuv*dir (Mayersga ko'ra ma'lumotlar bo'yicha birikuv). Bunda ma'lumotlar, modulga murojaat qilinganda, uning parametrlarining qiymati sifatida uzatiladi, yoki bo'lmasa ushbu modulning biron-bir funksiyani yechish uchun boshqa modulga murojaati natijasi sifatida uzatiladi.

Modulning mustaqilligi (rutinnost modulya) bu uning o'ziga avvalgi murojaatlardan mustaqilligini bildiradi. Agar modulga murojaat etish natijasi (effekti) faqatgina shu modul parametrlariga bog'liq bo'lsa (ya'ni unga bo'lgan avvalgi murojaatlarga bog'liq bo'lmasa), bunday modul *mustaqil* hisoblanadi. Ayrim xollarda modulga murojaat etish natijasi (effekti) ushbu modulning, unga bo'lgan avvalgi murojaatlar natijasida o'zgarishi mumkin bo'lgan ichki horlatiga bog'liq bo'ladi. Bunday modul *avvalgi murojaatlarga bog'liq modul* deb ataladi. Mayers avvalgi murojaatlarga bog'liq (bashorat qilib bo'lmaydigan) modullardan foydalanishni tavsiya qilmaydi, chunki ular dasturda "mug'ombir" (tutqich bermaydigan) xatolar kelib chiqishiga sabab bo'ladi. Ammo bu tavsiya konstruktiv emas, chunki ko'p hollarda aynan avvalgi murojaatlarga bog'liq bo'lgan modul axboriy jihatdan mustahkam modulni yaxshiroq ishga solishi mumkin. Shuning uchun bu o'rinda quyidagi (ehtiyotkorroq) tavsiyaga amal qilish maqsadga muvofiqdir:

- agar mustaqil modulni qo'llash nomuvofiq modullar birikuviga olib kelmasa, har vaqt shunday moduldan foydalanish zarur;

- parametrik birikuvni ta'minlash uchun zarur bo'lib qolgandagina, avvalgi murojaatlarga bog'liq bo'lgan modullardan foydalanish mumkin;

- bu bog'liqlik bunday modul spetsifikatsiyasida shunday aniq ifodalangan bo'lmog'i lozimki, bunda ushbu modulga kelajakdagi turli xil murojaatlarda modulning xatti-harakatlarini istiqbollash mumkin bo'lsin.

So'nggi tavsiya bilan bog'liq holda mustaqil (ya'ni o'ziga bo'lgan avvalgi murojaatlarga bog'liq) modul holatlari haqidagi inson bilimlaridan kelib chiqqan tashqi tasavvurlarga berilgan ta'rif diqqatga sazovordir. Bu holda modul orqali amalga oshiriladigan har bir funksiya (operatsiya)ning bajarilish effektini mana shu tashqi tasavvurlar atamalari vositasida tavsiflash lozim. Bu, o'z navbatida, ushbu modul xatti-harakatining istiqbolini belgilashni ancha osonlashtiradi.

3.1.3. Dastur tuzilishini ishlab chiqish usullari

Yuqorida ta'kidlab o'tilganidek, dasturning modulli tuzilmasi sifatida daraxtsimon tuzilma (shu jumladan shoxlari birikib tugunlar hosil qilgan daraxtlar tuzilmasi) dan foydalanish qabul qilingan. Bunday daraxt tugunlarida dasturiy modullar joylashadi, yo'naltirilgan yoylar (strelkalar) esa modullarning statik tobe'ligini ko'rsatadi, ya'ni har bir yoy o'zi chiqib kelgan har bir modul matnida yana kirib ketadigan modulga ishora borligini ko'rsatadi. Boshqacha qilib aytganda, har bir modul o'ziga tobe' bo'lgan modullarga murojaat qilishi mumkin, ya'ni shu modullar orqali ifodalanadi. Bunda dasturning modulli tuzilmasi, pirovard natijada, ushbu dasturni hosil qilgan modullar spetsifikatsiyasi majmuini o'z ichiga olmog'i lozim. Dasturiy modul spetsifikatsiyasi quyidagilarni o'z ichiga oladi:

- modul kirishlarining sintaktik spetsifikatsiyasi (u ushbu modulga hamda uning har qanday kirishiga qo'llanilayotgan dasturlash tilida to'g'ri sintaktik murojaatni qurish imkonini beradi);

- modulning funksional spetsifikatsiyasi (ushbu modul o'zining har biri kirishi bo'yicha bajaradigan funksiyalarining semantik, ya'ni mazmun jihatdan tavsifi).

Modulning funksional spetsifikatsiyasi ham xuddi dasturiy vositaning funksional spetsifikatsiyasi kabi tuziladi.

Dasturni ishlab chiqish jarayonida uning modul tuzilmasi dasturlash tartibini aniqlash va ushbu tuzilmada ko'rsatilgan modullarni sozlashda turlicha shakllanishi va qo'llanishi mumkin. Shuning uchun dastur tuzilmasini ishlab chiqishning turli xil usullari haqida gap yuritish mumkin. Odatda maxsus adabiyotlarda ikki xil usul haqida gap boradi: yuqorilovchi ishlab chiqish usuli va pasayuvchi ishlab chiqish usuli.

*Yuqorilovchi ishlab chiqish usuli*ning mohiyati quyidagichadir. Avval dasturning daraxt ko'rinishidagi modelli tuzilmasi yaratiladi. Keyin navbatma-navbat dastur modullari (eng quyi sathdagi moduldan boshlab) dasturlanadi. Bunda har bir dasturlanayotgan modul uchun ushbu modul murojaat qilishi mumkin bo'lgan boshqa barcha modullar dasturlashtirib bo'lingan bo'lmog'i lozim. Dasturning barcha modullari dasturlashtirib bo'lingach, ular navbatma-navbat test sinovlaridan o'tkaziladi va sozlanadi. Bunda dastur modullari qanday tartibda dasturlangan bo'lsa (yuqorilab boruvchi), ularning sinovi va sozlanishi ham shu tartibda amalga oshiriladi. Dasturni ishlab chiqishning bunday tartibi bir qarashda juda tabiiy ko'rinishi mumkin: har bir modul dasturlash paytida o'ziga bevosita tobe' bo'lgan va dasturlab bo'lingan modullar orqali ifodalanadi, test sinovlaridan o'tkazishda esa sozlab bo'lingan modullar qo'llanadi. Biroq hozirgi zamon texnologiyasi dasturni ishlab chiqishning bunday tartibidan foydalanishni tavsiya qilmaydi. Birinchidan, biron-bir modulni dasturlash uchun ushbu modul foydalanadigan boshqa modullarning matnlari bo'lishi shart emas, buning uchun har bir qo'llanayotgan modul spetsifikatsiyalangan bo'lishi kifoya (bunda spetsifikatsiya hajmi ushbu modulga to'g'ri murojaatni tuzish imkonini berishi kerak). Modulni test sinovidan o'tkazish uchun esa qo'llanayotgan modullarning o'rniga ularning imitatorlari (zaglushkalari) dan foydalanish mumkin (xatto foydali). Ikkinchidan, har bir dastur o'zi uchun ichki, ammo o'zining modullari uchun global (umumiy) bo'lgan tamoyillar (ishga tushirish, taxmin qilish, ma'lumotlar tuzilmasi va h. k.)ga qaysidir ma'noda bo'yinsunadi. Bu tamoyillar esa dasturning kontseptual butligini belgilab beradi hamda ishlab chiqish jarayonida shakllanadi. Yuqorilovchi ishlab chiqish usulida ushbu global axborot quyi sathdagi modullar uchun hali to'la xajmda aniq bo'lmaydi. Shuning

uchun boshqa modullarni dasturlashda ushbu global axborotni aniqlashtirish jarayoni kechayotganda quyi sathdagi modullar qayta dasturlanadi. Uchinchidan, yuqorilab boruvchi testda har bir modul uchun (bosh moduldan tashqari) yetakchi dastur (modul)ni yaratishga to'g'ri keladiki, bu yetakchi dastur (modul) test sinovidan o'tkazilayotgan modul uchun zaruriy axborot muhiti holatini tayyorlab berishi hamda unga talabdagi murojaatni amalga oshirishi lozim. Bu jarayon dasturni sozlash ishlari xajmlarini oshirib yuboradi. Inchunin, test sinovlari ushbu modullar aynan ishchi holatda yuzaga keladigan sharoitlarda sinovdan o'tkazilganligiga hech qanday kafolat bermaydi.

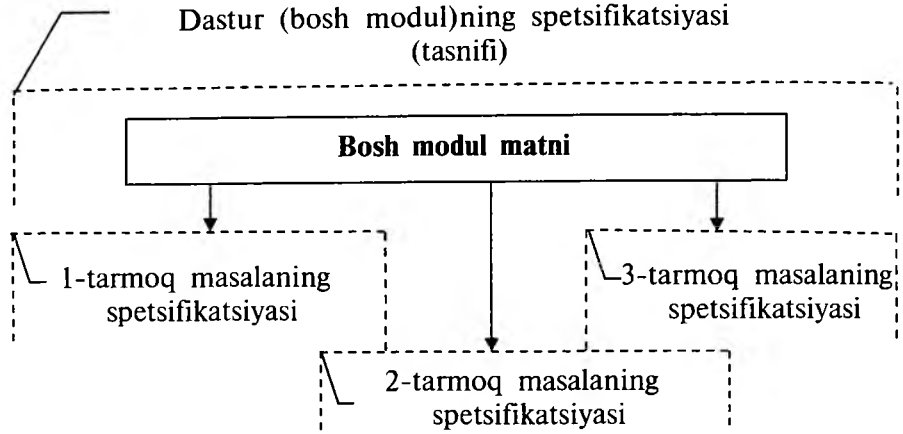
*Pasayuvchi ishlab chiqish usuli*ning mohiyati quyidagichadir. Avvalgi usulda bo'lganidek, bu yerda ham oldin dasturning daraxt ko'rinishidagi modul tuzilmasi yaratiladi. Keyin navbatma-navbat dastur modullari dasturlanadi. Bunda dasturlash eng yuqori sathdagi (bosh) moduldan boshlanadi. Keyin boshqa biron-bir modulni dasturlashga o'tiladi. Bunda ushbu modulga murojaat qiladigan modul dasturlab bo'lingan bo'lishi lozim. Dasturning barcha modullari dasturlab bo'lingach, ular xuddi shunday pasayuvchi tartibda navbatma-navbat test sinovlaridan o'tkaziladi va sozlanadi. Bunda birinchi bo'lib dasturning bosh moduli test sinovidan o'tkaziladi, chunki u testdan o'tkazilayotgan dasturning hammasiga tegishli bo'ladi hamda shuning uchun test sinovi ushbu dastur amal qiladigan axborot muhitining "tabiiy" holatida o'tkaziladi. Yana bunda bosh modul murojaat qilishi mumkin bo'lgan modullar ularning imitatorlari (zaglushkalari) ga almashtiriladi. Har bir *modul imitatori* bu oddiy dasturiy fragment (qism) bo'lib, u asosan imitatsiya-lanayotgan modulga murojaat haqida xabar (signal) beradi, dasturning to'g'ri ishlashi uchun zarur bo'lgan qiymatlar va kirish parametrlariga ishlov beradi (ba'zida bu ma'lumotlarni bosma holda chiqarib ham beradi) hamda, zaruratga ko'ra, avvaldan tayyorlab qo'yilgan tegishli natijani chiqarib beradi. Bosh hamda navbatdagi har qanday modul test sinovidan o'tkazilib va sozlab bo'lingach, ushbu daqiqada o'zining imitatori (agar bular bo'lsa) ga ega bo'lgan modullardan biri test sinovidan o'tkaziladi. Buning uchun test sinovidan o'tkazilishi tanlab qilingan modul imitatori o'rniga modulning o'zi olinadi. Bundan tashqari bu sinovga test sinovidan o'tkazilishi tanlab qilingan modul murojaat qilishi mumkin bo'lgan modullar imitatorlari qo'shiladi. Shunday qilib, yuqorilovchi test sinovidan

o'tkazish paytidagi katta hajmdagi "sozlovi" dasturlash o'rniga dastur modullarida qo'llanadigan anchayin sodda imitatorlarni dasturlash amalga oshiriladi. Bundan tashqari, imitatorlar chiqarib berayotgan kerakli natijalarni berish yo'li bilan testlarni tanlab olish jarayoniga moslashish uchun ham imitatorlardan foydalanish qulaydir. Dasturni ishlab chiqishning bunday tartibida barcha zaruriy global axborot o'z vaqtida shakllanadi, ya'ni modullarni dasturlashdagi noxush xatoliklar manbai yo'q bo'ladi. Pasayuvchi ishlab chiqish usulining dasturni qo'llashda ma'lum qiyinchiliklar tug'ilishiga olib keluvchi kamchiligi shundan iboratki, bunda qo'llanayotgan dasturlash tilining bazaviy imkoniyatlari mavhumlashtiriladi, ya'ni mavhum (abstrakt) operatsiyalar o'ylab topiladiki, bu operatsiyalarni keyinchalik dasturda ajratib olindan modullar yordamida amalga oshirishga to'g'ri keladi. Biroq, shuning bilan birga, bunday mavhumlashtirishlar (abstraktsiyalar) katta dasturiy vositalarni ishlab chiqishning zaruriy shartidir. Shuning uchun ularni rivojlantirish muhimdir.

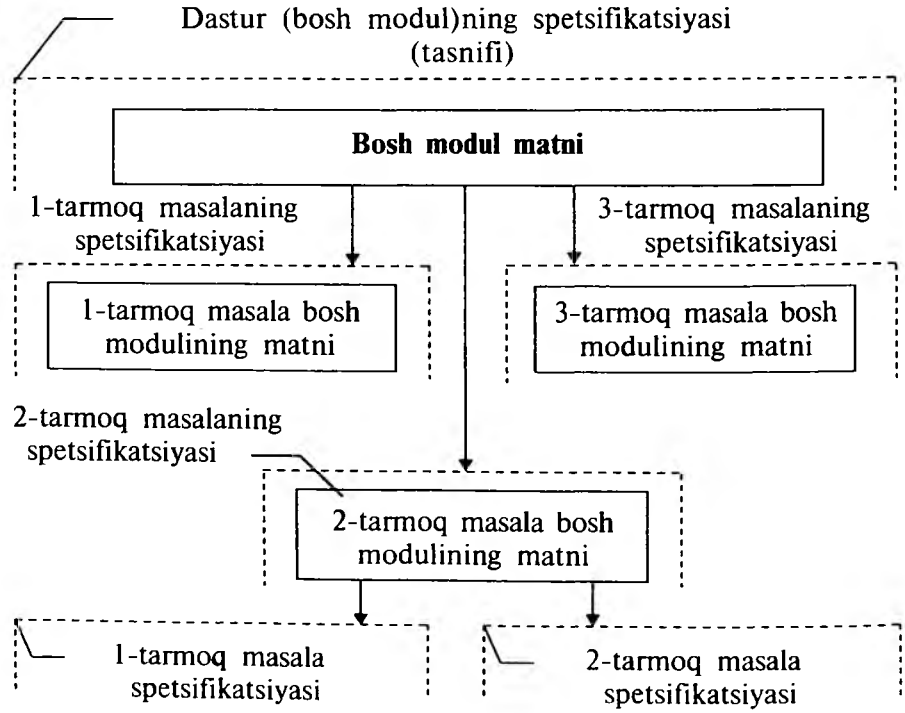
Ko'rib chiqilgan yuqorilovchi va pasayuvchi ishlab chiqish usullarining (biz ularni *klassik usullar* deb ataymiz) o'ziga xos xususiyati shundaki, bunda dasturning modul tuzilmasi modullarning dasturlanishi (kodlanishi)dan oldin ishlab chiqilishi talab qilinadi. Bu talab dasturiy vositani ishlab chiqishning shalolasimon yondoshuviga to'liq mos keladi, chunki dasturning modul tuzilmasini ishlab chiqish va uni kodlash dasturiy vositani ishlab chiqishning turli bosqichlarida amalga oshiriladi: modul tuzilmasini ishlab chiqish DV ni konstruktsiyalash bosqichini nihoyasiga etkazadi, kodlash esa kodlash bosqichini boshlab beradi.

Bu usullar e'tirozlarga ham duch keldi. Chunki avval modullarni dasturlamay turib, dastur tuzilmasini aniq va chuqur ishlab chiqish mumkinligi shubha tug'diradi. Agar shalolasimon yondashuv bir oz takomillashtirilsa, amalda bunday qilish shart emas. Quyida dasturlarni ishlab chiqishda konstruktiv va arxitektura yondoshuvlari taklif qilinadiki, ularda modulli tuzilma modullarni dasturlash (kodlash) jarayonida shakllanadi.

Boshqa har qanday modulni dasturlashda ham xuddi shunday xatti-harakatlar amalga oshiriladi. Ushbu dasturlanadigan modu spetsifikatsiyalangan, ammo hali dasturlanmagan modullar ichidar dastur daraxtining joriy holatidan tanlab olinadi. Buning natijasid dastur daraxtining navbatdagi yanada ko'proq shakllanishi (masalar 4.2-rasmda ko'rsatilganidek) amalga oshiriladi.



3.1-rasm. Konstruktiv yondashuv paytida dasturning modulli tuzilmasini shakllantirishning birinchi qadami

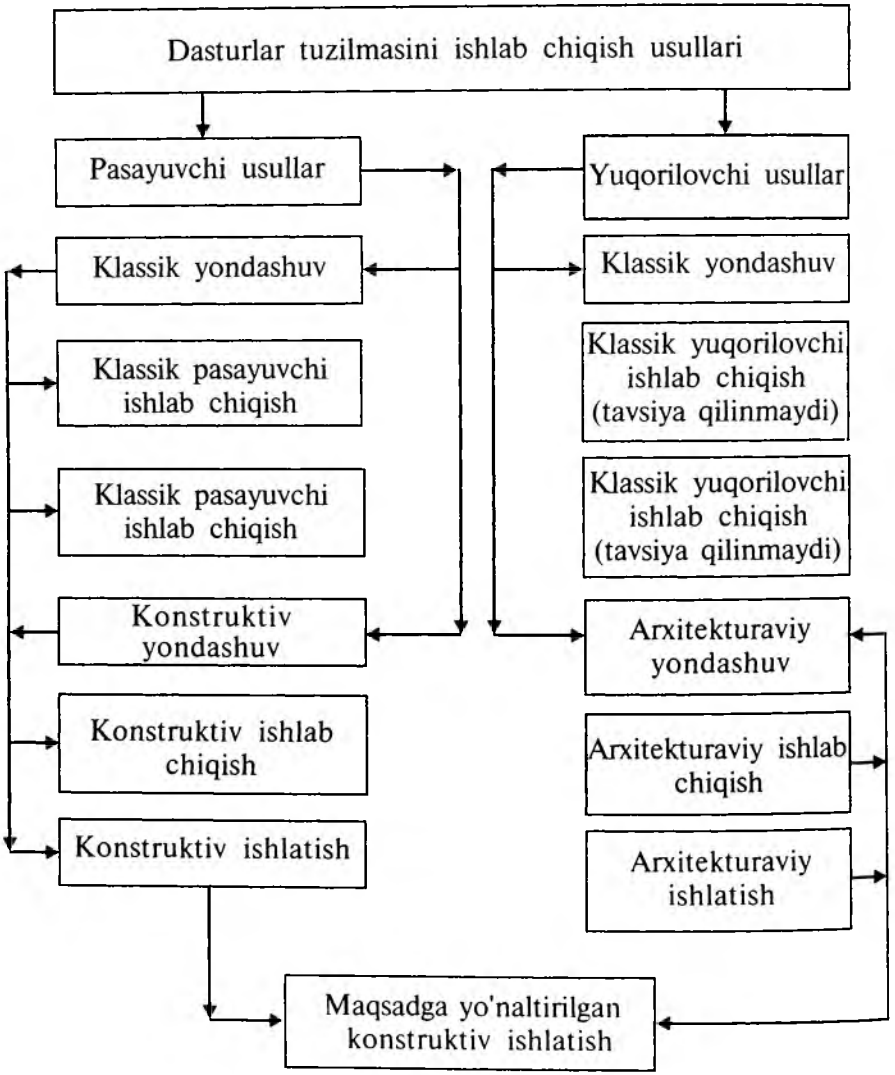


3.2-rasm. Konstruktiv yondashuv paytida dasturning modulli tuzilmasini shakllantirishning ikkinchi qadami

Dasturni ishlab chiqishdagi arxitektura yondoshuvi yuqoriiovcni ishlab chiqishning modifikatsiyasi bo'lib, bunda dasturning modulli tuzilmasi modulni dasturlash jarayonida shakllanadi. Biroq bunda dasturni ishlab chiqishdan butunlay boshqa maqsad, ya'ni aniq dasturni ishlab chiqish emas, qo'llanayotgan dasturlash tili darajasini ko'tarish ko'zlanadi. Bu nimani bildiradi? Shuni bildiradiki, berilgan soha uchun har biri shu sohadagi turli masalalarni yechishda qo'llanishi mumkin bo'lgan tipik(namunaviy) funksiyalar ajratiladi hamda spetsifikatsiyalanadi, keyin esa ushbu funksiyalarni bajaradigan dasturiy modullar dasturlanadi. Bunday funksiyalarni ajratish jarayoni berilgan sohadagi masalalarni yechish tajribasini to'plash va umumlashtirish bilan bog'liq bo'lgani uchun, odatda avval soddaroq modullar alohida modullar sifatida ajratib olinadi va ishga tushiriladi, keyinchalik esa asta-sekin avval ajratib olingan modullarni qo'llaydigan modullar paydo bo'ladi. Bunday modullar to'plami shunga mo'ljallab yaratiladiki, berilgan sohadagi u yoki bu dasturni ishlab chiqishda konstruktiv yondoshuv doirasida ushbu modullarning ayrimlarini qo'llash mumkin bo'lib qoladi. Bu muayyan dasturni ishlab chiqishga ketadigan mehnat sarflarini avvaldan tayyorlangan va tekshiruvdan o'tgan quyi sathga mansub modul tuzilmalarini mana shu dasturga ulash yo'li bilan ancha qisqartirish imkonini beradi. Bunday tuzilmalar turli xildagi muayyan dasturlarda ko'p martalab qo'llanishi mumkin bo'lganligi uchun, arxitektura jihatdan yondashuv dasturlashda qaytariqlar bilan kurashish yo'l sifatida olib qaralishi mumkin. Buning bilan bog'liq holda arxitektura jihatidan yondoshuv doirasida yaratiladigan dasturiy modullar odatda ushbu modullarni parametrlarga sozlash yo'li bilan ularning qo'llanishini kuchaytirish uchun parametrlanadi.

Pasayuvchi ishlab chiqishning klassik usulida ishlab chiqilayotgan dasturning barcha modullarini avval dasturlab olish va shundan so'nggina ularning pasayuvchi test sinovlaridan o'tkazilishini boshlash tavsiya etiladiki, bu yana o'sha sharsharasimon yondoshuvga to'la mos keladi. Biroq ishlab chiqishning bunday tartibi yetarlicha asoslangan bo'lolmaydi: modullarni test sinovlaridan o'tkazish va sozlash tobe modullar spetsifikatsiyasining o'zgarishiga va xatto butun dasturning modul tuzilmasining o'zgarishiga olib kelishi mumkin. Shuning uchun bunday holda ayrim modullarni dasturlash befoyda amalga oshirilgan ishga aylanadi. Dasturni ishlab chiqishning boshqa tartibi nazarimizda, maqsadga muvofiqroq ko'rinadi. Bu usul adabiyotlarda

pasayuvchi usul nomi bilan ma'lum bo'lib, u sharsharasimon yondoshuvning ma'lum bir modifikatsiyasidir. Bu usulda dasturlangan modul, boshqa modulni dasturlashga o'tmay turiboq, test sinovidan o'tkaziladi.



3.3-rasm. Dastur tuzilmasini ishlab chiqish usullarining tasnifi

Bu usullarning hammasi, o'z navbatida, daraxtsimon dastur tuzilmasini ishlab chiqish jarayonida uning uzel (modul)lari qanday ketma-ketlikda aylanib chiqilishiga qarab, turli ko'rinishlarga ega

bo'ladi. Buni, masalan, qavatma-qavat (keyingi sathga o'tishdan oldin, bir sathdagi modullarni ishlab chiqish yo'li bilan) amalga oshirish mumkin. Pasayuvchi ishlab chiqishda dastur daraxtini yana leksikografik tartibda (yuqoridan pastga, chapdan o'ngga) aylanib chiqish mumkin. Daraxtni aylanib chiqishning boshqa variantlari ham mavjud. Masalan, konstruktiv yondoshuvda dastur daraxtini aylanib chiqish uchun Fuksman g'oyalariga amal qilish maqsadga muvofiqdir. Bu g'oyalardan Fuksman o'zi taklif qilgan vertikal qatlamlash usulida foydalangan. Bunday aylanib chiqish mohiyati quyidagilardan iborat. Konstruktiv yondoshuv doirasida avval dasturning eng sodda varianti uchun zarur bo'lgan modullar ishlab chiqiladi. Bunda ushbu dastur kirish ma'lumotlarining juda cheklangan to'plamlari uchungina normal bajarilishi mumkin. Bunday dasturda iqtiboslar (silkalar) mavjud bo'lgan boshqa modullar o'rniga bu modullarning imitatorlarigina kiritiladi. Bu imitatorlar asosan bunday xususiy holat chegarasidan chiqish haqida signal berilishini ta'minlaydi. Keyin ushbu dasturga boshqa ayrim modullarning ishlanmalari (jumladan, ayrim mavjud imitatorlar o'rniga) qo'shiladi. Bu modullar ishlanmalari kirish ma'lumotlarining boshqa biron to'plamlari uchun normal ish bajarilishini ta'minlaydi. Bu jarayon ham talabdagi dastur to'la ishlab chiqilguncha bosqichma-bosqich davom etadi. Shunday qilib, dastur daraxtini aylanib chiqishdan ko'zlanadigan maqsad normal faoliyat ko'rsatayotgan dasturning u yoki bu variantining eng qisqa yo'l bilan ishga tushirilishini amalga oshirishdan iborat. Shuning uchun ham konstruktiv ishlab chiqishning bunday turi *maqsadga yo'naltirilgan konstruktiv ishlab chiqish usuli* nomini oldi. Bu usulning afzal tomoni shundaki, ishlab chiqishning dastlabki bosqichidayoq ishlab chiqilayotgan dasturning ishchi varianti yaratiladi. Psixologik jihatdan bu ishlab chiquvchi mehnatining samaradorligini oshiruvchi doping vazifasini o'taydi. Shu tufayli bu usul diqqatni o'ziga jalb qiilish kuchiga ega.

Aytilganlarga yakun yasab, 4.3-rasmda dastur tuzilmasini ishlab chiqish usullarining ko'rib chiqilgan umumiy tasnifi keltiriladi.

3.1.4. Dastur tuzilishining nazorati

Dastur tuzilishini nazorat qilishda uchta usuldan foydalanish mumkin:

- statik nazorat;

- yondosh nazorat;
- mufassal nazorat.

Statik nazoratda dastur tuzilishiga baho beriladi: yuqorida ko'rib o'tilgan asosiy modul tavsiflarining qiymatlarini hisobga olgan holda dastur modullarga qay darajada unumli bo'lingan.

Yuqoridan yondosh nazorat bu dasturiy vosita arxitekturasi va tashqi tavsifining ishlab chiquvchilar tomonidan olib boriladigan nazoratidir. Pastdan yondosh nazorat bu modullar spetsifikatsiyasining ishlab chiquvchilar tomonidan olib boriladigan nazoratidir.

Mufassal nazorat avvaldan ishlab chiqilgan testlarni bajarishda dastur tuzilmasini xayolan "ko'rib chiqish" (tekshirish)ni bildiradi. Dasturiy vositaning funksional spetsifikatsiyasi yoki arxitekturasining qo'lda amalga oshiriladigan imitatsiyasi kabi, dinamik nazorat turidan biri bo'lib xizmat qiladi.

Shuni ham ta'kidlash lozimki, dastur tuzilmasining aytib o tilgan nazorati DV ni ishlab chiqishning sharorasimon yondashuvi doirasida amalga oshiriladi. Konstruktiv va arxitektura yondashuvida dastur tuzilmasining nazorati muvofiq keladigan vaqt daqiqalarida modullarni dasturlash jarayonida amalga oshiriladi.

3.2. DASTURIY MODULNI ISHLAB CHIQISH

Tarkibi

Dasturiy modulni ishlab chiqish tartibi. Tuzilmaviy dasturlash va qadamma-qadam detallashtirish. Soxta kod haqida tushuncha. Dasturiy modul ustidan nazorat.

3.2.1. Dasturiy modulni ishlab chiqish tartibi

Dasturiy modulni ishlab chiqishda quyidagi tartibga rioya qilish maqsadga muvofiqdir:

- modul spetsifikatsiyasini (tasnifini) o'rganish va tekshirish, dasturlash tilini tanlash;
- algoritm va ma'lumotlar tuzilmasini tanlash;
- modulni dasturlash (kodlash);
- modul matnini qiyomiga yetkazish;
- modulni tekshirish;
- modulni yig'ish (kompilyatsiya qilish).

Dasturiy modulni ishlab chiqishdagi birinchi qadam asosan

dastur tuzilmasining pastdan yondosh nazoratidan iborat: modul spetsifikatsiyasi (tasnifi)ni o'rganar ekan, ishlab chiquvchi bu tasnifning unga tushunarli ekaniga va ushbu modulni ishlab chiqish uchun yetarli ekaniga ishonch hosil qilishi kerak. Bu qadamning oxirida dasturlash tili tanlab olinadi: garchi dasturlash tili butun DV uchun avvaldan belgilab olingan bo'lishi mumkin bo'lsa-da, biroq ayrim hollarda (dasturlash tizimi yo'l qo'ysa) ushbu modulni ishga tushirish uchun ko'proq mos keladigan boshqa til (masalan, assembler tili) tanlanishi mumkin.

Qo'yilgan masala yechimini topish uchun biron-bir algoritmlar ma'lum bo'lishi mumkin. Dasturiy modulni ishlab chiqishning ikkinchi qadamida mana shuni aniqlash lozim bo'ladi. Agar to'g'ri keladigan algoritm mavjud bo'lsa va u topilsa, undan foydalanish maqsadga muvofiq bo'ladi. Modulning o'z funksiyalarini bajarishda qo'llanadigan tegishli ma'lumotlar tuzilmasini tanlab olish ko'p o'rinda ishlab chiqilayotgan modulning mantig'i va sifat ko'rsatkichlarini avvaldan belgilab beradi. Shuning uchun bu tanlashga g'oyat mas'uliyatli yechim sifatida qarash lozim.

Uchinchi qadamda tanlab olingan dasturlash tilida modul matnini tuzish amalga oshiriladi. Modul tasnifi (spetsifikatsiyasi)da ko'rsatilgan funksiyalarni amalga oshirishda hisobga olinishi lozim bo'lgan turli xil detallar shu qadar ko'pki, bu xatolar va noaniqliklarga to'lib toshgan anchayin chalkash matnning yuzaga kelishiga sabab bo'lishi mumkin. Bunday modulda xatolarni izlab topish va ularga to'g'rilashlar kiritish g'oyat ko'p kuchni talab qiluvchi masala bo'lishi mumkin. Shuning uchun modul matnini tuzishda texnologik jihatdan asoslangan va amalda tekshirilgan dasturlash tartibidan foydalanish g'oyat muhimdir. Bu masalaga birinchilardan Deykstra e'tibor qaratdi hamda tuzilmaviy dasturlashning asosiy tamoyillarini asoslab berdi. Amaliyotda keng qo'llanib kelayotgan ko'plab dasturlash tartiblari mana shu tamoyillarga asoslanadi. Eng keng tarqalgan intizomlardan biri bu qadamma-qadam detallashtirish tartibi bo'lib, u qo'llammaning 3.2.2- va 3.2.3-bo'limlarida mufassal muhokama qilinadi.

Modulni ishlab chiqishning navbatdagi qadamida modul matni DV sifati spetsifikatsiyasi (tasnifi)ga muvofiq keladigan tugal holatga keltiriladi. Modulni dasturlashda ishlab chiquvchi asosiy e'tiborini modul funksiyalarini to'g'ri amalga oshirilishiga qaratadi. Bunda u izohlarga ko'p ham e'tibor bermaydi va dastur uslubiga qo'yiladigan

talablarda ham ayrim noaniqliklarga yo'l qo'yadi. Modul matnini qiyomiga yetkazishda esa u matndagi mavjud izohlarni tahrir etishi hamda talabdagi sifat primitivlarini ta'minlash maqsadida, unga qo'shimcha izohlar kiritishi lozim. Xuddi shu maqsadlarda matn uslubiga qo'yiladigan talablarni bajarish uchun ham ma'lum tahrir ishlari amalga oshiriladi.

Modulni tekshirish qadami modulni sozlash, ya'ni uni kompyuterda bajarishdan avval, uning ichki mantig'ini qo'lda tekshirishdan iborat bo'lib, bu qadamda DV ishlab chiqilishining har bir bosqichida qabul qilinayotgan qrorlarni nazorat qilish zarurligi to'g'risidagi umumiy tamoyil amalga oshiriladi. Modulni tekshirish usullari qo'llanmaning 3.2.4-bo'limida muhokama qilinadi.

Va, nihoyat, modulni ishlab chiqishning so'nggi qadamida modulni tekshirish (kompilyator yordamida) tugallanadi hamda modulni sozlash jarayoniga o'tiladi.

3.2.2. Tuzilmaviy dasturlash

Modulni dasturlashda u nafaqat kompyuterga, balki insonga ham tushunarli bo'lishini hisobga olish kerak: modulni ishlab chiquvchilar ham, uni tekshiruvchi shaxslar ham, modulni sozlash uchun test tuzuvchi testchilar ham, modulga talab qilingan o'zgarishlarni kirituvchi DV kuzatib boruvchilari ham modul ishi mantig'ini qayta-qayta tahlil etishga majbur bo'ladilar. Hozirgi zamon dasturlash tillarida ushbu mantiqni g'oyat chalkashtirib yuboradigan vositalar ko'p bo'lib, ular modulni inson uchun tushunilishini qiyinlashtiradi. Shuning uchun to'g'ri keladigan til vositalarini tanlash choralarini ko'rish hamda ma'lum dasturlash tartibiga rioya qilish zarur. Buning bilan bog'liq holda Deykstra dasturni bir nechta turdagi boshqarish konstruktsiyalari (tuzilmalar)dan iborat kompozitsiya sifatida qurishni taklif qildiki, bu boshqarish konstruktsiyalari dastur ishi mantig'ini tushunarliroq qilish imkonini beradi. Faqat shunday konstruktsiyalardan foydalangan dasturlash *tuzilmaviy* (strukturali) dasturlash degan nom bilan ataldi.

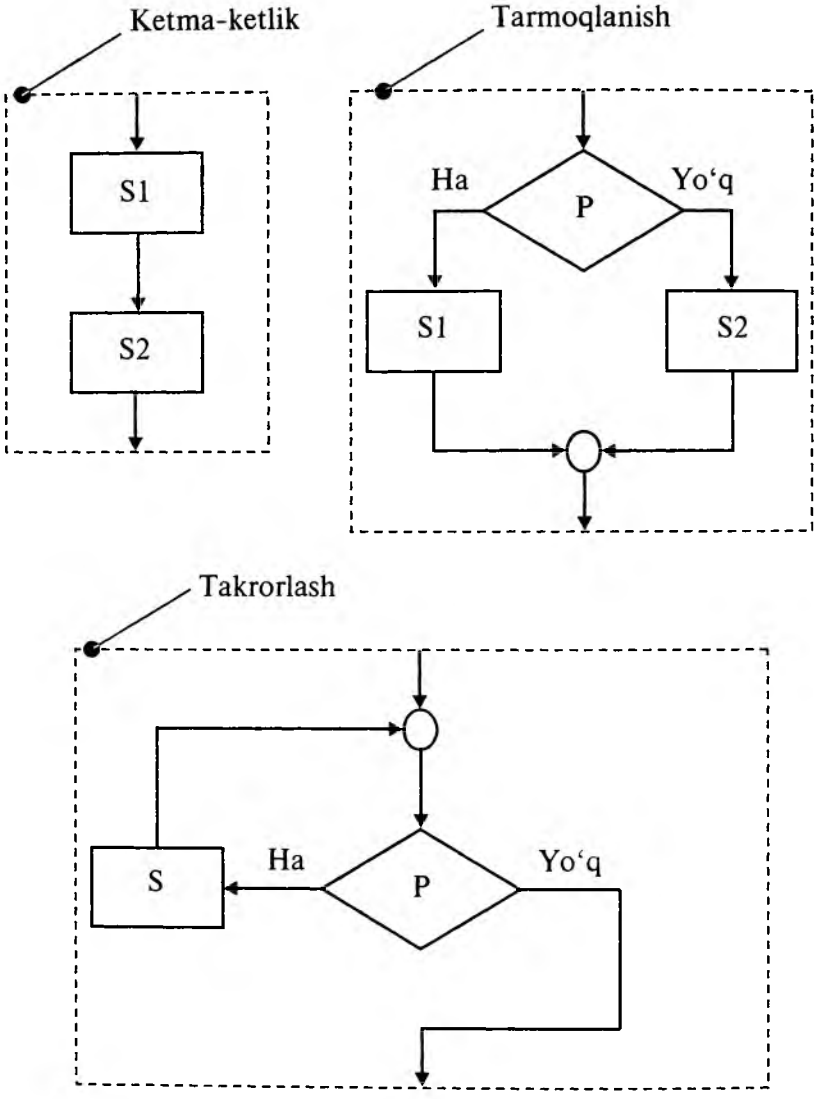
Strukturali (tuzilmaviy) dasturlashning asosiy konstruktsiyalari quyidagilardir: ketma-ketlik, tarmoqlanish va takrorlanish (4.4-rasmga qarang). Umumlashma operatorlar (ishlov berish uzellari) - S, S1, S2 hamda predikat (shart) - R ushbu konstruktsiyalarning tarkibiy qismlari (komponentlaridir). Bunda yo qo'llanayotgan

dasturlash tilining oddiy operatori (o'zlashtirish, kirish, chiqish va protseduraga murojaat qilish operatorlari), yoki tuzilmaviy (strukturali) dasturlash asosiy boshqaruv konsturktsiyalarining kompozitsiyasi bo'lgan dastur fragmenti umumlashma operator sifatida xizmat qilishi mumkin. Muhimi shundaki, har bitta konstruktsiya boshqaruv bo'yicha faqat bitta kirish va bitta chiqishga ega.

Yana shunisi ham muhimki, bu konstruktsiyalar hozirdanoq matematik ob'ektlardirlar (aynan shu tuzilmaviy dasturlashning muvaffaqiyatini ta'minlaydi). Har bitta tuzilmaviy bo'lmagan dastur uchun funksional ekvivalent bo'lgan (ya'ni aynan bitta masalani yechadigan) tuzilma holiga keltirilgan dasturni yaratish mumkin. Tuzilma holiga keltirilgan dasturlar uchun ma'lum bir xossalarning matematik isbotini berish mumkin. Bu esa dasturdagi ayrim xatolarni aniqlash imkonini beradi. Bu masalaga alohida ma'ruza bag'ishlanadi.

Tuzilmaviy dasturlash ba'zida "GO TO siz dasturlash" deb ham ataladi. Biroq bu yerda gap operator GO TO da emas, balki undan betartib foydalanishdadir. Ko'p hollarda tuzilmaviy dasturlash ayrim dasturlash tillarida (masalan, FORTRAN da) aks ettirilganda, o'tish operatori (GO TO) tuzilmaviy konstruktsiyalarni ishlatish uchun qo'llanadi. Bu tuzilmaviy dasturlash tamoyilini buzmaydi. Dasturni aynan "tuzilmaviy bo'lmagan" o'tish operatorlari chalkashtiradi, bunda ayniqsa matn modulida bajarilayotgan o'tish operatoridan yuqorida (avval) joylashgan operatorga o'tish chalkashtiradi. Shunday bo'lsa-da, o'tish operatorini chetlab o'tishga urinsak, tuzilmaviy dasturlar nihoyatda katta va qo'pol bo'lib ketishi mumkin. Bu esa ularning aniqligiga zarar yetkazadi va matn modulida qo'shimcha xatolarning paydo bo'lish xavfini tug'diradi. Shuning uchun mumkin bo'lgan o'rinda o'tish operatorini chetlab o'tishni maslahat berish mumkin, ammo bu dastur aniqligiga zarar yetkazmasligi kerak.

O'tish operatorini qo'llashning foydali jihatlari ham bor. O'tish operatori yordamida tsikldan, ya'ni muayyan tuzilmaviy birlik (umumlashma operator) ishini tugallaydigan alohida shart-sharoit bo'yicha bajariladigan protseduradan chiqishni amalga oshirish mumkin. Buning bilan u dastur tuzilmasini faqat lokal (ya'ni boshqalarga ta'sir qilmaydigan bitta o'rinda) buzishi mumkin xolos. Favqulodda (odatda xato) vaziyatlarga nisbatan yuzaga keladigan reaktsiya (munosabat)ni dastur tuzilmasida ishga solish ancha qiyinchiliklar tug'diradi.



3.4-rasm. Tuzilmaviy dasturlashning asosiy boshqarish konsturktsiyalari

Chunki bunda tuzilmaviy birlikdan muddatdan oldin chiqishnigina emas, balki bu vaziyatga tegishli ishlov berishni ham amalga oshirish (masalan, to'g'ri keladigan tashxislash axborotini chiqarib berish) talab qilinadi. Favqulodda vaziyatning ishlovchisi dasturning to'g'ri kelgan tuzilmaviy bosqichida turishi mumkin, unga murojaat esa turli quyi bosqichlardan turib amalga oshirilishi mumkin.

Favqulodda vaziyatlarga reaktsiya (munosabat) quyidagicha amalga oshiriladi. Favqulodda vaziyatlarning ishlovchilari u yoki bu tuzilmaviy birlik oxiriga joylashadi hamda har bir shunday ishlov beruvchi shunday dasturlanadiki, ishini tugatgach, o'zi o'rnashtirilgan tuzilmaviy birlik oxiridan chiqishni amalga oshiradi. O'tish operatori shunday ishlov beruvchiga ushbu tuzilmaviy birlikdan turib murojaatni amalga oshiradi.

3.2.3. Qadamma-qadam detallashtirish hamda soxta kod (psevdokod) haqida tushuncha

Tuzilmaviy (strukturali) dasturlash modul matni qanday bo'lishi kerakligi haqida maslahatlar beradi. Bunday matnni tuzish uchun dasturchi qanday yo'l tutishi kerak degan savol tug'iladi. Odatda modulni dasturlash ishini uning blok-sxemasini tuzishdan boshlaydilar. Bu blok-sxema modul ishi mantig'iga umumiy tavsif beradi. Biroq zamonaviy dasturlash texnologiyasi tegishli kompyuter yordamisiz bu ishni amalga oshirmaslikka maslahat beradi. Garchi blok-sxemalar modul ishi mantig'ini ancha aniq tasavvur qilish imkonini bersa-da, ularni qo'lda kodlashda dasturlash tilida g'oyat o'ziga xos xatolar manbai yuzaga keladi: asosan ikkio'lchamli bo'lgan blok-sxemalarni chiziqli matn (lineyniy teks) da aks etishi, modul ishi mantig'ini buzib ko'rsatish havfi bo'ladi. Blok- sxemalarni tuzishda grafik muharrir (redaktor) dan foydalanilsa, yoki bu blok-sxemalar bo'yicha matn avtomatik ravishda dasturlash tiliga generatsiya qilinadigan darajada formallashgan (qoliplangan) bo'lsa (xuddi R-texnologiyada bo'lganidek), bu holat bundan mustasno.

Modul matnini tuzishning asosiy usuli sifatida hozirgi zamon dasturlash texnologiyasi *qadamma-qadam detallashtirish*ni tavsiya etadi. Bu usulning mohiyati shundaki, modul matnini ishlab chiqish jarayoni bir qator qadamlarga bo'linadi. Birinchi qadamda modul ishining umumiy sxemasi chiziqli matniy shaklda (ya'ni g'oyat yirik tushunchalardan foydalangan holda) tavsiflanadi. Bunda bu tavsif to'liq formallashmagan hamda uning inson tomonidan idrok etilishiga qaratilgavn bo'ladi. Har bir navbatdagi qadamda avvalgi qadamlardan birida ishlab chiqilgan biron-bir tavsifdagi tushuncha (uni *aniqlanayotgan tushuncha* deb ataymiz) yanada aniqlashtiriladi va detallashtiriladi. Bunday qadam natijasida tanlab olingan va aniqlanayotgan tushunchaning tavsifi yaratiladi. Barcha aniqlanayot-

gan tushunchalar aniqlangach (ya'ni pirovard natijada bazaviy dasturlash tilida ifodalab berilgach), bu jarayon o'z nihoyasiga yetadi. So'nggi qadamda bazaviy dasturlash tilidagi matn yuzaga keladi hamda tuzilmaviy dasturlash konstruktsiyalarining barcha kirishlari ushbu dasturlash tili vositalari bilan ifodalanadi. Bunda yana aniqlanayotgan tushunchalarning barcha kirishlari ularning berilgan tavsiflariga almashtiriladi.

Qadamma-qadam detallashtirishda ko'rsatib o'tilgan tavsiflarni taqdim etish uchun *soxta kod* nomini olgan qisman formallashgan tildan foydalaniladi. Bu til tuzilmaviy dasturlashning barcha konstruktsiyalaridan foydalanish imkonini beradi. Bu konstruktsiyalar esa umumlashma operatorlar va shart-sharoitlarni taqdim etish uchun tabiiy tildagi formallashmagan fragmentlar bilan birgalikda formallashgan (qoliplangan) tarzda rasmiylashtiriladi. Umumlashma operatorlar va shart-sharoitlar sifatida shuningdek bazaviy dasturlash tilidagi tegishli fragmentlar ham berilishi mumkin.

Bazaviy dasturlash tilidagi modulning tashqi tavsifini soxta koddagi bosh tavsif deb hisoblash mumkin bo'lib, uning tarkibiga quyidagilar kiradi:

- bazaviy tildagi modulning boshi, ya'ni ushbu modulning birinchi gapi yoki sarlavhasi (spetsifikatsiyasi);
- bazaviy tildagi tavsiflar bo'limi (majmui), bunda protseduralar va funksiyalarning tavsiflari o'rniga ularning faqat tashqi tavsiflari keltiriladi;
- modul tanasi operatorlari ketma-ketligining bitta umumlashma operator sifatida noformal ifodalanishi;
- bazaviy tildagi modulning so'nggi gapi (oxiri).

Protsedura yoki funksiya tavsifi tashqi tomondan bir xil shakllantiriladi. Darvoqe, Deykstraga amal qiladigan bo'lsak, tavsiflar bo'limini bu o'rinda formal bo'lmagan ifoda bilan taqdim etish afzalroqdir. Bunda formal bo'lmagan ifoda alohida tavsif sifatida detallashtiriladi.

Umumlashma operatorning soxta koddagi formal bo'lmagan ifodasi bu ifoda mazmunini umumiy tarzda ochib beradigan ixtiyoriy gap bilan tabiiy tilda amalga oshiriladi. Bunday ifodani shakllantirishga qo'yiladigan yagona formal talab quyidagichadir: bu gap bitta yoki bir nechta grafik (bosma) satrni to'liq egallashi hamda nuqta (yoki buning uchun maxsus ajratilgan boshqa biron belgi bilan tugallanishi) lozim.

```
Ketma-ket kelish
        umumlashma_operator
        umumlashma_operator
Tarmoqlanish:
        AGAR shart BU HOLDA
          umumlashma_operator
        AKS HOLDA
          umumlashma_ operator
        HAMMA AGAR
Kaytariq:
         HOZIRCHA shart-sharoit QILMOQ
         umumlashma_operator
         HAMMA HOZIRCHA
```

3.5-rasm. Soxta kodda tuzilmaviy dasturlashning asosiy konstruksiyalari

Har bir formal bo'lmagan umumlashma operator uchun tuzilmaviy dasturlash konstruktsiyasining kompozitsiyasi hamda boshqa umumlashma operatorlar yordamida uning ishi mantig'ini ifodalab beradigan (uning mazmunini detallashtirib beradigan) alohida tavsif yaratilishi kerak. Bunday tavsifning sarlavhasi sifatida detallashtirilayotgan umumlashma operatorning formal bo'lmagan ifodasi kelishi kerak. Tuzilmaviy dasturlashning asosiy konstruktsiyalari quyidagi ko'rinishda taqdim etilishi mumkin (3.5-rasmga qarang).

Soxta koddagi umumlashma operator sifatida yuqorida ko'rsatib o'tilgan o'tish operatorining juz'iy holatlaridan foydalanish mumkin (3.6-rasmga qarang).

```
Qaytariq (tsikl)dan chiqish:
        CHIQMOQ
Protsedura (funksiya)dan chiqish:
        QAYTMOQ
Favqulodda vaziyatni ishlashga o'tish:
        QO'ZG'ATMOQ istisno_ismini (?        )
```

3.6-rasm. Umumlashma operator sifatidagi o'tish operatorining juz'iy holati

Favqulodda vaziyatlar (istisnolar) ishlov beruvchilarining ketma-ketligi modul yoki protsedura tavsifining oxirida beriladi. Shunday ishlov beruvchilarning har bittasi quyidagi ko'rinishga ega:

ISTISNO istisno_nomi
    umumlashma operator
BARCHA ISTISNO

Favqulodda vaziyatga ishlov beruvchining parametrlarsiz protseduradan farqi shundaki, protsedura bajarilgach, boshqaruv bu protseduraga murojaatdan keyin keluvchi operatorga qaytadi, istisno bajarilgandan so'ng esa boshqaruv modul yoki protsedura (funksiya) ga murojaatdan so'ng keladigan operatorga qaytadi, bunda ushbu istisno gap borayotgan modul yoki protsedura oxirida joylashtirilgan bo'ladi.

Detallashtirishning har bir qadamida ancha mazmunli tavsif berilishi tavsiya etiladiki, bu tavsif tushunishga oson, ko'rgazmali bo'lishi, ya'ni bir varaqli matnga sig'ishi lozim. Odatda bu shuni bildiradiki, bunday tavsif tuzilmaviy dasturlashning besh-oltita konstruktsiyasi uchun kompozitsiya vazifasini o'tamog'i lozim. Oraga suqib kiritilgan konstruktsiyalarni o'ng tomonga bir necha pozitsiyaga surib joylashtirish tavsiya etiladi (3.7-rasm). Natijada blok-sxemalar bilan bemalol raqobatga kirisha oladigan ish mantig'i tavsifiga ega bo'lish mumkin. Bu tavsif xatto afzallikka ham ega: bunda tavsifning chiziqliligi saqlanadi.

```
FAYLDAGI YOZUVLARNI BERILGAN FILTRNI
QONIQTIRADIGAN BIRINCHI YOZUVGACHA O'CHIRISH:

  FAYL BOSHINI O'RNATISH.
  FAYL TUGAMASDAN
   NAVBATDAGI YOZUVNI QILISH O'QISH.
    AGAR NAVBATDAGI FAYL FILTRNI QONIQTIRSA,
          U HOLDA
  CHIQISH

AKS HOLDA
NAVBATDAGI YOZUVNI FAYLDAN O'CHIRISH
HAMMASINI, AGAR
HAMMASINI, -GACHA
AGAR YOZUVLAR O'CHIRILMAGAN BO'LSA, U HOLDA
"YOZUVLAR O'CHIRILMADI" DEB YOZIB QO'YISH
AKS HOLDA
"n-TA YOZUVLAR O'CHIRILDI" DEB YOZIB QO'YISH
HAMMASINI, AGAR
```

3.7-rasm. Soxta kodda detallashtirishning bitta qadamiga misol

Qadamma-qadam detallashtirish g'oyasining muallifi Deykstra degan fikr mavjud. Biroq Deykstra nazarimizda modul matnini tuzishning printsipial farq qiladigan yanada mukammalroq va istiqboli porloq bo'lgan usulini taklif qiladi. Birinchidan, u operatorlarni aniqlash bilan birga qo'llanayotgan ma'lumotlar tuzilmasini ham asta-sekin (qadamma-qadam) aniqlashni taklif qiladi. Ikkinchidan, Deykstra har bir qadamda detallashtirish uchun qandaydir virtual mashinani yaratishni hamda uning atamalari vositasida barcha aniqlanayotgan tushunchalarni detallashtirishni taklif qiladi.

Shunday qilib, Deykstra mohiyat e'tibori bilan gorizontal qavatlar bo'yicha detallashtirishni taklif qilganki, bu qavatma-qavat tizimlar haqidagi g'oyani modulni ishlab chiqish darajasiga olib o'tilishini bildiradi. Modulni ishlab chiqishning bunday usuli hozirgi paytda ADA paketlar tili va ob'ektli yo'naltirilgan dasturlash vositalari tomonidan qo'llab-quvvatlanadi.

3.2.4. Dasturiy modul ustidan nazorat

Dasturiy modulni nazorat qilishning quyidagi usullari qo'llanadi:
- modul matnining statik tekshiruvi;
- mufassal kuzatuv;
- dasturiy modul xususiyatlarini isbotlash.

Modul matnining statik tekshiruvida ushbu matn moduldagi xatolarni topish maqsadida boshdan oyoq ko'rib chiqiladi. Odatda bunday tekshiruvga modul ishlab chiquvchisidan tashqari yana bitta yoki xatto bir nechta dasturchilar jalb qilinadi. Bunday tekshiruv paytida aniqlangan xatolarni shu topda emas, balki modul matnini o'qish tugaganidan so'ng to'g'rilash tavsiya qilinadi.

Mufassal kuzatuv modulni dinamik nazorat qilish turlaridan biridir. Bunda ham bir nechta dasturchi ishtirok etib, ular modulning bajarilishini biron-bir matnlar to'plamida sinab ko'radi.

Dasturiy modul xususiyatlarining isbotiga uni verifikatsiylash orqali amalga oshiriladi. Ushbu usul hozircha juda kam qo'llanishini qayd etish bilan cheklanamiz.

3.3. DASTURIY VOSITALARNING ISHLAB CHIQILISHIGA OB'EKTLI YONDASHUV

Tarkibi: ob'ektli dasturlash maqsadi va uning xususiyatlari

3.3.1. Dasturlashda ob'ektlar va munosabatlar. Dasturiy vositalarning ishlab chiqilishiga ob'ektli yondashuvning mohiyati

Bizni o'rab turgan olam ob'ektlar hamda ular o'rtasidagi munosabatlardan iborat. V. Dal lug'atiga ko'ra ob'ekt (predmet) bu his-tuyg'u vositasida yoki aqlan idrok etiladigan barcha narsalardir (ya'ni moddiy ob'ekt yoki aqliy ob'ekt). Shunday qilib, *ob'ekt* o'zida biron-bir mohiyatni aks ettiradi hamda vaqt o'tishi bilan o'zi bilan biron-bir munosabatlarga kirishgan boshqa bir ob'ekt ta'sirida o'zgarishi mumkin bo'lgan qandaydir holatga ega bo'ladi. U ichki tuzilishga ega bo'lishi, ya'ni, aytaylik, o'zlari ham o'zaro qandaydir munosabatlarga kirishgan boshqa ob'ektlardan tashkil topishi mumkin. Bundan kelib chiqqan holda, dunyoning ob'ektlardan iborat tabaqaviy tuzilishini qurib chiqish mumkin. Biroq, bizni o'rab turgan olamni har gal konkret ko'rib chiqadigan bo'lsak, ayrim ob'ektlar bo'linmas bo'lib chiqadi, bunda ko'rib chiqish maqsadlari bilan bog'liq holda bunday (bo'linmas) ob'ektlar tabaqaning turli darajalariga mansub bo'lishi mumkin. *Munosabat* ayrim ob'ektlarni bog'laydi: bu ob'ektlarning o'zaro bog'lanishi biron-bir xususiyatga ega deb hisoblash mumkin. Agar munosabat n ta ob'ektni bog'layotgan bo'lsa, bu holda bunday munosabat n o'rinli (n-li) deb ataladi. Biron-bir konkret munosabat bilan bog'lanishi mumkin bo'lgan ob'ektlarning birlashgan har bitta o'rinda turli ob'ektlar (biroq aniq ob'ektlar) mavjud bo'lishi mumkin (bunday hollarda ular *ma'lum sinfga mansub ob'ektlar* deb ataladi). Bir o'rinli munosabat *ob'ektning oddiy xususiyati* deb ataladi. Ob'ektlarning ko'p o'rinli munosabatlarini *ob'ektning assotsiatsiyali xususiyati* (agar ushbu ob'ekt ushbu munosabatda ishtirok etsa) deb ataymiz. Ob'ektning holati ushbu ob'ektning oddiy yoki assotsiatsiyali xususiyatlarining ma'nolari (qiymatlari) ga ko'ra o'rganilishi mumkin. Biron-bir umumiy xususiyatlarga ega bo'lgan barcha ob'ektlar to'plami *ob'ektlar sinfi* deb ataladi.

Bizni o'ragan olamni idrok etish yoki o'zgartirish jarayonida

biz hamma vaqt ushbu olamning u yoki bu soddalashtirilgan modelini olib qaraymiz hamda unga bizni o'ragan olamdan o'zimizni qiziqtirgan biron-bir sinflarning ob'ektlari va munosabatlarini kiritamiz.

Ichki tuzilishga ega bo'lgan har bir ob'ekt o'zining modelli olamidan iborat bo'lib, ushbu tuzilmaning ob'ektlari va ularni bog'lab turgan munosabatlarni o'z ichiga oladi. Shunday qilib, atrof olamni modelli olamlarning tabaqaviy tuzilmasi (aniqrog'i, shunga yaqinroq) sifatida olib qarash mumkin.

Hozirgi paytda bizni o'rab turgan olamni idrok etish yoki o'zgartirish jarayonida turli xildagi axborotni ishlash uchun kompyuter texnikasi keng qo'llanadi. Buning bilan bog'liq holda ob'ektlar va munosabatlarning kompyuter (axborotli) taqdimoti qo'llanadi. Har bir ob'ekt axboriy jihatdan ushbu ob'ektning holatini aks ettiruvchi biron-bir tuzilma vositasida taqdim etilishi mumkin. Bu ob'ektning oddiy xususiyatlari bevosita ushbu tuzilmaning alohida komponentlari (tarkibiy qismlari) ko'rinishida yoki ushbu ma'lumotlar tuzilmasi ustidagi maxsus funksiyalar vositasida berilishi mumkin. Assotsiatsiyali munosabatlar (n>1 uchun n o'rinli munosabatlar) ni yo aktiv (faol) shaklda yoki passiv shaklda taqdim etish mumkin. Aktiv shaklda n o'rinli munosabat biron-bir dasturiy fragment (qism) ko'rinishiga ega bo'lib, u yo n o'rinli funksiyani (tegishli ob'ektlar birlashmasi xususiyatining qiymatini belgilaydigan funksiyani), yoki taqdim etilayotgan munosabatlar vositasida bog'lanayotgan ob'ektlarning holati bo'yicha ushbu ob'ektlarning ayrimlarining holatlarini o'zgartiradigan protsedura (amal) ni ishga soladi. Passiv shakldagi bunday munosabat konkret (muayyan, aniq) munosabatlarga bog'liq bo'lmagan umumiy protseduralar bo'yicha qabul qilingan bitimlar asosida talqin etiladigan biron-bir ma'lumotlar tuzilmasi ko'rinishiga ega bo'lishi mumkin. Qanday holatda bo'lmasin, munosabat taqdimoti ma'lumotlarni ishlash bo'yicha biron-bir xatti-harakatlarni belgilaydi.

Modelli olamni tadqiq etishda foydalanuvchilar kompyuterdan axborotni turlicha yo'llar bilan olishlari (yoki olishni xohlashlari) mumkin.

Bir o'rinda ularni muayyan ob'ektlarning ayrim xususiyatlari haqidagi axborot olish yoki modelli olamning ayrim ob'ektlari o'rtasidagi qandaydir natijalar qiziqtirib qolishi mumkin. Bunday talablarni qondirish maqsadida foydalanuvchilarni qiziqtirgan

funksiyalarni bajaruvchi tegishli DV lar yoki foydalanuvchilarni qiziqtirgan munosabatlar haqidagi axborotni chiqarib bera oladigan axborot tizimlari ishlab chiqiladi. Kompyuter texnikasi taraqqiyotining boshlang'ich davrida (kompyuterlar quvvati hali uncha yuqori bo'lmagan paytda) modelli olamga bunday yondashuv tabiiy bir hol edi. Aynan shunday yondashuv DV ning ishlab chiqilishiga avvalgi ma'ruzalarda batafsil ko'rib chiqilgan *funksional* (*relyativ*) yondashuvni keltirib chiqardi. Bu yondashuvning mohiyati shundaki, DV (shu hisobda dastur matnlari) tuzilishining tavsifi va qurilishi uchun muntazam ravishda *funksiya* (*munosabat*) *dekompozitsiyasidan* foydalaniladi. Bunda buyurtma berilayotgan va amalga oshirilayotgan funksiyalar yuklangan modelli olam ob'ektlarining o'zlari alohida qismlarda (ya'ni ushbu funksiyalarni bajarish uchun zarur bo'lgan xajmda) hamda ushbu funksiyalarni amalga oshirish uchun qulay shaklda taqdim etilgan. Shu yo'l bilan talabdagi funksiyalarning samarali bajarilishiga erishilgan, biroq foydalanuvchini qiziqtirgan modelli olamning yaxlit va adekvat kompyuter ko'rinishi yaratilmagan. Ushbu modelli olam haqida DV dan olish mumkin bo'lgan axborot xajmini va ko'rinishini ozgina kengaytirishga urinish ham DV ning o'zida jiddiy yangiliklar kiritishni talab qilishi mumkin edi.

Boshqa o'rinlarda foydalanuvchini modelli olam ob'ektlari holatlarining o'zgarishi ustidan kuzatish olib borish qiziqtirishi mumkin. Bu esa bunday ob'ektlarning tegishli axborot modellaridan foydalanishni, modelli olam ob'ektlarining o'zaro muloqot jarayonlarini modellashtiradigan dasturiy vositalarning yaratilishini, foydalanuvchiga ushbu axborot modellari (*foydalanuvchilik ob'ektlari*)ga kirish huquqining berilishini talab qiladi. Ishlab chiqishning an'anaviy usullaridan foydalanib bu ishni bajarish ancha qiyin masala edi. Bu masalani to'laroq hal qilishda DV ishlanmasiga ob'ektli yondashuv ko'proq qo'l keladi. Uning mohiyati DV ning tavsifi va tuzilishida *ob'ektlar dekompozitsiyasi*dan muttasil ravishda foydalanishdan iborat. Bunda ushbu DV bajarayotgan funksiyalar (munosabatlar) boshqa darajalar ob'ektlarining munosabatlari orqali ifodalanadi, ya'ni ularning dekompozitsiyasi ob'ektlar dekompozitsiyasiga ko'p darajada bog'liq bo'ladi.

DV ishlab chiquchilarining nuqtai nazaridan kelib chiqsak, ob'ektlar (hamda ularning sinflari)ning quyidagi kategoriyalarini farqlash lozim:

- modelli (moddiy yoki aqliy) olam ob'ektlari;

- real olam ob'ektlarining axboriy (ya'ni axborot beruvchi) modellari (ularni *foydalanuvchilik ob'ektlari* deb ataymiz);

- dasturlarning bajarilish jarayoni ob'ektlari;

- DV ni ishlab chiqish jarayoni ob'ektlari (*texnologik dasturlash ob'ektlari*).

Bundan tashqari, modelli olam bilan foydalanuvchi o'rtasidagi o'zaro aloqalar xususiyatlarining kompyuterda taqdim etilish usulidan kelib chiqib, passiv va aktiv ob'ektlarni ham farqlash mumkin. *Passiv ob'ekt* muayyan turga mansub turli ma'lumotlarni saqlay oladigan (bu ma'lumotlar ushbu ob'ektning turli *holatlarini* bildiradi) hamda biron-bir operatsiyalar to'plamining bajarilishi bilan bog'liq bo'lgan axborot muhitining biron-bir fragmenti (qismi)dan iboratdir. Bunday ob'ektga nisbatan qo'llanadigan operatsiyalar tashqi aktiv kuch ta'sirida amalga oshirilib, bu kuch yo foydalanuvchidan, yoki biron-bir dasturiy fragmentni bajarish jarayonida mana shu fragmentdan keladi. *Aktiv ob'ekt* bu passiv ob'ektning shunday kengaytirilgan ko'rinishiki, unda axboriy muhit fragmenti bajarilish jarayonida (*aktiv holatda*) tura oladigan dasturiy fragmentlarni ham saqlay olishi mumkin. Biron-bir dasturiy fragmentlari aktiv holatda turgan aktiv ob'ekt o'zi joylashtirilgan operatsiya muhitidan xabarlar yoki signallarni qabul qilib olish hamda ushbu xabarlar va signallarga javob tariqasida biron-bir operatsiyalarni mustaqil bajarish imkoniyatiga ega. Shunday qilib, aktiv ob'ekt *ichki aktiv kuchga* ega deb hisoblash mumkin.

DV ishlanmasiga ob'ektli yo'naltirilgan yondoshuv haqida gap borganda, modelli olam ob'ektlariga tavsif berishga hamda ularning axborot modellarini tuzishga yo'naltirilgan yondashuv nazarda tutiladi. Bunda DV ishlanmasining ko'pchilik jarayonlari o'ziga xos ("ob'ektli") xususiyatlar kasb etadiki, ular quyidagilardan iborat:

- ob'ektlar va ularning sinflarini tavsiflash imkonini beradigan tushunchalar tizimidan foydalanish;

- ob'ektlar dekompozitsiyasi DVni soddalashtirishning asosiy vositasi bo'lib xizmat qiladi;

- ishlab chiqish jarayonlarini soddalashtirish uchun dasturdan tashqari abstraktsiyalar qo'llanadi;

- funksiyalarning ishga tushirilishidan ko'ra ma'lumotlar tuzilmasini ishlab chiqishga yetakchi o'rin berish.

DV ni ishlab chiqishdagi o'ziga xos xususiyatlarning asosiylarini sharsharaviy texnologiya modeli doirasida ko'rib chiqamiz.

3.3.2. Dasturiy vositaning tashqi tavsifini ishlab chiqishda ob'ektli yondashuv xususiyatlari

Ob'ektli yondashuvda DVning tashqi tavsif bosqichi relyatsion yondashuvga nisbatan ancha sig'imi keng va mazmundor bo'lib chiqadi.

Talablarni aniqlash foydalanuvchi talab qilinayotgan DV vositasida o'rganmoqchi yoki shunchaki qo'llamoqchi bo'lgan modelli olamga noformal tavsif berishdan iborat. Bunda prototiplashning ahamiyati ortadi. Bu yondashuvda prototiplash bosqichi ko'p hollarda DV ishlanmasining keyingi bosqichlarida ish hajmining kamayishi hisobiga qoplanadi.

Sifat spetsifikatsiyasi o'z mazmunini saqlab qoladi.

Talablar spetsifikatsiyasi jarayonining mazmuni ancha o'zgaradi: DV ning funksional spetsifikatsiyasini ishlab chiqish o'rniga modelli olamning uch qismdan iborat bo'lgan formal tavsifi yaratiladi:

- ob'ektli model tavsifi;
- dinamik model tavsifi;
- funksional model tavsifi.

Ushbu qismlarning vazifasi nimadan iborat? Buni quyidagicha ta'riflash mumkin: ob'ektli model nimadir sodir bo'lishi mumkin bo'lgan nimanidir aniqlaydi; dinamik model sodir bo'lish vaqtini aniqlaydi; funksional model nima sodir bo'lishini aniqlaydi.

Ob'ektli model ishlab chiqilayotgan DV taqdim etishi lozim bo'lgan modelli olamning statik ob'ektli tuzilmasini ko'rsatadi. U qo'llanayotgan ob'ektlar sinflarining hamda ushbu sinflar o'rtasidagi munosabatlarning ta'riflarini, shuningdek, ushbu sinflarning qo'llanayotgan ob'ektlari ta'rifini hamda bu ob'ektlar o'rtasidagi munosabatlarni o'z ichiga oladi.

Ob'ektli modelda *ob'ektlar sinfi* odatda uchlik ko'rinishida bo'ladi: sinf nomi, atributlar (belgilar) ro'yxati, operatsiyalar ro'yxati.

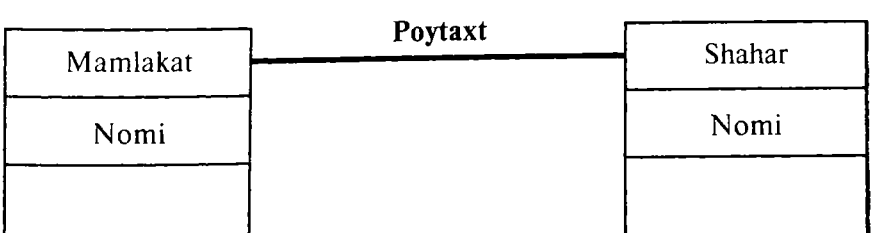
Har bir atribut biron nom oladi hamda muayyan tur ma'nolariga ega bo'lishi mumkin. Mohiyatan, sinf atributi ushbu sinf ob'ektlarining biron-bir oddiy xususiyatini ifodalaydi. Ob'ektlarning biron-bir oddiy xususiyatlarini atribut sifatida olib qarash, ayniqsa, ushbu atributlar mazmuniga ko'ra ob'ektlar tasnifi amalga oshirilayotganda g'oyat qulay. Sinf taqdimotida ko'rsatiladigan operatsiyalar ushbu sinf ob'ektlarining boshqa xususiyatlarini (xoh oddiy bo'lsin, xoh assotsiativ bo'lsin) aks ettiradi. Ular ushbu sinf

ob'ektlarini nima qilish mumkinligi (yoki ushbu ob'ektlarning o'zlari nima qilishlari mumkinligi) ni ko'rsatadi.

Ob'ektli modelda ob'ektlar o'rtasidagi munosabatlar ushbu ob'ektlar mansub bo'lgan sinflar o'rtasidagi munosabatlar sifatida umumlashtiriladi. Bunda odatda faqat bir o'rinli va ikki o'rinli ob'ektlararo munosabatlar qo'llanadi. Bundan ko'ra murakkabroq munosabatlar ob'ektli modellarning murakkablashuviga olib keladi. Boshqa tomondan esa bunday munosabatlar qo'shimcha sinflarni aniqlash hisobiga har vaqt ikki o'rinli munosabatlarga kelitirilishi mumkin. Bir o'rinli munosabatlar **atributlar** deb ataladi. Bunda ob'ektning ayrim atributlari sinf atributlaridan, ularga aniq ma'nolar (qiymatlar) taqdim etish yo'li bilan hosil bo'ladi. Ikkita (va undan ortiq) ob'ektlar o'rtasidagi munosabat *aloqa* deb ataladi, ularning umumlashmasi (sinflararo munosabatlar) esa **assotsiatsiyalar** deb ataladi. Ob'ektli modelda assotsiatsiyalar muhim rol o'ynaydi - ular ob'ektlar o'rtasida yo'l qo'yilishi mumkin bo'lgan aloqalarni belgilab beradi. Assotsiatsiyalarning quyidagi turlari mavjud:

- ob'ektlar holatlarining o'zaro aloqalari;
- ob'ektlar agregatsiyalashuvi (tuzilmalashuvi);
- sinflar abstraktsiyalashuvi (tug'ilishi).

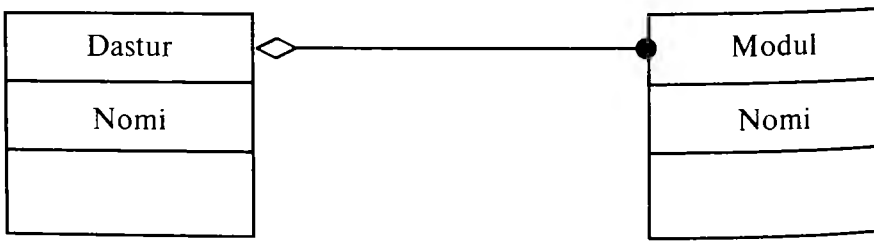
"O'zaro aloqa" assotsiatsiyasi, mohiyatan, bunday munosabatga kirishgan sinf ob'ektlari biron-bir operatsiyalarning parametrlari bo'lishi mumkinligini bildiradi. "Agregatsiyalashish" assotsiatsiyasi bunday aloqaga kirishgan sinflardan birining ob'ekti ushbu sinflardan boshqasining ob'ektini o'z ichiga olishini (yoki olishi mumkinligini) bildiradi. "Abstraktsiyalashish" assotsiatsiyasi bunday munosabatlarga kirishgan sinflardan biri ushbu sinflardan boshqasining xususiyatlarini meros qilib olishini hamda boshqa (qo'shimcha) xususiyatlarga ham ega bo'lishi mumkinligini bildiradi.



3.8-rasm. Ob'ektlar sinflari o'rtasidagi munosabatlarga misol

Ob'ektli modelni taqdim etishda odatda ob'ektlar spetsifikatsiyasining grafik tillari (masalan, UML tili) qo'llanadi. Bunday tillarda sinflar va ob'ektlar to'g'ri to'rtburchaklar ko'rinishida beriladi va ularda ushbu ob'ektlarni spetsifikatsiyalayotgan axborot beriladi. Ikkita sinf o'rtasida munosabatlar o'rnatish uchun bu sinflarga mos keluvchi to'g'ri to'rtburchaklar chiziq bilan birlashtiriladi va chiziq tepasida turli grafik belgilar va ayrim yozuvlar keltiriladi. Grafik belgilar ushbu sinflar o'rtasidagi munosabatlar turini spetsifikatsiyalaydi, yozuvlar esa bu munosabatni to'liq identifikatsiya qiladi (ya'ni aniqlashtiradi). Masalan, 4.8-rasmda mamlakat va shahar sinflari o'rtasida berilgan munosabat "birga bir" ko'rinishiga ega. Yanada aniqroq qilib aytganda, bu munosabat shuni bildiradiki, mamlakat sinfining har bir ob'ekti shahar sinfining faqat va faqat bitta ob'ekti bilan "poytaxtga ega" munosabati bilan bog'liq, shahar sinfining ushbu ob'ekti esa mamlakat sinfining boshqa bironta ob'ekti bilan bunday munosabat bilan bog'liq emas.

Ob'ektlar dekompozitsiyasini tavsiflash uchun agregatlashish ko'rinishidagi munosabat qo'llanadi. Masalan, 4.4-rasmda "dastur bitta yoki bir nechta moduldan iborat" munosabati ko'rsatilgan.



*3.9-rasm.*Ob'ektlar sinflari o'rtasida agregatlashish munosabatiga misol

Shuni ta'kidlash lozimki, relyatsion yondashuvda ob'ektli model tashqi axborot muhitining tavsifini to'liq o'z ichiga oladi.

Dinamik model DV taqdim etishi lozim bo'lgan modelli olamning ob'ektli modelidan ob'ektlar holatining yo'l qo'yiladigan o'zgarish ketma-ketliklarini ko'rsatadi. Bu model tashqi signallar (o'zaro aloqalar) ga javoban operatsiyalar ketma-ketligini tavsiflaydi, ammo bunda bu operatsiyalarning nima qilayotgani ko'rilmaydi. Agar tegishli ob'ektli model aktiv ob'ektlarga ega bo'lsa, bu holda dinamik modelga zarurat bo'ladi.

Dinamik modelning asosiy tushunchalari - hodisalar va ob'ektlar holatlaridir. *Hodisa* deganda bu yerda bir ob'ektning ikkinchi ob'ektga muayyan vaqtda sodir bo'ladigan elementar ta'siri tushuniladi. Bir hodisa mantiqan boshqasining ortidan kelishi yoki boshqasi bilan bog'liq bo'lmasligi mumkin. Boshqacha qilib aytganda, dinamik modelda hodisalar *qisman tartibga solingan* bo'ladi. *Ob'ekt holati* deganda bu yerda ob'ekt atributlari ma'nolari (qiymatlari)ning majmuyi hamda ushbu ob'ekt joriy aloqalarining boshqa ob'ektlar bilan tegishli tomonlari tushuniladi. Ob'ekt holati ushbu ob'ektga ta'sir etadigan biron-bir ikkita hodisa o'rtasidagi vaqt intervaliga bog'liqdir. Biron-bir hodisadan ta'sirlanish natijasida ob'ekt bir holatdan ikkinchi bir holatga *o'tadi.*

Buning bilan bog'liq holda dinamik modelda aktiv ob'ektlarning har bir sinfi uchun o'z *holatlar diagrammasi* tuziladi. Bu diagramma graf ko'rinishiga ega bo'lib, uning uchlari holatlarni bildiradi, yoylari esa ushbu holatlar o'rtasidagi o'tishlarni bildiradi. Bu o'tishlar hodisalar nomlari bilan belgilanadi. Ayrim o'tishlar ushbu o'tishlarga ruxsat beruvchi shart-sharoitlar bilan bog'iq bo'lishi mumkin. *Shart-sharoit* bu ayrim ob'ektlar atributlarining ma'nolari (qiymatlari)ga bog'liq bo'lgan predikatdir. Har bir shart-sharoit yoyda ko'rsatiladi, ushbu shart-sharoit yoy bo'ylab o'tishni boshqaradi. Shunisi e'tiborliki, holatlar diagrammasida ayrim holatlar yoki hodisalar muayyan operatsiyalar bilan bog'lanadi. Hodisa bilan bog'lanayotgan operatsiya ob'ektning ushbu hodisaga reaktsiyasini (munosabatini) bildiradi hamda ushbu operatsiya soniyada (biron-bir vaqt intervali nuqtasida) bajariladi deb hisoblanadi. Bunday operatsiya *xatti-harakat* deb ataladi. Holat bilan bog'lanayotgan operatsiya ushbu holat bog'liq bo'lgan vaqt intervali doirasida bajariladi (ya'ni ushbu interval bilan chegaralangan davomiylikka ega bo'ladi). Bunday operatsiya *faoliyat* deb ataladi. Holatlar diagrammasi ushbu operatsiyalar aktivlashuvini boshqarish yo'llarini belgilab beradi. Shunday qilib, holatlar diagrammasi bitta ob'ektlar sinfining xatti-harakatini tavisflaydi.

Dinamik model sinflar o'rtasidagi hodisalar yordamida barcha holatlar diagrammalarini birlashtiradi.

Funksional model nimani ko'rsatadi? U chiqish qiymatlari qanday qilib kirish qiymatlaridan kelib chiqib, yana ushbu qiymatlar hisoblab chiqariladigan qatorni ko'rsatmay turib, hisoblanishini ko'rsatadi. Bu model ob'ektli va dinamik modellarda qo'llanadigan barcha operatsiyalar (*tashqi operatsiyalar*), shart-sharoitlar

va cheklanishlarni belgilab beradi. DV ishlanmasiga relyatsion yondashuvda funksional model tashqi funksiyalar ifodasiga mos keladi.

Funksional modelda yirik operatsiyalarni belgilashda *oqimli diagrammalar* (*ma'lumotlar oqimi diagrammasi*) qo'llanadi. Ular bu operatsiyalarni ancha sodda operatsiyalar vositasida ifoda etish imkonini beradi. *Jarayonlar, ob'ektlar va ma'lumotlar oqimi* oqimli diagrammalarning asosiy tushunchalari bo'lib xizmat qiladi. Oqimli diagramma bu graf bo'lib, ob'ektlar yoki jarayonlar uning uchlari, ma'lumotlar oqimlari esa uning yoylari hisoblanadi. Jarayonlar bir ob'ektlardan kelib tushayotgan va saqlash uchun boshqa ob'ektlarga yuborilayotgan ma'lumotlarni qayta ishlaydi. Bu jarayonlar *ichki* operatsiyalarni tashkil qiladi. Ichki operatsiyalar orqali esa ushbu oqimli diagramma taqdim etgan operatsiya ifodalanadi. Ob'ektlar passiv (*ma'lumotlar omborlari*) va aktiv (*agentlar*) bo'lishi mumkin. Passiv ob'ektlar faqat ma'lumotlarni saqlash uchungina qo'llanadi, aktiv ob'ektlar esa ma'lumotlarni saqlash uchun ham, ularni qayta ishlash uchun ham qo'llanadi. Ma'lumotlar oqimi ma'lumotlar harakatining yo'l qo'yiladigan yo'nalishlarini hamda ma'lumotlar harakatining turlarini belgilab beradi. Jarayonlar bevosita aniqlanadigan terminal operatsiyalar orqali yoki boshqa oqimli diagrammalar yordamida ifodalanishi mumkin. Shunday qilib, oqimli digrammalar tabaqaviy ko'rinishga ega.

Terminal operatsiyalar relyatsion yondashuvdagidek aniqlanadi. Darvoqe, ma'lumotlar oqimlari diagrammalari ham relyatsion yondashuvda qo'llanishi mumkin.

Shunday qilib, ob'ektli yondashuvda tashqi tavsif bosqichining asosiy mazmunini *ob'ektli modellashtirish* tashkil etadi. Bunda formal spetsifikatsiyalash tillari, shu jumladan grafik tillar ham keng qo'llanadi. Hozirda eng keng qo'llanadigan shunday tillardan biri bu UMB tilidir.

3.3.3. Dasturiy vosita loyihasini tuzish bosqichida ob'ektli yondashuv xususiyatlari

Ob'ektli yondashuvning loyiha tuzish bosqichida ob'ektli modellashtirish jarayoni davom etadi: tashqi tavsif bosqichida qurilgan modellarga, dasturiy tizimlar tavsifining atamalariga aniqliklar kiritiladi, ob'ektlarning dekompozitsiyasi davom ettiriladi.

DV ob'ektli arxitekturasini ishlab chiqish jarayonida foydalanuvchi bevosita ishlamoqchi bo'layotgan ob'ektlarning axborot modellari ajratib olinadi hamda ularning dasturiy spetsifikatsiyasi tugallanadi, shuningdek ularning foydalanuvchi interfeysi aniqlanadi. Bunday ob'ektlarni foydalanish ob'ektlari deb ataymiz. Bunday ob'ektlar sinflari yoki alohida aktif ob'ektlar arxitektura tarmoq tizimlari (tizimchalari)ni hosil qiladi. Ushbu tarmoq tizimlar o'rtasidagi o'zaro aloqalar usullari belgilanadi.

Ob'ektli yondashuvda aktiv ob'ektlardan foydalanilganda arxitekturalarning asosiy keng sinfi sifatida *parallel faoliyat ko'rsatuvchi dasturlar jamoasi* xizmat qiladi, bunda dasturlar vazifasini aynan shu abstrakt ob'ektlar bajaradi. "Mijoz-server" arxitekturasi mana shunday sinf arxitekturasining namunaviy misoli bo'la oladi. Bunday tizimda "server" deb ataluvchi aktiv ob'ektlardan biri "mijoz" deb ataluvchi boshqa aktiv ob'ektlarning so'rovi bo'yicha muayyan dasturiy xizmatlar ko'rsatadi. Bunday so'rov serverga mijozdan keluvchi xabar yordamida uzatiladi, server bajargan so'rov natijasi esa mijozga boshqa xabar vositasida uzatiladi.

Dasturiy tarmoq tizimlar (tizimchalar) tuzilmasining ishlab chiqilishini davom ettirish va ularni dasturlash tillarida kodlash bundan buyon relyatsion yondashuv doirasida davom ettirilishi mumkin. Bunda relyatsion yondashuvga yo'naltirilgan dasturlash tillari qo'llanadi. Foydalanuvchi ushbu tizimchalarning ichki tuzilishini endi "ko'ra olmaydi". Biroq ushbu tarmoq tizimlarning ob'ektli dekompozitsiyasini davom ettirish zarurligini isbotlovchi kuchli dalillar mavjud. Bu tarmoq tizimlarning ob'ektli tuzilmasi, ularning relyatsion yondashuvdagi tuzilmasiga qaraganda, *ishlab chiquvchi* uchun ancha tushunarliroq bo'lishi mumkin.

Bundan tashqari, DV ni ishlab chiqishda ob'ektli dekompozitsiyaning davom ettirilishi va ob'ektli yondashuvning asosiy tushunchalari va usullaridan foydalanish "tabiiy" holdir, chunki bunda ishlab chiqishning barcha jarayoni bir butun yagona tus kasb etadi (kontseptual jihatdan bir butun ko'rinishga ega bo'ladi). Bunda endi boshqa turdagi dasturlash tillarini - ob'ektli yo'naltirilgan dasturlash tillarini qo'llashga to'g'ri keladi. Arxitektura tarmoq tizimlarining bunday dekompozitsiyasida dasturlarda yuzaga keladigan ob'ektlarni *dasturlarni bajarish jarayoni ob'ektlari* deb ataymiz.

3-bo'limga oid nazorat savollari

1. Dasturiy modul nima?
2. Dasturiy modul mustahkamligi deganda nimani tushunasiz?
3. Dasturiy modul birikuvi nima?
4. Tuzulmaviy dasturlash deganda nimani tushunasiz?
5. Dasturiy modulning qadamma-qadam detarlashtiruvchisi nima?
6. Soxta kod nima ekanligini tushuntiring.
7. Dasturiy vositalarning ishlab chiqilishiga ob'ektli yondashuvning mohiyati nima?
8. Dasturiy vositaning tashqi tavsifini ishlab chiqishda ob'ektli yondashuv xususiyatlari nimadan iborat.
9. Dasturiy vosita loyihasini tuzish bosqichida ob'ektli xususiyatlar nimadan iborat?

4. DASTURLAR ISHLAB CHIQISHNING TEST USULLARI

O'quv maqsadi

O'quvchi eng muhim test usullarini biladi va uni misollarning biri orqali tushuntirib bera oladi

Tarkibi

- Yozuv stoli oldidagi test
- Black Box <> White Box
- Test ma'lumotlari
- Modul testi
- Umumiy test

Dastur ishlab chiqilayotganda, ya'ni dasturning oxirgi, tarjima qilsa bo'ladigan va umuman olganda ishlaydigan kodi (Quellcode) mavjud bo'lsa, unda dasturning ushbu kodi istalgan talablarga muvofiq kelish-kelmasligi tekshirib ko'rilgan bo'lishi kerak. Bu **test nazorati** (testlash) deb ataladi.

Testlash sifat kafolatining bir qismidir. Test nazoratining ko'p ko'rinish va usullari mavjud. Bu bobdagi namunalarda eng muhim usullar ko'rsatiladi va izohlanadi.

Testlash - bu havaskor dasturchilar qo'llashni yaxshi ko'radigan sinov emas. Professional dasturlar muntazam ravishda testdan o'tib turishlari, funksionallik tasnif (spesifikatsiya) bilan mosligini hujjat bilan tasdiqlash uchun esa test natijalari hujjatlashtirilishi kerak.

Hech bir test dasturning 100% aniqligini tasdiqlay olmaydi. Har bir test ma'lum shart-sharoitlarda ishlab chiqilgan dasturiy ta'minot istalgan natijani berishnigina tasdiqlashi mumkin. Dasturiy ta'minot sohasida dasturning funksionalligi qanchalik aniqroq bayon qilingan bo'lsa, uning ishlashini shunchalik aniqroq tekshirish mumkin.

Testlar dasturchilar tomonidan ham, dasturlash bilimlariga ega bo'lmagan foydalanuvchilar tomonidan ham o'tkaziladi. Ko'pincha foydalanuvchilar dasturchilarning xayollariga ham kelmagan shunday yo'llar bilan ham dasturdan foydalanadilar.

Shu tarzda xatolar topilishi va tuzatilishi mumkin.

Testlashning turli xil usullarini kengroq tushintirish uchun quyida masalani oydinlashtiradigan oddiy misol tanlaymiz: "Bubblesort"-saralash algoritmi.

4.1. ODDIY MISOL: BUBBLESORT

Ma'lumotlar maydonini tez-tez ko'tariluvchi va pasayuvchi yo'nalish bo'yicha navlab turish zarur (Array). Buning uchun bir nechta algoritmlar mavjud. Bubblesort algoritmi juda tez algoritim hisoblanmaydi, lekin u yetarlicha sodda bo'lgani tufayli endigina ushbu sohaga kiruvchilarga ham tushunarli. Bubblesort deb atalishiga sabab - bunda yirik elementlar sovun ko'pigi kabi ko'piradi, natijada eng yirik elementlar yuqorida va eng kichik elementlar esa pastga cho'kadi.

Vazifa

Bubblesort algoritmida har doim ikkita yonma-yon joylashgan unsurlar bir-biri bilan qiyoslanadi. Bir unsur ikkinchisidan katta, unda har ikkala belgilar o'rni almashtiriladi. So'zimizga kichkina misol.

Dastlabki holat:

19 5 32 8

Birinchi ko'rikdan o'tkazish

19 **5** 32 8 >> 5 19 32
8

5 **19** **32** 8 >> 5 **19** **32**
8

5 19 **32** **8** >> 5 19 **8**
32

Ikkinchi ko'rikdan o'tkazish

5 **19** 8 32 >> **5** **19** 8 32

5 **19** **8** 32 >> 5 **8** **19** 32

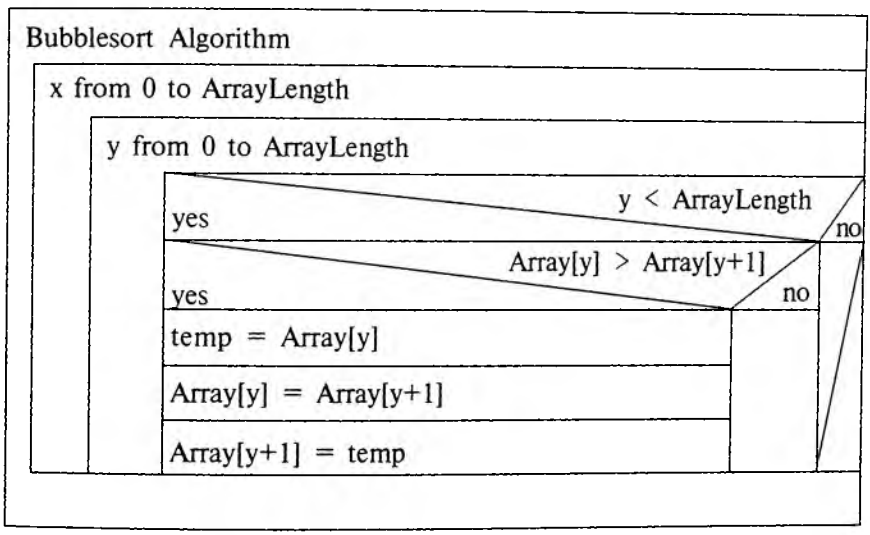
5 8 **19** **32** >> 5 8 **19** **32**

Har bir o'tkazishda ma'lumotlar maydonining alohida elementi (Arrayelement) keyingisi bilan qiyoslanadi. Ma'lumotlar maydonining har bir elementida bitta tashqi va bitta ichki sikl bajariladi. Bu holatda bu yerda to'rtta sikllik bajariladi. To'g'ri, ma'lumotlar maydonining bu namunasida (Array) ikkita ko'rik o'tkazuvidan keyinoq

maqsadga muvofiq to'g'ri ajratilgan. Tegishli nazorat tekshiruvi yordamida endi navlarga ajratishni to'xtatsa ham bo'ladi, biroq "asosiy versiya" uchun birinchi marta bu kerak emas.

Dasturning bajarilishi

Bubblesort-algoritm funktsionalligini yaxshi Nassi/Shneidermann-Diagramm diagrammasi bilan ham ko'rsatish mumkin edi, unga mos kodlash (deyarli) har qanday dasturlash tilida amalga oshishi mumkin.



4.1-rasm. Bubblesort dasturining Nassi/Shneidermann-Diagramm diagrammasi

Element o'zidan keyingi elementdan katta emasligini tekshirishdan navbatdagi tekshirish amalga oshiriladi. Ikkinchi so'roqda ma'lumotlar maydonining y+1 indeksi foydalanishi sababli y indeksi ma'lumotlar maydonining eng yuqori indeksidan (ArrayLength) kichik bo'lishi kerak. y eng katta element, deb faraz qilamiz, unda y+1 ma'lumotlar maydoni chegarasidan tashqaridagi xotira doirasida yotadi, biroq u mumkin bo'lgan qiymatga ega bo'lmaydi. Ba'zi kompilyatorlar (Compiler) bu o'rinda ushbu so'roqdan o'tkazish bo'lmasa darrov bajarilgan bo'ladi (bu yerda u juda ma'qul va foydalidir).

Shunday qilib "agar" (if-so'rov) ichki so'rovda eng muhim element uning ortitagi elementdan kattami, degan savol tekshiriladi. Agar shunday bo'lsa, har ikkala qiymat o'rni almashtiriladi.

Bu namunada ko'tariluvchini ketma-ketlikda saralash o'tkaziladi. Agar saralashni pasayuvchi ketma-ketlikda olib borish zarur bo'lsa, qiyoslashni atigi "nisbatan katta"dan "nisbatan kichikka o'zgartirish kerak bo'ladi".

Dasturiy kod

Nassi/Shneidermann-Diagramm diagrammasidagi dasturiy funktsiya yuqori darajadagi istalgan tilda tuzilishi mumkin. Namuna sifatida bu o'rinda C dasturlash tilidan foydalanish mumkin. Funktsiyaga ma'lumotlar maydonining birinchi elementi uzatilishi mumkin (indeks kabi), biroq u keyin funktsiyada odatdagiday burchak qavslari bilan ajratilishi mumkin. Keyinda ma'lumotlar maydonining yuqorigi indeksi talab qilinadi (ArrayLength).

```
void BubbleSort(int *Array, int ArrayLength)
{
  int x, y;
  int temp;
  for(x=0;x<=ArrayLength;x++)
  {
        for(y=0;y<=ArrayLength;y++)
        {
              if(y < ArrayLength)
              {
                    if(Array[y] > Array[y+1])
                    {
                          temp = Array[y];
                          Array[y] = Array[y+1];
                          Array[y+1] = temp;
                    }
              }
        }
  }
}
```

Endi, dastur bekamu ko'stmi, u istalgan talablarga javob beradimi, degan savolga javob berish uchun uni tekshirib ko'rish kerak.

4.2. YOZUV STOLI OLDIDAGI TEST

Yozuv stoli oldidagi test haqiqiy tafakkur testi bo'lib, kompyutersiz sodir bo'ladi va dasturchilarning o'zlari tomonidan o'tkaziladi.

Shu zayl, dasturchi o'z stoli oldida o'tiradi va uning dasturi bekamu ko'stligi haqida yaxshilab o'ylab ko'radi.

Hujjatlashtirish va yordamchi vositalar sifatida uning ixtiyorida quyidagilar mavjud:

- Talablar tahlili.
- Amalga oshirish rejasi (bu yerda: Nassi/Shneidermann-Diagramm diagrammasi).
- Amalga oshirish (bu yerda: dastur kodini bosmadan chiqarish (listing)).
- Qog'oz va qalam.

Talablar tahlili, bu degani, dastur maqsadi ravshan: sonlarning joylashgan qatori o'suvchi ketma-ketlikda saralanishi kerak.

Dasturchi Nassi/Shneidermann-Diagramms diagrammasi yoki dastur kodi yordamida testlaydigan qator savollar qo'yilishini o'ylab chiqadi va bu bilan dasturning, bekamu ko'stligini ko'rsatadi. Savollar ta'riflanmagan va talablar tahlili esa dasturchi fantaziyasiga bog'liq. "Bubblesort" dasturiga quyidagi savollar qo'yilishi mumkin:

a) qator to'g'ri saralanganmi (7 23 12 128 0 15)?

b) agar qator uzunligi (Arraylength) noto'g'ri berilgan bo'lsa, agar u o'ta katta, agar u o'ta kichik bo'lsa, nima sodir bo'ladi?

c) agar qator salbiy sonlarga ega bo'lsa, nima sodir bo'ladi?

d) agar qator vergulli sonlarga ega bo'lsa, nima sodir bo'ladi?

e) agar qator faqat bitta yagona sondan iborat bo'lsa, nima sodir bo'ladi?

f)agar qator bir nechta bir xil sonlarga ega bo'lsa nima sodir bo'ladi?

g) agar qator son o'rniga o'zgaruvchan miqdorlar yoki ifodalar yoki simvollardan (o'zgaruvchan miqdorlar tipli belgi (Character) yoki satr (String)) iborat bolsa, nima sodir bo'ladi?

Alohida savollar qo'yilishi uchun savollarning shunday yaxshi jadvali yaratilishi kerakki, unda dastur kodida uchraydigan barcha o'zgaruvchi miqdorlar taqsimlangan va dasturning fikran o'tishida ularning ma'nolari o'zgarishi nuqtai nazaridan o'rganilgan bo'ladi:

a) Masalaning qo'yilishi quyidagi tarzda ifodalanishi mumkin:

x	y	Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]	Arraylength
0	0	**7**	**23**	12	128	0	15	6
0	1	7	**23**	**12**	128	0	15	6
0	2	7	12	**23**	**128**	0	15	6
0	3	7	12	23	**0**	**128**	15	6
0	4	7	12	23	0	**15**	**128**	6
0	5	Exit of innermost control structure						6
1	0	**7**	**12**	23	0	15	128	6
1	1	7	**12**	**23**	0	15	128	6
1	2	7	12	**0**	**23**	15	128	6
1	3	7	12	0	**15**	**23**	128	6
1	4	7	12	0	15	**23**	**128**	6
1	5	Exit of innermost control structure						6
2	0	7	12	0	15	23	128	6
....								

4.2-rasm. Yozuv stoli oldidagi test uchun test ma'lumotlari ketma-ketligi

Bunga o'xshash jadvallar boshqa masalalar uchun ham tuzilishi mumkin. Agar masala qo'yilishini tekshirish (tadqiq qilish) natijasida, masalan, o'zgaruvchan miqdorlarning belgilanmagan qiymatlari yoki cheksiz sikl (dasturning o'ta sikllanib ketishi) sababli dasturda xatolar aniqlansa, unda dastur kodi sikldan chiqishining tegishli shartlari bilan to'ldiriladi

Yozuv stoli oldidagi testdan dasturning samaradorligi haqidagi birinchi mulohazani chiqarish mumkin. Jadvallardan kelib chiqadiki, har bir sikldagi saralab bo'lingan sikllar qaytadan ko'rikdan o'tkaziladi. Bu sodda, lekin samarasiz Bublblesort algoritmi bilan tushuntiriladi. Dasturni bajarish vaqti (Arraylenth)?, o'z navbatida vaqtning aniq o'lchovi dasturiy kodlardagi vaqtni kirgizish moslamasi yordamida amalga oshirilishi mumkinligini o'z ichiga oladi.

Dastur samaradorligining (yoki ta'sirchanligini) test nazoratining predmeti hisoblangan uning korrektligi (dasturni bexato ishlashi) bilan hech qanday umumiylikka ega emas, biroq u sifatning eng muhim mezonlaridan biridir (6-bo'limga qarang).

Yozuv stoli oldidagi test dasturchi tomonidan, texnik yordamchi vositalarisiz o'tkazilishi sababli mazkur usul test nazoratining

unchalik samarali usuli hisoblanmaydi. Ushbu sabab dasturning bexatoligini (korrektligini) tasdiqlash uchun faqat birgina yozuv stoli oldidagi testning o'zi kifoya qilmaydi. Albatta, yozuv stoli oldidagi test boshqaruvchi tuzilmalarning (masalan tsikillar, "agar-leolda" shartlari) korrektligi bo'yicha muhim ko'rsatmalar beradi hamda dasturchilarda turli xil shartlar va kirituvchi miqdorlar algoritmlar asosida kodlash (dasturlash) orqali o'tishini tushinishga juda yordam beradi.

Quyida ko'rsatiladigan Black Box va White-Box testlar **nazoratining dinamik usuliga** tegishli, yozuv stoli oldidagi testlash esa **nazoratining statistik usuli** sinfiga oiddir. Dasturlashdagi mavjud kodlarning tahlili (Codereview), nazorat varaqlari (Checklisten) va dasturni tekshirish (verifikatsiya) kabi nazorat usullari ko'proq yozuv stoli oldidagi testning har xil ko'rinishlari va qo'shimchalari bo'lib, ular o'z navbatida nazoratning boshqa statistik usullarini boshqacha ko'rinishidir.

Yozuv stoli oldidagi test tafakkur testi bo'lib, unda dasturchi jarayon diagrammalari va dasturning oxirgi kodining matni (listing) yordamida turli xilda qo'yilgan masalalarni testlaydi. Yozuv stoli oldidagi test turli xil katta miqdorlarda dasturlashtirilgan algoritimlar va boshqaruvchi tuzilmalarni tushinishga yordam beradi. Yozuv stoli oldidagi test samarali emas va dastur to'kisligini tasdiqlash uchun nazoratning yagona usuli sifatida unga yo'l qo'yilmaydi.

4.3. BLACK-BOX-TESTI

Black-Box-testi dasturiy ta'minotning nazorat usulini bildirib, unda testlar testlanayotgan tizimning ichki harakat tamoyillariga oid bilimlarisiz ishlab chiqiladi. U testlar yordamida funktsionalligiga mo'ljallangan sinov bilan chegaralanadi, bu shuni bildiradiki, test variantlarini aniqlash uchun testlanayotgan ob'ekt amalga oshuvidan emas, balki tasnif yoki talablar tahlili(istalgan natija)dan foydalanilishini ko'rsatadi. Bu usulda dasturning aniq ahvoli ko'rib chiqilmaydi, aksincha unga nisbatan "qora qutti" (Black Box) kabi munosabatda bo'linadi. Faqat tashqi ochiq-oydin xususiyatlargina testga qo'shiladi.

Maqsad dasturiy ta'minot tizimi va spetsifikatsiyasi (uning tasnifi yoki boyoni) muvofiqligini tekshirishdir. Rasmiy va norasmiy

tasniflardan kelib chiqib, testlar variantlari ishlab chiqiladi, ular talab qilingan funktsiyalar miqdoriga amal qilinganini qayd qiladi. Testlanishi lozim bo'lgan tizim bunda yaxlit butunlik sifatida qaraladi, testlash natijalarini baholash uchun faqatgina uning tashqi xususiyatlari talab qilinadi.

Testlar variantlarini norasmiy tasnifidan olib chiqish - bu xiyla qimmat turadigan mashg'ulot va tasnif aniqligining darajasiga bog'liq holda va ma'lum sharoitlarda imkoni yo'q. Shuning uchun ham ko'pincha to'liq Black-Box-test matni to'liq White-Box-test matni singari kam rentabellidir.

Hatto muvafaqqiyatli Black-Box-test matni dasturiy ta'minot aniqligining kafolati bo'la olmaydi, chunki dasturiy ta'minot loyihasining ilk fazalarida ishlab chiqilgan tasniflar ancha keyingi qismlar va amalga oshirish qarorlarini ma'lun qila olmaydi.

Black-Box-test matn dasturchining "o'z xatolari" atrofida testlar ishlab chiqishi va shu bilan realizatsiyadagi kamchiliklarni ko'zdan qochirishdan chetlashishga imkon tug'diradi. Dasturiy tizim ishlashining tamoyillarini biladigan dastur myallifi uning spetsifikatsiyasidan tashqarida bo'lgan qandaydir qo'shimcha taxminlar yo'li bilan, bilmasdan turib testlarda ba'zi bir hollarni esdan chiqarishi mumkin yoki ularni dastur spetsifikatsiyasiga nisbatan boshqacha tassavvur etishi mumkin. Black-Box-testining yana bir foydali xossasi shundan iboratki, bunda u spetsifikatsiya'ni to'laligi bo'yicha tekshirish uchun qo'shimcha yordam sifatida yaroqli bo'ladi, chunki to'la bo'lmagan spetsifikatsiya testlarni yaratishda ko'plab savollarning tug'ilishiga olib keladi.

Testlarni tuzuvchilarga tizim ishlashining ichki tamoyillari to'g'risidagi bilimlari (ma'lumotlar) kerak emas. Black-Box-testlarini amaliyotga tadbiq etishda testlarni yaratish uchun maxsus komanda zarur bo'ladi. Ko'plab korxonalarda buning uchun hatto testlash bo'yicha maxsus vakolatga ega bo'lgan bo'limlar mavjud.

Bubblesort dasturining ko'rib chiqilgan namunasida sinovni o'tkazayotgan shaxs ixtiyorida dasturiy ta'minotning kompilyatsiyalangan va silliqlangan versiyasi mavjud.

Dasturga turli xil sonlar qatorining ba'zi bir miqdori beriladi va saralangan natijalar o'rganib chiqiladi. Agar sonlarning ma'lum qatorida xatolar yoki tasnifdan chetga chiqish aniqlansa, ular hujjatlar bilan asoslanadi va dasturchiga oxirigacha ishlash uchun beriladi.

Ayniqsa dasturiy ta'minotning keng ko'lamdagi tizimli holatida test variantlari miqdori tizimli va testlarning to'liq ishlab chiqilgan ketma-ketligi amaliyot uchun katta ahamiyatga ega. Bu miqdorni qisqartirishning quyidagi imkoniyatlari mavjud:

- Eng so'nggi qiymatlar va maxsus qiymatlar testlanadi.
- Barcha funktsiyalar keyinroq birgalikda harakatda bo'luvchi chastotaga muvofiq testlanadi (algoritmlar, qoniqarsiz tasnifli qismlar, tajribasiz dasturchilar guruhi).
- Shunday funktsiyalarning test nazorati ayanchli, ya'ni ularda xatolar, ayniqsa, jiddiy bo'ladi (masalan, keng ko'lamli ma'lumotlarni soxtalashtirish yoki ularning buzilishi).

Black-Box-matn dasturiy ta'minotning nazorat usuli bo'lib, unda testlar test qilinayotgan tizimlarning ichki tamoyillari bilmasdan turib ishlab chiqiladi. Talablar tahlili asosida dastur ishining faqat istalgan tamoyili ko'rib chiqiladi. Test nazoratining ushbu usuli dasturchi sezmay qolishi mumkin bolgan xatolarni topish uchun qo'l keladi. Nazoratning bu usuli yordamida amalga oshirishning ko'rib chiqilmaganligi tufayli ko'rsatilayotgan buzulishlar topilishi mumkin emas.

4.4. WHITE-BOX-TESTI

White-Box-testi (shuningdek **Glass-Box-test**, ya'ni shaffof quti usuli bilan testlash) tushunchasi dasturiy ta'minotning nazorat usulini bildirib, unda testlar testlanayotgan tizim ishlashining ichki tamoyil bilimlari asosida ishlab chiqiladi. Shunday qilib, Black-Box-testidan farqli ravishda ushbu test uchun dasturning boshlang'ich kodiga nazar tashlashga ruxsat berilgan, bu kod tekshiriladi.

White-Box-testi namunasi dasturning o'tishiga tegishli bo'lgan test nazorati bo'lib, unda oldingi o'rinda uning tarkibiy diagrammasi (Struktogramm) yoki ma'lumotlar oqimining o'tish chizmasi (blok-chizma) turadi. Testlash maqsadi bu - dasturning boshlang'ich kodi to'laqonligiga doir test variantlari yetarliligi (ta'minlanganlik)ning ba'zi bir mezonlarini qoniqtirishni ta'kidlashdir. Shu bilan birga nazoratning eng kichik hajmi mavjud:

- Operatorlar birlashuvi: barcha operatorlarning bajarilishi.
- Kantlar birlashuvi: Tarkibiy diagramma tarmoqlanishning barcha mumkin bo'lgan xoshiyalarini yoki ma'lumotlar oqimining o'tish chizmasi (blok chizmalar) to'plamidir.

Agar, hatto, dasturiy ta'minot tizimi **yetarlilik (ta'minlanganlik) mezonlariga** nisbatan testdan muvafaqqiyatli o'tgan bo'lsa-da, bu hol unda xatolar borligini istisno qilmaydi. Buni White-Box-testi tabiati shunaqaligi bilan tushuntirish mumkin. Bunda quyidagi sabablar bo'lishi mumkin:

• White-Box-testi variantlarini dastur tasnifidan emas, balki dasturning o'zidan chiqarib olinadi. Faqat tizim bexatoligi testlanishi mumkin, u talab qilingan semantika shartlariga javob bera olishi kerak.

• Shuningdek, agar dasturning barcha yo'nalishlari (marshrutlari) testlanib bo'lingan bo'lsa ham bu hol dasturning xatosiz ishlashini bildirmaydi. Ma'lumotlarning nazorat oqimi ustunida kantlar (hoshiyalar) bo'lmagan hollar qaralmaydi.

Shuningdek, agar tizimni uning nim tizimlarida sinash talab etilsa, unda buning uchun testlanayotgan tizim ishlashining ichki jarayonlarini bilish kerak bo'ladi. White-Box-testi paydo bo'lgan xatolarni lokallashtirishda (cheklashda), ya'ni xatolarni tug'diruvchi komponentalarni identifikatsiyalash uchun ayniqsa qo'l keladi.

Testlarni ishlab chiqaruvchi testlanayotgan tizim ishlashining ichki jarayonlariga oid bilimga ega bo'lishi kerakligi sababli, White-Box-testlari o'sha komanda tomonidan tuziladi. Ko'pincha testlanishi lozim komponentalar mazkur komanda dasturchilari tomonidan amalga oshiriladi. Qoida bo'yicha White-Box-testlari uchun testlash bo'yicha maxsus bo'limlar tashkil etilmaydi, chunki bu vazifa uchun maxsus qo'yilgan tekshiruvchidan keladigan foyda ko'pincha murakkabligi tufayli tizim mohiyatiga kirish natijasini chippakka chiqaradi.

Bubblesort dasturining ko'rib chiqilgan namunasi uchun x va y indekslari orqali uzulish mezonlariga ega har ikkala ichki (qo'yilgan) sikllar testi muhim ahamiyatga ega. Bu sikllar barcha mumkin bo'lgan test ma'lumotlari bilan tizimli ravishda ko'rikdan o'tkaziladi va shu bilan birga dasturning o'zini tutishi o'rganiladi. Aniqlangan xatolar identifikatsiya qismi ishida va to'g'ridan-to'g'ri dasturiy kodda bartaraf qilinishi mumkin.

White-Box-testi (shuningdek Glass-Box-test) test nazorati usulidir. Unda testlar testlanayotgan dastur harakati ichki tamoyillari bilimlari bilan ishlab chiqiladi. Bevosita dastur kodi testdan o'tkaziladi. Usul bevosita dasturning boshlang'ich kodida xatolar identifikasiyasi uchun yaroqlidir. Bu usul yordamida dasturning talablar taxminiga muvofiq kelishini tekshirish mumkin emas.

4.5. BLACK-BOX-TESTI VA WHITE-BOX-TESTINI QIYOSLASH

Black-Box-testlari White-Box-testlarini almashtirishi mumkin emas, va aksincha. White-Box-testlari spetsifikatsiya bo'yicha xatolarni topish uchun qo'llaniladi, biroq bu testlarda ma'lum komponentalardagi va hatto xatolarni belgilovchi komponentalarning o'zlarining xatolarini identifikatsiyalash uchun ishlatilishi dargumon. Buning uchun White-Box-testlari zarur.

Shuni hisobga olish kerakki, ikkita qismdagi komponentalardagi ikkita xato o'zaro vaqtincha korrekt tuyiladigan umumiy tizimni to'ldirishi mumkin. Buni White-Box-testi yordamida topish osonroq. White-Box-testlarini Black-Box-testlari bilan qiyoslaganda, keyingi usul orqali testlarni o'tkazish ancha qimmatroqdir, chunki ular uchun katta tashkiliy infratuzilma (xususiy komanda)talab qilinadi.

Black-Box-testlarning White-Box-testiga nisbatan afzalliklari:

- Umumiy tizimning White-Box-test yordamidagiga nisbatan yaxshiroq tekshirish (verifikatsiya).
- Tasnifning tekshirilishi.
- "Xatolar atrofida" test nazorati.
- Tegishli tasniflashda semantik tavsiflarning test nazorati.
- Platformaga bog'liq bo'lmagan realizatsiyalarda yaratiladigan testlar ketma - ketligining (qatorlarning) tizimli ravishdagi moyilligi.

Black-Box-testining White-Box-testiga nisbatan kamchiliklari:

- Ancha jiddiy tashkiliy harakatlar (ko'p mehnat talab qilishlik).
- Amalga oshirishda qo'shimcha ravishda kiritilgan vazifalar faqat tasodifan testlanadi.
- Yetarli bo'lmagan testlar ketma - ketligi yaroqsizdir.

White-Box-testining Black-Box-testiga nisbatan afzalliklari:

- Tarkibiy qismlar va ular ishlashining ichki tamoyillarining test sinovi.
- Nisbatan kamroq tashkiliy xarajatlar (ko'p mehnat talab qilishlik).
- Elektron asboblarning yaxshi saqlanganligi tufayli avtomatlashtirish.

White-Box-testining Black-Box-testiga nisbatan kamchiliklari:

- Tasniflash shartlari bajarilishining tekshirilmasligi.
- "Xatolar atrofida" test nazorati.

Black-Box-testlari White-Box-testlarini almashtira olmaydi va

aksincha. Qo'llanishi kerak bo'lgan usullarni tanlash ko'p omillarga bo'g'liq. Avval Black-Box-test yordamida dastur tizimini uning funktsionalligiga tekshirib ko'rish, undan so'ng esa White-Box-test yordamida xatolar identifikatsiyasini o'tkazishning ahamiyati katta bo'ladi.

4.6. TEST MA'LUMOTLARI

Oldingi bo'limlarda bayon qilingan testlardan birini o'tkazish uchun test ma'lumotlari talab qilinadi. Tizim dasturlarining boshqarilishi va natijalarini tekshirish uchun test ma'lumotlari mumkin va mumkin bo'lmagan barcha ma'lumotlarni qamrab olishi kerak.

Test ma'lumotlari quyidagilardan tashkil topadi:

- Standart qiymatlar.
- Ekstremal qiymatlar.
- Xato qiymatlar.

Shu bilan birga test ma'lumotlari testlanuvchi algaritmlarga bog'liq bo'ladi. Quyida berilgan jadval test ma'lumotlarining har xil turlarini ko'rsatadi.

Algoritm	Standart qiymatlar	Ekstremal qiymatlar	Xato qiymatlar
Butun sonlarni kiritish	Butun sonlar	Juda kichik va juda katta sonlar	Harflar (simvollar, belgilar)
Bitta sohaga oid butun sonlarini qo'shish	Soha chegarasidagi sonlar	Soha chegarasidagi sonlar	Vergulli sonlar
Indeks sikllarini bajarish	Indekslar	Boshlang'ich qiymat va indekslar uzilishining qiymatlari	Ma'lum soha doirasidan tashqaridagi indekslar

4.3-rasm. Test ma'lumotlari turlari

Algoritm uchun mumkin bo'lgan barcha hollarni hisobga oladigan test ma'lumotlarini aniqlaydigan hol juda qimmatga tushadi. Murakkab algoritmlar uchun ko'p hollarda buni amalga oshirish-

ning iloji yo'q. Quyidagi oddiy misol bo'lishi mumkin bo'lgan barcha harakatlarning ko'rib chiqilishida test ma'lumotlari qanchalik keng qamrovli bo'lishi mumkinligini oydinlashtirib beradi.

Berilgan uchta a, b, c tomonlarning uzunliklari orqali uchburchak to'la aniqlanadi. Kiritilgan a, b va c lar bo'yicha dastur uchburchakni hisoblashi kerak. Bunda geometriyadan ma'lum quyidagi shartlar yordam beradi:

1) a,b va c lar 0 dan katta son bo'lishi kerak.

2) a+b>c

Keyingi jadval esa a, b va c qiymatlarining test qiymatlari jadvalida qanday kombinatsiyalari bo'lishi mumkinligini ko'rsatib beradi.

a tomon	b tomon	c tomon	testning varianti
3	4	5	to'g'ri burchakli uchburchak
3	3	4	Teng yonli uchburchak
3	3	3	Teng tomonli uchburchak
3	3	0	uchburchak emas
3	4	-2	uchburchak emas
10	5	5	uchburchak emas
2	5	8	uchburchak emas
a	4	5	uchburchak emas
3		5	uchburchak emas
va h.k.(30 dan ortiq test variantlari)			

4.4-rasm. Uchburchak yasashning test ma'lumotlari jadvali

Agar nazariy jihatdan har bir algoritmning tizimli dasturi uchun test ma'lumotlari ketma-ketligini tuzish mumkinligi o'ylab ko'rilsa, unda ayniqsa yirik dasturlarda bu ishni maqbul xarajatlar bilan bajarish mumkin emasligi ayon bo'ladi.

Shuning uchun test ma'lumotlari (qatorlari) tipik qiymatlar, shuningdek, eksterimal va xato qiymatlarini tanlash bilan chegaralanish orqali tekshiriladi. Ishning muhimligi test ma'lumotlari ketma-ketligi (qatori) bundan oldingi bo'limlarda bayon qilib berilgan. Test nazoratining turli xil usullariga muvofiq dasturchilarning hamda

foydalanuvchilarning o'zlari tomonidan ham dasturlashga oid bilimlarisiz ishlab chiqiladi.

Shu bilan birga barcha xatolarni 99 %gacha identifikatsiyalaydigan va cheklaydigan puxta testlar o'tkazish mumkin. Qoida bo'yicha hatto test ma'lumotlarining keng ko'lamiligi (qatorlari) sababli ham, dasturning 100% aniqligini kafolatlab bo'lmaydi.

Test ma'lumotlariga standart qiymatlarga, eksterimal qiymatlarga va xato qiymatlarga ajratiladi. Test ma'lumotlari qatori aytilgan qiymatlar tanlovidan iborat va dasturchilar tomonidan ham, foydalanuvchilar tomonidan ham dasturlashga doir bilimlarsiz ishlab chiqiladi. Qoida bo'yicha hatto keng ko'lamli testlar yordamida ham dasturning 100% aniqligini kafolatlab bo'lmaydi.

4.7. MODUL TESTI VA UMUMIY TEST

Modul testining maqsadi - bu ma'lumotlarning xato turlarini qo'llash asosida xatolar topishdan iborat. Bundan tashqari, modul testida xato qidirish shartlari asosida algoritmlari va xatolarga ishlov berishning o'zida ham yuz beradi. Qisqacha aytganda, modul testi kodlashning tipikxatolarini topishi kerak.

Qoida bo'yicha dasturchi tomonidan dasturlash jarayonida o'tkaziladi. Shu bilan birga u testlanuvchi ma'lumotlariga javob beradi. Bundan tashqari, dasturchi test variantlari haqida o'ylaydi va o'zi testlar o'tkazadi.

Modulli testlash uchun uni boshqa modullar aloqasini simulyatsiya qilish imkoniyatiga ega bo'lishni kafolatlovchi va to'ldirish joyi ko'rsatkichi sifatida turgan Stub deb nomlangan test kerak bo'ladi.

Modul testiga kelganda, gap Black-Box-test va White-Box-testdagi kombinatsiyalar haqida ketadi. Shu bilan birga Black-Box-testi modul bajaradigan interfeysni testlash uchun o'tkaziladi. So'ngra White-Box-test o'tkaziladi. Ushbu test koddagi xatolarni aniqlashidan iborat. Bu xatolar esa faqat bevosita dasturning boshlang'ich kodi ishlab chiqilayotganda va iloji boricha barcha nazorat yo'nalishlari kamida bir marta bajarilgandagina topilishi mumkin.

Umumiy test nihoyat, dasturiy ta'minot tizimining yakuniy testiga olib boradi. Bu yerdagi shart modul testlarining muvafaqqiyatli o'tkazilishidan iborat. Umumiy testda uch xil testlar farq qilinadi:

- Birlashtiruvchi test.
- Tizimli test.

• Qabul qilish testi.

Birlashtiruvchi test barcha qismlarning birgalikdagi ishlashini sinaydi. Modullar interfeyslari boshqa modullar bilan bog'liqlikda testlanishi kerak. Bu o'rinda topiladigan xatolar odatda, dizayn xatolari bo'ladi.

Buning ustiga katta miqdordagi mumkin bo'lgan turli xil usullar mavjud. Birinchidan Top-Down yoki Bottom-Up-testi bor. Shu bilan birga modullar yoki pastdan yuqoriga qarab tayyor dasturga to'planadi va har qadamda testdan o'tkaziladi yoki aksincha. Buning ustiga, masalan, ma'lumotlar bazasi darajasiga "quyi" sifatida, foydalanuvchi interfeysiga esa "yuqori" sifatida qarash mumkin.

Amalga oshirishning keyingi moduli funksional mo'ljalli test nazoratidir. Bunda modullar ularning vazifalariga muvofiq ravishda dastur negiziga birlashadi va testdan o'tkaziladi. Bu usulning afzalligi shundaki, ko'zda tutilmagan vazifalarni texnik bajarish mumkinligi tezroq tekshiriladi. Bundan tashqari, foydalanuvchi loyihaning o'zgarmas taraqqiyotini ko'radi. So'nggi mulohaza, albatta foydalanuvchi testlashga kirishgandagina yoki foydalanuvchi va mijozga kamida test natijalari ko'rsatilgan hollardagina to'g'ridir.

Keyingi imkoniyat esa darajaga mo'ljallangan usuldir. Bunda modullar alohida darajalar bo'yicha joylashadi. Bu holda darajalar alohida darajalar uchun vakil qilingan komanda tomonidan testlanadi. Bu o'rinda, masalan, klassik uch darajali model qo'llanilishi mumkin. Darajalar bo'yicha alohida sinovlarning afzalligi shundaki, testlar yonma-yon o'tkazilishi mumkin. Biroq bunda ko'p test darajalari (Kuchaytiruvchi-shakllantiruvchilar) va Stubs kerak. Bundan tashqari, oxirida darajalar birlashtirilishi undan keyin esa ularning birgalikidagi harakati tekshirilishi kerak bo'ladi.

Agar test paytida jiddiy xatolar topilmasa integratsion test tugallangan hisoblanadi. Bunday xatolarga dasturning inqirozli buzilishi, dasturlarning siklda tushib qolishi, resurslarning avariyali saqlanishi, yoki kritik funktsional xatolar.

Tizimli test qabul qiluvchi test oldidagi umumiy testni o'z ichiga oladi. U buyurtmachining barcha talablari bajarilgan, bajarilmaganligini ko'rsatadi. Tizimning muhim shart-sharoiti shundaki, mahsulot barqaror holatda bo'ladi, dasturlash bo'yicha barcha ishlar bajarilgan va jiddiy kamchiliklar bartaraf qilingan bo'ladi. Tizimli testda dasturning ishchi xususiyatlariga alohida ahamiyat beriladi. Ushbu xatolarning ko'pchiligi dizayn va tasnifning tipik xatolaridir, shu

bilan birga kaskad modelidagi shunday xatolardan biridir, shuningdek, barcha fazalar yuqoridan bajarilishini bildiradiki, bu esa kalendar reja va byudjet bo'yicha katta zarba bo'ladi.

Qabul qiluvchi test to'gridan-to'g'ri mijoz oldida o'tkaziladi, bu shuni bildiradiki, tizim mijoz guvohlanadi (installanadi). Shu bilan birga yangi tizim yoki to'liq, yoki qadamma-qadam installanadi, natijada qaytadan tizimli test o'tkaziladi. Ko'pincha bunda agar yangi tizimda jiddiy xatolar yuzaga kelishi mumkin bo'lgan hollar uchun mijozning eski tizimi bundan keyin yangi tizim bilan bab-baravar ishlay olishni ta'minlanishi kerak. Shunday qilib, jiddiy xato bo'lgan taqdirda bu ishlab chiqarishning turib qolishiga olib kelmaydi. Qabul qiluvchi testning maqsadi - bu yangi tizim ishlab chiqarish sharoitlarida mijozda hech qanday muammosiz ishlashni ko'rsatishdir.

Black-Box-test va White-Box-testdan kombinatsiyalar ko'rinishidagi modulli testlari dasturchi tomonidan dasturlash davomida o'tkaziladi. Umumiy test birlashtiruvchi test, tizimli test, va qabul qiluvchi testlarga bo'linadi va dasturiy ta'minotning tayyor tizimini insallyatsiyalash va uni mijozga uzatishni maqsad qiladi.

4-bo'limga oid nazorat savollari

1. Black box -testi nima? U nima maqsadda qo'llaniladi?
2. White box -testi nima? U nima maqsadda qo'llaniladi?
3. Siz modul testining qaysi birlarini bilasiz, ularning o'ziga xosligi va vazifasi nimadan iborat?
4. Testlanuvchi ma'lumotlar qanday tanlanadi?

5. DASTURIY TA'MINOT MAHSULOTLARINING SIFAT KO'RSATKICHLARI

O'quv maqsadi

O'quvchi sifat belgilari tushunchasini o'zlashtiradi va ularni namunalardan birida tushuntirib bera oladi.

Mazmuni

- Korrektlilik
- Ishonchlilik
- Foydalanuvchi uchun qulaylik
- Xizmat ko'rsatishning qulayligi
- Samaradorlik
- Ko'p marotaba foydalanish imkoniyati
- Uyg'unlik

5.1. SIFAT BELGILARI AXBOROTI

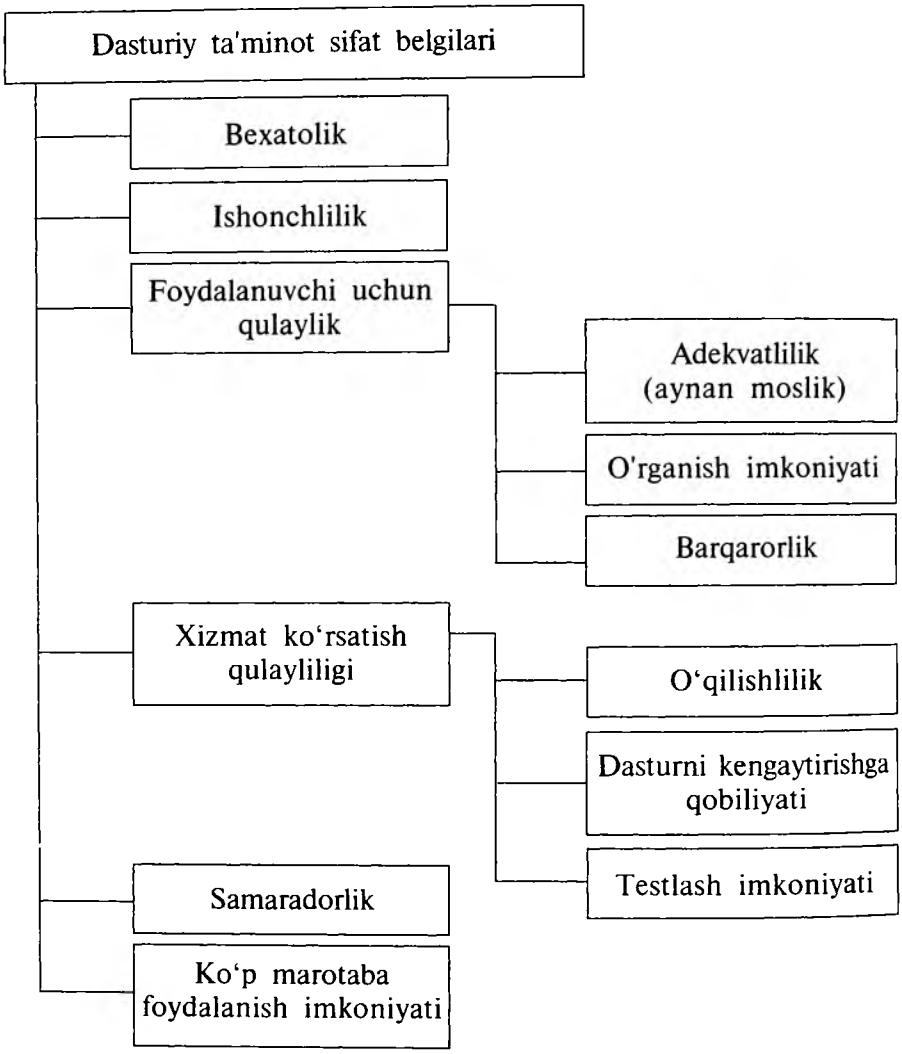
Agar dasturiy mahsulot baholanmoqchi bo'lsa, unda sifat darajasini belgilovchi mezonlar zarur. Biroq dasturiy ta'minot sifati aniq belgilangan tushuncha (atama) bo'lib barqarorlashmagan, shuning uchun yaxshi dastur qanday bo'lishini aniq va qabul qilingan shaklda aytib bo'lmaydi.

Sifatning turli xil belgilari talablar asosida aniqlangan, ularning ma'lum sharoitlardagi ta'riflari talablar tahlilida berilgan. Biroq bu belgilar qanday qilib o'lchanishi mumkin va bundan sifatga qo'yiladigan talablar uchun qanday xulosalar kelib chiqadi?

Keyingi rasmlarda amalda ahamiyatga ega deb qabul qilingan sifat belgilari keltirilgan (5.1-rasm).

5.2. KORREKTLILIK, ISHONCHLILIK VA FOYDALANUVCHI UCHUN QULAYLIK

Bexatolik tushunchasi bu yerda dastur tasnifiga muvofiq to'g'ri ma'nolar berishini bildiradi. To'g'rilikning isboti uchun tegishli test ma'lumotlari hamda Black-Box-testlarining o'tkazilishi to'g'ri keladi. Shu bilan birga asos faqat uning tasnifi (talablar tahlili) bo'ladi.



5.1-rasm. Sifat belgilari tushunchalari

Agar talablar tahlili noaniq bo'lsa yoki u butunlay bo'lmasa, unda dastur garchi foydalanuvchi uning ishlash qobiliyatidan norozi bo'lishiga qaramasdan to'g'ri bo'lishi mumkin.

Ishonchlilik deganda dastur to'g'ri ishlashi kerak bo'lgan ma'lum vaqt oralig'i tushuniladi. Ma'lumotlar kiritilishi va apparat vositalari benuqson ishlaydi degan shart bajarilishi kerak. Ishonchlilik bu kiritishning ma'lum shart-sharoitlarida ma'lumot va kiritish hollarining berilgan miqdori uchun ajratilgan vaqt oralig'i paytida

vazifaning to‘g‘ri bajarilishi ehtimolidir. Ishonchlilikka, shuning dek, apparat vositalari ishida uzulishlar bo‘lgan va foydalanuvchi tomonidan bekorchi kiritishlar qilingan hollarda dastur shunga muvofiq javob berishi (masalan, xatolar haqida axborot uchun berilgan xabarlar yordamida) ham tegishli bo‘ladi. Hech qanday ziyon (masalan, azaliy to‘xtab qolish natijasida ma’lumotlarning yo‘qolishi) keltirmaydigan usul bilan ekspluatatsiya xatolarini va ma’lumotlarini noto‘g‘ri kiritishni to‘xtatib qolish dastlabki shartlarning buzilishi bo‘yicha chidamliligiga bog‘liqligini ham ta’kidlab o‘tish lozim bo‘ladi. Shunday qilib, dasturning ishonchliligi va chidamliligi o‘zaro chambarchas bog‘liq.

Foydalanuvchi uchun qulaylik tushunchasi ostida 6.1-rasmga muvofiq quyidagi 3 belgi yotadi: o‘xshashlik, o‘rganish imkoniyati va dastlabki shartlar buzilishiga chidamlilik.

Shunday qilib, o‘xshashlilik tushunchasi ostida dasturiy ta’minot tizimining tushunarli, sodda va standart operatsiya muhiti ko‘zda tutiladi. Unga yana dasturni ishlab chiqarishning so‘raladigan talablariga javob berish kerakligi va ma’lumotlarning tushunarli shaklda havola qilinishi ham tegishlidir.

Foydalanuvchi uchun tushunarli qo‘llanma va menyu rejimidagi hissiy boshqaruv **o‘rganish imkoniyati** tushunchasi ostida berkinadi.

Bardoshlilik shunday ahvolni belgilaydiki, bunda dastur ekspluatatsiya xatolariga qaramasdan, o‘z ortidan tizimning fojiali ravishda ishlashdan bosh tortishi, elektr tarmog‘i buzilishi va asboblar xatolari kabi, tashqi ta’sirlarda esa ma’lumotlar yo‘qolishi va ma’lumotlarning chidamsizligi ham sodir bo‘ladi.

Bexatolik dasturning tasnifi bilan muvofiqligini bildiradi. Ishonchlilik bu dastur to‘g‘ri ishlaydigan ma’lum vaqt oralig‘idir. Foydalanuvchi uchun qulaylikka o‘xshashlik, o‘rganishga imkoniyat va chidamlilik beradi. Chidamlilik xatolar to‘xtatib qolinishini bildiradi va u ishonchlilik tushunchasi bilan chambarchas bog‘liq.

5.3. XIZMAT KO‘RSATISHNING QULAYLIGI

Dasturiy ta’minotning xizmat ko‘rsatishi va kelgusidagi takomillashuvining mufassal bayoni 6-bo‘limda keltirilgan.

Xizmat ko‘rsatish qulayligi dasturiy ta’minotning kelgusidagi rivojlanishi bilan o‘zaro bog‘liq.

- Xatolar sabablarining cheklanishi.

- Xatolarni tuzatishni amalga oshirish imkoniyati.
- Dastur vazifalarining to'ldirilishi yordamida tavsiflash.

```
void BubbleSort(int *Array, int ArrayLength)
{
    int x, y;
    int temp;
    for(x=0;x<=ArrayLength;x++)
    {
        for(y=0;y<=ArrayLength;y++)
        {
            if(y < ArrayLength)
            {
                if(Array[y] > Array[y+1])
                {
                    temp = Array[y];
                    Array[y] = Array[y+1];
                    Array[y+1] = temp;
                }
            }
        }
    }
}
```

5.2-rasm. Yaxshi tarkiblashgan dasturiy kod

```
void BubbleSort(int *Array, int ArrayLength)
{
int x, y;
int temp;
for(x=0;x<=ArrayLength;x++)
{ for(y=0;y<=ArrayLength;y++)
{ if(y < ArrayLength)
{ if(Array[y] > Array[y+1])
{ temp = Array[y];
Array[y] = Array[y+1];
Array[y+1] = temp;
}}}}}
```

5.3-rasm. Yomon tarkiblashgan dasturiy kod

Xizmat ko'rsatish qulayliklari ham 3 kichik belgilarga: tushunarlilik, dasturning kehgayishga qobiliyati va testlash imkoniyatlariga bo'linadi.

Dasturning **o'qilishliligi deganda** dasturiy kodning tarkiblashtirilgan dasturlash shartlariga muvofiq ishlab chiqilganligi tushiniladi (6.2 va 6.3-rasmlarga qarang). Hujjatlar yuksak talablarga javob berishi kerak va dasturni nazorat qilish maqsadida asboblar vositasida tahlil qilinishi mumkin.

Xizmat ko'rsatish qulayligiga shu xoski, dasturning boshlang'ich kodi yaxlit ishlab chiqilgan va o'tish operatoriga (GOTO) ega emas. Har bir dasturiy satr faqat bitta operatorga ega bo'lishi kerak.

Namunali dasturchi eng avvalo dasturchi bilan gaplashmasdan, har doim boshqa dasturchini tushunadi va uning dasturiy kodini ishlab chiqishi mumkin degan tasavvur bilan dasturlaydi. Bu shuni bildiradiki dasturiy kodda dasturchi fikri tarziga, shuningdek o'zgaruvchan miqdorlar tasnifi bevosita dasturning boshlang'ich kodida sharhlar satrida hujjatlashtirilgan bo'ladi.

Hamma gap aynan sharhlarga iltifotsiz munosabatda bo'luvchi dasturchilarning tabiatidadir. Bu ruhiy tomondan tushuntiriladi, chunki dasturchi o'z g'oyalri o'z algoritmlari va o'z tuzilmalarini "o'z tasavvurlari va qarashlari" kabi ko'rishni yoqtiradi va ularni biron kimsa bilan baholashni sira ham xohlamaydi. Biroq tajriba shuni ko'rsatidiki dasturchi, shuningdek, yana tez unutadi ham. Agar bir necha oy o'tgach o'z dasturiy bloklaridan birini o'zgartirishi yoki kengaytirishi kerak bo'lsa, u dasturlash paytida eng avval tasavvur qilganlarini qayta tiklash uchun anchagina ko'p vaqt sarf qiladi. Bu shuni ko'rsatadiki, tarkiblangan dasturlash va dasturning boshlang'ich kodidagi puxta hujjatlar nafaqat sifatning muhim belgisi, balki u dasturchining shaxsiy manfaatlariga ham mosdir.

Dasturning kengayish qobiliyati deganda shuni tushunish kerakki, dasturiy ta'minotdagi o'zgarishlar oddiy maqsadga qaratilgan holda va imkoni boricha noma'qul qo'shimcha ta'sirlarsiz o'tkazilishi mumkin. Dasturning kengayish qobiliyatiga komplekslilik (murakkablik) sezilarli darajada ta'sir ko'rsatadi. Dasturning, modulning yoki sinfning o'lchami kattalashib borishi bilan o'zgarishlar borgan sari murakkab bo'la boradi.

"Dasturiy ta'minotning katta tizimi ko'pincha ulkan, lekin nozik konstruktsiyaga o'xshaydi, undan bitta g'ishtni ham butun san'at asarini yakson qilmasdan olib bo'lmaydi".

Bundan dasturning kengayish qobiliyatini yaxshilash uchun konstruksiya qurishning faqat ikkita tamoyilini keltirib chiqarish mumkin:

- Oddiy arxitekturalar yaratish.
- Sodda tuzilmada dasturlar murakkab tuzilmalardagiga nisbatan aralashuvni yengilroq amalga oshiradi.
- Modullarning keng yechimi.
- Modullarning boshqa modullar bilan birlashuvi murakkablikni oshiradi.
- Biron-bir modulning o'zgarishi boshqa modullarga moslashishni talab qiladi. Bu xatolarga olib kelishi mumkin va bundan imkoni boricha modulning avtonom tuzilishi yordamida qochish kerak.
- Kapsulyatsiya" va "polimorfizm" konseptsiyasi tufayli ob'ektga mo'ljallangan dasturlash dasturning kengayishiga yuqori qobiliyatni ta'minlash uchun eng yaxshi shart-sharoitlarni yaratib beradi. Kuchli modullashni (protseduraga oid dasturlash) har doim yaxshi dasturiy kodining sifat belgisidir.

Testlashning yaxshi imkoniyati dasturni bajarishning aniqligini ko'zda tutadi. Dasturni aniq bajarish tufayli xatolarni cheklash osonroq. Testlash imkoniyati asosan moduliligi va tarkiblash darajasiga bog'liq bo'ladi (masalan, sinflar va protseduralarni tarkiblash).

Modulli tuzilmalar imkoniyatiga qarab mustaqil ishlovchi qismlar bilan alohida qismlarning xatosiz ish ko'lami bo'yicha tekshiruvini yengillashtiradi. Ob'ektga mo'ljallangan tizimlar kapsulyatsiyalar va ularning yuksak modulli tuzilmalari sababli testlash imkoniyatini ta'minlash uchun ayniqsa qo'l keladi.

Tushunarlilik, dasturning kengayishiga qodirlik va testlash imkoniyati foydalanuvchiga qulay dasturiy kodning asosiy sifat belgialaridir. Dasturchi tuzilmalashtirlgan dasturlash qoidalariga rioya qilishi, izoh satrlarida dasturning o'z boshlang'ich kodini yaxshi hujjatlashtirishi va yuqori modulli tuzilimali o'z dasturini ishlab chiqishi kerak.

5.4. SAMARADORLIK VA TAKROR FOYDALANISH IMKONIYATI

Samaradorlik tushunchasi ostida barcha resurslardan eng yaxshi foydalanishni o'z ichiga oluvchi dastur tushuniladi. Resurslar tushunchasi xotira sohasiga, aloqalar kanallariga va periferiy jihozlarga taa'lluqlidir.

Samaradorlik tezlik bo'yicha va tizimli dasturning xotira jabhasidan foydalanish bo'yicha o'lchanadi. Tezlik, jumladan, samarali algoritmlardan foydalanishga bog'liq. Dasturiy ta'minotning testlanishida tezlik vaqt datchigini bevosita dastur kodiga kiritish va o'lchangan vaqtni berish yo'li bilan o'lchanadi. Vaqt datchiklari, ayniqsa, sikllarni ko'rikdan o'tkazishda, tarmoqlanishda va protseduralarga rekursiv munosabatlarda ma'noga ega. Ko'pincha xotira sohasini ko'p iste'mol qilish ortiqcha jadvalli tuzilmalar yoki ortiqcha o'lchamli o'zgaruvchan miqdorlar testlariga ega ma'lumotlar bazalarida sodir bo'ladi. Shuning uchun ma'lumotlar bazalarini ishlab chiqishda meyorlashtirish juda muhim.

Hozirgi paytda chaqqon protsessorlar va eslab qoluvchi qurilmalar (RAM, qattiq disk) juda arzon bo'lib qolganligi sababli, dasturlashda samaradorlikka borgan sari kamroq ahamiyat berilmoqda. Shunday bo'lsa-da, dasturlashning yaxshi uslubi samarali algoritmlar ishlab chiqishda va ma'lumotlar bazalarining meyorlashtirilgan tuzilmalarida foydalanish bilan ajralib turadi.

Tizimli dasturning **takror foydalanish imkoniyati dasturiy bloklarning meyorlashuvi bilan** chambarchas bog'liqdir.

Standartlashtirilgan qismlarning afzallik tomoni shuki, ular arzonroq, ishonchliroq va qoida bo'yicha agar ularning nuqsoni bo'lsa, qismlarni ta'mirlash yoki almashtirish osonroq bo'ladi. Xuddi shu narsa dasturiy ta'minot meyorlashtirilgan (standartlashtirilgan) qismlaridan takror foydalanishga ham xoski, bu dasturiy ta'minot mahsuldorligi va sifatini oshirishning eng samarali imkoniyatlaridan biridir.

Dasturiy ta'minot ishlab chiqishda takror foydalanish turli shakllarda yuz beradi.

- Dasturning mavjud dastlabki kodlarini nusxalash yordamida takror foydalanish.
- Takror foydalanishning ushbu eng oddiy shakli uchun dasturning tuzilishi va mantig'ini aniq bilish zarur. Bu harakat usuli katta xarajatlarini talab qiladi va ta'mirga yomon yaroqlilikka olib keladi.
- Kutubxonalardan foydalanish.
- Takror foydalanish bu yerda ham eng avvolo dasturning boshlang'ich kodidan qayta tiklab foydalanishga asoslanadi. To'g'ri mavhumlashtirish yuz beradi, protsedura kutubhonalaridagi proetsedura mavhumlashtirishi guruhlarning ob'ektga mo'ljallangan kutubxonalarda tasniflanadi. Dasturning boshlang'ich kodi ichki mantiqning aniq bilimlarisiz takror foydalanish uchun yaroqlidir. Kutubxonadan

foydalanishda faqat uning mazmuni va masalalar obzoriga ega bo'lish, shuningdek, ma'lumotlar uzatish ko'rsatkichlarini bilish yetarli.

Dasturiy ta'minot ishlab chiqarishdagi samaradorlik bu ma'nbalardan eng yaxshi foydalanishdir. Standart dasturiy bloklardan foydalanish (protseduralar kutubxonasi) tizimli dasturdan ko'p marotaba foydalanish imkoniyatiga ko'maklashadi.

5.5. DASTURIY TA'MINOT SIFATI VA XARAJATLARI

Dasturiy ta'minotning yaxshi sifati ishlab chiqishga ortiqroq xarajatlarni talab qiladi. Keng iste'mol tovarlarinig boshqa sohalarida

Belgi \ ga ta'sir qiladi	Bexatolik	Ishonchlilik	O'xshahslik	O'rganish imkoniyati	Barqarorlilik	Tushunarlilik	Dasturni o'zgartirish va kengaytirishga layoqat	Testlash imkoniyati	Samaradorlik	Ko'p marotaba foydalanish imkoniyati	**Ishlab chiqish vaqti**	**Hayot sikli**	Tajriba konstruktorlik ishlariga xarajatlar	Ishlab chiqarish xarajatlari	Texnik xizmat ko'rsatishga xarajatlar	Ko'chirishga xarajatlar
Bexatolik		+	0	0	+	0	0	0	0	0	-	+	-	+	+	0
Ishonchlilik	0		0	0	+	0	0	0	-	0	-	+	-	+	+	0
O'xshahslik	0	0		+	0	0	0	0	+	-	-	0	-	+	+	-
O'rganish imkoniyati	0	0	0		0	0	0	0	-	0	-	0	-	+	0	0
Barqarorlilik	0	+	+	0		0	0	+	-	0	-	+	-	+	+	0
O'qilishlilik (tushunarlilik)	+	+	0	0	+		+	+	-	+	+	+	+	0	+	+
Dasturni o'zgartirish va kengaytirishga layoqat	+	+	0	0	+	0		+	-	+	-	+	+	0	+	+
Testlash imkoniyati	+	+	0	0	+	0	+		-	+	+	+	+	0	+	+
Samaradorlik	-	-	+	-	-	-	-	-		-	-	+	-	+	-	-
Ko'p marotaba foydalanish imkoniyati	0	0	-	0	0	0	+	0	-		-	+	-	-	+	+

(+ ijobiy ta'sir, − salbiy ta'sir, 0 hech qanday ta'sir yo'q).

5.4-rasm. Sifat belgilari, vaqt va xarajatlarning o'zaro bog'lanishi

bo'lganidek dasturiy ta'minot uchun ham sifat uzoq muddatga o'zini oqlaydi va shunchaki "arzon", lekin sifati pastligi nisbatan foydaliroq.

Dasturiy ta'minot ishlab chiqarish vaqtida sifatga, investitsiyaga, qoida bo'yicha, texnik xizmat ko'rsatish va dasturiy mahsulot ekspluatatsiyasiga, xarajatlar va vaqt tejalishiga sabab bo'ladi.

Sifat belgilarining bir-biri bilan o'zaro ta'siri, shuningdek, ishlab chiqarish vaqtiga (muddatiga) va ishlab chiqarish uchun xarajatlarga ta'sirini ko'rsatib beradi.

5-bo'limga oid nazorat savollari

1. Dasturlarning xizmat ko'rsatish qulayligi belgilari qanday aniqlanadi?
2. Dasturning korrektligi, ishonchliligi va turg'unligi nima?
3. Dasturning samaradorligini nima aniqlaydi?
4. Dasturning qayta ishlatilish imkoniyatini ta'minlash uchun nima kerak bo'ladi?

6. DASTURNING TA'MINOT XIZMATI VA UNING KELGUSIDAGI RIVOJLANISHI

O'quv maqsadi

O'quvchi xizmat ko'rsatishning tahlili va borishini, shuningdek, hujjatlashtirishni yaxshi o'zlashtiradi.

Mazmuni

- Xizmat ko'rsatish bo'yicha (tahrir qiluvchi, takomillashtiruvchi, moslashtiruvchi) harakatlar.
- ABC- tahlil va Portfolio-tahlil.
- Xizmat ko'rsatishning borishi.
- Hisobot berish/xizmat ko'rsatish bo'yicha hujjatlar.
- Texnik xizmat ko'rsatish haqidagi shartnomalar (bitimlar) turlari, rasmiylashtirilishi.

Dasturiy ta'minot xizmati dasturiy ta'minotni ishchi, texnik, dolzarb holatda tutishi va uni iste'molchining o'zgarib turadigan talablariga moslanuvchan holda saqlashi kerak. Bunda xizmat ko'rsatishning quyidagi uchta vazifasi yuzaga keladi:

- Xatolarni tuzatish.
- Dastlabki talablarning o'zgarishi.
- Funktsionallik va mahsuldorlikning yaxshilanishi.

Ko'pincha xizmat ko'rsatishni faqat xatolarni tuzatish, deb tushunishadi. Biroq xizmat ko'rsatish - bu ancha ko'proq narsani bildiradi, chunki dasturiy ta'minot ishlab chiqarishning har qanday mahsuloti yoki vositasi sifatida eskiradi va dasturiy ta'minot tizimini texnikaning darajasida egallashi uchun vaqt va xarajatlar talab qilinadi.

Xizmat ko'rsatish uchun manbalar eng boshdan rejalashtirilishi zarur. Ular agar dasturiy ta'minot o'z tashkilotida ishlab chiqilgan bo'lsa vaqt va inson kuchini, yoki dasturiy ta'minot maxsus firma buyurtmasi bo'yicha ishlab chiqilgan bo'lsa - tegishli byudjetni o'z ichiga oladi. Keyingi holatda dasturiy ta'minot xizmat ko'rsatishi haqida shartnomalar tuzish tavsiya qilinadi, shu zaylda xarajatlarni ham, mahsuldorlikni ham rejalashtirish mumkin.

6.1. XIZMAT KO'RSATISH BO'YICHA HARAKATLAR

Xizmat ko'rsatish bo'yicha quyidagi harakatlarni farqlash kerak:

- Tahrirlovchi xizmat.
- Takomollashtiruvchi xizmat.
- Moslashtirish xizmati.

Tahrirlovchi xizmat xatolarini topish va ularni bartaraf qilish uchun zarur.

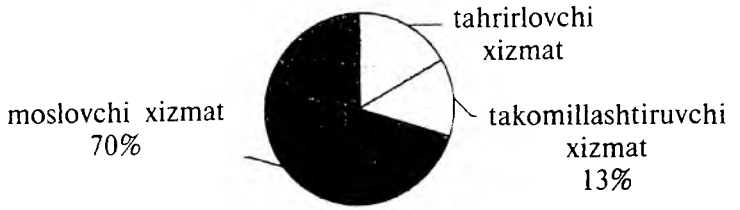
- Ishlov xatolari: dasturning kutilmaganda tugashi (avariyali ishdan bosh tortish), xatoga ega dasturdan ma'lumotlar chiqarish, kiritilayotgan ma'lumotlarni tekshirishning qatnashmayotgan turi yoki sohasi.
- Dasturni bajarishdagi to'siqlar: javob qaytarish vaqtining uzoqligi va mahsuldorlikning yetarli emasligi (o'tkazishga layoqat);
- Ekspluatatsiyaga kiritishdagi kamchiliklar: standartning shikastlanishi, dasturning bardoshsiz yoki noto'liq loyihasi.

Dasturiy ta'minot ekspluatatsiyasi davomida tahrirlovchi xizmatidan foydalanish har xil hajmda bo'lishi zarur. Ekspluatatsiya'ning ma'lum davridan keyin dasturning bajarilish variantini muvofiqlashtirish uchun tahrirlovchi xizmat takomillashtiruvchi xizmat bilan almashtiriladi.

Takomillashtiruvchi xizmat mahsuldorlikni oshirish, dasturning xossalarini o'zgartirish yoki yangisini qo'shish, shuningdek, dasturning bo'lg'usi ta'mirga yaroqliligini yaxshilash vazifalariga ega.

Moslashtiruvchi xizmatdan dasturiy ta'minotni apparat vositalari va yoki dasturiy muhitning o'zgarishlariga moslashtirish uchun foydalaniladi. Bunda so'z:

- Ma'lumotlar muhitining o'zgarishi: masalan ma'lumot tashuvchining o'zgarishi yoki izchil fayllar va indeksli faylning ma'lumotlar bazasini boshqarish tizimiga (DBMS) ko'chirilishi kabi o'zgarishlar va;



6.1-rasm. Xizmat bo'yicha vazifalarning taqsimlanishi

• Ishlov berish muhitining o'zgarishi: masalan, apparat vositalarining yangi muhitiga yoki yangi operatsion tizimiga ko'chirish singari o'zgarishlar haqida boradi.

Tadqiqotlar hayratlanarli natijalar shuni ko'rsatdi, xarajatlarning ko'pchiligi tahrirlovchi xizmat (masalan, xatolar qidiruvchi) tufayli emas, balki moslashtiruvchi xizmat (masalan yangi operatsiya muhitiga moslanish) tufayli kelib chiqar ekan.

Xizmat ko'rsatish bo'yicha harakatlarning uchta turi ajraladi: tahrirlovchi xizmat xatolarni topadi va yo'qotadi, takomillashtiruvchi xizmat mahsuldorlikni oshiradi va moslashtiruvchi xizmat yangi operatsion tizimi singari yangi muhitga moslashtiradi.

6.2. XIZMAT KO'RSATISHGA LOYIQ DASTURNI ANIQLASH

6-bo'limda tavsiflangan dasturiy ta'minot sifat belgilari xizmat ko'rsatishga loyiq dasturlarni aniqlab olish uchun eng muhim indikatorlardir. Biroq bular yetarli emas va shunday qilib, dasturga xizmat ko'rsatishi kerakmi yoki yo'qligini aniqlab olish uchun ikkita usul ajratiladi:

• Portfolio-tahlil.
• ABC-tahlil.

6.2.1. Portfolio-tahlil

Portfolio-tahlilda tashkilotning foydalanayotgan dasturlari tahlil qilinadi va Portfolio-diagrammasiga kiritilgan ikkita mezon bo'yicha baholanadi (6.4-rasmga qarang):

• Texnik sifat.
• Texnik-iqtisodiy sifat.

Har ikkala mezon ikkala tushunchani aniqroq tavsifalashga yordam beradigan yana boshqa mezonlar bilan aniqlashtirishga muhtojdir.

Texnik sifatga quyidagilar tegishli:

• Dasturiy kodning modul tuzilishi (o'qishlilik, o'zgarishlar kiritishning yangiligi).
• Boshlang'ich kodning tarkiblashtirilgan tuzilishi.
• Testlashtirish imkoniyati.
• To'lalik.
• Ko'p marotaba foydalanish imkoniyati.

- Bexatolik.

Texnik iqtisodiy sifatga quyidagilar kiradi:

- Tashkilot uchun muhimlik.
- O'xshashlik.
- O'rganish imkoniyati.
- Chidamlilik.

Sifatning aytilgan belgilari o'zgarmas emas va turli tashkilotlarda turlicha bo'lishi mumkin.

Buni keyingi misolda tushuntirib berish mumkin:

Tashkilot ekspluatatsiyaga 10 ta tizimli dastur kiritadi. Har bir tizimli dastur xizmatining qanday ustunligi bo'yicha zarurligini aniqlash kerak.

Birinchidan har bir dastur bo'yicha uning texnik sifati ahamiyati va texnik iqtisodiy sifati uchun ahamiyati aniqlanadi. Buni quyidagi rasmda ko'rsatilgan jadvalda hisoblab chiqiladigan 10 ballik tizim (10-juda yaxshi sifat, 1-juda yomon sifat) bilan aniqlash mumkin.

Dasturiy ta'minot A			
	Baho	**Omil**	**Natija**
Texnik sifatlar			
Modulli tuzilish (modulyarlash)	5	0,1	0,5
Tarkiblash	5	0,1	0,5
Testlash imkoniyati	6	0,1	0,6
To'lalik	7	0,1	0,7
Ko'p marotaba foydalanish imkoniyati	7	0,2	1,4
Bexatolik	7	0,4	2,8
Texnik sifat miqdori			**6,5**
Texnik - iqtisodiy sifat			
Muhimlik (1=juda muhim, 10=muhim emas)	3	0,5	1,5
O'xshashlik (adekvatlilik)	8	0,1	0,8
O'rganish imkoniyati	8	0,1	0,8
Chidamlilik	8	0,3	2,4
Texnik iqtisodiy sifat miqdori			**5,5**

6.2-rasm. Tizimli dastur xizmati muhimligini baholash jadvali

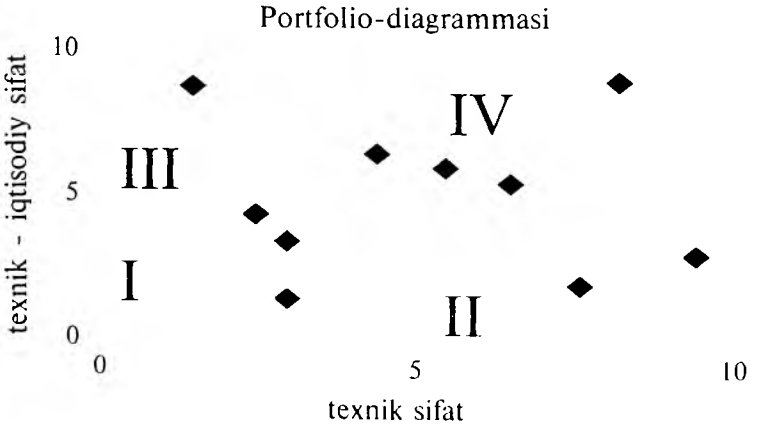
Sifat belgisining muhimligi omil bilan tartibga solinadi. Bitta baholovchi guruhdagi barcha omillar miqdori 1.0 ga teng bo'lishi kerak. Shu zaylda tahlil qilingan 10 tizimli dasturning har biriga bu bilan, oxirida, tahlilnoma jadvalida ifodalangan texnik va texnik-iqtisodiy qiymat beriladi:

Bundan keyin shu zaylda olingan barcha qiymatlar Portfolio-diagrammasiga kiritiladi.

Kvadratlardagi raqamlar tizimli dasturlar xizmatining ustunligini ko'rsatadi.

	Texnik sifat	Texnik-iqtisodiy sifat
Dastur 1	6,5	5,5
Dastur 2	8,2	9,0
Dastur 3	9,4	3,0
Dastur 4	3,0	3,5
Dastur 5	4,4	6,5
Dastur 6	2,5	4,4
Dastur 7	7,6	2,0
Dastur 8	5,5	6,0
Dastur 9	1,5	8,8
Dastur 10	3,0	1,5

6.3-rasm. Portfolio-dasturini baholash jadvali



6.4-rasm. Portfolio-diagrammasi

Dastlab past texnik - iqtisodiy va texnik sifatli dasturlar xizmat ko'rsatadi (kvadarat - 1). Ustunlik bo'yicha ikkinchi bo'lib past texnik iqtisodiy sifatga ega yoki yuqori ahamiyatli dasturlar xizmat qilishi kerak (kvadrat II). Keyin III kvadratda joylashgan dasturlar keladi. IV kvadratdagi dasturlar texnik ham, texnik iqtisodiy nuqtai nazardan ham joyida, shuning uchun ular oxirgi xizmat ko'rsatish ustunligiga ega.

Portfolio-tahlilda tashkilotning barcha dasturlari ularning texnik va texnik - iqtisodiy sifati bo'yicha tahlil qilinadi. Ushbu dasturlar xizmat ko'rsatishning ustunligi Portfolio-diagrammadan kelib chiqadi.

6.2.2. ABC-tahlil

ABC-tahlil xizmat qilish lozim bo'lgan dasturlarni aniqlashning sodda shakli. Tashkilotda foydalanilayotgan dasturlarning har biri quyidagi uch mezon bo'yicha tekshiriladi:

- Dastur tashkilot uchun qanchalik muhim?
- Keyingi yillar ichida dasturni qanchalik ko'p o'zgartirishdi?
- Dasturdan qancha xodim foydalanadi?

Dastur ishlab chiqarish jarayoni va tashkilot mahsuloti uchun nima bilan ko'proq zarur bo'lsa, u shunchalik muhim. Masalan, mashinasozlikda SDB (sonli dasturiy boshqaruv) stanoklarni dasturiy boshqaruv juda muhim. Agar u ishdan chiqsa, ishlab chiqarish to'xtab qoladi.

Dasturning muhimligi bilan yana unnig so'nggi yillar ichida qanchalik ko'p o'zgargani va yangilanganligiga bog'liq. Muhim dasturlar tez-tez o'zgarib turadi.

Dasturdan foydalanuvchi xodimlar soni ham shuingdek, muhimlik indekatoridir. Ko'p sonli xodimlar foydalanadigan nuqsonli dasturlar vaqt va resurslarni talab etadi, shu sababli yuqoru ustunlik bilan xizmat qilishlari kerak.

Ushbu tahlildan foydalanilayotgan dasturlarning ABC-tasnifi kelib chiqadi:

A - ishlab chiqarish jarayoni yoki mahsulot uchun zarur dasturlar

B - boshqa muhim dasturlar

C - ahamiyati yo'q dasturlar

Bu ABC izchillik bilan foydalanilayotgan dasturning (texnik) xizmat ko'rsatish ustunligini belgilaydi.

6.3. XIZMAT KO'RSATISHNING BORISHI

Dasturiy ta'minot xizmat ko'rsatishda ko'pincha kuzatish mumkinki, bunda texnik xizmat ko'rsatish talab qilinishida darrov "bir amallab" nimanidir dasturlashga urinishadi. Bu harakat ta'moyili samarali emas va masalan, har qo'shimcha foydalarini e'tiborga olmasa, xizmat ko'rsatishga tushgan buyurtmaga bo'lganiga qadar undan ham yomonroq ahvolga tushib qolishi mumkin.

Xizmat ko'rsatish o'z-o'zidan xizmat ko'rsatishni o'tkazishning ma'lum tasnifiga amal qilishi kerak. U xizmat boshlangunga qadar tasniflangan va rejalashtirilgan bo'lishi kerak. Shunda ma'lum tasnifga rioya qilinganligi tufayli xizmatni yaxshilab hujjatlashtirish mumkin (6.4-bobga qarang).

Xizmat ko'rsatishning borishida quyidagi uslubiyat tamoyili taklif qilinadi:

Dasturiy ta'minotni tushunish	→	Ko'p hollarda xizmat ko'rsatish lozim bo'lgan dasturiy ta'minot xatolar bilan yoki chala holda hujjatlar bilan asoslangan (hujjatlashtirilgan) bo'ladi. Shuning uchun dasturiy ta'minotga xizmat ko'rsatishga kirishishdan oldin eng avvalo o'zining tuzilishi bo'yicha, o'zining ichki mantig'iga, o'z algoritmlariga ko'ra va tizim qismlarining (interfeyslar) o'zaro aloqasi tushunarli bo'lishi kerak. Bu jarayon, aslini olganda, ba'zan xizmat ko'rsatishga qancha vaqt ketsa shuncha vaqt oladi.
Muammolar verifikatsiyasi	→	Ishning ahvolini aniqlash.
Muammo-larning ajratilishi	→	Muammolarni cheklash, bu demak, vazifalarni ro'yxatlash va tizimning o'zgartirish kerak bo'lgan bloklarini tenglashtirish.
Muammolar ishlab chiqarish (faqat tahrirlobchi xizmatda)	→	Dasturchi muammoni uzil-kesil payqashi va bu bilan o'zgartirishlarni to'g'ri amalga oshirishning isbotiga ega bo'lishi uchun xatolar manbai tiklanishi kerak.
Muammolarni yechish	→	O'zgarishlarni amalga oshirish xizmat ko'rsatish jarayonining oxirgi boshqichida amalga oshiriladi. Shuningdek hujjatlarni o'zgartirish va versiyalarni boshqarishni o'tkazish kerak.

6.5-rasm. Xizmat ko'rsatish bosqichlari

6.5-rasmda ifodalangan harakat ta'moyilini xizmat ko'rsatishning barcha turlari (tahrir qiluvchi, takomillashtiruvchi, moslashtiruvchi) uchun qo'llash lozim. Ma'lumotlarning chiqarilishi faqat tahrirlovchi xizmatida amalga oshiriladi va takomillashtiruvchi va moslashtiruvchi xizmatlarda o'tkazilmaydi.

Odatda esa, aynan yirik tashkilotlarda bitta emas, balki katta miqdordagi xizmat ko'rsatish hollari yuzaga keladiki, ularni albatta bir vaqtning o'zida o'tkazib bo'lmaydi. Shuning uchun xizmatni aniqlab olgandan keyin bu hollarni tasniflash maqsadga muvofiq. Qoida bo'yicha, bu masalalar tizim ma'muri vazifasiga kiradi.

Texnik xizmat ko'rsatish masalalarini tasniflashda shuni hisobga olish kerakki, 1-ustun bo'yicha xizmat ko'rsatish chiqimlari oldindan ko'rib chiqilmasdan amalga oshiriladi. Aynan tashkilot uchun jiddiy xatolar uni yo'qotish uchun xarajatlar ikkinchi darajali rol o'ynaydi va nima bo'lganda ham bu xarajatlarni ko'tarish kerak.

Ustunlik	Belgilar	Choralar
1	Tez tahrir qilish va moslashtirish, ishlab chiqarish jarayonidagi to'siqlarga yoki ma'lumotni to'laligicha ololmaslikka olib keladigan xatolar	**Qisqa muddatli rejalashtirish va amalga oshirish** Xizmatlarni rejalashtirish, zarurat tug'ilganda quvvat yaratish uchun xizmat bo'yicha boshqa harakatlarni to'xtashish
2	Katta samaradorlik va unumdorlokka olib boruvchi tahrirlash, moslashtirish, va yaxshilash	**O'rta muddatli rejalashtirish va amalga oshirish** Talablar tahlili, chiqimlar va samaradorlik tahlili, mavjud moliyaviy va shaxsiy resurslar asosida xizmat ko'rsatishni rejalashtirish
3	Uzoq muddatga zarur bo'ladigan tahrirlash, moslashtirish va yaxshilash	**Uzoq muddatli rejalashtirish va amalga oshirish** Xizmat arxivida saqlash, xizmat arxividagi xizmat bo'yicha masalalarni vaqti-vaqti bilan tekshirish va bo'sh quvvatlar mavjudligida ustunlikning 3-dan 2- prioritetga o'zgarishi. Xizmat bo'yicha harakatlar bo'yicha yondashuv

6.6-rasm. Texnik xizmat ko'rsatish masalalari tasnifi

Shuning uchun tashkilot byudjetini rejalashtirishda eng boshidan moliyaviy vositalarni (texnik) xizmat ko'rsatishning qisqa muddatli vazifalarini rejalashtirish va byudjetdan 2-ustun bo'yicha xizmat ko'rsatish choralarini alohida mablag' bilan ta'minlash kerak.

Xavf sinfi	Xavf tavsifi	Hal qilish va nazorat
A	Dasturiy ta'minot yoki xato boshqa komponentalarga (qismlarga) ta'sir qilmaydi	O'z-o'zini nazorat
B	Dasturiy ta'minot yoki xato o'z qo'llanish sohasi ichida (mas. Moliyaviy buxgalterlik hisobi tizimida) ta'sir ko'rsatishi mumkni	Manfaatdor soha rahbari tomonidan hal qilish va nazorat
C	Dasturiy ta'minot yoki xato boshqa qo'llanish sohalariga ta'sir ko'rsatishi mumkin (mas. Texnologik jarayon ishlab chiqarishga)	Manfaatdor soha rahbari tomonidan hal qilish va nazorat

6.7-rasm. Texnik xizmat ko'rsatish masalalari xavflari tasnifi

Agar tashkilot AT-xizmatlar yetkazib beruvchi bilan texnik xizmat ko'rsatish haqida shartnoma tuzgan bo'lsa (8.5-bobga qarang), unda bu xarajatlarni rejalashtirish albatta joriy bir martalik buyurtmali hollarga nisbatan xiyla yengilroq bo'ladi.

2-ustun prioriteti bo'yicha (texnik) xizmat ko'rsatish vazifalarini rejalashtirishdagi xavf sinflari bo'yicha dasturiy ta'minotning quyidagi taqsimlanishiga asoslanuvchi hal qilish tamoyiliga amal qilish chuqur ma'noga ega.

Xavf sinflari bo'yicha tasnif yordamida mayda o'zgarishlarning tasdiqlash jarayoni va nazorat ostiga olinishidan va bu bilan keraksiz byurokratik chigalliklar harakatga kelishdan saqlanishi kerak.

Texnik xizmat ko'rsatish vazifalari ustunlik raqamlarini bo'lish yordamida tasniflanadi. Rejalashtirish, chiqimlarni ko'rib chiqish va xizmatni o'tkazish ustunlikka bog'liq, shu bilan birga faqat tashkilot uchun qiyin bo'lgan xizmat rejalashtiriladi va oldindan chiqimlarni tahlil qilmasdan topshiriladi. Xizmatning bevosita o'tkazilishi jarayonining belgilangan uslubiy qadamlari (bosqichlari) ga muvofiq amalga oshiriladi, shu bilan birga dasturiy ta'minotni tushunish uchun kerakli vaqtni hisobga olish lozim bo'ladi.

6.4. DASTURIY TA'MINOT HUJJATLARI

Ko'pincha hujjatlashtirishga mensimay munosabatda bo'lishadi, biroq aynan dasturiy ta'minot xizmatida u xatolarni qidirib topish va statistik baholarni o'zgartirish hamda qayta ishlab chiqish uchun muhimdir.

Dasturchilar uchun maqbullik mulohazalar nuqtai nazaridan dasturiy ta'minotning kuzatib boruvchi (xizmat ko'rsatish) hisobot tizimi imkoni boricha sodda va quyidagi elementlarga ega bo'lishi kerak.

- Identifikatsiya raqami / xizmat raqami.
- Dastur va modulning nomi.
- Dasturchi.
- Bildirilgan vaziyat.
- Xizmatga buyurtma sanasi.
- Xizmatning boshlanish sanasi.
- Xizmatning tugash sanasi va ekspluatatsiyaga kiritish.
- Xizmat ko'rsatish sababi/o'zgarishlar.
- Xizmatning qisqacha tavsifi/o'zgarishlar.
- Avval boshidan baholangan mehnat xarajatlari.
- Haqiqiy mehnat xarajatlari.
- Xizmat ko'rsatish ta'sir qilgan dasturlar, modullar/o'zgarishlar.
- Boshqa eslatmalar.

Hisobot tuzilgandan keyin xizmat samaradorligi baholanishi mumkin. Xarajatlarni tahlil qilish uchun xizmatning o'rtacha davomiylikda ishlashiga baho berishni amalga oshirish mumkin.

6.5. TEXNIK XIZMAT KO'RSATISH BO'YICHA BITIM

Ma'lumotlarga zamonaviy ishlov beruvchi tashkilot apparat vositalarining va dasturiy ta'minotning beto'xtov ishlashini ta'minlab turadi. Agar, misol uchun, avtomatlashtirilgan ishlab chiqarishda apparat vositalari yoki dasturiy ta'minot safdan chiqsa, unda mahsulotni to'la ololmaslik natijasida juda tez yuqori xarajatlar yuzaga keladi va ular ma'lumotlarning ishlash dasturi bahosini oshirib yuborishi mumkin. Texnik xizmat ko'rsatish bitimi mavjudligidan tashkilot tegishlicha malakali AKT xizmati yetkazib beruvchi yoki dasturchining zimmasiga ma'lumotlarga ishlov berish dasturining beto'xtov ishlashi uchun jon kuydirish va buning uchun javobgarlik hissini ham yuklaydi.

Texnik xizmat ko'rsatish haqidagi bitimlarning quyidagi turlari ajratiladi:

• Texnik xizmat ko'rsatish shartnomasi dasturiy ta'minotning ushbu bobda bayon etilgan tahrirlovchi, takomillashtiruvchi, moslashtiruvchi xizmatlariga tegishlidir. Apparat vositalari, shuningdek, shartnomaga kiritilishi yoki alohida shartnomada ko'zda tutilishi mumkin.

• Qo'llab-quvvatlash shartnomasi dasturiy ta'minot ekspluatatsiyada foydalanuvchining telefon orqali, xat orqali yoki shaxsan qo'llab-quvvatlashiga oiddir (Hotline). Qo'llab-quvvatlash va xizmat ko'rsatish bir-biridan mustaqil bitta shartnomada birga qo'shila oladigan ikkita alohida sohani bildiradi.

• Servis xizmati shartnomasi-bu texnik xizmat ko'rsatish va qo'llab-quvvatlash shartnomasining o'zaro uyg'un birikmasidir.

Foydalanuvchilar ko'pincha xizmat ko'rsatish va qo'llab-quvvatlashni farqlay olmaydilar, shuning uchun AKT xizmati ta'minlovchilari ko'pincha bitta servis xizmati shartnomasida har ikkala xizmatni taklif qiladi. Ushbu bobda yolg'iz dasturiy ta'minot xizmatiga oid bo'lishiga qaramasdan bundan keyin qo'llab-quvvatlash shartnomasi (Hotline- shartnoma) sohasi qisqacha tasvirlab beriladi.

6.5.1. AT shartnomalar ta'moyillari

Texnik xizmat ko'rsatish, qo'llab-quvvatlash yoki servis xizmatini ko'rsatish shartnomasi, har qanday boshqa shartnoma singari xizmatlar ta'minlovchisi (mas. AKT xizmatlari ta'minlovchisi yoki dasturchi) va mijoz o'rtasidagi majburiyat yuklovchi bitimdir. Xizmat ta'minlovchisi xizmat ko'rsatish majburiyatini oladi, mijoz esa xizmat uchun haq to'lash majburiyatini oladi.

Quyidagi elementlar shartnomalarning barcha turlari uchun umumiydir:

• Xizmat ko'rsatish ta'minotchisi va mijozni manzili, telefon raqami, elektron pochtasi (E-mail) bilan ko'rsatish.

• Shartnoma mavzusi: gap nima haqida borayapti?

• Xizmat ko'rsatish ta'minotchisi xizmatlari, masalan, ma'lum vaqt oralig'ida telefon yordami.

• Shartlashayotgan tomonlarning huquq va burchlari: bunda xizmatlar aniqlashtiriladi, xizmat ko'rsatish tartibi bayon qilinadi

va masalan mijoz tomonidan yordam berish majburiyati tartibga solinadi.

- Shartnomaning kuchga kirishi shartnomaning amal qilish va uning bekor qilinishi.
- Haq to'lashning tartibga solinishi.
- Barcha shartlashayotgan tomonlar imzolari.

6.5.2. Qo'llab-quvvatlash shartnomasi (Hotline)

Qo'llab-quvvatlash shartnomasi yoki Hotline-bitim foydalanuvchining ekspluatatsiyasi davomida dasturchining yoki AKT xizmati ta'minotchisining yordamini olishini boshqaradi. Bu yordam, qoida bo'yicha, telefon orqali yoki elektron pochta bilan amalga oshiriladi. Foydalanuvchiga yordamning alohida turi, masalan, Microsoft Netmeeting yordamida Remote Desktop (uzoqlashtirilgan ish stoli) bilan ishlashdir. Bunda xizmatlar ta'minotchisi foydalanuvchinnig Desktopiga ro'yhatga yoziladi, unnig roziligi bilan masofali nazoratni vaqtincha qarz olib turadi va shunday qilib nafaqat ma'lum xizmatkor va ish usullarini tushuntirib bera oladi, balki bevosita ularni ko'rsatib beradi.

Hotline-bitim "xizmatlari, huquq va burchlari" bo'limida quyidagilar tartibga solinadi:

- Narsa (predmet) – bu qanday dasturlar bo'yicha yordam ko'rsatayapti deganidir.
- Xizmat ta'minotchisiga yeta olish (telefon, e-mail).
- Maxsus to'g'ri chiziq bo'yicha aloqa vaqti (masalan, s 08.00 dan 16.00 gacha).
- Maxsus to'g'ri chiziq bo'yicha aloqa hajmi (masalan, 1 oyda 10 soat).
- Xizmat ta'minotchisining javob qaytarilish vaqti (masalan, Hotline-rasmiy talabidan so'ng ikki soat davomida).
- Mijoz dasturiy ta'minotining shart-sharoitlari, masalan, Remote Desktop yordamida.
- To'lov (Har oy uchun yoki bajarilgan ish hajmiga bog'liq ravishda).

Foydalanuvchilar, ko'pincha Hotline ning "dolzarb" yo'li orqali o'qitishni talab qilishga moyildir. Biroq yordamning mohiyati **o'qitilgan** foydalanuvchiga joriy ish davomida yordam ko'rsatishdan iboratdir.

Bu o'rinda AKT xizmat ta'minotchisi mijozning xodimlari dasturiy ta'minot ekspluatatsiyasini o'rgangan bo'lishlari haqida qayg'urishi va yordam shartnomasida Hotline hajmini chegaralash yordamida tegishli kafolatlarini ko'zda tutishi kerak.

6.5.3. Texnik xizmat ko'rsatish shartnomasi

Texnik xizmat ko'rsatish shartnomasi yordam shartnomalariga (Support-shartnomalari) nisbatan keng ko'lamli va murakkabroq. Ushbu bobda bayon etilgan fikrlar takrorlanuvchi, takomillashtiruvchi va moslashtiruvchi xizmatlarga tegishli.

Texnik xizmat ko'rsatish shartnomalari ishlatilayotgan dasturiy ta'minotning:

- Xatosiz ishlashi yo' li bexatolikni ta'minlovchi xizmat.
- Texnika darajasida qolishni takomillashtiruvchi xizmat.
- Yangi operatsion tizimlariga yoki yangi apparat vositalariga moslashtirilgan bo'lishini ta'minlashi kerak (moslashtirish xizmati).

Texnik xizmat ko'rsatish shartnomalari qanchalik muhimligini quyidagi misolda tushuntirish mumkin:

O'zbekmashinasozlik tashkiloti XYZ avtomabil ishlab chiqarish uchun ehtiyot qismlar tayyorlaydi. Taskiliy ma'lumotlarga elektron ishlov berishdan foydalanayotganda, uning yordamida nafaqat ishlab chiqarish, balki ishlab chiqarishning butun moddiy texnik ta'minoti (logistika) va hisob- kitoblari boshqariladi. UNIFIN dasturi yordamida mijozlar va ta'minotchilar bilan tuziladigan barcha shartnomalarga umumiy hisobchilikka va buxgalteriya hisobiga ishlov beriladi.

O'zbekiston Respublikasi soliq islohatini o'tkazadi va barcha soliq turlari va soliq miqdorlarini o'zgartiradi. Ma'lum kunlardan boshlab barcha hisob-kitoblar faqat yangi soliq tizimi sharoitlarida amalga oshirilishi mumkin.

Shunday qilib, UNIFIN dasturini yangi soliq tizimiga to'g'rilash kerak. Tashkilotning rasmiy talabiga AKT xizmat ta'minotchisi javob beradi: "Biz bilan texnik xizmat ko'rsatish shartnomasini tuzgan mijozlar avtomatik tarzda dasturning yangi versiyasini oladilar, shuning uchun yangi soliq tizimiga to'g'rilashni o'z vaqtida o'tkazish mumkin. Texnik xizmat ko'rsatish shartnomalariga ega bo'lmagan mijozlar dasturni yangidan sotib olishlari kerak. Afsuski biz narxlarni oshirishimiz kerak. Sizdan shunga tushunib yondashishingizni so'ray-

mizki, biz avval o'z mijozlarimizni texnik xizmat ko'rsatish shartnomasi bo'yicha ta'minlaymiz, qolgan mijozlar uchun hozirgi vaqtda talab yuqoriligi sababli ta'minlash muddati 6 oyni tashkil qiladi".

Bu tipik senariy. XYZ mashinasozlik tashkiloti ko'p sohalarda foydalanilayotgan UNIFIN dasturiga bog'liqdir. Dasturning benuqson ishlashi tashkilot uchun o'ta muhimdir.

UZS dan CAR ga yuqorida eslatilgan to'g'rilash oldindan rejalashtirilishi va amalga oshirilishi valyuta - moliyaviy islohotning belgilangan kundan oldinroq tugallanishi lozim bo'lgan zarur tahrirlovchi yoki takomillashtiruvchi xizmatlariga namunadir.

Texnik xizmat ko'rsatish shartnomasining mavjudligi xizmatlar taklif qilayotgan AKT tashkilotga dasturning o'z vaqtida moslashuvini amalga oshirish va uni mijozlar ixtiyoriga havola qilishi majburiyatini yuklaydi. Texnik xizmat ko'rsatish shartnomasiz dasturning har bir o'zgarishiga alohida buyurtma qilish kerakki, bu ma'lum hollarda katta va rejalashtirilmagan chiqimlarga olib kelishi mumkin. Belgilangan to'lovni ko'zda tutadigan texnik xizmat ko'rsatish shartnomasi tufayli bu xarajatlarni rejalashtirsa bo'ladi va ular alohida yuzaga kelmaydi.

Texnik xizmat ko'rsatish shartnomasining "xizmatlar, huquq va majburiyatlar" bo'limida quyidagilar boshqariladi:

Shartnoma mavzusi

Shartnoma mavzusi xizmat ko'rsatishga loyiq dasturiy ta'minotdir. Texnik xizmat ko'rsatish shartnomasi, shuningdek, namunali shartnoma kabi tuzilishi, xizmat qilishi lozim bo'lgan dasturlar esa shartnomaga ilova sifatida alohida sanab o'tilishi mumkin. Bu shartnomaning xizmat ko'rsatishini yengillashtiradi, shuning uchun dasturiy ta'minot hajmi yoki xizmat qilishi lozim bo'lgan dasturlar o'zgargan holda, yana yangidan umumiy shartnoma tuzishga zarurat yo'q, balki shunchaki ilovani dolzarblashtirishgina kerak bo'ladi.

Xizmatlar odatda

- Xatolarni bartaraf qilish.
- Dasturiy ta'minotning qonuniy o'zgarishlarga moslashuvi.
- Xizmatlarning yaxshilanishi va to'ldirilishi (kengaytirilishi).
- Service Packs-xizmat paketlarining oraliq o'zgarishlarini yetkazib berishini.
- Asosiy o'zgarishlarni (masalan, yangi operatsiya tizimidagi) yetkazib berishni o'z ichiga oladi.

Jumladan kelgusi aniq o'zgarishlarning bo'lg'usi xizmatlari kelishib olinadi.

Agar maxsus to'g'ri chiziq bo'ylab aloqa (Hotline) yoki maslahat kabi xizmatlar kelishib olinadigan bo'lsa, unda ularni vaqt bo'yicha chegaralash maqsadga muvofiq (masalan, 1 oyda 10 soat Hotline va 10 soat maslahat).

Shuningdek, bu yerda dasturiy ta'minotning boshlang'ich kodi betaraf muassasada (masalan notarus) qoldirilishi haqida farmoyish berilishi kerak. Bu, masalan, dasturni ishlab chiqaruvchi bankrotga uchragan holda joriy qilingan dasturiy ta'minot mijoz ixtiyorida qolishi uchun mijozni kafolatlash ehtiyojiga xizmat qiladi.

Huquq va majburiyatlar

• "Huquq va majburiyatlar" bo'limida xizmatni o'tkazish jarayoni, shuningdek, AKT xizmati ishlab chiqaruvchisining kafolatlari tartibga solinadi.

• Odatdagi tarkib quyidagicha:

• AKT xizmati ta'minotchisi dasturiy ta'minotning ishga yaroqliligini kafolatlaydi.

• Foydalanuvchi ma'lumotlar himoyasini ta'minlashni amalga oshiradi.

• Foydalanuvchiga dasturiy ta'minotni mustaqil o'zgartirishga ruxsat berilmaydi.

• Foydalanuvchi xizmat ko'rsatish bo'yicha harakatlar uchun mashina vaqtini taqdim qiladi.

• Foydalanuvchi uzoqlashtirilgan xizmat uchun modem kirish yo'li yoki VPN-tunelini (agar bu kelishilgan bo'lsa) ishga soladi.

• Shartnoma sirini oshkor qilmaslik uchun ikki tomonlama majburiyat belgilab olinadi.

• Bu bo'limda, shuningdek, xizmat ko'rsatish vaqti va reaksiya javob vaqti kelishilgan bo'lishi kerak.

• Javob vaqti – bu foydalanuvchining xati haqidagi xabari bilan xatoni bartaraf qilishning boshlanishi o'rtasidagi xatdir. Ba'zi bir shartnomardagi "kengaytirish senariylari"da javobning tegishlicha har xil vaqtlari bilan ajratiladi.

• Qurshab oluvchi xatolar (javob vaqti 24 soatdan kamroq).

• Fojeali xatolar.

• Tashqi turdagi nuqsonlar.

Xizmat ko'rsatish vaqti – bu xato haqidagi xabar qabul

qilinadigan va ishlov beriladigan vaqt. Qoidaga ko'ra, u byuroning odatdagi ish vaqtiga mos keladi. Bir nechta smena bilan (almashuv bilan) ishlaydigan ishlab chiqaruvchi korxonalarda ko'pincha kunutun xizmat ko'rsatish ma'qul. Bu alohida kelishib olinishi kerak.

Boshqa shart-sharoitlar

Texnik xizmat ko'rsatish shartnomalarining bundan keyingi mazmuni (shartnomaning kuchga kirishi, shartnomaning amal qilish muddati, shartnomaning bekor qilinishi) umumiy AT - shartnomalariga muvofiq keladi. To'lov sifatida paushal miqdor yoki xizmat qilishi lozim dasturiy ta'minotning xarid narxidan foiz miqdori qabul qilingan va AQSHda kelishilgan xizmatlar hajmiga bog'liq ravishda xizmatga yillik to'lov sifatida dasturiy ta'minot narxining 8 %idan 25 % igacha miqdori qabul qilingan.

Xizmat uchun to'lovni oldindan undirish va AKT xizmat ta'minotchisi foydalanuvchi tomonidan to'lov kechiktirilgan holda xizmatni cheklashi yoki to'xtalishi mumkinligi haqida kelishuv to'g'risida shartlashib olish tavsiya qilinadi.

Shartnoma bitimlarining texnik dasturiy ta'minoti (tahrirlovchi, takomillashtiruvchi va moslashtiruvchi) xizmat ko'rsatish shartnomalari va qo'llab-quvvatlovchi (foydalanuvchi yordami Hotline) shartnomalari farq qiladi. Servis xizmati ko'rsatish shartnomalari xizmat ko'rsatishdan ham, yordam ko'rsatishdan ham iborat. Texnik xizmat ko'rsatish shartnomalari mijozni foydalanilayotgan dasturiy ta'minot ishlash layoqati bilan ta'minlaydi va dasturiy ta'minot xizmat ko'rsatishni rejalashtirish va xarajatlari byudjetini tuzish uchun yaxshi asos bo'ladi.

6-bo'limga oid nazorat savollari

1. Dasturlarning texnik xizmat ko'rsatish shartnomasining komponentlari qanday bo'ladi?
2. Portfolio-tahlil nima?
3. Texnik xizmat ko'rsatish bosqichlari va masalalari nimadan iborat?
4. Dasturlarning xizmat ko'rsatish bo'yicha asosiy harakatlari nimadan iborat?

7. DASTURIY TA'MINOT BO'YICHA HUJJATLAR (DASTURLARNI TAVSIFLASH)

O'quv maqsadi

O'quvchi dasturiy ta'minot bo'yicha turli xildagi hujjatlarni yaxshi o'zlashtirdi va misollar asosida alohida turlar o'rtasidagi tafovutlarni tushintirib berishi mumkin.

Tarkibi

- Foydalanuvchi hujjatlari (installatsiya / ekspluatatsiya)
- Dasturiy ta'minot bo'yicha hujjatlar
- Loyiha hujjatlari

7.1. HUJJAT TURLARI

Ko'pincha dasturchilar butun diqqatlarini tobora ko'p miqdordagi va tobora murakkab vazifalar ishlab chiqarishga qaratadi va hujjatlar bilan ishlashga keragicha e'tibor bermaydi. Biroq aynan yaxshi hujjatlashtirish tizimli dasturning sifat belgisidir va shuning uchun hujjatlashtirishga avvaldan yetarlicha vaqt va resurslar rejalashtirilgan bo'lishi kerak, uni ishlab chiqish esa sinchiklab amalga oshirilishi kerak.

Dasturni ishlab chiqarishda hujjatlarning quyidagi turlari farqlanadi:

Hujjat turlari	Mo'ljallangan guruh	Shakl
Foydalanuvchi hujjatlar	Foydalanuvchi	Ma'lumotnomadan onlayn-yordam, CD
Dasturiy ta'minot bo'yicha hujjatlar (tizim hujjatlari)	Dasturchi tizim ma'muri	Boshlang'ich kod hujjatlari, listinglar, tasniflar, diagrammalar
Loyiha hujjatlari	Buyurtmachi, menejer	Axborotlar, loyiha hisobotlari

7.1-rasm. Hujjat turlari, shakllar va mo'ljallangan guruhlar

Hujjatlashtirishdan maqsad dasturni o'zi ishlab chiqmagan, aloqasi yo'q shaxslar ishlab chiqilgan dasturiy ta'minot xizmatlari va qo'llanishi qanday bo'lsa, uning qurilish va ichki tuzilishi ham shunday ekanligini tushunib tasavvur qila olishdadir.

7.2. FOYDALANUVCHI HUJJATLARI

Dasturiy ta'minot foydalanuvchi hujjatlarining foydalanuvchisi va dasturiy ta'minot qo'llanilayotgan tashkilotning tizim ma'muri uchun tuziladi. Odatda, bu hujjatlar bir yoki bir nechta ma'lumotnoma ko'rinishida tuziladi yoki lazer diskiga (CD) yoziladi va 3 qismga ajratib chiqiladi.

- Installyatsiya bo'yicha ma'lumotnoma.
- Xizmat ko'rsatish bo'yicha ma'lumotnoma.
- Boshqarish bo'yicha ma'lumotnoma.

Installyatsiya bo'yicha ma'lumotnoma dasturiy ta'minotni apparat vositalarida to'g'ri Installash uchun xizmat qiladi. Yaxshi sifatli dasturiy ta'minot bugun bevosita foydalanuvchining o'zi tomonidan installyangan bo'lishi mumkin. Ko'pincha buning uchun (qisman) avtomatik installyatsiya (installyatsiya ustasi) foydalaniladi, u apparat vositalari joylashuvini mustaqil tarzda aniqlab oladi va foydalanuvchini ekspluatatsiyaga kiritish bo'yicha zarur ma'lumotlarni interaktiv to'ldirishga taklif qiladi (installyatsiya yo'li, qismlar). Ushbu holda foydalanuvchi uchun hujjatlar doirasidagi installyatsiya bo'yicha ma'lumotnoma eng kamida to'ldirilishi lozim ko'rsatkichlar ko'rsatilgan, installyatsiya paytida paydo bo'ladigan oynalar (Screenshots)ga ega bo'lishi kerak.

Xizmat ko'rsatish bo'yicha ma'lumotnomada foydalanuvchi dasturiy tizim bilan tanishadi. Barcha xizmatlar va ularning ma'lumotlar fondiga ta'siri tushuntirib beriladi. Xizmat ko'rsatish bo'yicha yaxshi ma'lumotnoma foydalanuvchiga oddiy, lekin kompleks (mufassal) misolda asosiy xizmatlar hajmi va dastur xizmati paytidagi tegishli harakat usulini tushuntirib beruvchi sistemaga kirishdan iborat.

Boshqarish bo'yicha ma'lumotnomada dasturiy ta'minot tizimini bitta kompyutorda yoki tarmoqda boshqarish uchun xizmat qiladi. Unga tizimni boshlah yoki tugallash foydalanuvchilarni boshqarish va huquqlarni berish, shuningdek, tegishli qarorlar bilan birga, tizimning to'xtab qolish signalizatsiyasi kabi jarayonlar tegishli.

7.3. DASTURIY HUJJATLAR

Dasturiy – shuningdek tizim hujjatlari deb ataluvchi hujjatlar - dasturiy ta'minot tuzilishini va uning tuzilmasini tavsiflab beradi. U xatolarni qidirishda yordamlashadi va tizimni o'zgartirish va kengaytirish uchun zarur ma'lumotlarni taqdim qiladi. Dastur yoki tizim hujjatlari shunchalik to'la va shunday qurulishi kerakki, aloqasi yo'q, dasturni o'zi dasturlamagan kishi ham uni tushuna va qayta tiklay olishi mumkin bo'lsin.

Dasturiy hujjatlar dasturni ishlab chiqarishning eng boshlang'ich bosqichida boshlangan bo'lishi va doim dolzarb ahvolda tutilishi kerak. Quyidagi tarkib muhim hisoblanadi:

- Masalaning qo'yilishini tavsiflash (tizim tasnifi).
- Amalga oshirish tavsifi.
- Foydalanuvchi ma'lumotlari tavsifi.
- Testli bayonlashtirish.
- Barcha dasturlar listingi.

Dasturiy hujjatlaring muhim tarkibiy qismi, eng avvalo, tizim tasnifidir. U buyurtna haqida masala qo'yilishidan kelib chiqadi va 2.3 bo'limda ko'rsatilgan elementlardan tarkib topadi. Dasturiy hujjatlarninng bu qismi uchun hujjatlashtirishning 3.1 va 3.2 texnikalar (tuzilish diagrammari, diagrammalar, ma'lumotlar oqimining o'tish tizmasi, ob'ekt munosabat turidagi modellar (ER - modellar) va boshqalar) foydalanishi mumkin.

Hujjatlarning har doim dolzarb holda tutilishi muhim, bu shu demakki, dastur tuzilishida yoki dasturning boshlang'ich kodida o'zgarishlar bo'lganida, ular dasturiy hujjatlarda ham aks etgan bo'lishi lozim.

Foydalanilgan massiv ma'lumotlarni tavsiflashi, sinovlar natijalarini protokollash va barcha dasturlarni listinglari dasturiy hujjatning asosiy qismi deb hisoblanadi. Ular texnik xizmat ko'rsatish va dastiriy tizimni modifikasiyalash imkonini beradi va osonlashtiradi.

Ma'lumotlar massivi ularning nomlarini, mundarijasini, yozuv tuzilishini, fayl yaratishni, shuningdek, imkon bolgan yozish va o'qish amallarini ko'rsatish orqali tavsiflanadi.

Boshlang'ich dastur kodi izoh satri yordamida qancha yaxshi hujjatlashtirilgan bo'lsa, shuncha dasturning tavsifi modul tahlili, parametr va interfeyslarning tavsifi chegaralangan bo'lishi mumkin.

Umuman olganda dastur hujjatlari qisqa, aniq va ko'rgazmali

bo'lishi, va albatta u yaratilgan dasturiy ta'minot tizimi to'liq yoritilgan bo'lishi kerak.

7.4. LOYIHA HUJJATLARI

Agarda foydalanuvchi va dasturiy ta'minot hujjatlari mukammal tuzilgan bo'lsa, u holda lohiha hujjatlarini tugatish umuman qiyin bo'lmaydi.

Dasturiy hujjatlar texnik qismlarning katta bilimlariga ega bo'lmagan va/yoki dasturning har bir qismi bilan shug'ullanishga vaqti yo'q buyurtmachi yoki menedjer uchun mo'ljallangan.

Shuning uchun loyiha hujjatlari umumiy tizim tahlilnomasini ko'rsatish bilan chegeralanadi, shunday ekan, nafaqat talablar tahlilidan va foydalanuvchi hujjatlaridagi funksionallik bilan, balki dasturiy huggatlardagi umumiy tuzilma va ichki tuzilish bilan ham chegeralanadi. Tahlilnomalarda qismlardan voz kechiladi va foydalanuvchi hujjatlari yoki dasturiy hujjatlarning tegishli qismlaridan dalil keltiriladi.

Dasturiy hujjatlarda masalalar qo'yilishi va olingan natijalarni qiyoslash, shuningdek, moliyaviy xarajatlar va ishchi kuchi xarajatlarini ko'rsatish muhimdir. Buyurtmachi va menedjer uchun bu muhim hal qiluvchi elementlardir.

Hujjatlashtirishning 3 ta asosiy turi mavjud: foydalanuvchi hujjatlari (ma'lumotnoma, onlayin-CD) dasturiy ta'minot foydalanuvchisi uchun dastur xizmatlarini tavsiflab beradi. Dasturiy (shuningdek, tizim hujjatlari deb nomlanuvchi) hujjatlar dasturiy ta'minotning ichki tuzilishini va texnik qismlarini dasturchi uchun tavsiflab beradi. Loyiha hujjatlari buyurtmachi va menedjer uchun jamlovchi tahlilnomadir. Yaxshi hujjatlar shart bo'lib, bu dasturiy ta'minot sifatining belgisidir.

7-bo'limga oid nazorat savollari

1. Hujjatning uchta bosh turi nimadan iborat va ular nimaga xizmat qiladi?
2. Foydalanuvchi hujjati qanday ma'lumotnomalardan iborat?
3. Dasturiy hujjat nimalardan tashkil topadi?
4. Loyiha hujjati nima?

1-QISM BO'YICHA RASMLAR RO'YXATI

Darslikning bu qismida Visual Basic tili mufassal yoritiladi. Mavzularga oid misollar keltirilgan. Misollarni dasturini tuzishda darslikning birinchi qismida keltirilgan dasturlash usullari, texnikalari va texnologiyalaridan keng foydalanilgan. Olingan natijalarning visualizatsiyasi ham keltirilgan.

8. VISUAL BASIC MUHITI

8.1. VISUAL BASICNI ISHGA TUSHIRISH

Windows asosiy menyusidan dasturni ishga tushirish uchun quyidagilarni bajarish kerak:

1. Ekranning quyi qismida joylashgan **"Пуск" (Start)** tugmasini bosing.

2. Windowsning asosiy menyusidagi "Программы" **(Programms)** ni tanlang. Menyuda ishga tushirish buyrug'i paydo bo'ladi.

3. **Microsoft Visual Studio 6.0** optsiyasini tanlang.

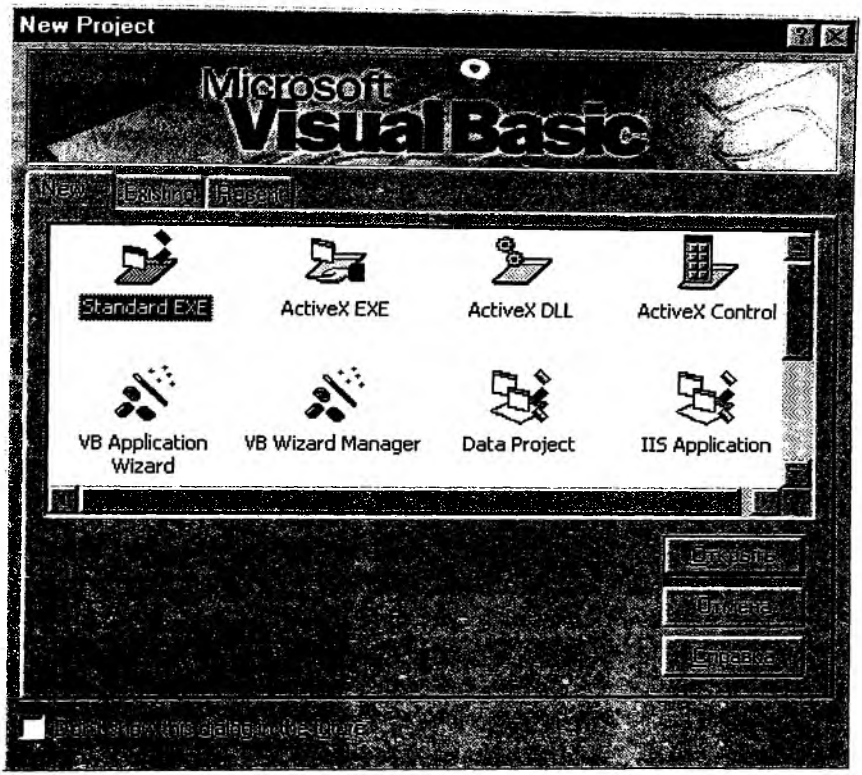
4. Navbatdagi menyu bandidan **Microsoft Visual Studio 6** ni tanlang.

Visual Basic 6 ni ishga tushirganda ekranda **New Project** muloqot oynasi chiqadi. Uning yordamida yangi loyiha uchun shablon tanlash, loyiha yaratish masterini ishga tushirish yoki mavjud bo'lgan loyihani ochish mumkin.

Bu oyna uchta ilovadan tashkil topgan:

- **New** (Yangi) - shablonlardan va yangi loyiha yaratuvchi loyiha masteridan tashkil topgan.
- **Existing** (Mavjud) – ilgari yaratilgan loyihani va Visual Basicning loyiha-misollarini ochishga imkon beradi. Ushbu ilova yoyiladigan menyuga ega va u yordamida kompyuterda barcha mavjud bo'lgan loyiha papkalarining birini tanlash mumkin.
- **Recent** (Yaqinda yaratilgan) – Oxirgi ish vaqtida ochilgan loyihalarni o'z ichiga oladi.

Yangi loyihani yaratishda **New** ilovasi ishlatiladi. Unda mavjud bo'lgan loyihaning shablon turlarini tanlash mumkin, lekin boshlang'ich bilim olish maqsadida standart ilovani tanlaymiz:



8.1-rasm

Standard EXE - bajariladigan standart ilovalar.

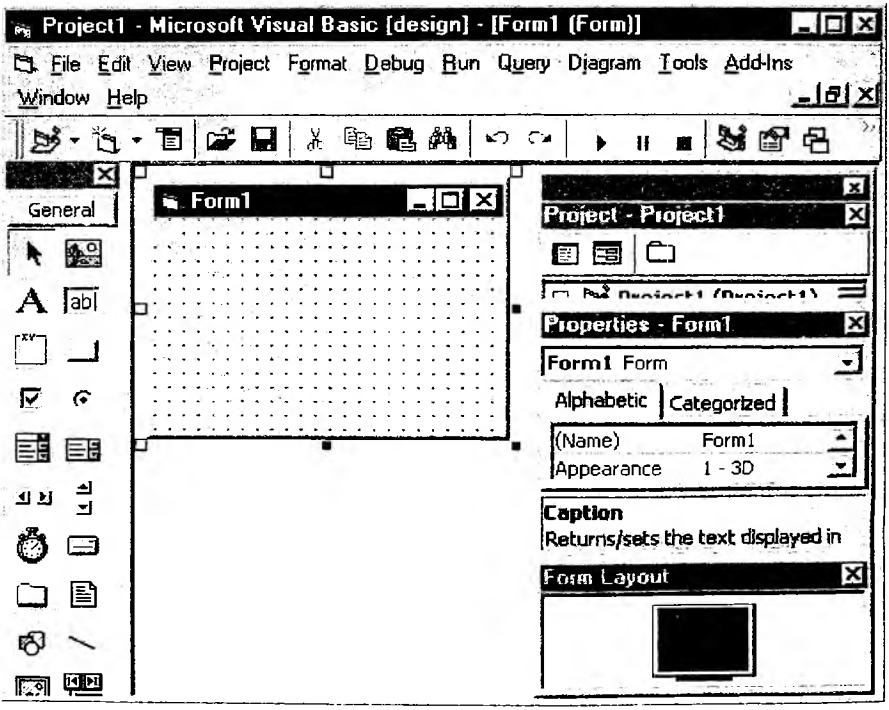
8.2. ISHLANMANING INTEGRALLASHGAN MUHITI

Ishlab chiqishning integrallashgan muhiti (IDE) bizga ma'lum bo'lgan Microsoft ilovalarining boshqa turdagi grafik interfeysini namoyon qiladi. Uning tashqi korinishi 8.2-rasmda ko'rsatilgan.

Loyihalash muhitining tarkibiga quyidagi asosiy elementlar kiradi:

- asosiy menyu;
- asbob-uskunalarning standart paneli (**Standard**);
- boshqarish elementlari paneli;
- loyiha yurituvchi oyna (**Project**);
- forma konstruktori;
- menyu tahrirlagichi (**Menu Editor**);

- xossalar oynasi (**Properties**);
- forma maketi oynasi (**Form Layout**);
- ob'yektlarni ko'rish oynasi (**Object Browser**);
- dastur kodini tahrirlagich.



8.2-rasm

8.2.1. Asosiy menyu

Asosiy menyu Microsoft ilovalaridagi kabi ochilib-yopiluvchi qism menyularidan iborat qatordan tashkil topgan.

U quyidagi asosiy buyruqlardan iborat:

- File (fayl)
- Edit (Tahrirlash)
- View (Ko'rinish)
- Project (Loyiha)
- Format (Format)
- Debug (Rostlash)
- Run (Ishga tushirish)
- Query (So'rov)

- Diagram (Diagramma)
- Tools (Servis)
- Add-Ins (Sozlash)
- Window (Oyna)
- Help (Yordam)

Asosiy menyu ko'rinishi 8.3-rasmda keltirilgan.

8.3-rasm

Asosiy menyuning ko'pchilik buyruqlari Windows ilovalarida Ishlatiladi (masalan, Microsoft Word yoki Microsoft Excel). Shu Sababdan ular alohida ko'rib chiqilmaydi. Zarur bo'lgan tushintirishlar materiallarni yetkazib berish jarayonida beriladi.

Menyu pastida joylashgan, ko'proq ishlatiladigan buyruqlar menyu standart asboblar panelida kichik rasmchali tugmalar shaklida tasvirlangan.

8.2.2. Standart asbob-uskunalar paneli

Standart asboblar paneli asosiy menyu ostida joylashgan. Asboblarning standart panelida ko'p ishlatiladigan menyu buyruqlarini chaqirish uchun tugmachalar joylashgan.

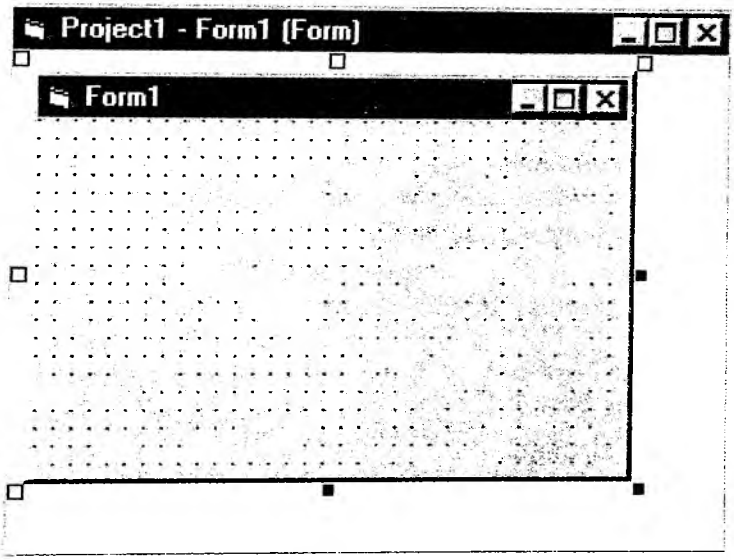
8.4-rasm

Standart asboblar panelining ko'pchilik tugmachalari tavsiyaLari boshqa Windows ilovalari tavsiyalari bilan mos keladi. Qator tugmachalar visual dasturlash muhitining o'ziga xos maxsus funksiyalarni bajaradi (masalan,  **Menyu Editor** (Tahrirlagich menyusi) yoki  **Toolbox** (Boshqarish elementlari paneli)). Quyida standart panelning tugmachalarining tavsiyalari jadvali keltirilgan.

Tugma	Nomi	Tavsiya qilinishi
	**Add Standard EXE Project** (Standart loyihaqo'shish)	Standart exe-loyihaqo'shadi
	**Add Form** (Formaqo'shish)	Loyihagaformani qo'shadi
	**Menu Editor** (Tahrirlash menyusi)	Tahrirlash menyusini chaqiradi
	**Open Project** (Loyihani ochish)	Loyihani ochadi
	**Save Project** (Loyihani saqlash)	Loyihani saqlaydi
	**Cut** (Kesib olish)	Almashish buferigaaxborotni qirqib oladi
	**Copy** (Nusxalash)	Almashish buferiganusxaoladi
	**Paste** (Qo'yish)	Almashish buferidan axborot qo'yadi
	**Find** (Qidirish)	Kontekst bo'yichaaxborot qidirishni amalgaoshiradi
	**Can't Undo** (Oldingini bekor qilish)	Oldingi holatni bekor qiladi
	**Can't Redo** (Takrorlashni bekor qilish)	Bekor qilingan holatni tiklash
	**Start** (Ishgatushirish)	Dasturni ishgatushiradi
	**End** (Yakunlash)	Dastur bajarilishini yakunlaydi
	**Break** (To'xtatish)	Dastur bajarilishini to'xtadadi
	**Project Explorer** (Loyihalar sharhlovchisi)	Loyihalar sharhlovchisining oynasini ochadi
	**Properties Window** (Xususiyatlar oynasi)	Xususiyatlar oynasini ochadi
	**Form Layout Window** (Formamaketi oynasi)	Formamaketi oynasini ochadi
	**Object Browse** 1.2.3. (Ob'yektlar brauzeri)	Ob'yektlar brauzeri oynasini ochadi
	**Toolbox** (Boshqarish elementlari paneli)	Boshqarish elementlari panelini ochadi
	**Data View Window** (Ma'lumotlarni ko'rish oynasi)	Ma'lumotlarni ko'rish oynasi ochadi
	**Visual Component Manager** (Vizual komponentlarini boshqarish)	Visual Component Manager vizual komponentlarini boshqarish oynasini ochadi

8.2.3. Forma konstruktori oynasi

Ilovalarni vizual loyihalashda forma konstruktori oynasi asosiy ishchi oyna hisoblanadi (5-rasm). Bu oynani chaqirish uchun asosiy menyuning **View** (Ko'rinish) menyusidagi **Object** (Ob'yekt) buyrug'i yoki loyihalar sharhining **Forms** guruhidagi ob'yektning kontekst menyusining **View Object** buyrug'i tanlanadi.



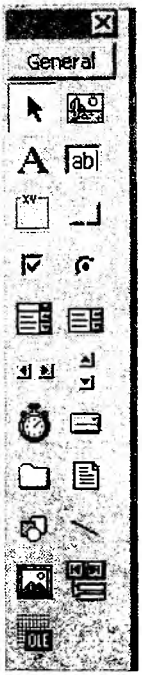
8.5-rasm

Formalar konstruktori oynasidagi barcha formalar va ilova ob'yektlari ishlab chiqiladi. Formadagi ob'yektlarning aniq pozitsiyalarini aniqlashga imkon beradigan to'r hosil bo'ladi.

Formaning va sichqonchaning ajratgich markerinidan foydalanib oynadagi formaning o'lchamlarini o'zgartirish mumkin. Formaning o'lchamlarini o'zgartirish uchun sichqoncha ko'rsatkichini markerga o'rnatish kerak, u ikki yo'nalishli strelka ko'rinishiga ega bo'lganda uni talab etilgan o'lchamlargacha siljitiladi.

8.2.4. Boshqarish elementlari paneli

Boshqarish elementlari paneli - bu ilova formasini visual ishlab-chiqishda asosiy ishchi asbobi hisoblanadi (8.6-rasm). Boshqarish



8.6-rasm

elementlari paneli **View** (Ko'rinish) menyusidagi **Toolbox** buyrug'ini tanlash orqali chaqiriladi. Bu panelni chaqirishning yana bir usuli standart asboblar panelining **Toolbox** tugmasi bosiladi.

Boshqarish elementlar paneli tarkibiga formani boshqarishning asosiy elementlari - belgilar, matnli maydonlar, tugmalar, ro'yxatlar va forma maketini tezroq visual loyihalash uchun boshqa elementlar kiradi.

Boshqarish elementlarini formaga elementlar paneli yordamida joylashtirish quyidagicha amalga oshiriladi:

1. Sichqoncha yordamida talab qilingan boshqarish elementini tanlangiz.

2. Forma qurish (konstruktor) oynasiga o'tingiz. Shuning bilan sichqoncha ko'rsatkichi jo'ylashtirilgan ob'yekt joyini o'rnatishga imkon beradigan krest holatiga o'tadi. Sichqonchaning chap tugmasini bosgan holda yangi ob'yektning pozitsiyasi o'lchamlarini kiritingiz.

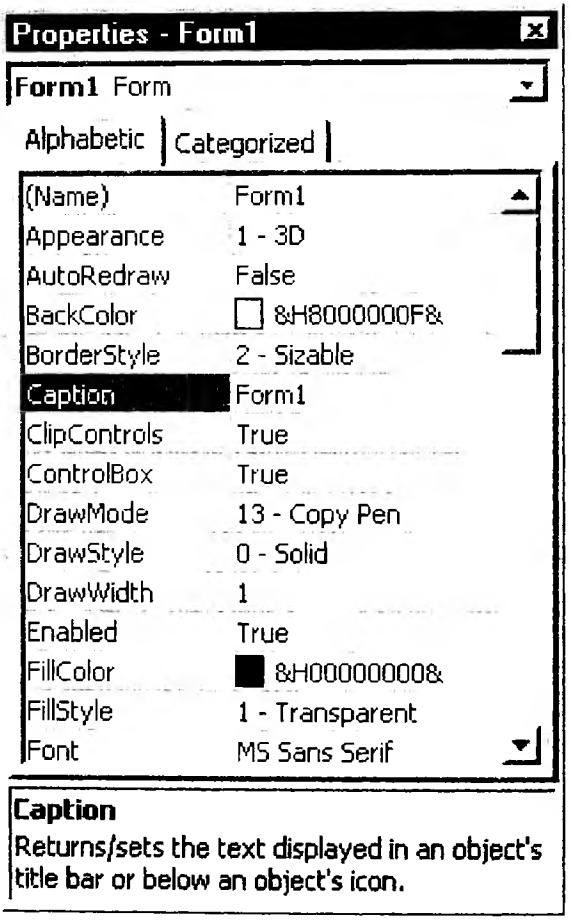
Boshqarish elementlari paneli tugmalari

Tugma	Nomi	Tavsiya etilishi
	Pointer (Ko'rsatkich)	Sichqoncha markerini (ko'rsatkichini) pozitsiyalash uchun ishlatiladi
	PictureBox (Grafik oyna)	Guruhga elementlarni birlashtirish uchun, unga grafik tasvirlarni, matnni, grafik elementlarni va animatsiyani chiqarish uchun mo'ljallangan grafik oynani formaga joylashtiradi
	Label (Belgi)	Formaga matnli axborotlarni, yozuvlarni va eslatmalrni chiqarish uchun mo'ljallangan ob'yektlarni joylashtiradi
ab|	**TextBox** (Matnli maydon)	Formaga matnli axborotlarni, sonlarni va vaqtlarni kiritish uchun foydalaniladigan matnli maydonlarni joylashtiradi
	Frame (Ramka)	Ob'yektlarni mantiqiy guruhga oladigan sarlavhali ramkani formaga joylashtiradi

Tugma	Nomi	Tavsiya etilishi
	**CommandButton** (Boshqarish tugmasi)	Formaga initsiyatsiya qilish, buyruqlarni bajarish, dasturni ishga tushirish uchun mo'ljallangan boshqarish tugmalarini joylashtiradi
	**CheckBox** (Bayroqcha)	Dasturning bajarilishi uchun shartlarni shakllantirish yoki "ha/yo'q" tamoyilida ishlovchi qandaydir sozlashlar uchun bayroqchalarni formaga joylashtiradi
	**OptionBufrton** (O'chirib-yondirgich)	Ishlash rejimini tanlash yoki dasturning bajarilishini sozlash uchun o'chirib-yondirgichlarni formaga joylashtiradi
	**ComboBox** (Ro'yxatli maydon)	Formada bir vaqtning o'zida kiritish maydonidan va ochilib-yopiluvchi ro'yxatdan tashkil topgan ob'yektni yaratadi
	**ListBox** (Ro'yxat)	Formada tavsiya etilgan ro'yxat qiymatlarining bittasini yoki birnechtasini tanlash uchun mo'ljallangan ro'yxat yaratadi
	**HScrollBar** (Gorizontal chizg'ich)	Formaga berilgan diapazondagi qiymatni tanlash uchun siljitgich sifatida ishlatiladigan vertical kesmani joylashtiradi
	**VScrollBar** (Vertikal chizg'ich)	Formaga berilgan diapazondagi qiymatni tanlash uchun siljitgich sifatida ishlatiladigan gorizontal kesmani joylashtiradi
	**Timer** (Taymer)	Formaga taymerni joylashtiradi
	**DriveListBox** (Qurilmalar ro'yxati)	Formada qurilmalar ro'yxatini yaratadi
	**DirListBox** (Papkalar ro'yxati)	Formada papkalarning daraxt ko'rinishini yaratadi
	**FileListBox** (Fayllar ro'yxati)	Formada fayllar ro'yxatini yaratadi
	**Shape** (Shakl)	Formada geometrik shakllarni - to'g'riburchaklik, kvadrat, aylana, ellips, doira, aylanaburchakli to'g'riburchaklik va kvadrat yaratadi
	**Line** (Chiziq)	Chiziqlar yaratadi
	**Image** (Tasvir)	Formaga grafik tasvirlarni joylashtirish uchun mo'ljallangan maydon yaratadi
	**Data** (Ma'lumotlar)	Formada yozuvlar bo'yicha siljishlar va navigatsiya natijasini tasvirlash uchun ma'lumotlar bazasida ma'lumotlarni boshqarish elmentlarini yaratadi

8.2.5. Xossalar oynasi

Xossalar oynasi (**Properties**) forma va unga joylashtirilgan ob'-yektlarning xossalarini sozlash va tasvirlash uchun qo'llaniladi. Unga, masalan, belgilab olingan ob'yekt xossasi formadagi joylashgan pozitsiyasi, balandligi, kengligi, rangidan tashkil topgan (8.7-rasm).



8.7-rasm

Xossalar oynasidagi **Caption** xossasiga "Yozuv matnini o'zgartirish" tugmasining sarlavhasini kiriting. Sarlavhani bosish jarayonida & belgi maxsus qiymatga ega va aks ettirilmaydi. ALT tugmasi va tagi chizilgan belgi yordamida siz ob'yektni sichqonchasiz tanlash imkoniga ega bo'lasiz. Bu tugmalarning ekvivalent kombinatsiyasi deyiladi.

8.3. BOSHQARISH ELEMENTLARI PANELINING BIRNECHTA ELEMENTLARINI ISHLATISH UCHUN MISOL

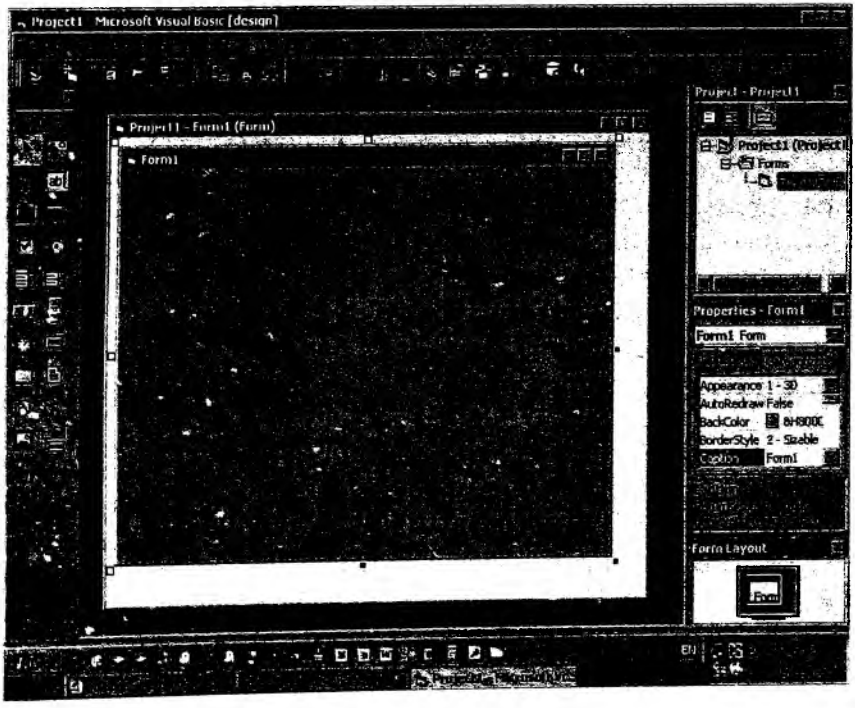
1-misol: "Salom olam" dasturi

Dasturlashning eskicha an'anasiga ko'ra Visual Basic ni o'rganishni ekranga "Salom olam" matnini chiqaradigan oddiy dasturni o'rganishdan boshlaymiz.

Biz Visual Basic da Matnli blokdan (**TextBox**) va ikkita buyruq tugmalaridan (**Command1**, **Command2**) tashkil topgan loyiha yaratamiz. "Matnni chiqarish" tugmasini bosganda "Salom olam" matni matnli blokda paydo bo'lishi va "Ishni yakunlash" tugmasi bosilganda dastur yakunlanishi kerak.

1. Visual Basic ni ishga tushiramiz va yangi standart ilova yaratamiz. Ekran ko'rinishi quyidagicha bo'ladi (8.8-rasm).

2. Boshqarish elementlari panelidan sichqonchaning chap tugmasini bosgan holda matnli blok tugmasini tanlaymiz.



8.8-rasm

Sichqonchaning ko'rsatkichini formaning ixtiyoriy nuqtasiga joylashtiramiz. Ko'rsatkich shu sababli krest shakliga o'tadi. Formada matnli blokni chizamiz (8.9-rasm).

3. Yuqoridagi usulda boshqarish elementlari panelidan "Buyruq" tugmasini (**Command Button**) bosamiz va uni bizning formamizda qayta chizamiz (8.10-rasm).

4. Yana bitta tugma yaratamiz (8.11-rasm).

5. Matnli blokning (**Text box**) xossasini o'zgartiramiz. Buning uchun matnli blok ob'yektiga sichqoncha yordamida bosamiz va ob'yektning xossalari oynasiga o'tamiz. **Text1** ob'yektining "Text" xossasini o'zgartiramiz (8.12-rasm).

Endi matnli blokda hech qanaqangi matn bo'lmaydi. Matn buyruq tugmasini bosgandan keyin oynaga chiqadi. Kerakli matnni buyruq tugmasining dasturiy kodiga yozamiz. Hozircha shrift va matn xarakteristikalarini o'zgartiramiz. Shriftni 2-Center - markaziga barobarlashtirish (Ochilib-yopiladigan ro'yxat oynasidagi strelkani bosish orqali kerakli band tanlanadi) bilan "Barobarlashtirish" (Alignment) xossasini o'rnatamiz. Shrift Font xossasida o'rnatiladi.

6. **Command1** buyruq tugmasining xossalarini o'zgartiramiz. Buning uchun **Command1** tugmasiga sichqoncha yordamida bosamiz va Caption xossasiga "Matnni chiqarish" jumlasi kiritiladi.

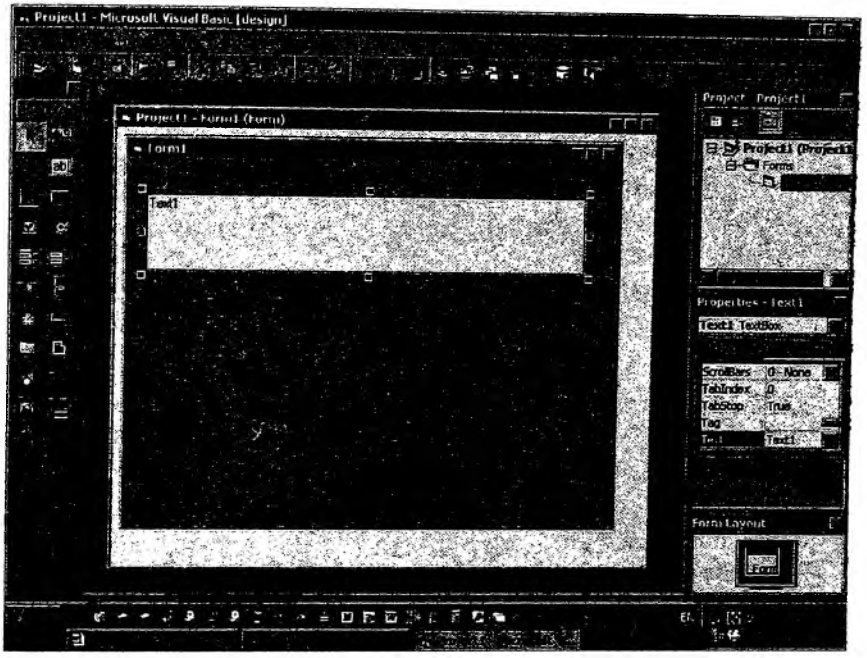
7. Ob'yektning kodlari oynasidagi (**Code**) **Private Sub** va **End Sub** qatorlar orasiga "Salom olam" jumlasini matnli blokga chiqaruvchi dastur matnini kiritamiz. Bu dastur matni **Text1** ob'yektining **Text** xossasini o'zgartiradi va quyidagi ko'rinishga ega bo'ladi: **Text1.Text = "Salom olam!"** (8.14-rasm).

7. **Command2** buyruq tugmasining xossalarini o'zgartiramiz. Buning uchun Caption xossasiga "Ishni yakunlash" jumlasi kiritiladi.

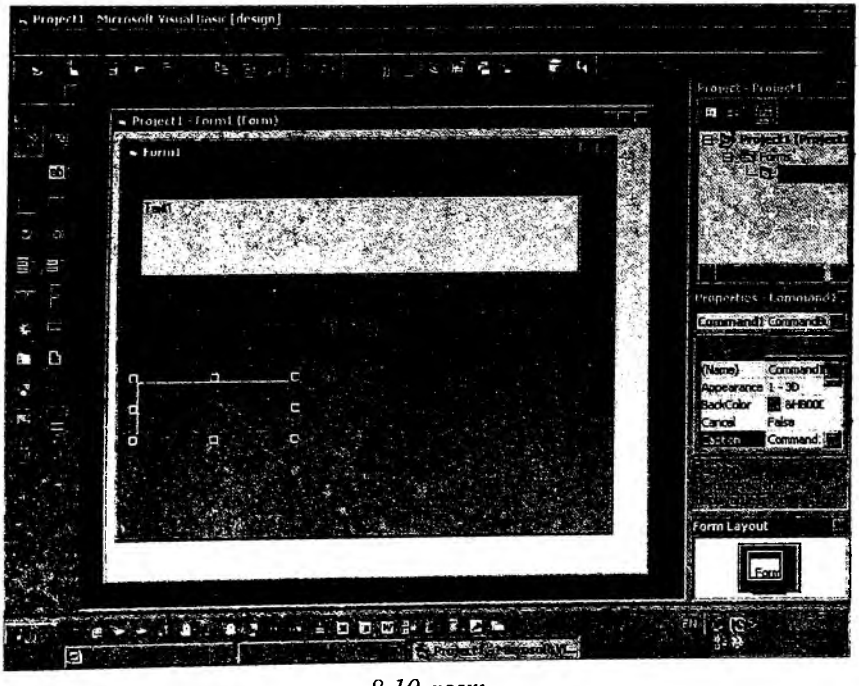
9. **Command2** ob'yektining kodlar oynasidagi (**Code**) **Private Sub** va End Sub qatorlar orasiga dastur ishini yakunlovchi **End** buyrug'ini yozamiz (8.16-rasm).

10. Dasturni ishga tushirish uchun asosiy menyudagi **Run** bo'limidan **Start** buyrug'i tanlanadi yoki asboblar panelidagi uchburchaklikli tugma bosiladi (8.17-rasm).

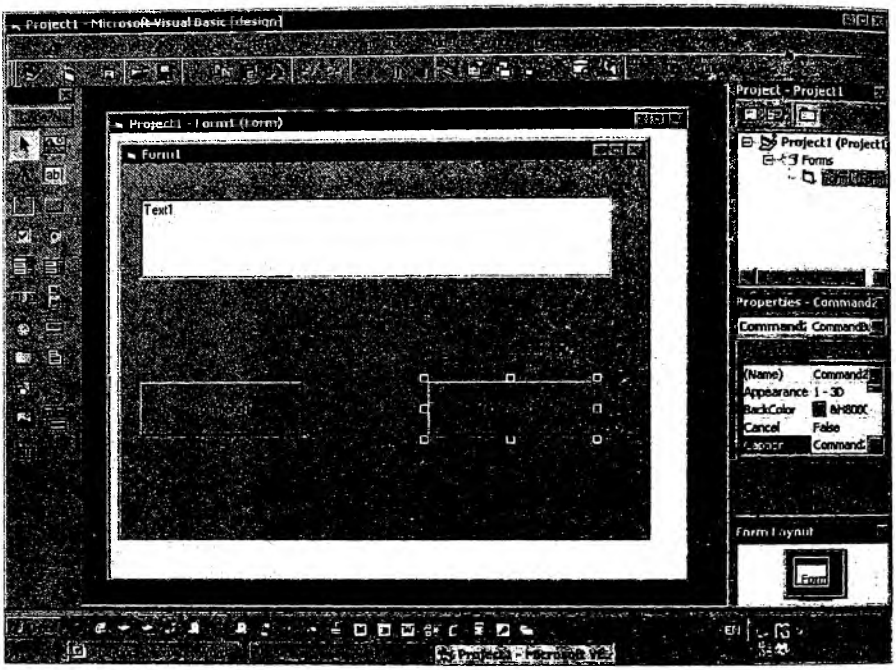
11. Ekranda biz yaratgan dastur ilovasi chiqadi. "Matnni chiqarish" tugmasini bosamiz va ekranda quyidagicha natijani olamiz (8.18-rasm).



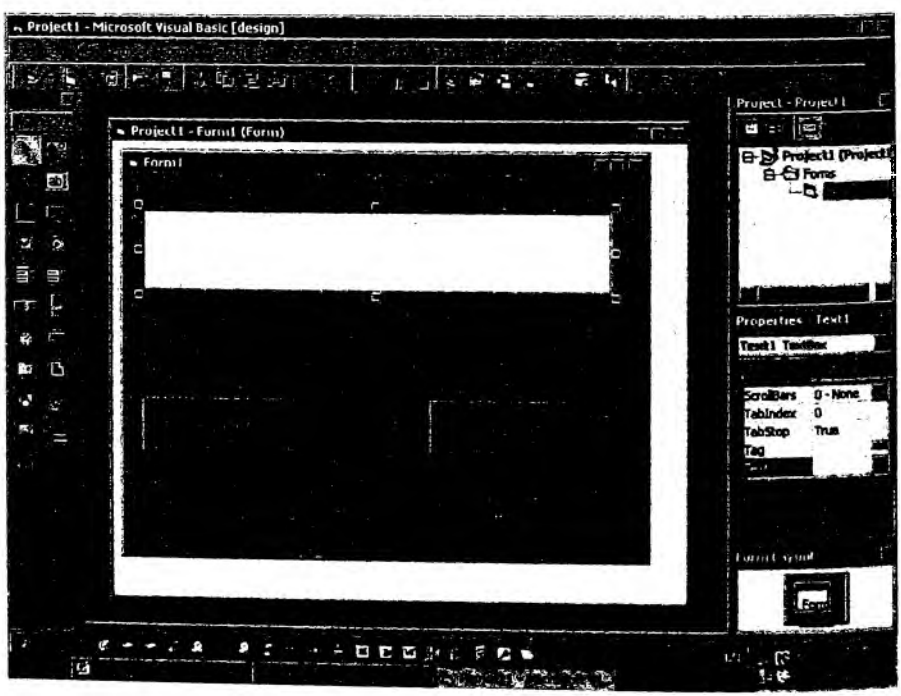
8.9-rasm



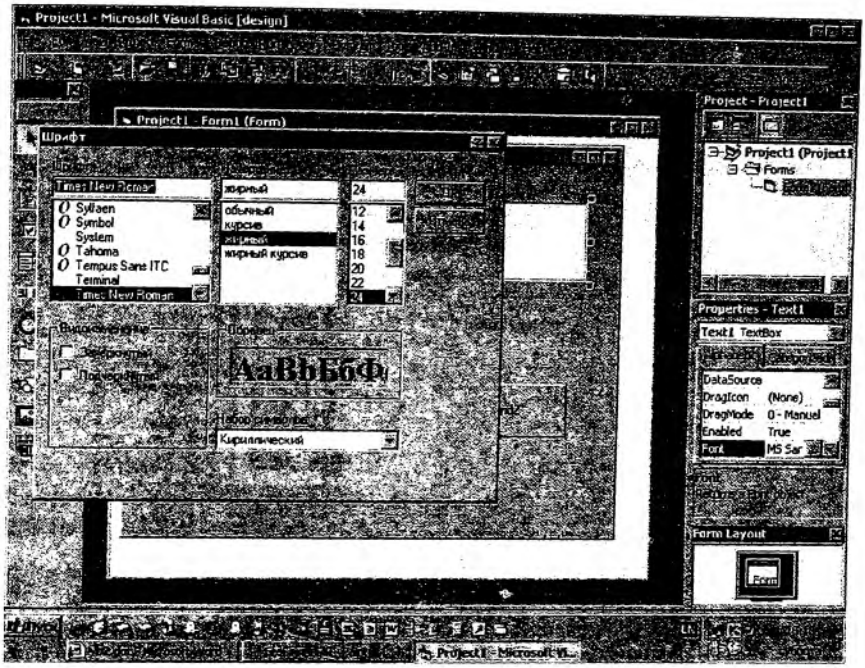
8.10-rasm



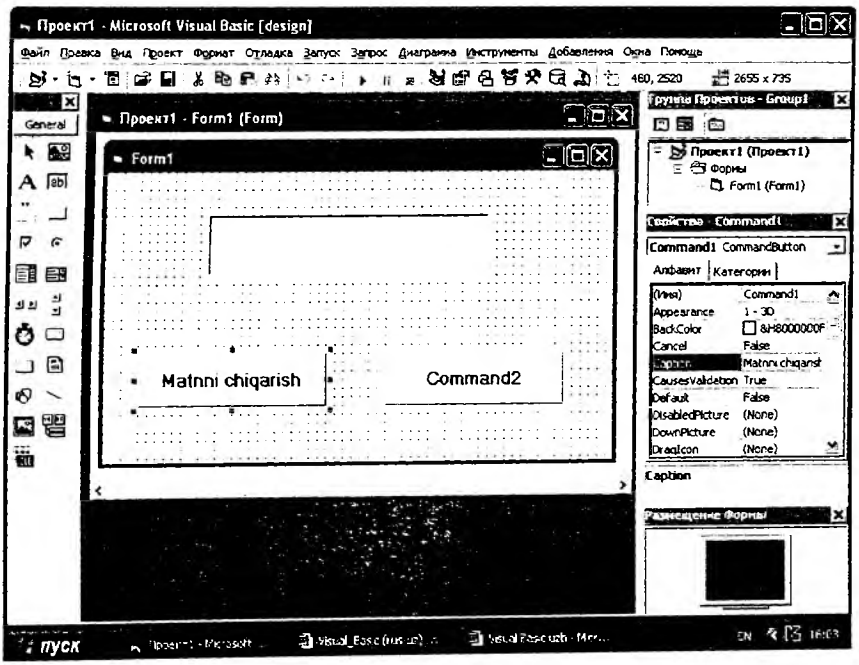
8.11-rasm



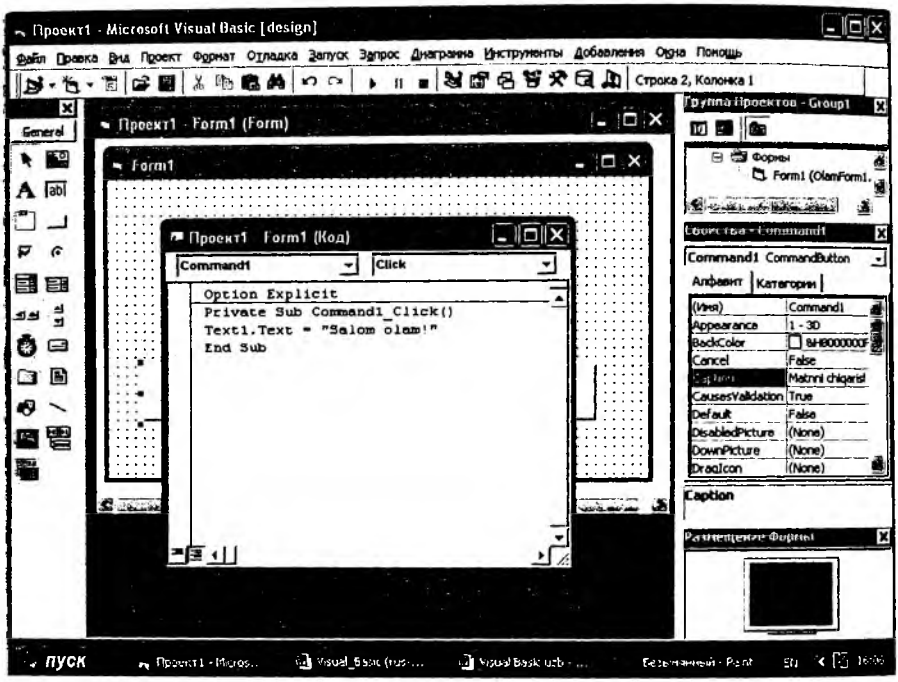
8.12-rasm



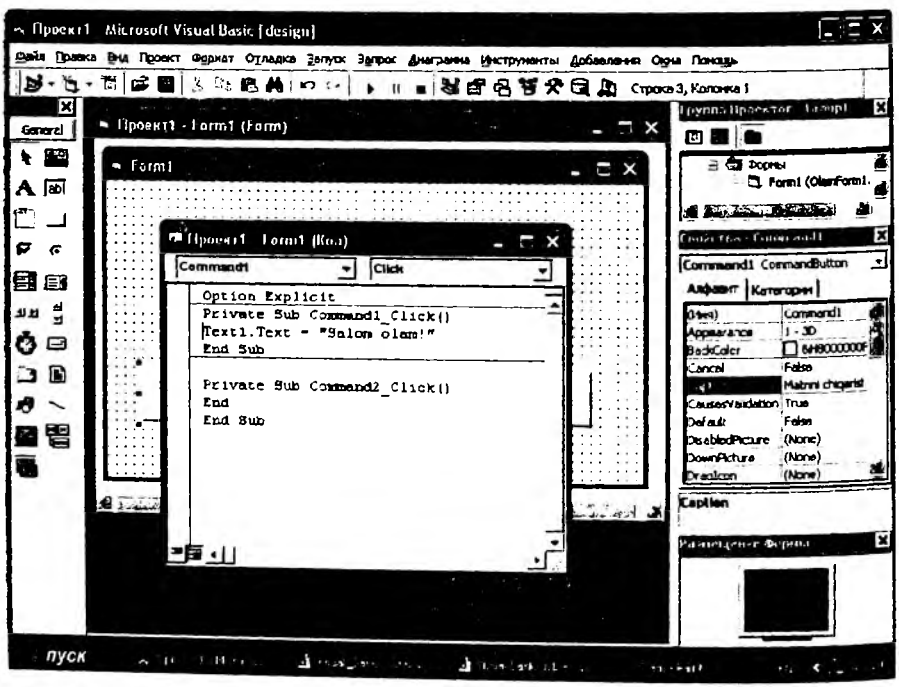
8.13-rasm



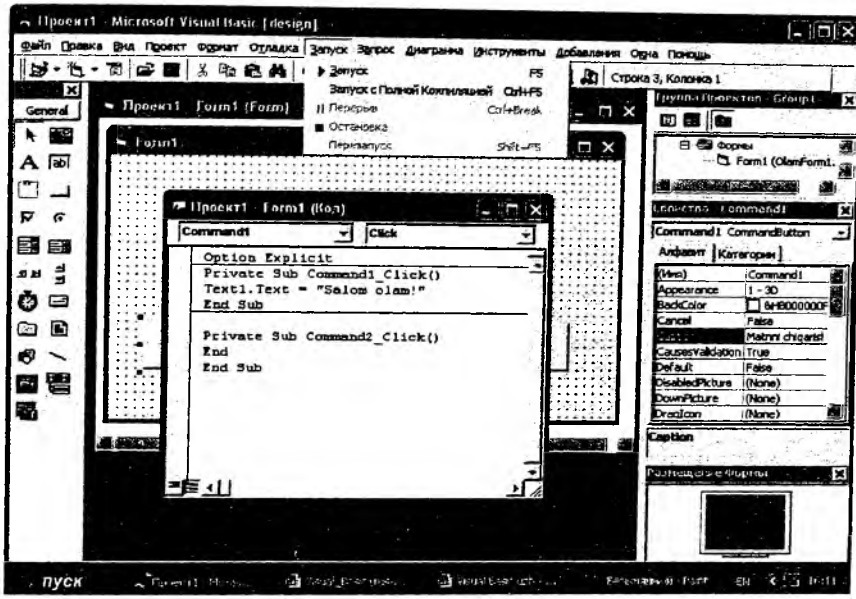
8.14-rasm



8.15-rasm



8.16-rasm



8.17-rasm

8.18-rasm

8-bo'limga oid nazorat savollari

1. Vizual Basic qanday ishga tushiriladi?
2. Vizual Basicning integrallashgan muhiti nimadan iborat?
3. Vizual loyihalash asoslari nimadan iborat?
4. Boshqarish elementlari panelini ishlatishga misollar keltiring?

9. O'ZGARUVCHILAR

Boshqa dasturlash tillari kabi VB tilida ham vaqtinchalik qiymatlarni saqlash, parametrlarni uzatish va hisoblashlar olib borish uchun o'zgaruvchilardan foydalaniladi. VB tilida o'zgaruvchilarni ta'riflash va ulardan foydalanish asosiy xususiyatlariga qisqacha to'xtalib o'tamiz. Odatda o'zgaruvchilarni ishlatishdan oldin, uni e'lon qilishadi, ya'ni shu Visual Basic va o'z dasturingizda o'zgaruvchilar nomlaridan foydalanishingiz haqida oldindan xabar berasiz, bunda o'zgaruvchida saqlanishi lozim bo'lgan ma'lumot turi ham e'lon qilinadi.

O'zgaruvchilar operativ xotirada ma'lumotlarni vaqtincha saqlash uchun o'zini rezervga olingan holda e'lon qilinadi. Har bir o'zgaruvchi shaxsiy nomlarga ega bo'ladi. O'zgaruvchiga qiymat o'zlashtirilgandan keyin biz uni va qiymatini dasturda ishlatishimiz mumkin.

O'zgaruvchilar **Dim** (dimension - o'lcham degan so'zdan kelib chiqqan) operatori yordamida e'lon qilinadi. **Dim** operatori o'zgaruvchi bilan operativ xotiraning aniqlangan sohasini rezervga oladi. O'zgaruvchini e'lon qilish uchun o'zgaruvchi nomini Dim operatoridan keyin yozish darkor. O'zgaruvchining nomidan keyin uning turini ko'rsatish maqsadga muvofiq keladi.

Masalan, Dim X - X o'zgaruvchisi turi ko'rsatilmasdan e'lon qilinadi. Dim X As Integer - X o'zgaruvchisining turi ko'rsatilgan holda ya'ni butun deb e'lon qiladi.

Visual Basic dasturlash muhitida ixtiyoriy turdagi ma'lumotlarni qabul qiladigan Variant turiga ega. Variant turiga mansub o'zgaruvchi foydalanishga juda qulay, lekin o'zgaruvchilarni aniq bir turda e'lon qilish operativ xotirani tejam ishlatilishiga imkon beradi va dasturning ishlashini tezlashtiradi.

9.1. O'ZGARUVCHINING NOMLARI

O'zgaruvchilar o'qishga qulay va birqancha ko'rgazmali bo'lishi uchun qandaydur ma'noga ega bo'lgan nomlar beriladi. O'zgaruvchilarga nom berishda bir nechta qoidalar mavjud:

- O'zgaruvchi nomi 255 dan oshmagan belgilardan iborat bo'lishi mumkin.

• O'zgaruvchi nomi ixtiyoriy harf va raqamlardan tashkil topishi mumkin.

• O'zgaruvchi nomining birinchi belgisi harf bo'lishi shart.

• O'zgaruvchi nomida probellar bo'lmasligi kerak.

• O'zgaruvchining nomi uzunligi unikal va ko'rish chegarasidan oshmasligi kerak.

Masalan, o'zgaruvchi nomlari quyidagicha bo'lishi mumkin: CurrentNum, Total, Date_of_birth. Navbatdagi o'zgaruvchi nomlari bo'lishi mumkin emas: 1Time, $Total, Date of birth.

Ma'lumotlarning turiga qarab o'zgaruvchi nomlarining prefikslari tavsiya qilinadi. Ular quyidagi jadvalda keltirilgan:

Ma'lumotlarning turi	**Prefiks**	**Misol**
Boolean	bin	binSuccess
Currency	cur	curPrice
Date	dtm	dtmFinish
Double	dbl	dblSum
Integer	int	intQuantity
Long	lng	lngTotal
Single	sng	sngLength
String	str	strLastname
Variant	vnt	vntValue

9.2. MA'LUMOTLARNING TURI

Visual Basic dasturiy muhitida quyidagi ma'lumotlarning asosiy turlarini ishlatishingiz mumkin:

- sonli (**Integer, Long, Single, Double, Currency**);
- qator (**String**);
- sana (**Date**);
- mantiqiy (**Boolean**);
- ixtiyoriy (**Variant**).

Ma'lumotlarning asosiy turlarining jadvali

Ma'lumotlarning turlari	O'lchami (baytlarda)	Qiymatlar diapazoni	Misollar
Integer (Butun)	2 bayt	-32 768 dan + 32 767 gacha	Dim intX as Integer intX = 23546
Long (Uzun butun)	4 bayt	-2 147 483 648 dan +2 147 483 647 gacha	Dim lngY as Long lngY = 1000000
Single (Bir karra aniqlikdagi siljuvchi nuqtali o'nlik son)	4 bayt	-3.402823E38 dan +3.402823E38 gacha	Dim sngA as Single sngA = 4.781
Double (Ikki karra aniqlikdagi siljuvchi nuqtali o'nlik son)	8 bayt	-1.79769313486 232D308 dan +1.79769313486 232aD308 gacha	Dim dblB as Double dblB = 1.00000000001
Currency (Pullik)	8 bayt	-922337203685 477.5808 dan +922337203985 477.5807gacha	Dim curM as Currency curM = 25456.20
String (Qator)	har bir belgi 1 bayt	1 dan 1 65535 belgigacha	Dim strS as String strS = "Basic"
Boolean (Mantiqiy)	2 bayt	True (Rost) yoki False (Yolg'on)	Dim binL as Boolean binL = True
Data (Sana)	8 bayt	January-1-100 (1.01.0100) dan December-31-9999 gacha	Dim dtmD as Data dtmD = #03-08-2005#
Variant (Variant)	16 bayt (son uchun) 22 bayt + bitta belgi 1 bayt (qator uchun)	Barcha turlar uchun	Dim vntV vntV = 13.45 yoki vntV = "Text"

9.3. O‘ZGARUVCHILARNI E’LON QILISH

Visual Basic dasturlash muhitida o‘zgaruvchilarni e’lon qilish quyidagi sintaksis bo‘yicha amalga oshiriladi:

Dim O‘zgaruvchiNomi [As Ma’lumotlar_turi]

Quyidagicha misollarni keltirish mumkin:

Dim bInSuccess As Boolean

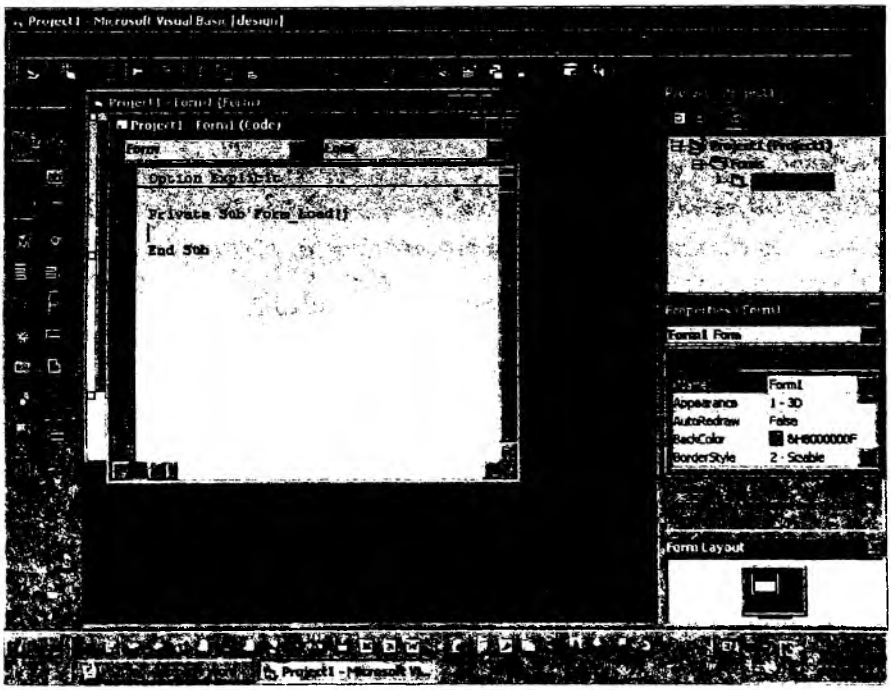
Dim strLastname As String, dblSum As Double

Fiksirlangan uzunlikdagi qatorlarni e’lon qilish uchun quyidagi sitaksis ishlatiladi:

Dim O‘zgaruvchi nomi As String * O‘zgaruvchi_uzunligi

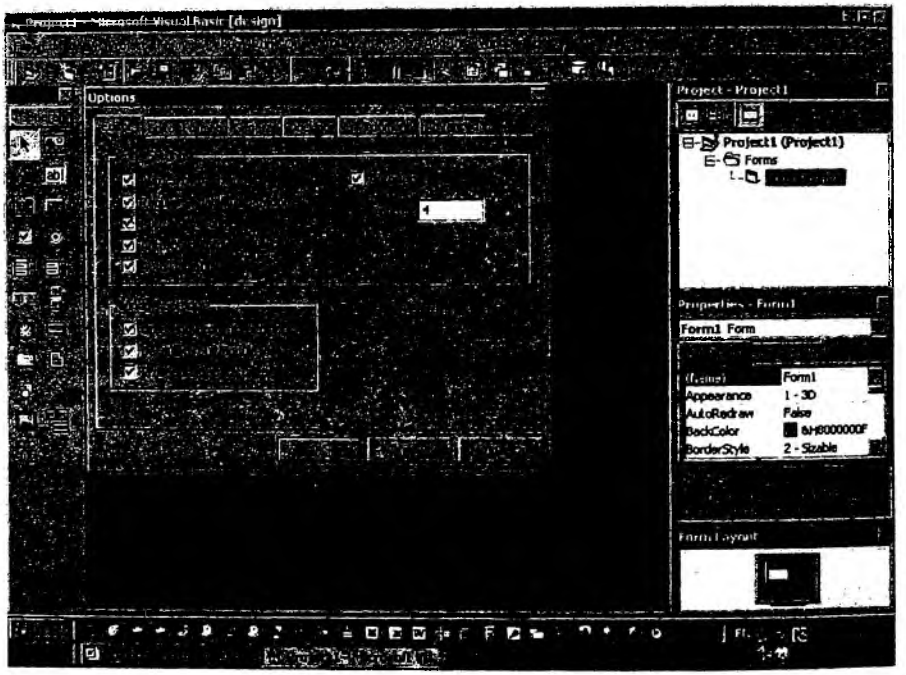
O‘zgaruvchi_uzunligi parametri belgilarning maksimal sonini ko‘rsatadi. (*) belgisi esa o‘zgaruvchi uzunligining fiksirlanganligini ko‘rsatadi.

O‘zgaruvchilarni aniq turda e’lon qilish orqali dastur translyatsiyasi rejimini o‘rnatish tavsiya etiladi. Buning uchun modulning boshida **Option Explicit** (Aniq turda e’lon qilish) operatorini kiritish kerak (9.1-rasm).



9.1-rasm

Bu operatorni Visual Basicning dastur oynasidagi barcha modullarga avtomat ravishda qo‘shish uchun **Tools** (Servis) menyusidagi **Options** (Parametrlar) buyrug‘i bajariladi. Options muloqot oynasi ochiladi va undagi **Editor** ilovasining **Require Variable Declaration** jumlasi to‘g‘risiga bayroqcha (flajok) o‘rnatiladi (9.2-rasm).



9.2-rasm

Agar Option Explicit operatorini modulga kiritmasangiz, unda siz o‘zgaruvchini noaniq e’lonidan foydalanishingiz mumkin. Bunday holatda o‘zgaruvchining turi birinchi o‘zlastirish operatoridayoq aniqlanadi va shu vaqtning o‘zida o‘zgaruvchiga xotiradan joy ajratiladi.

9.4. O‘ZGARUVCHIGA QIYMATLARNI O‘ZLASHTIRISH

Dasturda o‘zgaruvchini ishlatishdan avval unga qiymat berish zarur. O‘zlashtirishning eng sodda usuli "=" o‘zlashtirish operatorini ishlatishdan iborat. U quyidagi ko‘rinishga ega bo‘ladi:

O‘zgaruvchi = ifoda

O'zgaruvchi argumentiga o'zgaruvchi nomi beriladi va unga **ifoda** qiymati o'zlashtiriladi. Masalan,

sngFirst = 10
strLastname = "Ivanov"

Tenglik belgisidan o'ng tomonda faqat o'zgarmaslar joylashib qo'ymasdan bir qancha murakkab ifodalar ham joylashishi mumkin.

sngResult = sngFirst + 255
strName = "Иванов" & ": " & strTeam

O'zgaruvchining tavsifida ma'lumotlar turining ko'rsatmasi tushirib qoldirilishi mumkin. O'zgaruvchining turi bunda o'zgaruvchi nomining oxirgi simvoli bilan belgilanadi:@, =, %, &, !, $ (Currency, Double, Integer, Long, Single, String). Masalan, agar simvol $ qator ma'lumotlari turini aniqlash simvoli bo'lsa, u xolda text$ nomli o'zgaruvchi avtomatik xolda "simvollar qatori" "o'zgaruvchi turi" bo'lib qoladi

Keyinchalik bu maxsus simvol ma'lumotlar turining ko'rsatmasi tushirib qoldirilishi mumkin, lekin o'zgaruvchining nomida doimo turni aniqlash simvolining ishtirok etishi, qaysi ma'lumotlar turiga tegishli ekanligini eslatib turadi-bu esa bir-biriga zid ma'lumotlar turidan foydalanish xato qilmaslikka yordam beradi.

Agar oxirgi simvol yuqorida ko'rsatilganlarning hech biriga kirmasa va tur ko'rsatmalaridan foydalanilmasa, u holda o'zgaruvchilar ko'zda tutilgan ma'lumotlar turi Variantda belgilanadi, bu esa unda barcha ma'lumotlarni saqlashga imkon beradi.

Bir xil protsedurada bir-biridan faqat maxsus simvoli bilan ajralib turadigan o'zgaruvchilar tomonidan o'zgaruvchan oxirida foydalanish mumkin emas. Masalan, o'zgaruvchining barcha bir vaqtda ishlatilishi mumkin emas var$ va var%. Shuningdek, simvol aniqlash turi bo'lgan o'zgaruvchining nom oxirida e'lon qilinishi mumkin emas, tavsivchi yordamida As<tip peremennoy> (agar bu aniqlash oddiy simvol aniqlash turini qo'llashga qarshilik qilmasa). Masalan, agar siz xato haqida ma'lumot olsangiz, quyidagi barcha aniqlovchilarni kirgizib:

Dim var 1% as string
Dim var2% as integer

Protsedura yoki funksiyaning argumentlar ma'lumoti tipini aniqlash uchun protsedura yoki funksiyaning sarlavhasi qatorida ma'lumotlar tipi tavsifi foydalaniladi.

Masalan, protseduraning keyingi sarlavha satri uning parametrlarini o'zgaruvchining satr tipida tavsiflaydi:

Sub splitstr(str1 as string, str2 as string, str3 as string)

Ma'lumotlar tipini aniqlash qaytayotgan funksiyaning qiymati funksiyaning satrini yakunlaydi, masalan:

Function Find split space(strl as string)as integer

tavsiflaydi qaytayotgan funksiyaning qiymatini xuddi o'zgaruvchining kaltabutun tipi.

9.5. VARIANT TURIDAGI O'ZGARUVCHILARNI ISHLATISH AFZALLIKLARI

Variant turidagi o'zgaruvchilardan barcha turdagi ma'lumotlarni saqlash va amallarni bajarish uchun ishlatiladi. Ikkita holatni esdan chiqarmaslik kerak, ya'ni birinchidan, variant turidagi o'zgaruvchilar ustida arifmetik amallar va funksiyalarni faqat u sonli qiymatga ega bo'lgandagina ishlatish mumkin, ikkinchidan, qatorlarni konkatenatsiya qilishda "+" operatorining o'rniga & operatorini ishlatish kerak.

Variant turidagi o'zgaruvchilar **Empty**, **Null** yoki **Error** kabi maxsus qiymatlarga ham egadur.

• ***Empty qiymati***. **Variant** turidagi o'zgaruvchilarga 0 dan farqli qiymatlarni, bo'sh qatorni yoki **Null** qiymatini o'zlashtirish uchun **Empty** nomlanadi. **Empty** qiymatini aniqlash uchun **isEmpty** funksiyasini ishlatish mumkin:

```
If IsEmpty(x)  Then x= 0
```

• ***Null qiymati***. **Variant** turidagi o'zgaruvchilar ma'lumotlar bazasi bilan ishlaydigan ilovalardagi bo'sh ma'lumotlarni ko'rsatish uchun **Null** qiymatini ham ishlatish mumkin. **isNull** funksiyasi yordamida **Variant** turidagi o'zgaruvchi **Null** qiymatiga egaligini tekshiradi. **Variant** turidagi o'zgaruvchiga **Null** qiymatini o'zlashtirish uchun **Null** kalt so'zini ishlatish mumkin: y = Null **Variant** turidagi ega bo'lmagan o'zgaruvchiga **Null** qiymatini o'zlashtirish xato bo'ladi.

• ***Error qiymati***. **Variant** turidagi o'zgaruvchi proceduradagi hosil bo'lgan xatolar uchun Error qiymatini qabul qilishi mumkin. Lekin bu holatda ilova darajasidagi xatolar to'g'rilanmaydi. Ammo dasturchi uningh qiymatiga qarab aniq bir tuzatishlar kiritishi mumkin.

9.6. VARIANT TURIDAGI QIYMATLARNING ICHKI TASAVVURI

Variant turidagi o'zgaruvchilar unga saqlanadigan ma'lumotlarning ichki tasavvurini qo'llab-quvvatlaydi. **Variant** turidagi o'zgaruvchiga qiymatlar berishda Visual Basic muhiti bu qiymatlarning birqancha qulayliklarini qo'llaydi. Masalan, agar **Variant** turidagi o'zgaruvchiga kasr qismiga ega emas va katta bo'lmagan sonli qiymat o'zlashtirilgan bo'lsa, unda **Integer** tasavvuri ishlatiladi, agar kasr son o'zlashtirilgan bo'lsa, unda Double ichki tasavvuri ishlatiladi.

9.7. O'ZGARMASLAR

O'zgarmas deganda dasturning bajarilishi jarayonida o'zgarmaydigan ifoda elementi tushuniladi. Birnechta misollar keltiramiz:

- 142.34 - sonli o'zgarmas;
- 5.6E+4 - sonli o'zgarmas (5 600 ga teng);
- "Faylni o'qishda nosozlik" - simvolli o'zgarmas;
- #09/01/2005# - sana turidagi o'zgarmas;
- False - mantiqiy turdagi o'zgarmas.

9.8. O'ZGARMASLARNI E'LON QILISH

Dastur tezroq ishlashi uchun va kam xotira egallashi uchun, mumkin bo'lganda o'zgaruvchining konkret tiplaridan foydalanish tavsiya etiladi. Universal tip variant emas. O'zgaruvchining variant tipini qayta ishlash uchun faqat qo'shimcha xotira emas, balki qo'shimcha vaqt kerak bo'ladi;qaysi konkret ma'lumotlar tipiga qarashli ekanligini bu o'zgaruvchining qayta ishlash vaqtida talab qilinadi, kerak bo'lsa ma'lumotlar tipini o'zgartirish va bajarish lozim.

Nomlangan konstantalar tavsifi uchun operator Const qo'llaniladi o'zgaruvchi Dim tavsifi operatori bilan o'xshash. Bu operatorning sintaksisi:

Const<konstantanomi>(as<ma'lumotlar tipi>)=<ifoda>

<ifoda>-istalgan qiymat yoki konstanta sifatida foydalanishi kerak bo'lgan formula qaytayotgan qiymat. Masalan, keyingi operator butun konstanta tipini aniqlaydi:

const maxLen%=30
Const strErrorFile As String = "Faylni o'qishda nosozlik"

Xuddi o'zgaruvchidagi kabi konstantada ma'lmotlar tipining har xil qiymati bor, lekin bunda ular o'z qiymatini dastur bajarishda o'zgartirmaydi.

Agar siz programmangizda ma'lum bir konstantadan foydalanmoqchi bo'lsangiz, u xolda bu konstantaga ma'noli nom berish tavsiya etiladi va uni modul boshida tavsif etish, keyinchalik faqat nomlangan konstantalardan foydalanish tavsiya etiladi. Bu dasturni tushunarligina qilib qolmay, balki soddalashtiradi va sozlashda hamroh qiladi. Ko'pinchalik u yoki bu konstantaning qiymatini o'zgartirish talab qilinadi (hech bo'lmaganda sozlash davrida) va shunda faqatgina bitta qiymatni nomlangan konstantada o'zgartirish kifoya qiladi. Agarda dasturning matn kodida aynan shu qiymat foydalanilgan bo'lsa, unda u qiymatning hamma kelib chiqishini o'zgartirish ancha murakkab bo'ladi.

9.9. ASOSIY MATEMATIK OPERATORLAR

Matematik operatorlar sonlar ustida hisoblashlarni bajarishga imkon beradi.

Operatorlar	Bajariladigan amallar
+	Qo'shish
-	Ayirish
*	Ko'paytirish
/	Bo'lish
\	Butun sonli bo'lish
mod	Qoldiq
/\	Darajaga oshirish

Ustuvorliklar (Prioritetlar)

Matematik operatorlar ifodalarni kiritish uchun tavsiya qilingan. Ifoda bitta operatorga bog'liq bo'lgan o'zgaruvchilardan, qiymatlardan, funksiyadan tashkil topishi mumkin. Agar ifodada qavslar yo'q bo'lsa, operatorlar quyidagi tartib bo'yicha bajariladi:

1. Darajaga oshirish.

2. Ko'paytirish va bo'lish.
3. Butun sonli bo'lish.
4. Bo'linmadan olingan qoldiq.
5. Qo'shish va ayirish.

Ifodada hisoblash tartibini o'zgartirish uchun oddiy qavslar ishlatiladi. Masalan, (d-c*(a-b))/(a+b) formulasida avval a-b amali so'ng natijani c ga ko'paytmasi, olingan ko'paytma d dan ayirmasi, undan keyin esa a+b amali va nihoyat birinchi olingan natija ikkinchisiga bo'linadi. Ifodadagi qavslar soni chegaralanmagan, lekin shuni esda tutish kerak, ya'ni qavslar - juftli belgilar. Har bir ochilgan qavs uchun o'zining yopilgan qavsi ham bo'lishi shart.

Amallarning bajarilishi natijasining turi operandlarning turlariga bog'liq bo'ladi. Agar ikkala operand ham butun bo'lsa, qo'shish, ayirish, ko'paytirish va butun sonli darajaga oshirish amallari natijasi butun bo'ladi. Agar hech bo'masa bitta (yoki ikkita) operand haqiqiy bo'lsa, natija haqiqiy bo'ladi.

9.10. VISUAL BASICNING MATEMATIK FUNKSIYALARI

Matematik funksiyalar qiymatlar qaytaradi. Ular o'zgaruvchiga qiymat o'zlashtirilganda va ifodalarni hisoblashda ishlatilishi mumkin. n - funksiya argumenti va funksiya tarkibi sonli qiymatga ega bo'lishi yoki sonli turdagi o'zgaruvchi bo'lishi mumkin.

Funksiyalar	Bajaradigan vazifalari
Abs(n)	n ning absolyut qiymatini qaytaradi.
Atn(n)	n ning arktangensini radianlarda qaytaradi.
Cos(n)	n ning kosinus burchagini qaytaradi. n burchak radianlarda beriladi.
Exp(n)	n ning eksponentasini qaytaradi.
Rnd	0 dan 1 gacha bo'lgan intervaldagi tasodifiy sonni qaytaradi.
Sgn(n)	Agar n noldan kichik bo'lsa -1 ni, n nolga teng bo'lsa nolni va n noldan katta bo'lsa 0 ni qaytaradi.
Sin(n)	n ning sinus burchagini qaytaradi. n burchak radianlarda beriladi.
Sqr(n)	n dan chiqarilgan kvadrat ildizning qiymatini qaytaradi.
Tan(n)	n ning tangens burchagini qaytaradi. n burchak radianlarda beriladi.

9.11. SOLISHTIRISH OPERATORLARI

Solishtirish operatorlari shartli ifodalarda ishlatiladi. Shartli ifoda xossaning "rost" yoki "yolg'on" ligini dastur matnidagi boshqa bir elementining o'zgaruvchisi yordamida aniqlaydigan dastur operatorlarining bir qismi hisoblanadi. Masalan, **X>=0** ifodasi **True** qiymatini qabul qiladi, agar X o'zgaruvchisi musbat yoki nol qiymatiga ega bo'lsa, **False** qiymatini qabul qiladi, agar X o'zgaruvchisi manfiy qiymatiga ega bo'lsa.

Visual Basic ning solishtirish operatorlarining jadvali

Operatorlar	Vazifalari
=	Barobar
>	Katta
<	Kichik
<>	Teng emas
>=	Katta yoki teng
<=	Kichik yoki teng

9.12. MANTIQIY OPERATORLAR

True (Rost) qiymatini qabul qiluvchi yoki **False** (Yolg'on) qiymatini qabul qiluvchi ifodalar bulli ifodalar kabi bizga ma'lum, ammo **True** (Rost) yoki **False** (Yolg'on) natijalari esa bulli o'zgaruvchilariga yoki xossalariga o'zlashtiriladi. Bulli o'zgaruvchilar **As Boolean** tavsiflagichli Dim operatori yordamida e'lon qilinadi.

Visual Basic muhitining mantiqiy operatorlari jadvali

Operator	Amallar
And	Agar ikkala operandning qiymati **True** bo'lsa, **True** bo'ladi. Agar hech bolmaganda bitta operandning qiymati **False** bo'lsa, **False** bo'ladi.
Or	Agar hech bolmaganda bitta operandning qiymati **True** bo'lsa, **True** bo'ladi. Agar ikkala operandning qiymati **False** bo'lsa, **False** bo'ladi.
Not	Unar amal (bitta operand yordamida bajariladi) Agar operand **True** bo'lsa, **False** bo'ladi Agar operand **False** bo'lsa, **True** bo'ladi
Xor	Agar operandning qiymatlari o'zaro bir-biriga teng bo'lmasa (biri - **True**, ikkinchi - **False**), u holda True qiymatini oladi. Agar operandning qiymatlari o'zaro bir-biriga teng bo'lsa (ikkalasi ham - **True** yoki ikkalasi ham - **False**), u holda **False** qiymatini oladi.

9.13. QATORLAR USTIDA ISHLAYDIGAN FUNKSIYALAR

Qatorlar ustida ishlaydigan bitta amal - biriktirish amali yoki konkatenatsiya amali mavjud.

Konkatenatsiya amali & belgisi bilan belgilanadi. Ikkita qatorning konkatenatsiya amalining natijasi - biriktirilgan qator.

Masalan: Mayli uchta o'zgaruvchi strS1, strS2, strS0 dasturda quyidagi operatorlar yordamida tavsiflangan bo'lsin:

Dim strS1, strS2, strS0 As String

strS1 o'zgaruvchisining qiymati: strS1 = "Bizning kollej"

strS2 o'zgaruvchisining qiymati: strS2 = "Toshkentda joylashgan"

strS0 =strS1 & strS2 amallari bajarilgandan so'ng str S0 o'zgaruvchisining qiymati "Bizning kollej Toshkentda joylashgan" bo'ladi.

Visual Basic muhitida qatorlar ustuda bajariladigan amallar birnecha o'rnatilgan fuksiyalari yordamida amalga oshiriladi.

Qatorlar bilan ishlaydigan funksiyalar

Funksiya	Bajaradigan ishi
As C	Simvolning ASCII-kodini qaytaradi
Chr	ASCII-kodni simvolga o'tkazadi
InStr, InStrRev	Matn ichidan matnni qidirishni amalgam osiradi
LCase	Dastlabki qatordagi harf registrini kichigiga o'zgartiradi
Left	Qator boshidan simvollarning ko'rsatilgan sonini qaytaradi
Len	Qatordagi simvollarning umumiy sonini qaytaradi (qatorning joriy uzunligi)
LTrim, RTrim, Trim	Simvolli qatorning boshida, oxirida va ikkala tomonida joylashgan probellarni olib tashlaydi.
Mid	Qatorning ixtiyoriy joyidan berilgan sondagi simvollarini qaytaradi
Right	Qatorning oxiridan ko'rsatilgan sondagi simvollarini qaytaradi

Funksiya	Bajaradigan ishi
Str, CStr	Sonli qimatlarni qator turiga o'tkazadi
StrReverse	Qatordagi simvollarning joylashish tartibini teskariga o'zgartiradi
StrConv	Simvolli qatorning harflar registrini o'zgartiradi
Val	Qatorni son sifatida ifodalaydi
UCase	Dastlabki qatordagi harf registrlarini kattasiga o'zgartiradi

Bu funksiyalarning birnechtasini qarab chiqamiz:

• ***Str* funksiyasi. Str()** funksiyasi sonli qiymatlarni qator ko'rinishiga o'zgartiradi. Funksiyaning sintaksisi:

Str (Sonli ifoda). Sonli ifoda o'zgarmas, o'zgaruvchi, matematik operator va funksiya bo'lishi mumkin,

• ***Val* funksiyasi. Val()** funksiyasi simvolli qatorni sonli qiymatlarga o'zgartiradi. Funksiyaning sintaksisi: **Val (Simvolli qator)** Simvollar qatorini songa o'tkazishda barcha qatorda chapdan o'ngga qarab joylashgan raqamli simvollar hisobga olinadi. Qatorning boshida va oxiridagi probellar tashlab ketiladi. Qator ichiga probellar qo'yish mumkin emas. Agar ifodaning birinchi belgisi raqam bo'lmasa, Val funksiyasi nol qiymatini qaytaradi.

• **LTrim funksiyasi**. Simvolli qatorning boshidagi probellarni olib tashlaydi. Sintaksisi: **LTrim(Simvolli_ifoda)**

• **RTrim funksiyasi**. Simvolli qatorning oxiridagi probellarni olib tashlaydi. Sintaksisi: **RTrim(СимвольноеВыражение)**

• **Trim funksiyasi**. Simvolli qatorning boshida va oxiridagi probellarni olib tashlaydi.

Sintaksisi: **Trim(Simvolli_ifoda)**

• **Left funksiyasi**. Chap tomondan boshlab qatorni ajratib oladi.
Sintaksisi: **Left(Simvolli_ifoda, Simvollar_soni)**

• Right funksiyasi. O'ng tomondan boshlab qatorni ajratib oladi.
Sintaksisi: **Right(Simvolli_ifoda, Simvollar_soni)**

• **Mid funksiyasi**. Ixtiyoriy qismqatorni ajratib oladi. Sintaksisi: **Mid(Simvolli_ifoda, Pozitsiya_tartibi, Simvollar_soni)**

• **UCase funksiyasi**. Kichik harflarni katta harflarga almashtiradi. **UCase ()** funksiyasi **Variant** turidagi qiymat qaytaradi. **String** turidagi qiymat qaytarishi uchun **UCase$()** funksiyasini ishlatish zarur.

Funksiya Sintaksisi: **UCase (Simvolli_ifoda)**

• **LCase funksiyasi**. Katta harflarni kichik harflarga almashtiradi. **LCase ()** funksiyasi **Variant** turidagi qiymat qaytaradi. **String** turidagi qiymat qaytarishi uchun **LCase$()** funksiyasini ishlatish zarur.

Funksiya Sintaksisi: **LCase (Simvolli_ifoda)**

• **StrConv funksiyasi**. Kichik yoki katta harflardagi ifodani shaxsiy nomiga almashtiradi.

Funksiya Sintaksisi: **StrConv (Simvolli_ifoda)**

• **InStr funksiyasi**. Simvolli qator ichidan boshqa bir qatorni qidirishni amalga oshiradi.

Sintaksisi: **InStr (Dastlabki_qator, Qidirilayatgan_qator)**

9.14. MASSIVLAR

Massiv - bu o'zgaruvchi, unda bir vaqtda bir tipdagi bir necha qiymatlar saqlanadi. U o'zini bir tipdagi indeksi o'zgaruvchilar yig'indisi hisoblanadi. Massivning qo'llaniladigan indeksi soni har xil bo'lishi mumkin. Ko'pinchalik bir yoki ikki indeksli massivlar qo'llaniladi, ba'zan -uch indeksli, undan ko'p sonli indekslar kam uchraydi. VB 60tagacha indekslar qo'llashi mumkin. Massivning indekslar soni massivning razmerini belgilaydi. Bir indeksli massiv bir o'lchovli massiv deyiladi. Ikki indeksli -ikki o'lchovli va hokazo.

Katta sonli massivlar katta xotirani egallaydi, shuning uchun ularni qo'llashda ehtiyot bo'lish kerak. Massiv qo'llashdan avval uni operator Dim yordamida e'lon qilish lozim va unda massivda qiymatlar qaysi tipda saqlanishini ko'rsatish kerak. Hamma qiymatlar massivda bir tipdagi ma'lumotlar turiga qarashli bo'lishi majburiy. Bu chegaralashni amaliyotda chetlab o'tish mumkin, buning uchun massiv e'lon qilinganda Variant tipidan foydalanish mumkin, bunda massiv elementlari har xil tipdagi qiymatga ega bo'lishi mumkin.

Quyida massivni e'lon qilish operatorining sintaksisi:

Dim<massiv nomi>(<ko'lam1>, <ko'lam2>,...)
As<ma'lumotlar turi>

Bunda qavsda ko'rsatilgan kattaliklar <ko'lam1>, <ko'lam2> va hokazolar massiv kattaligini - indekslar soni shu indeks uchun

maksimal mumkin bo'lgan qiymatni beradi. Bunda massivning elementlar indeksatsiyasi o'rnatilgan bo'yicha noldan boshlanadi. Shunda **Dim Arrayl(9) as integer** ni e'lon qilinishi butun tipdagi o'zgaruvchi bo'lgan 10 elementdan iborat bir o'lchovli massivni aniqlaydi.

Dim Array2(4, 9) as Variant ning e'lon qilinishi esa Variant universal tipidagi o'zgaruvchi bo'lib (5*10) ellik elementdan iborat ikki o'lchovli massivni aniqlaydi. Pastki chegaralar standart qiymati sifatida mumkin bo'lgan qiymatlar indeksi uchun faqatgina nol qo'llanishi mumkin. Bu standart qiymatni operator Option Bass yordamida o'zgartirish mumkin. Agar modul boshiga Option Bass 1 operatorni joylashtirsak, u xolda massiv elementlari indeksatsiyasi o'rnatilgan bo'yicha noldan emas, birdan boshlanadi.

Massiv e'lon qilinganda faqatgina indeksning yuqori chegarasini emas, balki uning pastki chegarasini ham ko'rsatish mumkin, ya'ni massivning konkret indeksining diapazon o'zgarishi beriladi, bunda chegara ixtiyoriy butun son bo'lishi mumkin, nomanfiy bo'lishi shart emas.

Mana bu aniqlashning sintaksisi:

Dim<massiv nomi>(<min1>to<maks1>,...) As <ma'lumotlar tipi>

Masalan, agar siz meteorologik ma'lumotlar massivi bilan ishlamoqchi bo'lsangiz, oxirgi ikki xafta ichidagi o'rtacha kunduzgi haroratni ko'rsatuvchi, u xolda massivning quyidagi aniqlanishi qulay bo'ladi:

Dim Temperature(-14 to 0) As Single

Bunda, masalan temperature(-2) o'tgan kungi haroratga to'g'ri keladi, kerakli indeksni aniqlash uchun esa sizni qiziqtirayotgan kun uchun kerakli sana farqidan foydalanish kifoya.

Yuqorida ko'rilgan misollarda gap massivning mustahkam o'lchovlari haqida borgan edi, unda elementlar soni massivning operator Dim tavsifida aniq ko'rsatilgan. Bunday massivlar statik massiv deyiladi.

VB da dinamik massivlar ham qo'llanilishi ruxsat etiladi, bunda ularning o'lchamlari tavsifi qayd etilmaydi. Dinamik massiv o'lchamlari muayyan dastur bajarilayotganda o'rnatishili mumkin.

Dinamik massiv aniqlanayotganda operator Dim da massiv nomidan so'ngbo'sh qavslar turadi va o'zgaruvchan tipi tavsifi keladi. Indekslar soni va ular o'zgarishining diapazoni berilmaydi. Lekin massiv foydalanilishidan oldin, operator ReDim bajarilishi kerak, u dinamik massiv indeksining o'lchamlari va diapazon o'zgarishini beradi.

Dinamik massivning e'lon va o'lchamlari aniqlashining sintaksisi:

Dim<massiv nomi>() As <ma'lumotlar tipi >
ReDim<massiv nomi>(<ko'lam1>, <ko'lam2>,...)

Dinamik massivda e'loni, o'lchamlarini aniqlash va foydalanish, o'lchamlari va kattaligi o'zgarishi quyidagicha bo'lishi mumkin:

```
Dim dArray()As Variant
ReDim dArray(1, 2)
dArray(0, 0)=2
dArray(0, 1)=3
k=dArray(0, 0)+dArray(0, 1)
ReDim dArray(k)
dArray(0)="Stroka1"
```

Bu masalada massiv dArray boshida xuddi ikki o'lchamli olti elementdan iborat massivdek aniqlanadi so'ng boshqatdan xuddi bir o'lchamli massivdek aniqlanadi, bunda yuqori chegara indeksi o'zgaruvchi k ning qiymatida beriladi.

Massivning shu paytdagi yuqori va pastki chegaralarini aniqlash uchun **LBound** va **UBound** funksiyalaridan foydalanish mumkin, albatta.

Ko'zda tutilgan holda massivning o'lchamlari o'zgarganda unga yangidan xotira ajratiladi va uning elementlarining qiymatlari yo'qoladi.

Massivning joriy qiymatini yo'qotmaslik uchun uning o'lchovlari o'zgartirilganda Preserve so'zi ishlatiladi. Masalan, massivning dArray o'lchamini bir elementiga oshirish uchun bor elementlarning qiymatini yo'qotmagan xolda, quyidagidek bajarish mumkin.

```
ReDim Preserve dArray(UBound(dArray)+1)
```

9.15. PROTSEDURALAR VA FUNKSIYALAR, ULARNING VB CHAQIRILISHI VA PARAMETRLARNING UZATILISHI

VBda programmaning asosiy komponentlari protseduralar va funksiyalar hisoblanadi. Ular programma kodining qismi hisoblanadi, operatorlar Sub va EndSub Visual Basic yoki Function va EndFunction oralarida tuzilgan bo'ladi. VB protsedurasi quyidagi ko'rinishda bo'lishi mumkin:

Sub<protsedura nomi>(<argument1>, <argument2>,...)
<operatorVisual Basic1><operatorVisual Basic2>EndSub

funksiyaning protseduradan farqi shundaki, uning nomi o'zgaruvchi sifatida chiqadi va funksiyaning qaytishi chaqirish nuqtasi ma'nosida foydaliniladi. Bu shunday ko'rinishda bo'lishi mumkin:

Function<funksiya nomi>(<argument1>, <argument2>,...)
<operatorVisual Basic1><operatorVisual Basic2>
<funksiya nomi>=<kaytariluvchiQiymat >
EndFunction

Quyida ikki protsedura tavsifi berilgan:

```
Sub Main()
a=10
b=20
c=30
callExample(a, b, c)
call msg Box(a)
call msg Box(b)
call msg Box(c)
endSub
sub example(x, By val y, By RetZ)
x=x+1
y=y+1
z=z+1
call msg box(x)
call msg box(y)
call msg box(z)
endsub.
```

Yordamchi protsedura example foydalanadi, formal argumentlar sifatida uchta o'zgaruvchi har xil tavsif etilgan.

Agar loyihada har xil protsedura bir xil nom bilan bo'lsa, u holda nomini aniqlashtirish uchun protsedura chaqirilganda quyidagi sintaksis qo'llanadi:

<Modul nomi><Protsedura nomi>

Agar bu holda modul nomi bir necha so'zdan iborat bo'lsa, bu nomni kvadrat qavslarga olish kerak. Masalan, agar modul "Grafik protseduralar"-deb atalsa, protsedura-"Krestcha" deb atalsa, chaqiriq quyidagi shaklda bo'lishi mumkin:

[Grafik protseduralar].Krestcha

Boshqa loyihalarda joylashgan protseduralardan ham foydalanish mumkindir. Bu xolda nomni aniqlashtirish yana bir darajasi kerak bo'ladi

<Loyiha nomi><Modul nomi><Protsedura nomi>

9.16. O'ZGARUVCHINING VA KONSTANTANING HARAKAT SOHASI

Hamma protseduralar, funksiyalar, o'zgaruvchilar va konstantalar VBda o'z harakat sohasiga ega.

Bu ularning faqat dastur kodining ma'lum bir joyida - ayni ular tavsifida foydalanishi mumkinligini bildiradi.

Masalan, agar o'zgaruvchi A operator Dim yordamida tavsif etilgan bo'lsa, protsedura tanasida Proc1 nomi bilan aynan shu protsedura uning harakat sohasi hisoblanadi.

Shunday qilib, agar boshqa protsedura Proc2 bo'lsa, u xolda siz bunda shu o'zgaruvchidan foydalana olmaysiz. Agar siz shunga harakat qilib ko'rsangiz u xolda hato haqida tavsif qilinmagan o'zgaruvchidan foydalanish uchun ma'lumot olasiz (agar operator Option Explict-dan foydalanilsa) yoki boshqa o'zgaruvchini olasiz - xuddi shu nom bilan, lekin bir xil nomli o'zgaruvchi, birinchi protsedura bilan hech qanday bog'liqligi yo'q.

Dasturning qaysi joyida va aynan qanday tavsif etilgan o'zgaruvchi, buni uning harakat sohasi aniqlaydi va u qancha davr xotirada va o'zining qiymatini saqlaydi. O'zgaruvchining harakat sohasini aniqlashda uchta har xil daraja mavjud:

1. Protseduraning darajasi
2. Modul darajasi
3. Loyiha darajasi

Protsedura darajasida o'zgaruvchini aniqlash uchun uning tavsif shu protsedura tanasiga joylanadi, shunda bu lokal o'zgaruvchi bo'ladi

Protsedura darajasida o'zgaruvchini aniqlash va uni shu bilan birga tushunishni oson qilib, birgalikda modulning protseduralarida ishlatilishi uchun uning tavsifini modulning e'lonlar bo'limida biror bir protsedura yoki funksiya matni oldidan tanlanadi. Bunda yaqqol harakat sohasining yaqqol tavsifidan ham foydalanish mumkin. Din kalit so'zi **Private** kalit so'zi bilan almashtiriladi. Qaysi tavsifdan foydalanishning farqi yo'q va nihoyat o'zgaruvchining loyiha darajasida tavsif qilish uchun uning tavsifini modul loyihaining biron-bi e'lonlar bo'limida joylashtirish kerak va albatta kalit so'zi **Public** foydalanilishi shart. Shunday qilib, tavsif etilgan o'zgaruvchila loyihaning ixtiyoriy modulida ishlatilishi mumkin.

Yuqorida aytilganlarning hammasi harakat sohasining konstan va massivlarining tavsifiga va aniqlanishiga kiradi.

O'zgaruvchilar darajasini o'zgartirmaydigan, lekin o'zgaruvch qiymatini saqlab qoladigan protsedura darajasida tavsif qilingan protseduraning ishi tugagandan keyin o'zgaruvchilar uchun ula tavsifining yana bir usuli bor. Buning uchun tavsifchi statik o'z garuvchiligini aniqlab **Static**dan foydalanishi lozim. Bundan o'zgaruvchi o'zi uchun va o'zining qiymatini o'zi tavsif qilingan va foydalangan protsedura tamom bo'lgandan so'ng ham ajratilgan joyn xotirada saqlab qoladi. Shunga qaramay statik o'zgaruvchi boshq protseduralarda foydalanilmaydi. Uning faqatgina hayot davri o'zga radi, harakat sohasi emas (Agar shu protsedura qayta chaqirig'i bo'lsa)

Agar statik o'zgaruvchi tavsif etilgan protsedura qayta chaqirilsa oldingi chaqiriqda, ish yakunida qay holda bo'lsa unda bu o'zgaruvch o'zining oldingi qiymatini saqlab qoladi. Oddiy, statik bo'lmagan o'zgaruvchi har gal o'zgaruvchini tashabbusga chorlaydi va pro tseduraga kirishda bo'sh qiymat oladi.

9.17. PROTSEDURA VA FUNKSIYANING HARAKAT SOHASI

Protsedura va funksiyalar faqat ikki darajali harakat sohasiga ega: Modul darajasi va loyiha darajasi.

Loyiha darajasi yashirincha foydalaniladi. Shunday qilib, bu loyihada protsedura yoki funksiya ixtiyoriy boshqa protsedura yoki funksiya orqali chaqirilishi mumkin.

Protsedura va funksiya tavsifida loyiha darajasida nomajburiy kalit so'zi **Publicdan** foydalanish mumkin. Bu so'zning bor yoki yo'qligi ta'sir ko'rsatmaydi.

Agar faqat modul darajasida protsedura tavsif etilishi kerak bo'lsa buning uchun kalit so'zi **Private** qo'llaniladi. E'tiborga oling, bunday tavsif faqatgina protseduraning harakat sohasini toraytiribgina qolmay, balki va uning mustaqil protsedura sifatida foydalanishni taqiqlaydi, uni faqat boshqa protseduradan chaqirish mumkin. Nihoyat, protsedura yoki funksiyaning tavsifida kalit so'zi **Static** dan foydalanish mumkin. Bu hech ham shu protsedura yoki funksiyaning ichida tavsif qilingan protsedura harakat sohasiga ta'sir qilmaydi, lekin hamma o'zgaruvchiga ta'sir qiladi.

Bu holatda hamma lokal o'zgaruvchilar Static statusini oladi va shu bilan protsedura tugagandan so'ng uning xotirasida qoladi va chaqirilganda avvalgi qiymatini saqlab qoladi.

9-bo'limga oid savollar

1. Vizual Basicda ma'lumotlar tarkibi va turlarini qanday sinflarga bo'lish mumkin?
2. Vizual Basicda o'zgaruvchilar qanday e'lon qilinadi?
3.) Vizual Basicda o'zgaruvchilarga qiymatlarni o'zlashtirish qanday amalga oshiriladi?
4. Variant turidagi o'zgaruvchilarni ishlatish afzalliklari nimadan iborat?
5. Variant turidagi qiymatlarning ichki tasavvurini tushuntirib bering.
6. O'zgarmaslar va ularni e'lon qilish qanday amalga oshiriladi?
7. Asosiy matematik operatorlar.
8. Vizual Basicning matematik funksiyalari.
9. Solishtirish va mantiqiy operatorlar.
10. Qatorlar bilan ishlash funksiyalari.
11. Massivlar.
12. Protseduralar va funksiyalar.

10. DASTURNI BOSHQARISH VA NAZORAT QILISH TUZILISHI

10.1. IZOHLAR

Izoh dastur matniga tushuncha berish uchun ishlatiladi. Dastur matniga izoh kiritish uchun qatorning boshiga yoki izoh kiritilayotgan matn boshiga (') belgisi kiritiladi. Bu belgidan keyin kelgan ixtiyoriy mant izoh deb qabul qilinadi, ya'ni Visual Basic ularni transliyatsiya qilmaydi. Masalan,

```
' Qator boshidan boshlanuvchi izoh
X= Sin(a) + Cos(b) ' Bu operatordan keyingi izoh
```

10.2. OPERATORNI BIRNECHTA QATORGA JOYLASHTIRISH

Agar operator katta uzunlikga ega bo'lsa, uni qatorning davom etish belgilaridan foydalanib, bir nechta qatorga bo'lish mumkin. Bu belgilar probel va pastki chiziq (_) hisoblanadi.

Masalan, familiya, ism va sharflarni birlashtiruvchi qatorni bir nechta qatorga joylashtiramiz:

```
strName = strFamaly & strName & strFatherName
```

Buni quyidagicha yozish mumkin:

```
strName = strFamaly _
& strName & strFatherName
```

10.3. BIR NECHA OPERATORLARNI BIR QATORGA JOYLASHTIRISH

Qoida bo'yicha dasturni yozishda operatorlar alohida qatorga joylashtiriladi. Agar operatorlar unchalik katta bo'lmagan uzunlikga ega bo'lsa, Visual Basic ularni bitta qatorga oralarida ikki nuqta bilan yozishga imkon beradi.

Masalan: **x = a + b : y = c - d : z = a*b**

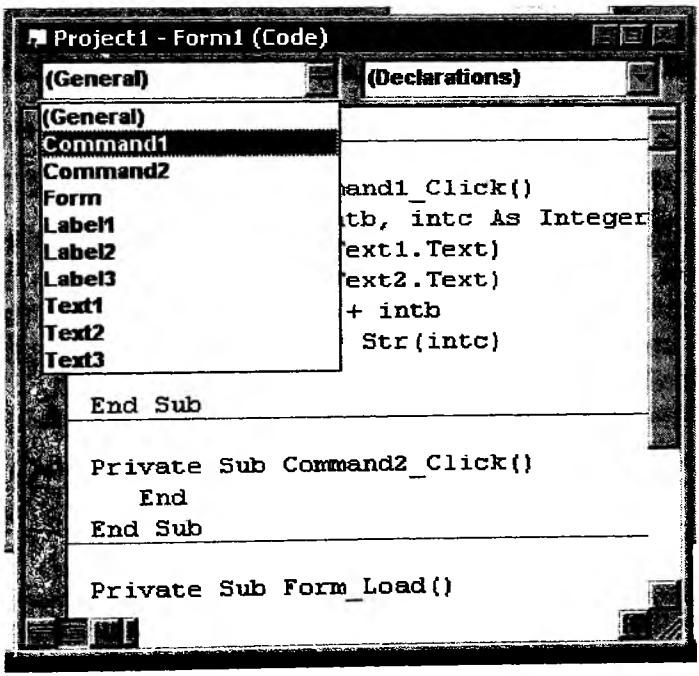
10.4. KODNI TAHRIRLASH

Visual Basic muhitida dastur kodini kiritish uchun kod tahrirlagichidan foydalaniladi. Uni ishga tushirish uchun **Project Explorer** oynasidagi kodini kiritiladigan formaga yoki ob'yektga kursorni o'rnatish kerak va quyidagi amallardan biri bajariladi:

- **View** (Ko'rinish) menyusidagi **Code** (Kod) buyrug'ini tanlash;
- tanlab olingan ob'yekt ustida sichqonchaning o'ng tugmasini bosganda hosil bo'lgan kontekstli menyudan **View Code** buyrug'ini tanlash;
- ob'yektning ustida sichqonchaning chap tugmasini ikki marta bosish.

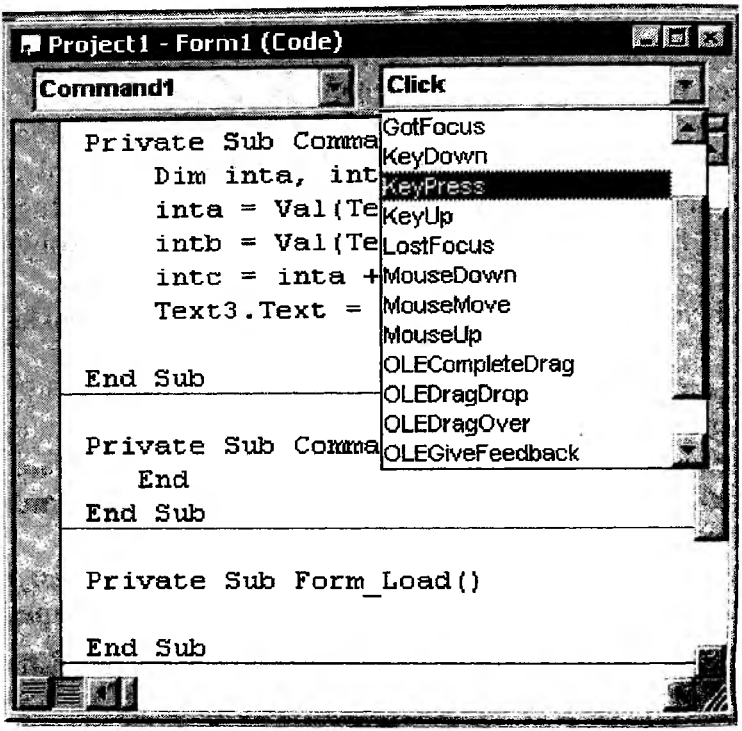
Bu amallarning bittasi bajarilganda dastur kodi kiritiladigan tahrirlagich oynasi ochiladi. Visual Basicning har bir moduli uchun ichki seksiyalarga ajratilgan alohida kod oynasi yaratiladi. Kod tahrirlagich oynasidan seksiya tanlash **Object** ro'yxati yordamida amalga oshiriladi.

Protsedura yaratish uchun ob'yekt tanlash: tahrirlovchi oynadagi **Object** (Ob'yekt) ro'yxatidan ob'yektlarning bittasi tanlanadi.



10.1-rasm

Procedure ro'yxatidan kerakli hodisa (Event) tanlanadi.



10.2-rasm

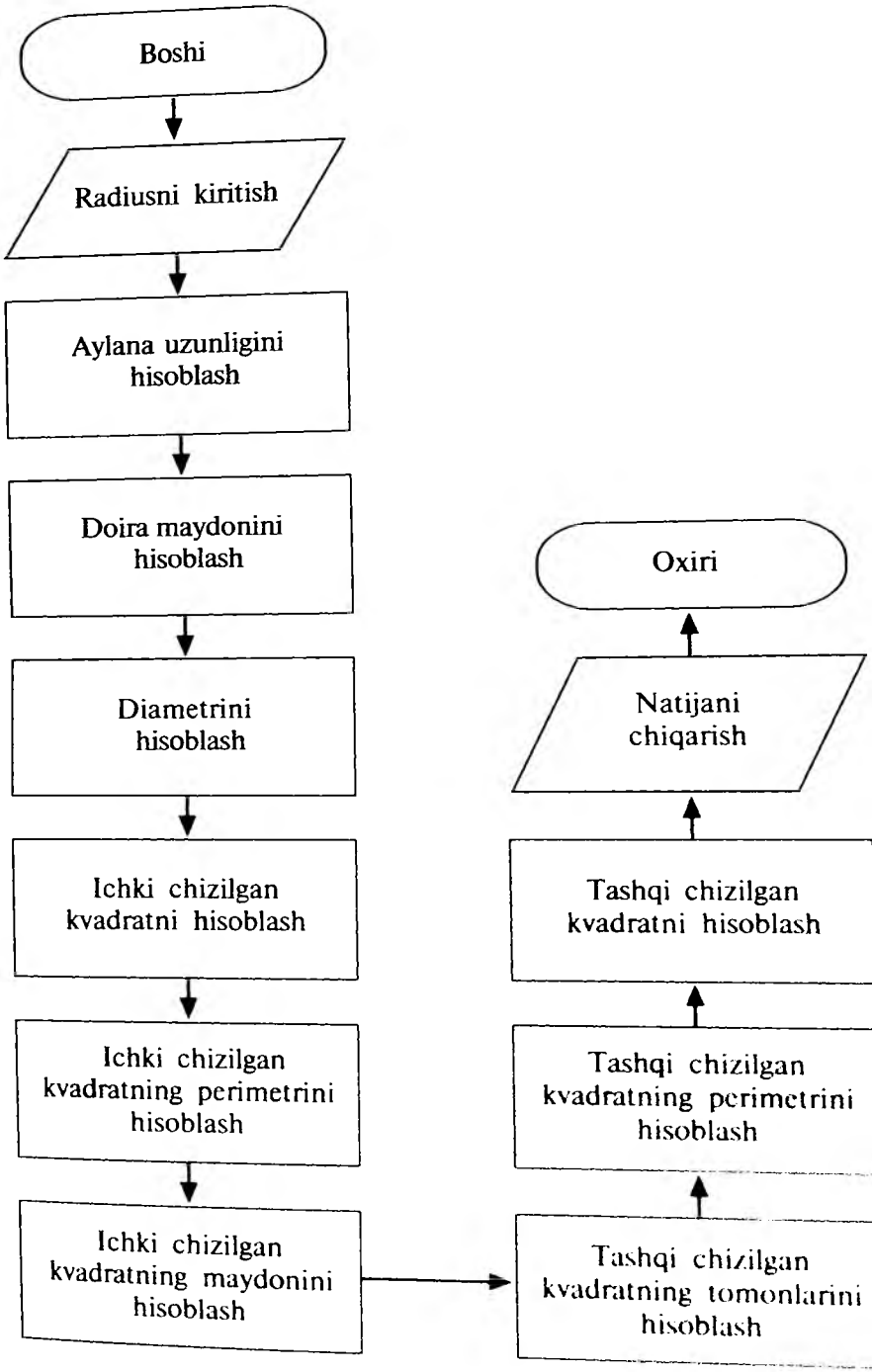
10.5. CHIZIQLI (KETMA-KET) ALGORITMLAR

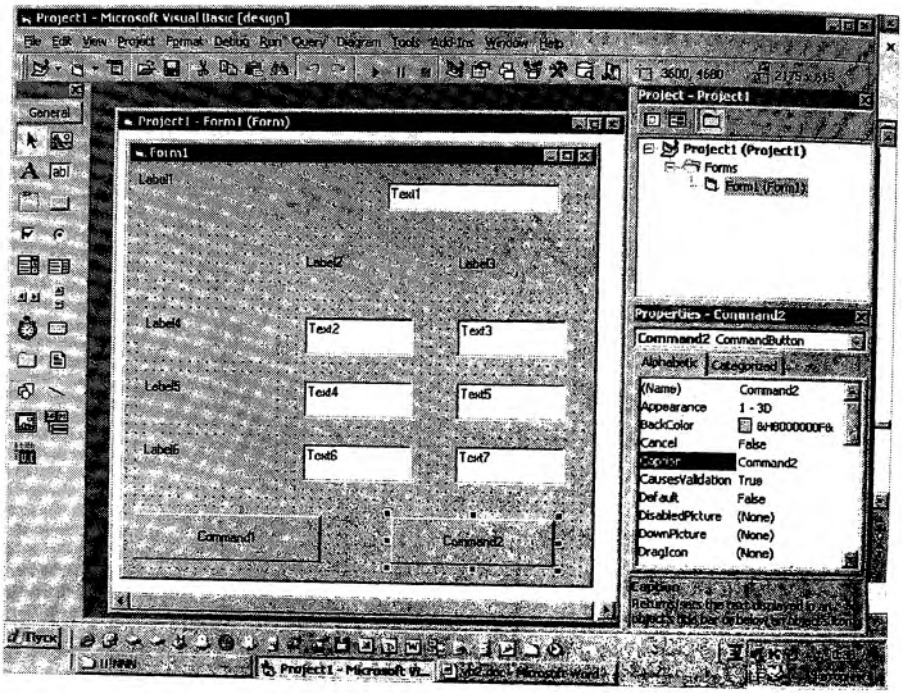
Ketma-ket algoritmlarda dasturning har bir qadami ketma-ket, birin-ketin bajariladi. Blok-sxemada ham ular qat'iy tartibda ketma-ket amallarning bajarilishini ko'rsatadi. Agar ular graf shaklida bo'lsa, unda u ketma-ket tugunlar ko'rinishida bo'lib, har bir tugundan kelasi tugunga kiradigan faqat bitta yoy chiqadi.

2-misol: Chiziqli algoritmli dastur

Aylananing radiusini ciqarish. Aylana uzunligini, doira yuzini, aylanaga ichki va tashqi chizilgan kvadratning yuzasini va perimetrini aniqlash.

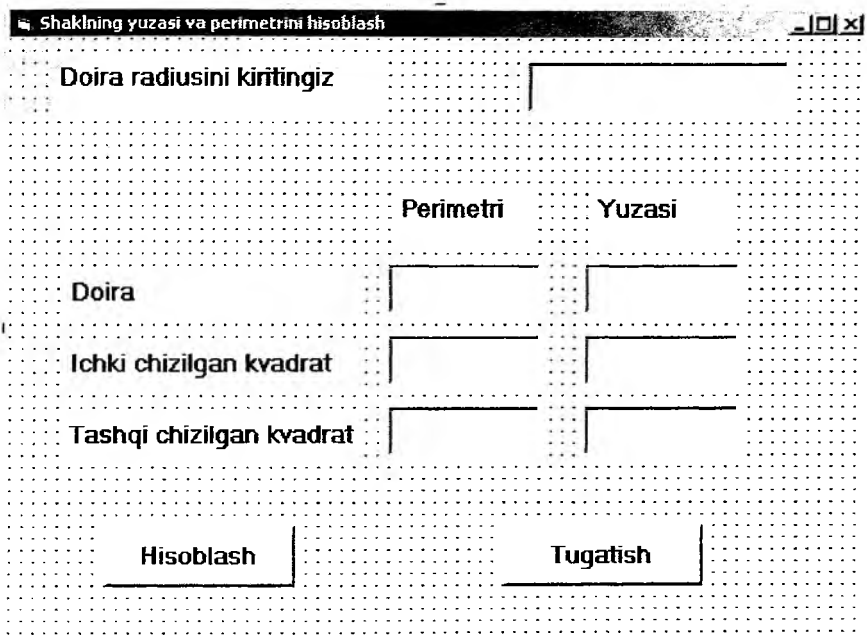
Blok-sxemasi





10.3-rasm

Ob'ekt	Xossa	O'rnatilgan qiymatlari
Form1	Caption	Shaklning yuzasi va perimetrini hisoblash
Label1	Caption	Doira radiusini kiritingiz
Label2	Caption	Perimetr
Label3	Caption	Maydon
Label4	Caption	Doira
Label5	Caption	Ichki chizilgan kvadrat
Label6	Caption	Tashqi chizilgan kvadrat
Command1	Caption	Hisoblash
Command2	Caption	Tugatish
Text1,	Text	Text xossasi maydonini tozalash
Text2,		
Text3,		
Text4,		
Text5, Text6		
Text7		

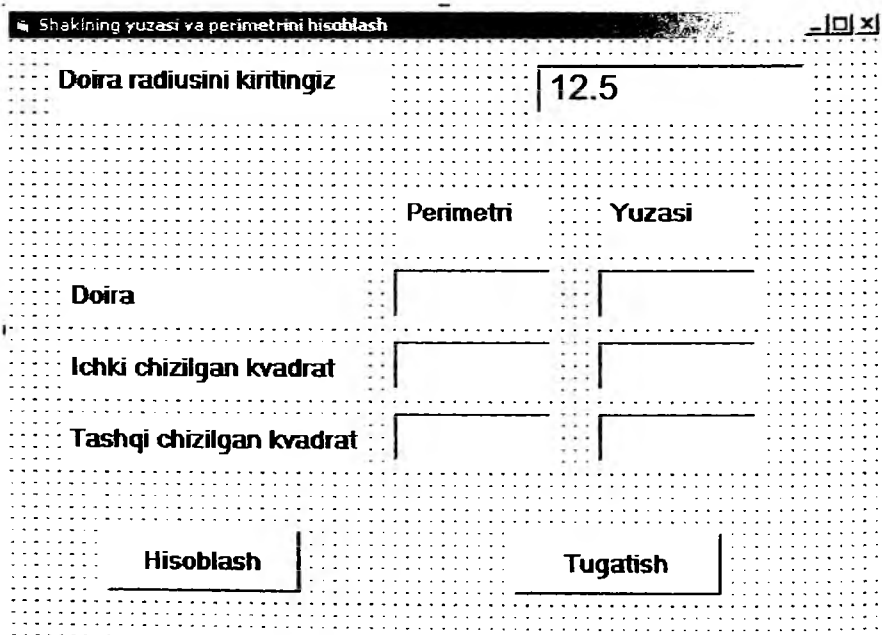


10.4-rasm

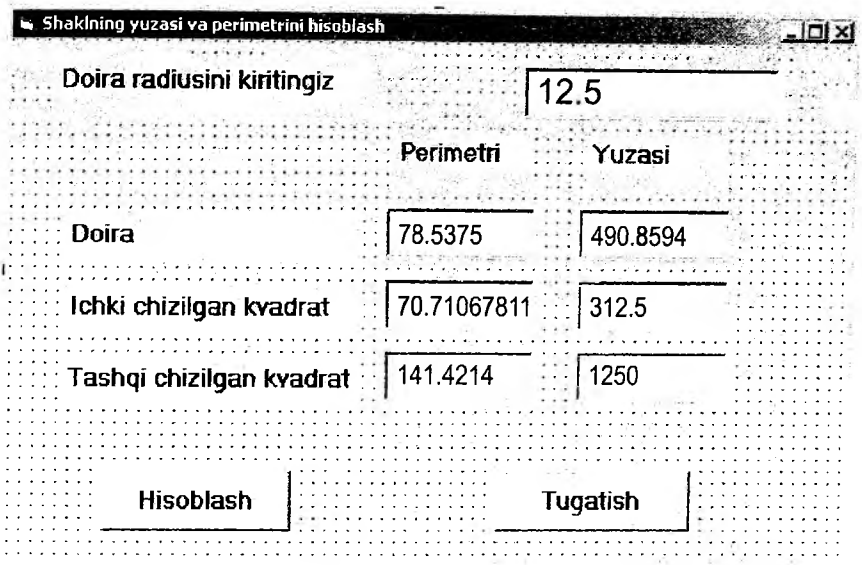
Dastur kodi

```
Option Explicit
Private Sub Command1_Click()
Const Pi As Single = 3.1415
Dim sngR As Single
Dim sngD, sngAV, sngAO As Single
Dim sngC1, sngC2 As Single
Dim sngKV1, sngKV2 As Single
Dim sngKO1, sngKO2 As Single
sngR = Val(Text1.Text) ' Radiusni aniqlash
sngC1 = 2 * Pi * sngR ' Aylana uzunligi
sngC2 = Pi * sngR ^ 2 ' Doira maydoni
sngD = 2 * sngR ' Diametr
sngAV = sngD / Sqr(2) ' Ichki chizilgan kvadratning tomoni
```

```
sngKV1 = 4 * sngAV ' Ichki chizilgan kvadratning perimetri
sngKV2 = sngAV ^ 2 ' Ichki chizilgan kvadratning maydoni
sngAO = sngD * Sqr(2) ' Tashqi chizilgan kvadratning tomoni
sngKO1 = 4 * sngAO ' Tashqi chizilgan kvadratning perimetri
sngKO2 = sngAO ^ 2 ' Tashqi chizilgan kvadratning maydoni
Text2.Text = Str(sngC1)
Text3.Text = Str(sngC2)
Text4.Text = Str(sngKV1)
Text5.Text = Str(sngKV2)
Text6.Text = Str(sngKO1)
Text7.Text = Str(sngKO2)
End Sub
Private Sub Command2_Click()
End
End Sub
```



10.5-rasm



10.6-rasm

10.6. IF, GO TO, SELECT BOSHQARUV TUZILMALARI

If va Select boshqaruv tuzilmalari **boshqaruv operatorlari yoki yechim qabul qiluvchi konstruktsiyalar** deb ataladi. Shu bois ular dastur operatorlarining odatdagi ketma-ket bajarilishini o'zgartiradi. Bu konstruktsiyalar ishlatilmaganda dastur birinchi operatordan oxirgi operatorgacha ketma-ket bajariladi. Yechim qabul qiluvchi operatorlarini qo'llagash dasturda hosil bo'lgan shartlarga boi'liq holda aniqlangan amallarni bajarishga imkon beradi.

3-misol: Tarmoqlanuvchi dastur

Cho'kayotgan kemadan seyfni chiqarib olish masalasi. Agar illyuminator R radiusli doira shakliga, seyf AxBxC o'lchamdagi to'g'riburchakli parallelopiped shakliga ega bo'lsa, seyfni chiqarib olish mumkinmi yoki yo'qmi, shuni tekshirishni dasturlash.

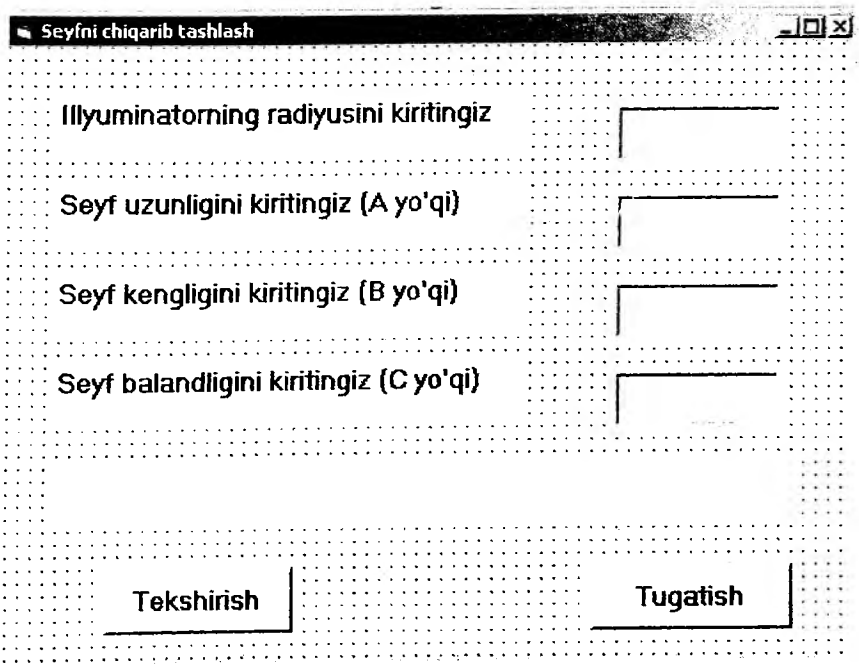
Agar uning eng kichik tomini diagonali illyuminator diametridan kichik bo'lsa seyfni chiqarib olish mumkin. Shu bois parallelepipedning yoqlari mos ravishda AxB, AxC va BxC bo'lsa, ularning ichidan diagonali illyuminator diametridan kichik bo'lgan minimal juftligini aniqlash yetarli. Buning uchun parallelopipedning yoqlari o'sish tartibida joylashtiriladi.

Algoritm

1. R, A, B, C larni kiritish.
2. A qiymati A,B,C qiymatlari ichidagi minimali ekanligini tekshirish.
3. Agar minimal bo'lsa, B ning C dan kichikligini tekshirish.
4. Agar kichik bo'lsa, A,B,C ketme-ketligini aniqlash.
5. Agar yo'q bo'lsa, A,C,B ketme-ketligini aniqlash.
6. Agar yo'q bo'lsa, B ning A,B,C qiymatlari ichida minimalligini tekshirish.
7. Agar minimal bo'lsa, A ning C dan kichikligini tekshirish.
8. Agar kichik bo'lsa, B, A, C ketma-ketligini aniqlash.
9. Agar yo'q bo'lsa, B, C, A ketma-ketligini aniqlash.
10. Agar yo'q bo'lsa, A ning B dan kichikligini tekshirish.
11. Agar kichik bo'lsa, C, A, B ketma-ketligini aniqlash.
12. Agar yo'q bo'lsa, C, B, A ketma-ketligini aniqlash.
13. Parallelopipedning kichik diagonali illyuminatorning diametridan kichikligini tekshirish.
14. Agar kichik bo'lsa, "Seyfni chiqarish mumkin" degan xabarni ko'rsatish.
15. Agar yo'q bo'lsa, "Seyf illyuminatordan chiqmaydi" degan xabarni ko'rsatish.

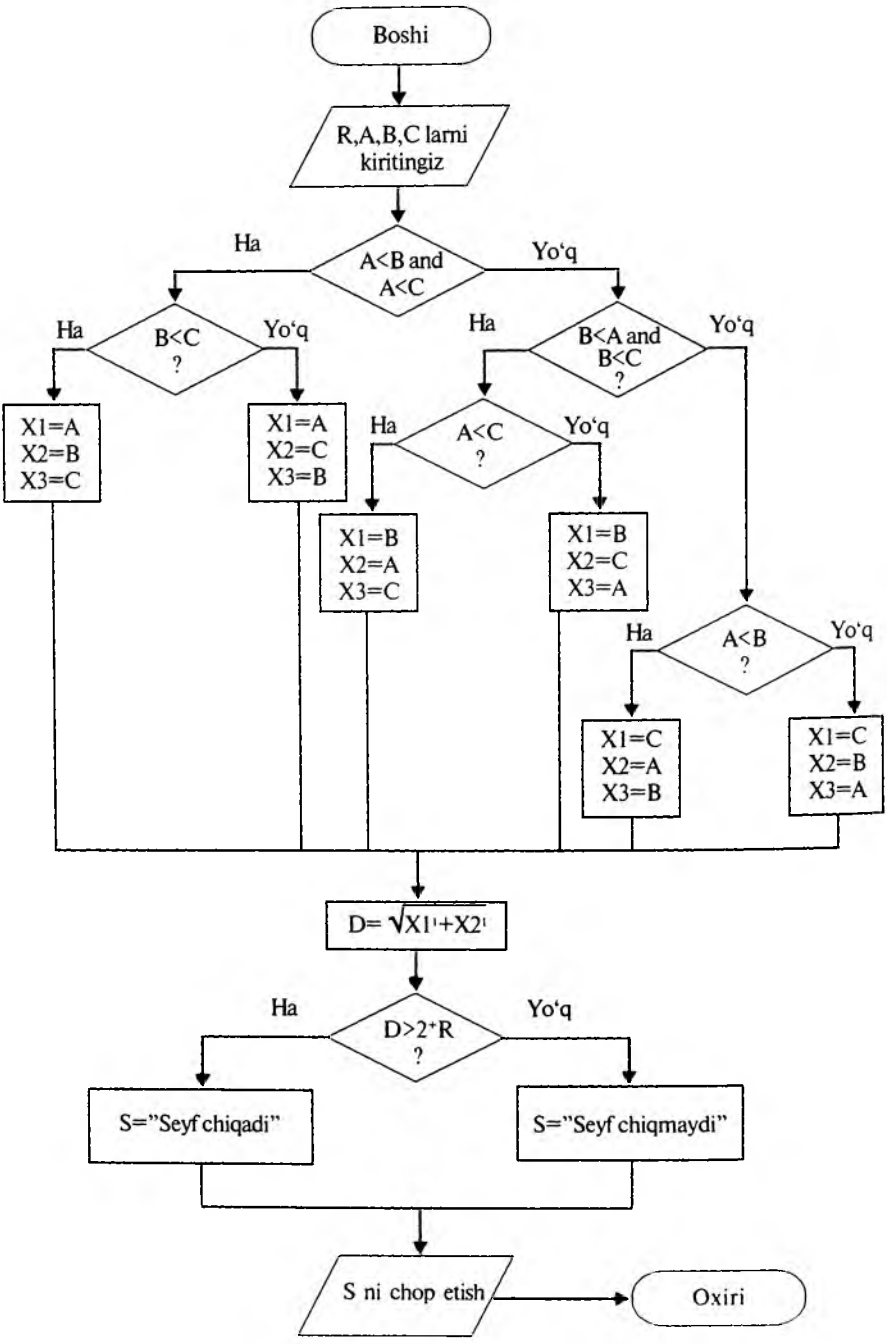
10.7-rasm

Ob'yekt	Xossa	O'rnatilgan qiymatlari
Form1	Caption	Seyfni chiqarib tashlash
Label1	Caption	Illyuminatorning radiyusini kiritingiz
Label2	Caption	Seyf uzunligini kiritingiz (A yo'qi)
Label3	Caption	Seyf kengligini kiritingiz (B yo'qi)
Label4	Caption	Seyf balandligini kiritingiz (C yo'qi)
Label5	Caption	Caption xossasi maydonini tozalash
Command1	Caption	Tekshirish
Command2	Caption	Tugatish
Text1, Text2 Text3, Text4	Caption	Text xossasi maydonini tozalash



10.8-rasm

Blok-sxemasi



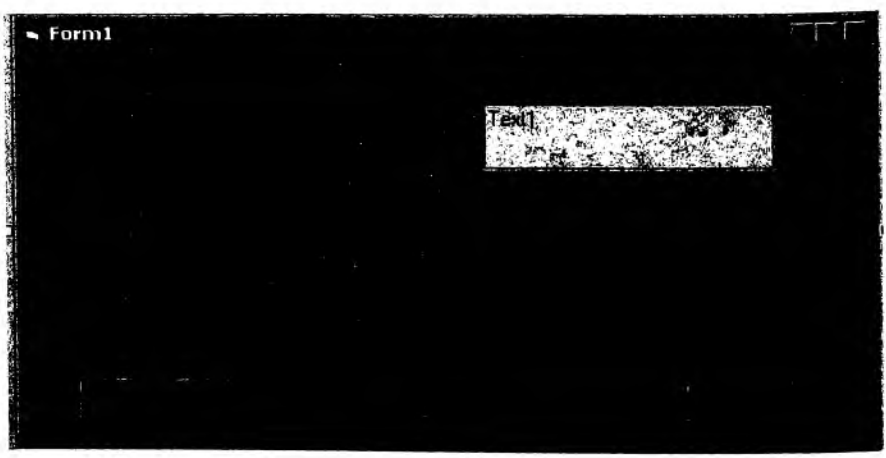
Dasturning kodi

```
Option Explicit

Private Sub Command1_Click()
    Dim R, A, B, C As Single
    Dim X1, X2, X3, D As Single
    Dim S As String
    R = Text1.Text
    A = Text2.Text
    B = Text3.Text
    C = Text4.Text
    If (A < B) And (A < C) Then
            If B < C Then
                    X1 = A
                    X2 = B
                    X3 = C
            Else
                    X1 = A
                    X2 = C
                    X3 = B
            End If
            ElseIf (B < A) And (B < C) Then
            If A < C Then
                    X1 = B
                    X2 = A
                    X3 = C
            Else
                    X1 = B
                    X2 = C
                    X3 = A
      End If
      Else
            If A < B Then
                    X1 = C
                    X2 = A
                    X3 = B
```

```
    Else
                X1 = C
                X2 = B
                X3 = A
    End If
    End If
    D = Sqr(X1 * X1 + X2 * X2)
    If D > 2 * R Then
        S = ""
    Else
        S = ""
    End If
    Label5.Caption = S
End Sub

Private Sub Command2_Click()
    End
End Sub
```

4-misol: Goto tuzilmasini ishlatish

Butun son kiritish. Uning juftligini tekshirish. Agar son juft bo'lsa, "Juft son" jumlasini chiqarish, agar son toq bo'lsa, "Toq son" jumlasini chiqarish.

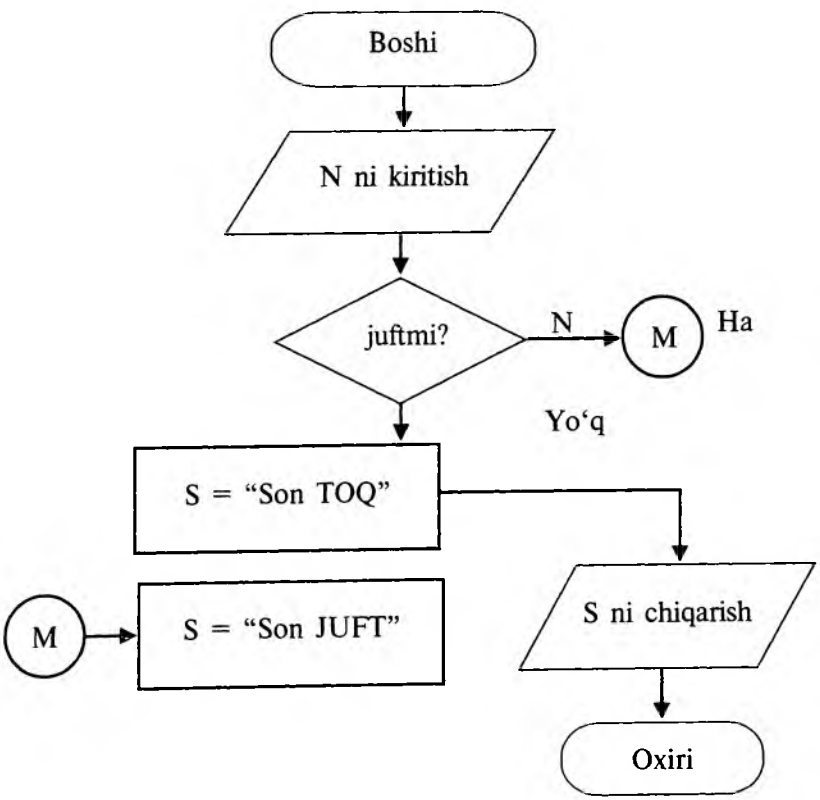


10.9-rasm

Ob'yekt	Xossa	O'rnatilgan qiymatlari
Form1	Caption	Shartsiz o'tish operatori
Label1	Caption	Butun son kiritingiz
Label2	Caption	Caption xossasi maydonini tozalash
Command1	Caption	Tekshirish
Command2	Caption	Tugatish
Text1	Text	Text xossasi maydonini tozalash

Shartsiz o'tish operatori

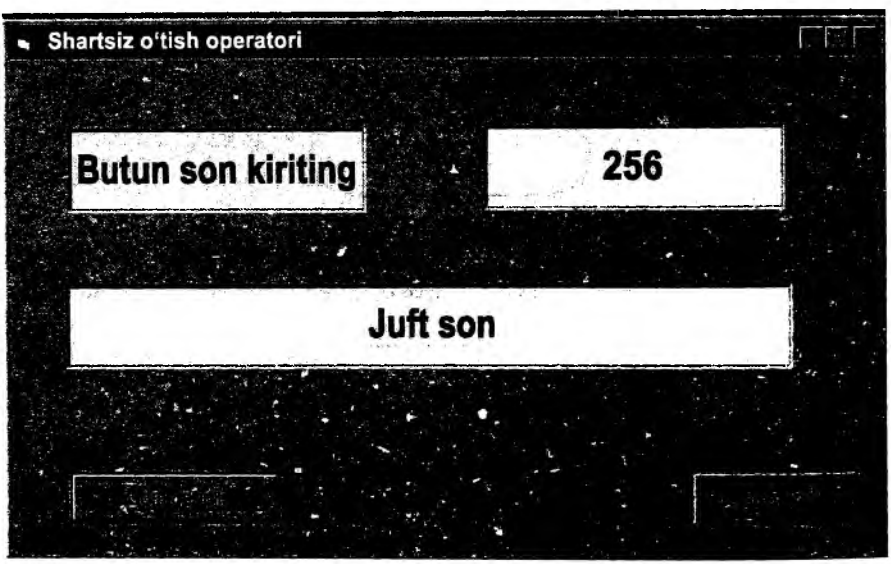
Blok-sxema



Dastur kodi

```
Option Explicit
        Private Sub Command1_Click()
            Dim N As Integer
```

```
        N = Val(Text1.Text)
        If (N Mod 2) = 0 Then GoTo Met1
        Label2.Caption = "Toq son"
        GoTo Met2
Met1: Label2.Caption = "Juft son"
Met2: End Sub

        Private Sub Command2_Click()
End
        End Sub
```



10.10-rasm

10.7. SELECT CASE SHARTLI OPERATORI

Select Case tuzilmasi dasturda bir necha shartlarni tekshirishga imkon beradi **If ...Then ... Else** konstruktsiyasi blokiga mos bo'ladi. Bu tuzilma tahlil qilinuvchi ifodalardan va har biri ushbu ifodaning qabul qiladigan qiymatlariga ega **Case** operatorlari majmuyidan tashkil topgan.

Struktura sintaksisi:

Select Case O'zgaruvchi
CASE qiymat1
konstruktsiya1

CASE qiymat2
konstruktsiya2
...
CASE qiymatK
konstruktsiyaK
CASE Else
Alternativ konstruktsiya
End Select

Visual Basic ifodaning konstruktsiyasida berilgan qiymatlarni hisoblaydi. So'ng olingan qiymat konstruktsiyadagi **Case** operatorlarida berilgan qiymatlari bilan solishtiriladi. Agar dastlabki qiymat topilsa, unda **Case** operatorida yozilgan buyruqlar bajariladi. Agar dastlabki qiymat topilmasa, unda alternativ **Case Else** operatorida (agar u ham bor bo'lsa) yozilgan buyruqlar bajariladi. Konstruktsiya bajarilishi yakunlangandan so'ng bosqarish **End Select** xizmatchi so'zidan keyingi operatorga beriladi. Konstruktsiya boshida **Select Case** xizmatchi so'zi joylashgan bo'lib, undan keyin joylashgan **"O'zgaruvchi"** parametri bir necha qiymatlarni tekshiradi. Davomi **Case** xizmatchi so'zi bilan boshlangan buyruqlar guruhidan iborat. Agar **"O'zgaruvchi"** parametri joriy **Case** operatorida ko'rsatilgan qiymatga teng bo'lsa, unda shu va keyingi **Case** xizmatchi so'zlari orasidagi buyruqlar bajariladi.

5-misol: Select Case tuzilmasini ishlatish

Tasodif sonlar datchigi yordamida o'yin zarralarini tashlashni dasturlash. Olingan natijalarni matn shaklida ko'rsatish kerak.

Alogritmi

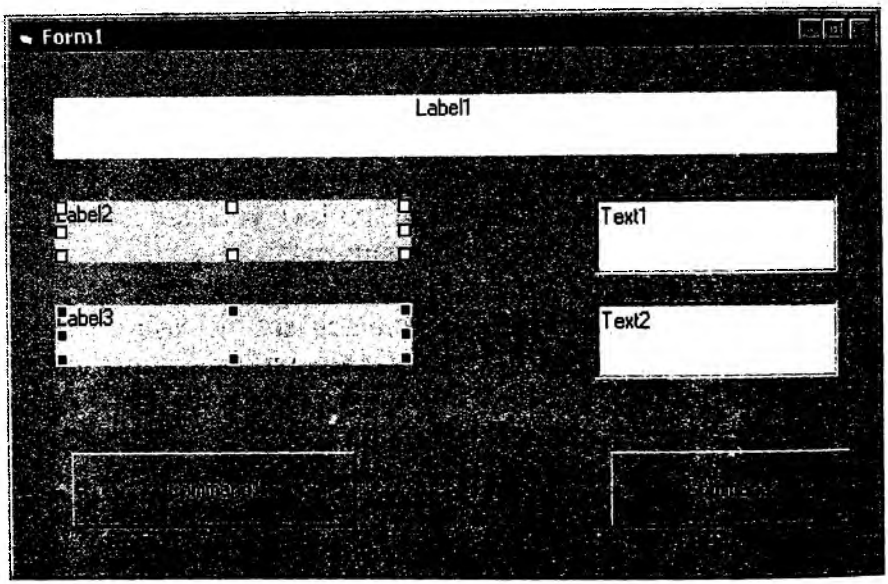
1. 1 dan 6 gacha bo'lgan R tasodif sonini olish
2. Agar R=1 bo'lsa, S qator o'zgaruvchisiga "BIR" matnni yozish kerak
3. Agar R=2 bo'lsa, S qator o'zgaruvchisiga "IKKI" matnni yozish kerak
4. Agar R=3 bo'lsa, S qator o'zgaruvchisiga "UCH" matnni yozish kerak

5. Agar R=4 bo‘lsa, S qator o‘zgaruvchisiga "TO‘RT" matnni yozish kerak

6. Agar R=5 bo‘lsa, S qator o‘zgaruvchisiga "BECH" matnni yozish kerak

7. Agar R=6 bo‘lsa, S qator o‘zgaruvchisiga "OLTI" matnni yozish kerak

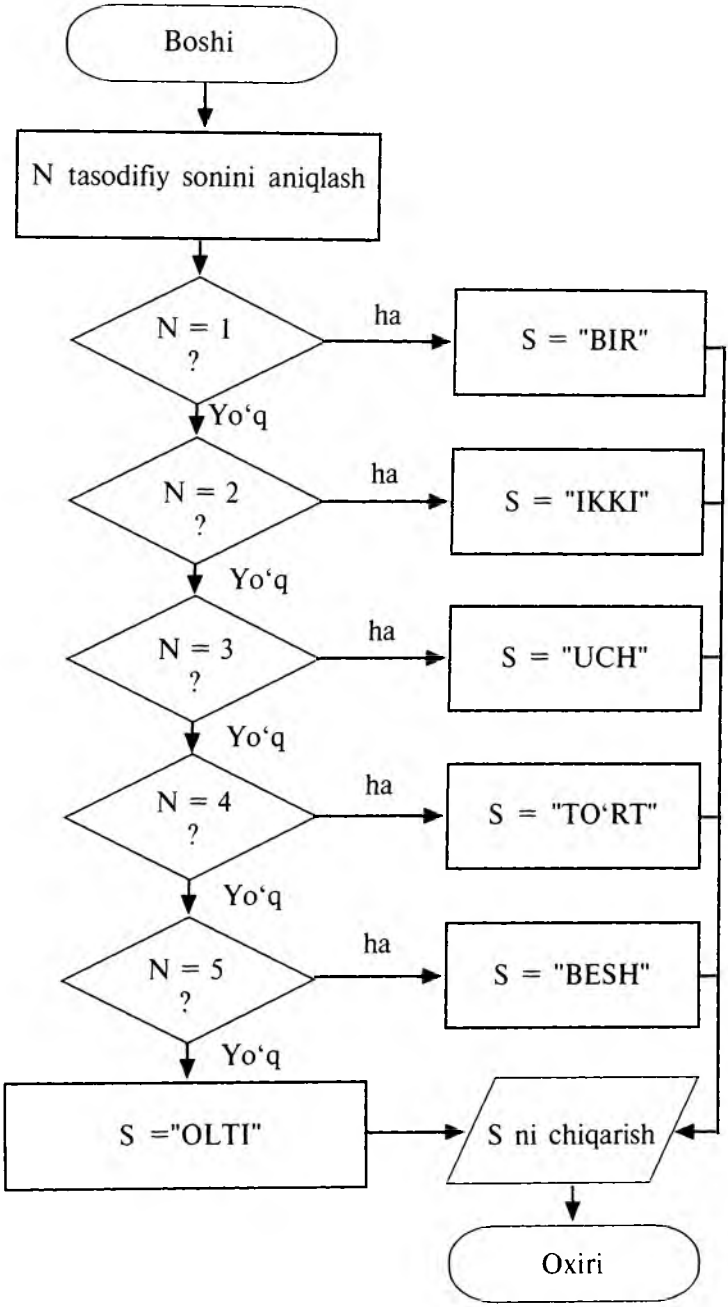
8. S ni chiqarish



10.11-rasm

Ob’yekt	Xossa	O‘rnatilgan qiymatlarii
Form1	Caption	Select Casetuzilmasi
Label1	Caption	1 dan 6 gacha bo‘lgan tasodif son
Label2	Caption	Sonli qiymati
Label3	Caption	So‘z bilan yoizilishi
Command1	Caption	Kubikni tashlash
Command2	Caption	Tugatish
Text1, Text2	Text	Text xossasi maydonini tozalash

Blok-sxema


Boshi
N tasodifiy sonini aniqlash
N = 1 ?
ha
S = "BIR"
Yo'q
N = 2 ?
ha
S = "IKKI"
Yo'q
N = 3 ?
ha
S = "UCH"
Yo'q
N = 4 ?
ha
S = "TO'RT"
Yo'q
N = 5 ?
ha
S = "BESH"
Yo'q
S ="OLTI"
S ni chiqarish
Oxiri

Dastur kodi

```
Option Explicit
                Private Sub Command1_Click()
                Dim R As Integer
                Dim S As String
                R = Int(6 * Rnd) + 1
  Select Case R
    Case 1
      S = "BIR"
    Case 2
      S = "IKKI"
    Case 3
      S = "UCH"
    Case 4
      S = "TO'RT"
    Case 5
      S = "BESH"
    Case Else
      S = "OLTI"
  End Select
  Text1.Text = Str(R)
  Text2.Text = S
End Sub

Private Sub Command2_Click()
  End
End Sub

Private Sub Form_Load()
  Randomize
End Sub
```

Select Case tuzilmasida solishtirish amallarini ishlatish mumkin. Buning uchun **Is** yoki **To** xizmatchi so'zini ifodaga qo'shish kerak. **Is** xizmatchi so'zi kompiliyatorga tekshirilayotgan o'zgaruvchining qiymatini ifodaning qiymati bilan solishtirishga ko'rsatmalar beradi. **To** xizmatchi so'zi qiymatlar diapazonini beradi. Mayli yuqoridagi misolda tasodifiy sonlar 1 dan 10 gacha bo'lgan inter-

valdan olinsin. Dastur kodini o'zgartiringiz va **Select Case**tuzilmasining ishlashini nazorat qiling.

Qayta ishlangan dastur kodi:

```
Option Explicit
Private Sub Command1_Click()
Dim R As Integer
Dim S As String
R = Int(10 * Rnd) + 1
Select Case R
  Case Is < 4
     S = "KAM"
  Case 4 To 6
     S = "KO'P"
  Case Else
     S = "BO'LISHI MUMKIN EMAS"
  End Select
  Text1.Text = Str(R)
  Text2.Text = S
End Sub

Private Sub Command2_Click()
  End
End Sub

Private Sub Form_Load()
  Randomize
End Sub
```

10.8. SIKLLI TUZILMALAR

Dasturlash tillarida takrorlanuvchi jarayonlarni bajarish uchun sikllituzilmalar ishlatiladi. Siklli tuzilmalarning uch xili mavjud:

1. Takrorlanishlar soni berilgan sikl (hisoblagichli sikllar).
2. Berilgan shart bajarilguncha bajariladigan sikl. Bunda shart sikl tanasi bajarilishidan oldin tekshiriladi (Yuqoridagi qator bilan boshqariladigan sikl).
3. Berilgan shart bajarilguncha bajariladigan sikl. Bunda shart sikl tanasi bajarilgandan so'ng tekshiriladi (Pastdagi qator bilan boshqariladigan sikl).

Sikl tanasi bo'yicha bir marta o'tish "iteratsiya" deyiladi.

10.8.1 FOR ... NEXT SIKL OPERATORI

For...Nexttuzilmasi buyruqlar ketma-ketligini bir necha marta bajaradi. Bunaqangi konstrusiyani sikl deb ataladi va u yordamida bajariladigan dastur kodilarini - sikl tanasi deb ataladi..

For...Nexttuzilmasining sintaksisi:

For ***Hisoblagich = Boshlang'ich_qiymat To Oxirg'i_qiymat Step Qadam***

konstruktsiya

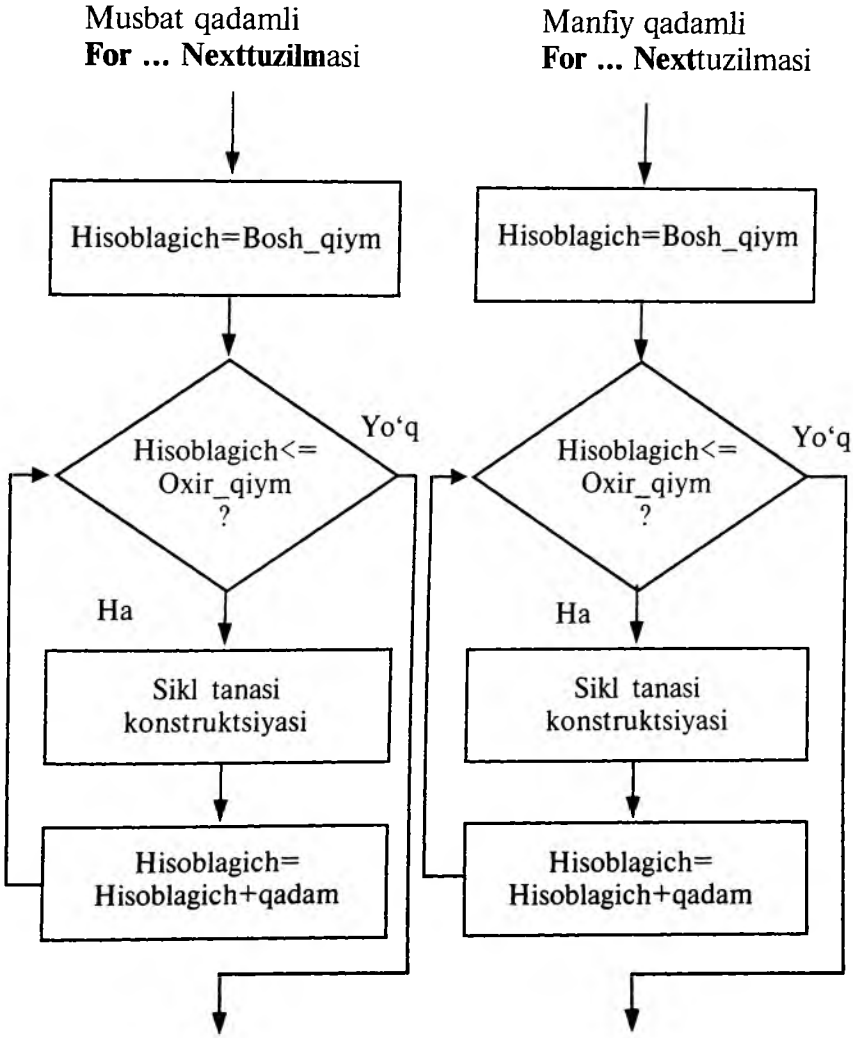
Next ***Hisoblagich***

Strukturaning birinchi argumenti - ***Hisoblagich*** siklning bajarilishlari sonini hisoblaydigan o'zgaruvchi nomini aniqlaydi. ***Boshlang'ich_qiymat*** parametri siklga birinchi marta kirishdan oldin ***Hisoblagich*** ga o'zlashtiriladigan boshlang'ich sonli qiymat bo'ladi. Sikl ***Hisoblagich*** ning qiymati **To** xizmatchi so'zidan keyingi ***Oxirg'i_qiymat*** ning qiymatidan oshib ketguncha bajariladi. Siklning har bir bajarilishidan so'ng ***Hisoblagich*** ning qiymati ***Qadam*** qiymatiga oshirilib boriladi. ***Next*** xizmatchi so'zi siklning oxirini ko'rsatadi. Siklning har bir bajarilishida ***Hisoblagich*** ning qiymati bilan ***Oxirg'i_qiymat*** argumentining qiymati solishtiriladi. Agar hisoblagichning qiymati ***Oxirg'i_qiymat*** ning o'rnatilgan qiymatidan oshib ketmasa, sikl tanasining konstruktsiyasi bajariladi. Aks holda boshqarish **Next** dan keyingi operatorga o'tadi. ***Hisoblagich*** ning qiymatini o'zgartiruvchi ***Qadam*** ning qiymati manfiy ham bo'lishi mumkin. Bunday holatda sikl ***Hisoblagich*** ning qiymati ***Oxirg'i_qiymat*** ning qiymatidan kichik bo'lguncha bajariladi va hisoblagichning boshlang'ich qiymati oxirgi qiymatidan katta bo'lishi lozim. **Step** xizmatchi so'zini ishlatmasa ham bo'ladi. Bunday holatda esa qadamning qiymati avtomat ravishda 1 ga teng bo'ladi. *Hisoblagich*, *Boshlang'ich_qiymat*, *Oxirg'i_qiymat*, *Qadam parametrlari* - o'zgaruvchi, o'zgarmas yoki butun turdagi ifoda bo'lishi mumkin.

Agar **Bosh_qiym** ning qiymati **Oxir_qiym** qiymatidan katta (musbat **qadam** da) yoki **Bosh_qiym** ning qiymati **Oxir_qiym** qiymatidan kichik (manfiy **qadam** da) bo'lsa, sikl tanasi bir marta ham bajarilmaydi.

Agar **Hisoblagich** ning qiymati sikl ichidagi shart yakunlanish holati bajarilmaydigan qilib o'zgartirilsa (masalan, iteratsiya oxirida

sikl **Hisoblagich**iga **Bosh_qiym** qiymati o'zlashtiriladi), unda "sikllashuv" hosil bo'ladi, ya'ni sikl cheksiz bajariladi.



6-misol: For ... Next operatorini ishlatish

Arifmetik progressiyaning dastlabki 100 ta elementinining yig'indisini hisoblash.

Arifmetik progressiya o‘zining a0 nollik elementi va d ayirmasi bilan beriladi. Arifmetik progressiyaning har bir kelasi elementi formulasi $\mathbf{a_k = a_{k-1} + d}$ bilan hisoblanadi.

Arifmetik progressiyaning dastlabki 100 ta elementinining yig‘indisi quyidagicha:

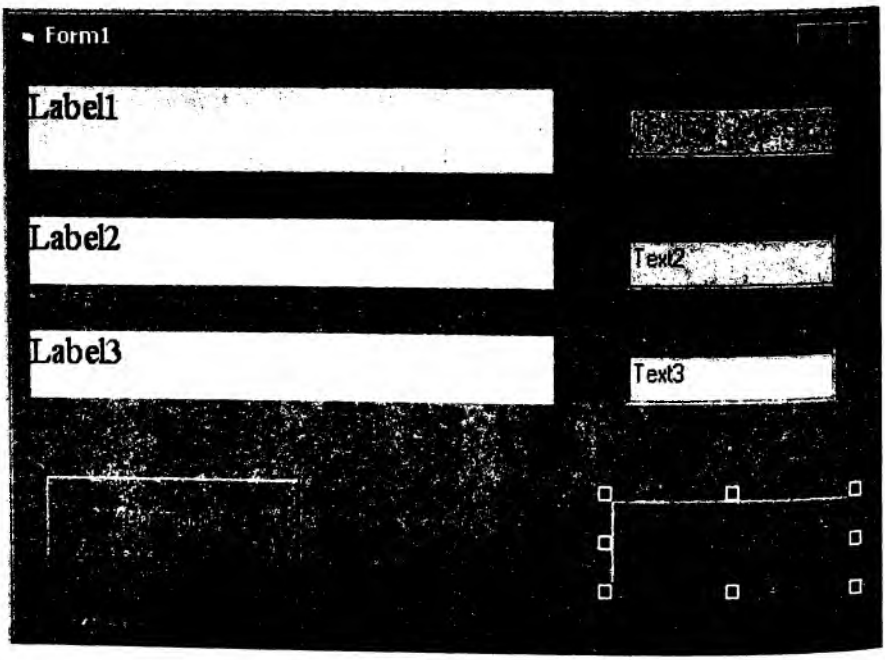
$\mathbf{S} = \sum \mathbf{a_k}$, bu yerda **k** 0 dan 100 gacha bo‘lgan qiymatlarni ketma-ket qabul qiladi.

Algoritmi

1. $\mathbf{a_0}$ va **d** larni kiritingiz.
2. Arifmetik progressiyaning yig‘indisi - S o‘zgaruvchisiga $\mathbf{a_0}$ ning qiymatini o‘zlashtiringiz.
3. For K = 1 To 100 ... Next siklida har bir iteratsiyada arifmetik progressiyaning har bir elementini va yig‘indining qiymatini quyidagi formula yordamida hisoblanadi:

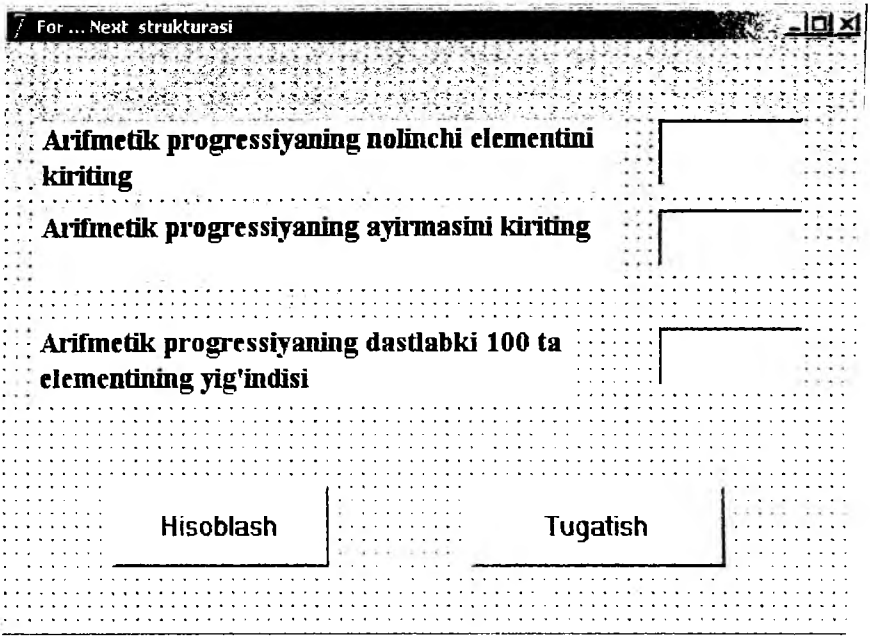
$$\mathbf{ak = ak\text{-}1 + d} \qquad \mathbf{S = S + ak}$$

4. S ni chiqarish



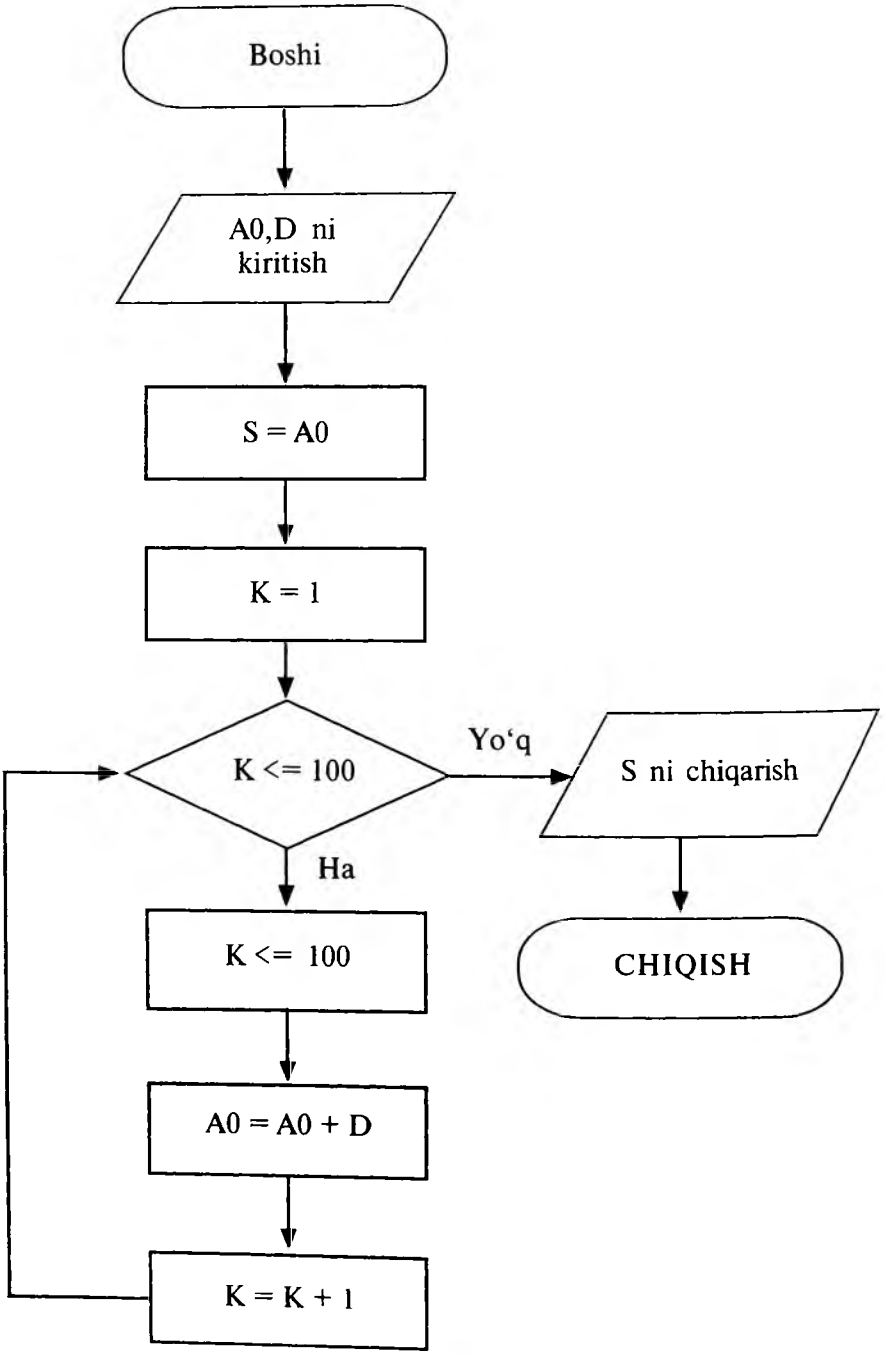
10.12-rasm

Ob'yekt	Xossa	O'rnatilgan qiymatlari
Form1	Caption	For ... Nexttuzilmasi
Label1	Caption	Arifmetik progresiyaning nolinchi elementini kiriting
Label2	Caption kiriting	Arifmetik progresiyaning ayirmasini
Label3	Caption	Arifmetik progressiyaning dastlabki 100 ta elementining yig'indisi
Command1	Caption	Hisoblash
Command2	Caption	Tugatish
Text1, **Text2,** **Text3**	Text	Text xossasi maydonini tozalash



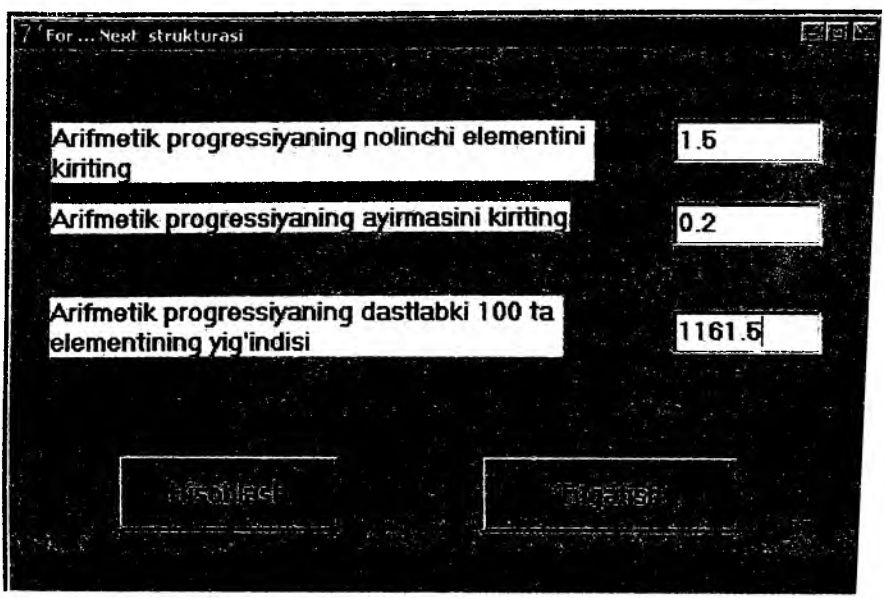
10.13-rasm

Blok-sxemasi



Dastur kodi

```
Option Explicit
                Private Sub Command1_Click()
                  Dim A0, D, S As Double
                  Dim K As Integer
                  A0 = Val(Text1.Text)
                  D = Val(Text2.Text)
  S = A0
  For K = 1 To 100
    A0 = A0 + D
    S = S + A0
  Next
  Text3.Text = Str(S)

End Sub

Private Sub Command2_Click()
  End
End Sub
```



10.14-rasm

10.9. SHARTI OLDINDA VA OXIRIDA BERILGAN SIKLLAR

Bazi bir holatlarda sikl tanasining necha marta bajarilishini oldindan aniqlab bo'lmaydi. Bunday holatlarda sikldagi shartning qiymati True bo'lguncha bajariladi.

Bunaqangi sikllarning ikki holati mavjud:

1. Sikl sharti iteratsiyaga har bir kirishidan oldin tekshiriladi (sharti oldinda berilgan sikllar)
2. Sikl sharti har bir iteratsiyaning bajarilishidan so'ng tekshiriladi (sharti oxirirda berilgan sikllar)

10.9.1. Sharti oldinda berilgan Do While siklining sintaksisi

Do While ***Shart***
Konstruktsiya
Loop

Do While ... Loop konstruktsiyasida berilgan sikl operatori undagi berilgan shartning qiymati **True** bo'lguncha davom etadi. ***Shart*** argumenti mantiqiy ifoda bo'lib, siklga har bir kirishda tekshiriladi. Agar uning qiymati **True** bo'lsa, **Do While** va **Loop** xizmatchi so'zlari orasidagi buyruqlar ketma-ketligi bajariladi. Bu konstruktsiya sikl tanasini tashkil qiladi. Agar siklga kirishda shartning qiymati **False** bo'lsa, sikldan chiqadi va boshqarish **Loop** dan keyingi konstruktsiyaga beriladi. Shunaqangi holat mavjudki, ya'ni sikl ichidagi operator bir marta ham bajarilmasligi mumkin. Bu holat sikl sharti birinchi marta tekshirilganda uning qiymati **False** bo'lsa.

10.9.2. Sharti oxirida berilgan siklning sintaksisi

Do
Konstruktsiyalar
Loop While ***Shart***

Bu sikl strtukturasini ishlatishda shart sikl oxirida tekshiriladi. Bu holatda sikl tanasi kamida bir marta bajariladi va undan keyin shart tekshiriladi.

10.9.3. Sharti oldinda berilgan sikl

Do ... Loop sikl konstruktsiyasining yana bir ko'rinishi bor. Ular avvalgi ko'rilgan sikllar bilan bir xil, lekin sikl shartining qiymati False bo'lguncha bajariladi.

Do Until ***Shart***
Konstruktsiya
Loop

10.9.4. Sharti oxirida berilgan sikl

Do
Konstruktsiya
Loop Until ***Shart***

Sikldan shartsiz chiqib ketish sharti oldinda va oxirida berilgan sikllarning tuzilmasi uchun **Exit Do** sintaksisiga ega. **Exit** buyrug'i siklning bajarilishini yakunlaydi va boshqarishni sikldan keyingi konstruktsiyaga beradi.

7-misol: Shartli sikl tuzilmasini ishlatish (shart sikl tanasi bajarilishidan oldin tekshiriladi)

Kvadrat berilgan. Kvadrat tomoni 2 ga teng. Kvadrat markazi dekart koordinatalari sistemasining (0,0) nuqtasida joylasgan. Siklda kvadrat ichidan tasodifiy nuqta olinadi. Nuqta markazlari (0,0) nuqtada va radiuslari R1=0,5; R2 =1,0 bo'lgan ikkita konsentrik aylanadan tashkil topgan halqaga tegishliligini tekshirish kerak. Quyidagi shartlarning biri bajarilsa, sikl to'xtatiladi.

- Tasodifiy olingan nuqtalarning soni 300 ga teng bo'lsa.
- Halqaga tegishli bo'lgan nuqtalar soni 50 ga teng bo'lsa.

Nuqtalarning umumiy soni va halqaga tegishli bo'lgan nuqtalarning soni chop etiladi.

Algoritmi

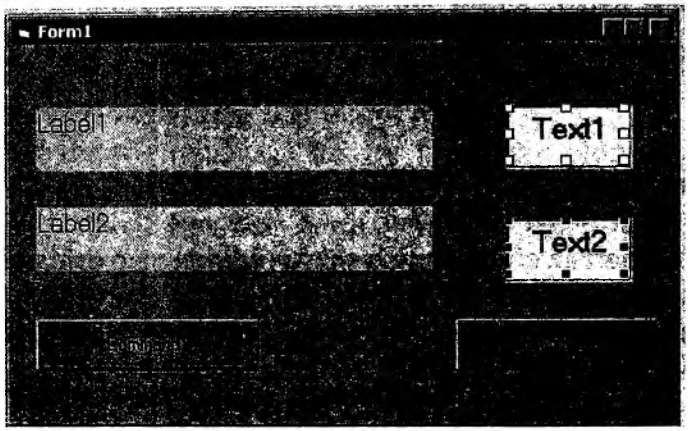
1. K1 - aniqlanadigan nuqtalarning tartibi. Siklga kirishdan oldi unga nol qiymatini berish lozim.
2. K2 - halqaga tushuvchi nuqtalar soni. Siklga kirishdan oldi unga nol qiymatini berish lozim.
3. Sikl sarlavhasi. K1 ≤ 300 va K2 ≤ 50 shartlarini tekshirish
4. Agar teksirish natijasi True bo'lsa, K1 =K1 + 1 iteratsiyasi _malgam oshirish.

(x,y) nuqtasi koordinatalarini topish

$-1.0 \le x \le +1.0$, $-1.0 \le y \le +1.0$

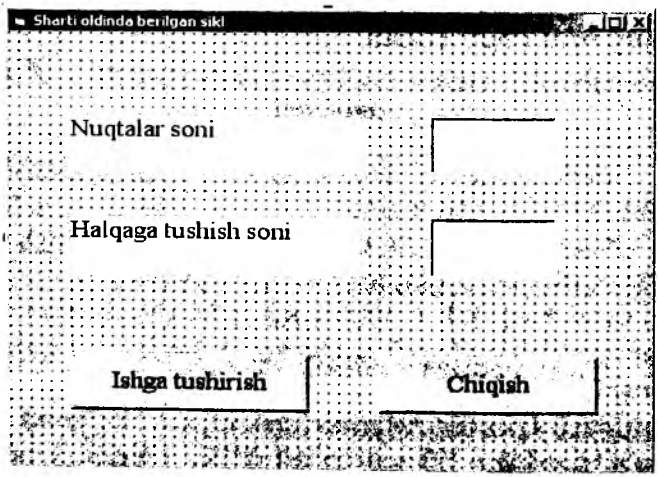
Nuqtaning $0.25 \le x^2 + y^2 \le 1.0$ halqasiga tegishliligini tekshir

Agar K2 =K2 + 1 sharti bajarilsa, 3 punktga o‘tish
5. K1, K2 larni chiqarish



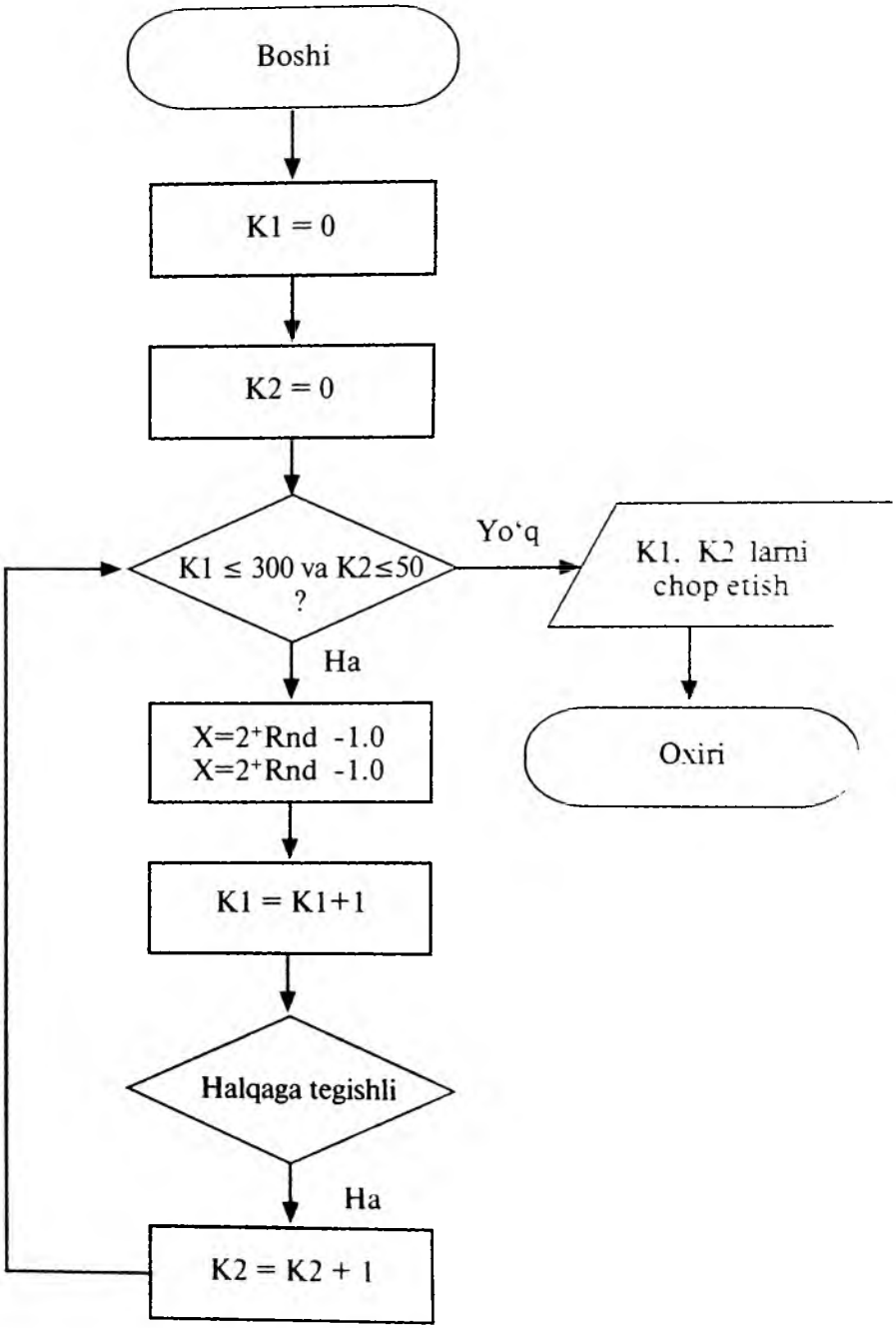
10.15-rasm

Ob’yekt	Xossasi	O‘rnatilgan qiymatlari
Form1	Caption	Sharti oldinda berilgan sikl
Label1	Caption	Nuqtalar soni
Label2	Caption	Halqaga tushishlar soni
Command1	Caption	Ishga tushirish
Command2	Caption	Dasturdan chiqish
Text1, Text2	Text	Text xossasi maydonini tozalash



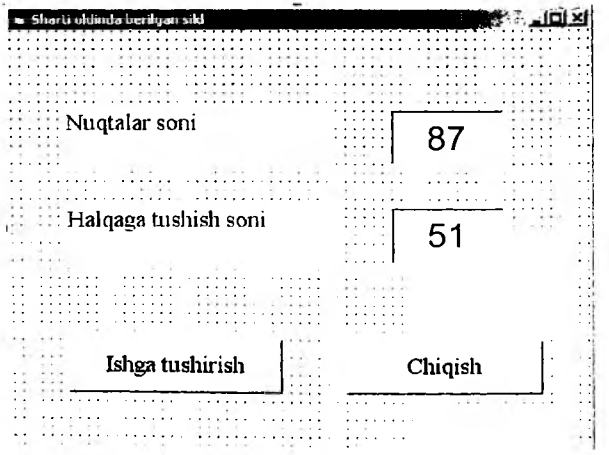
10.16-rasm

Blok-sxemasi



Dastur kodi

```
Option Explicit
Private Sub Command1_Click()
Dim K1, K2 As Integer
Dim X, Y, D As Single
K1 = 0
K2 = 0
Do While (K1 <= 300) And (K2 <= 50)
  X = 2# * Rnd - 1#
  Y = 2# * Rnd - 1#
  K1 = K1 + 1
  D = X * X + Y * Y
  If (D > 0.25) And (D < 1#) Then K2 = K2 + 1
Loop
Text1.Text = Str(K1)
Text2.Text = Str(K2)
End Sub

Private Sub Command2_Click()
  End
End Sub

Private Sub Form_Load()
  Randomize
End Sub
```



10.17-rasm

8-misol: Shartli sikllarning tuzilmasini ishlatish (sikl tanasi bajarilganidan so'ng shartni tekshirish)

Tasodif sonlar datchigi yordamida 0 dan 1000 gacha bo'lgan oraliqdan butun sonni olish. Urinishlar 950 dan katta bo'lgan qiymatni olguncha davom etadi. Olingan natijani va urinishlar sonini natijaga chiqarish kerak.

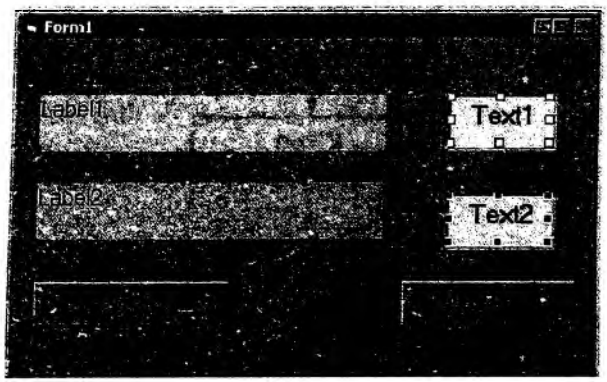
Algoritmi

1. K - urinishlar soni - siklga kirisgdan oldin unga nol qiymatini berish
2. Sikl sarlavhasini kiritish
3. Sikl iteratsiyasi formulasini kiritish

K =K + 1

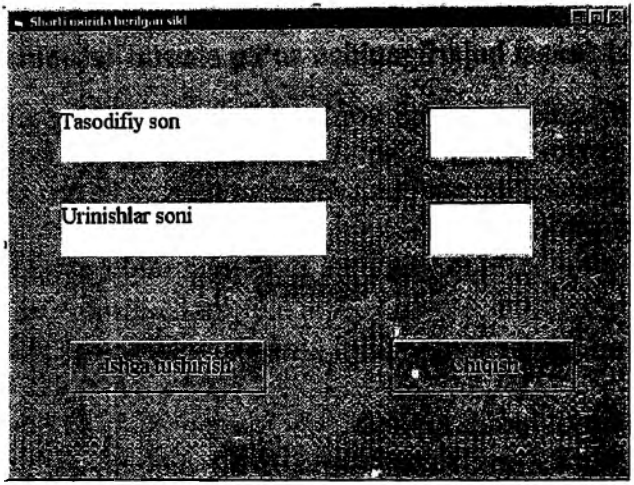
A =1000*int(Rnd) tasodif sonini aniqlash

4. Agar A ? 950 bo'lsa, 3-bosqichqa o'tish
5. A, K larni chiqarish



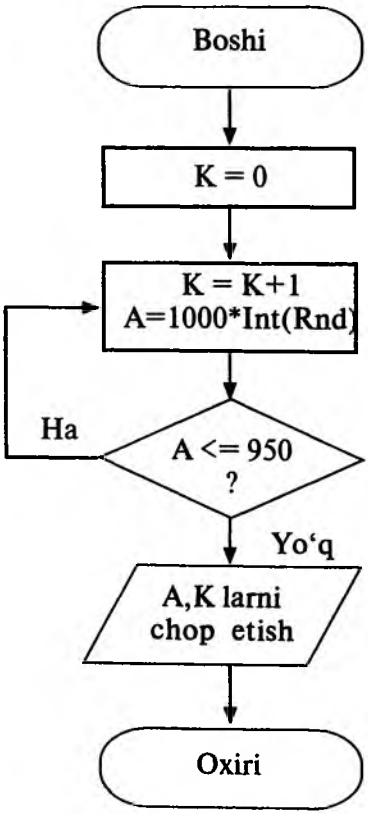
10.18-rasm

Ob'yekt	Xossasi	O'rnatilgan qiymatlari
Form1	Caption	Sharti oxirida berilgan sikl
Label1	Caption	Tasodifiy son
Label2	Caption	Urinishlar soni
Command1	Caption	Ishga tushirish
Command2	Caption	Chiqish
Text1, Text2	Text	Text xossasi maydonini tozalash



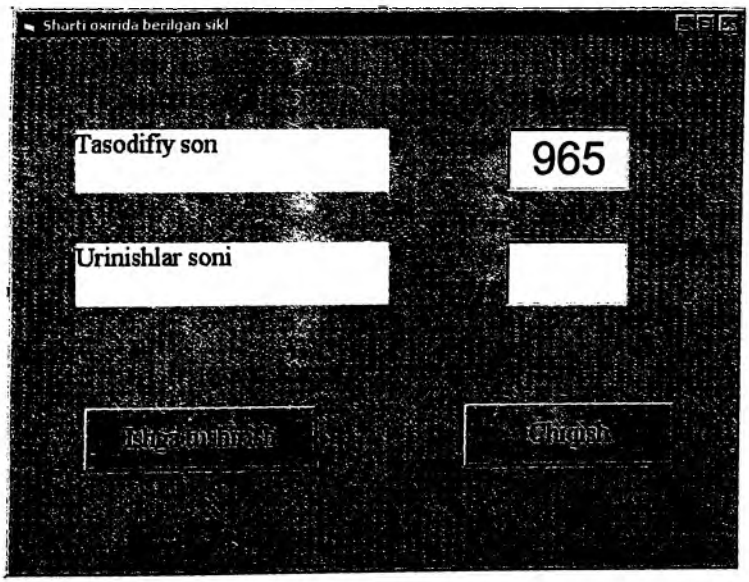
10.19-rasm

Blok-sxemasi



Dastur kodi

```
Option Explicit
Private Sub Command1_Click()
  Dim K, A As Integer
  K = 0
  Do
    K = K + 1
    A = Int(1000 * Rnd)
  Loop While A <= 950
  Text1.Text = Str(A)
  Text2.Text = Str(K)

End Sub

Private Sub Command2_Click()
  End
End Sub

Private Sub Form_Load()
  Randomize
        End Sub
```



10.20-rasm

10-bo'limga oid savollar

1. Vizual Basicda izohlar qanday tashkil qilinadi?
2. Operatorlarni bir nechta qatorga joylashtirish qanday tashkil etiladi?
3. Bir nechta operatorlarni bir qatorga joylashtirish qanday tashkil etiladi?
4. Kodni tahrirlash nima?
5. Chiziqli algoritm va ularning dasturlari.
6. If, Go to, Select boshqaruvchi tuzilmalarni tushuntirib bering.
7. Select Case shartli operatori va uni ishatish.
8. Siklli tugmalarni tushuntirib bering.
9. For ... Next sikl operatori va uni ishlatishni tushuntirib bering.
10. Sharti oldinda berilgan sikl operatori sintaksisini va ishlatilishini tushuntirib bering.
11. Sharti oxirida berilgan sikl operatori sintaksisini va ishlatilishini tushuntirib bering.

FOYDALANILGAN ADABIYOTLAR RO'YXATI

1. *Бухер, Патрик.* "Programmiersprachen" (Языки программирования), статья на сайте www.it-academy.cc, 10/2004.

2. *Бухер, Патрик.* "Der Bubble-Sort Algorithmus", статья на сайте www.it-academy.cc, 11/2004.

3. *Бёме*, д-р. Р. "Programmiersprachen C/C++" (Язык программирования C/C++), конспект лекций Института информатики Университета им. Мартина-Лютера, Халле-Виттенберг, 1996.

4. *Блессманн, Бюттнер, Дакс.* "Anwendungsentwicklung" (Программирование), Тройсдорф, 2002.

5. *Бройер, Клаус Ульрих* и *Мюллер, Карлхайнц.* "Umsetzungshilfen fur neue Prufungsstruktur der IT-Berufe" (Помощь в реализации новых экзаменационных стректур для ИКТ-профессий), заключительный отчет", Федеральное министерство образования и исследований (ответственный редактор), Бонн, 10/2000.

6. Федеральный институт образования (ответственный редактор), "Erlauterungen und Praxishilfen zur Ausbildungsordnung" (Пояснения и практическая помощь по положению об организации профессиональной подготовки), Берлин, 1998.

7. *Штабенов, Хельмут* и *Тодт, Петер.* "Informations- und Telekommunikationstechnik" (Информационная и телекоммуникационная техника), Бад Хомбург, 2001.

8. *Михельман, Норберт* и *Хеттвер, Рольф.* "Datenbankentwicklung und -anpassung mit MS Access und SQL" (Разработка баз данных и их адаптация к MS Access и SQL), Тройсдорф, 2001.

9. *Михельман, Норберт* и *Хеттвер, Рольф.* "Programmentwicklung mit C/C++ und HTML" (Программирование на C/C++ и HTML), Тройсдорф, 2001.

10. *Бойт, Маркус* и др. "Fachinformatiker/-in Systemintegration, IT-System-Elektroniker/-in, Prufungsvorbereitung" (Спе-

циалист по информатике, системной интеграции, специалист по электронике ИТ-систем, подготовка к экзамену), Тройсдорф, 2004.

11. *Лайтенбергер, Бернд.* "Die Entwicklung der Programmiersprachen" (Разработка языков программирования), Интернет-статья, Остфилдерн, 02/2006.

12. Clemson University. Department of Industrial Engineering, SC, USA, "Flow Charts", Web- Tutorial, deming.eng.clemson.edu.

13. Wirth, N. (1976). Algorithms + Data Structures = Programs, Prentice-Hall, Englewood Cliffs, N.J. (Русский перевод: Вирт Н. Алгоритмы + структура данных = программы. - М.б "Мир" , 1985.).

14. Федеративная Республика Германия, решение конференции министерства по делам образования, религии и культуры от 25.04.1997, положение об организации профессиональной подготовки специалистов по информатике, ИТ-электронике и коммерсантов ИТ-систем, опубликовано в федеральном вестнике Nr. 68 от 08.04.1998.

15. *Зайчик* и др., "Deutsch-Russisches Worterbuch fur EDV-Terminologie" (Немецко-русский словарь по компьютерной терминологии), Гамбург, 1998.

16. *Мизинина, Жильцов.* Англо-русский и русско-английский словарь компьютерной лексики, Москва, 2004.

17. *Брандт, Финн* и др, "T@ke IT", ключевые квалификации для ИТ-профессий, Гамбург, 2003.

18. Senator fur Bildung und Wissenschaft der Freien und Hansestadt Bremen (Сенатор по вопросам образования и науки свободного ганзейского города Бремен) (издатель), "Rahmenplan IT-Berufe" (Типовой план для ИТ-профессий), Бремен, 2000.

19. Thuringer Kultusministerium (Тюрингское министерство по делам образования, религии и культуры) (издатель), Umsetzung des KMK-Rahmenplans fur die Ausbildung zum Fachinformatiker Anwendungsentwicklung, Bad Berka, 2000.

20. *Ридлль, Альфред.* "Didaktik I Grundlagen" (Дидактика I. Основы) и "Didaktik II Berufliche Bildung" (Дидактика II. Профессиональное образование), конспект лекций Мюнхенского Технического Университета, 2003.

21. *Плате, Юрген.* "Algorithmen und Datenstrukturen,

Programmieren 1" (Алгоритмы и структуры данных, программирование 1), конспект лекций Мюнхенского высшего специального учебного заведения, FB 04, январь 2006-03-02.

22. Различные статьи с сайта de.wikipedia.org.

23. *Думке*, Р. "Algorithmen und Datenstrukturen" (Алгоритмы и структуры данных), и другие конспекты лекций Университета Otto-von-Guericke-Universitat, факультет информатики, Магдебург, 06/2005.

SHODMONQUL ABDIROZIQOVICH NAZIROV,
GYUNTER DIVALD

DASTURLASH ASOSLARI

Axborot-kommunikatsiya texnologiyalari sohasidagi kasb-hunar kollejlarining "Axborot-kommunikatsiya tizimlari (3521916)" mutaxassisligi talabalari uchun o'quv qo'llanma

«Sharq» nashriyot-matbaa
aksiyadorlik kompaniyasi
Bosh tahririyati
Toshkent – 2007

Muharrir *M. Saparov*
Badiiy muharrir *T. Qanoatov*
Texnik muharrir *D. Gabdraxmanova*
Sahifalovchi *I. Kravchenko*
Musahhih *N. Oxunjonova*

Bosishga 05.11.2007 da ruxsat etildi. Bichimi 60x90¹/₁₆. «Tayms UZ» garniturasi.
Ofset bosma. Shartli bosma tabog'i 12,5. Nashriyot-hisob tabog'i 12,5. Adadi 5000 nusxa.
Buyurtma № 4059. Bahosi kelishilgan narxda.

**«Sharq» nashriyot-matbaa aksiyadorlik kompaniyasi bosmaxonasi,
100083, Toshkent shahri, Buyuk Turon ko'chasi, 41.**

DASTURLASH ASOSLARI

ISBN 978-9943-00-212-8