O’ZBEKISTON RESPUBLIKASI OLIY VA O’RTA MAXSUS

TA’LIM VAZIRLIGI

Alisher Navoiy nomidagi Samarqand Davlat Universiteti mexanika-matematika fakultiteti Axborotlashtirish texnalogiyalari kafedrasi

“Tizimli va amaliy dasturlash” fanidan

5480100 – Amaliy matematika va informatika ta’lim yo’nalishi,

3-kurs talabalari uchun

Ma’ruzalar manti

Mualliflar: dots. Qobilov S. S.

k.o’q. Samatov J. A.

Samarqand – 2011

Ma’ruza № 1

Mavzu: Kirish. Programma ta’minoti strukturasi

Reja:

1. EHM arxitekturasi va programma ta’minoti.
2. Tizimli va amaliy programma ta’minoti.
3. Sistemaviy programmalash.
4. Mashina, operatsion tizim va vositalar.

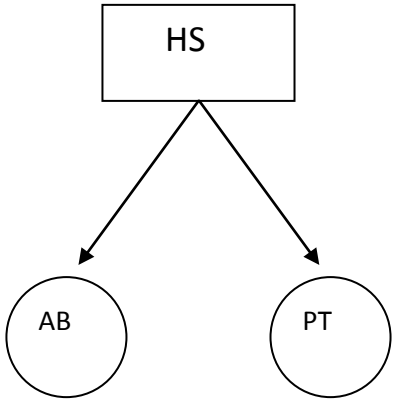
Tayanch iboralar: Hisoblash tizimi, arxitektura, programma ta’minoti, platforma, kontroller, tizimli programma, operatsion sistema.

Ma’ruza bayoni.

Sistemaviy programma taminoti strukturasi.

Programmalashtirish – bu ma’lumotlarni qayta ishlashni tashkil qilish jarayoni bo’lib programma tuzish ,u bilan bog’liq bo’lgan ma’lumotlar strukturasini tasvirlash kodlashtirishni o’z ichiga oladi.Programma esa komandalarning tartiblangan ketma-ketligi bo’lib konkret masalaning yechim algaritmini tasviridir yoki boshqacha qilib aytganda pragramma pragrammalash tilining konstruksiyalari ketma-ketligidir.Har bir programma hisoblash tizimi muhitida bajariladi.

Hisoblash sistemasi (HS) yoki tuzimi ikkita bir-biriga bog’liq bo’lgan, ajralmas qismlardan iborat, yani quydagi sxema yordamida ko’rsatish mumkin.

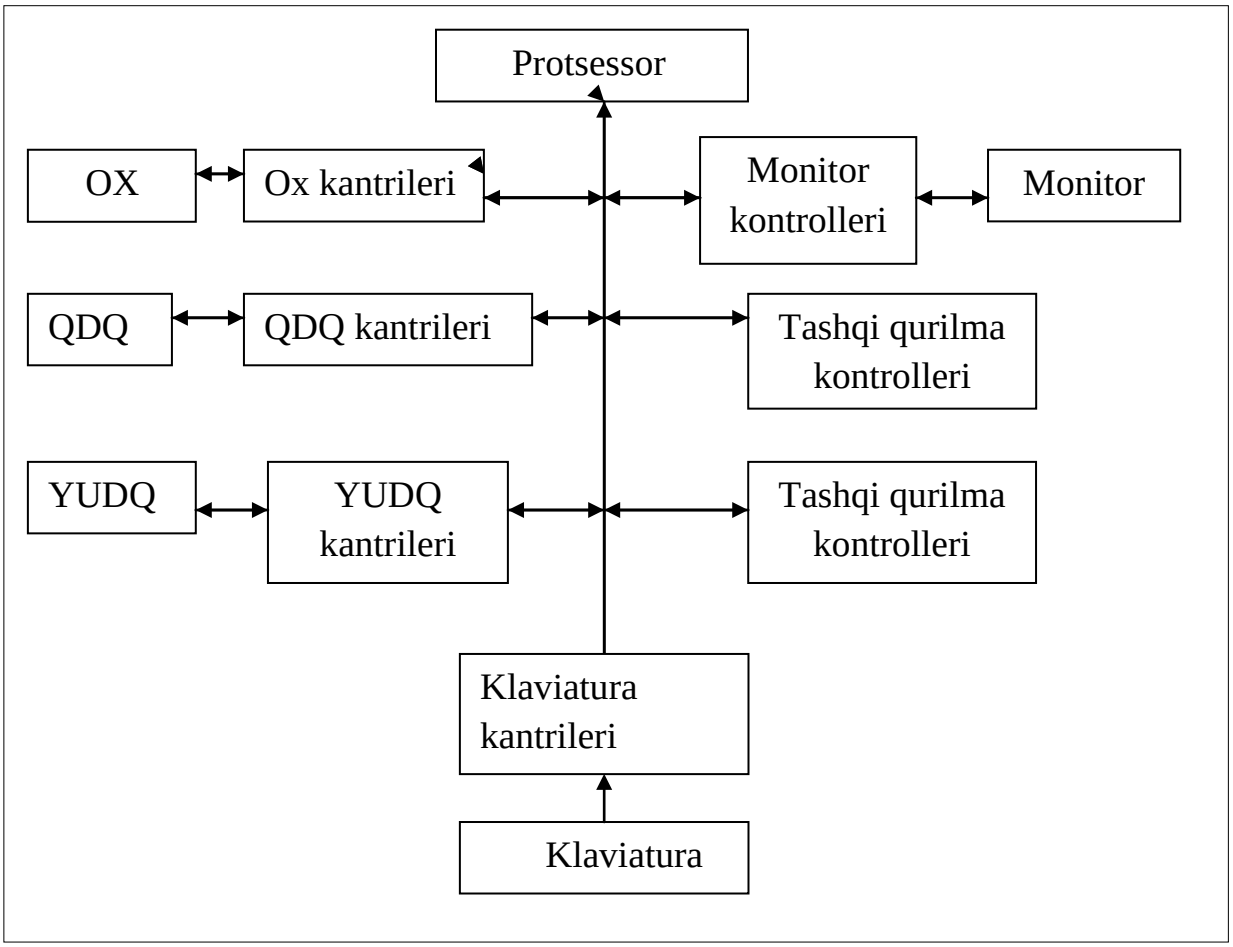


Bu yerda AB – aparat vositalari (hardware) va PT – programma taminoti (software).Hisoblash sistemasining har

bir kompanentasini yaratishning usuli vosita va texnalogiyalari mavjud.

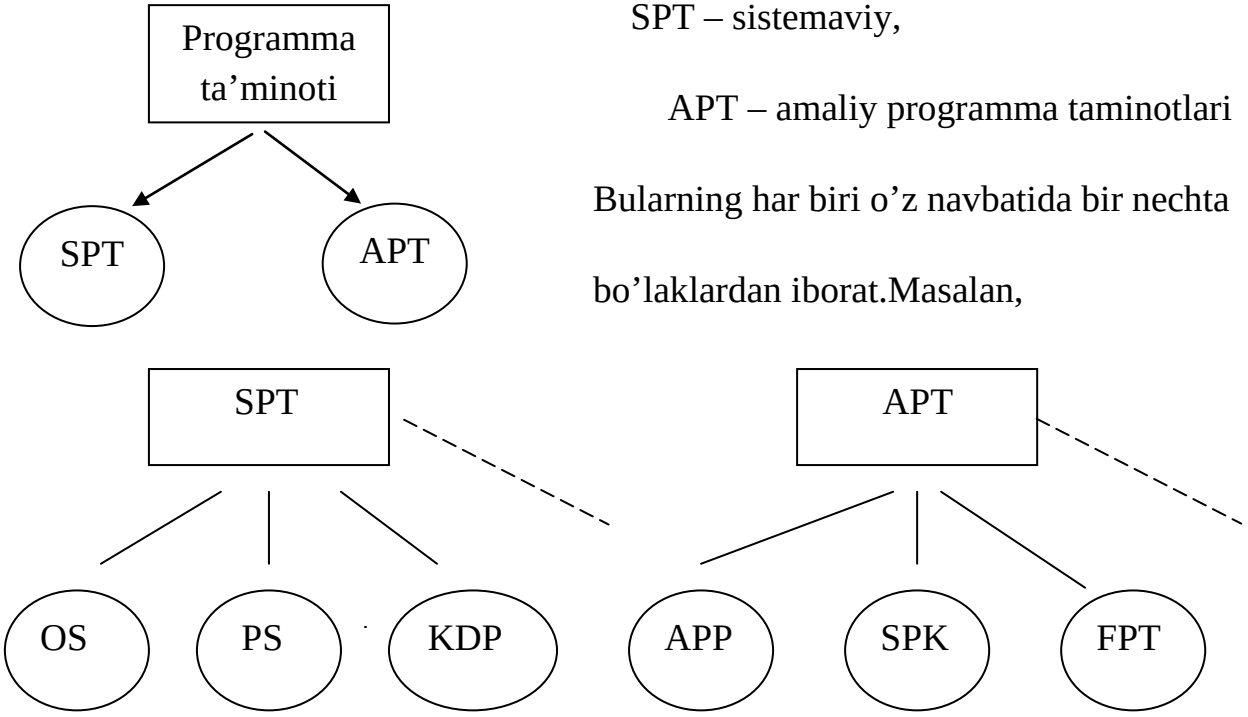
Aparat vositalarini tahlil qilishda tizim

arxitekturasini bilish kerak.Arxitektura so’zi EHM ,hisoblash sistemasi yoki programma taminotiga nisbatan tizimni umumiy mantiqiy tuzilishi , ma’lumotlarni kodlash, texnik vositalar va programmalar majmui, ularning o’zaro hamkorligi , xususiyatlari va qo’llash pirinsiplarini bildiradi.Jahondagi ShEHM parkini 85-87% tashkil qiluvchi IBM PCga o’rindosh kompyuterlarni arxitekturasini qarasak uning umumiy ko’rinishi quydagichadir



Bu yerda OX – operativ xotira QDQ – qattiq disk qurulmasi, YUDQ – yumshoq disk qurulmasi.**Kontroller maxsus protsessor bo’lib tashqi qurulmalarni bobqarish uchun ishlatiladi**.Shuning natijasida markaziy (bosh) protsessor quril-malarini boshqarishdan ozod bo’ladi.

Maskur fanni o’rganish jarayonida biz asosan programma taminoti,uning turlari, tashkil qilish usuli, vosita va texnalogiyalari bilan tanishamiz. Programma taminoti ikki turga bo’linadi – sistemaviy va amaliy programma taminoti.



SPT – sistemaviy,

APT – amaliy programma taminotlari

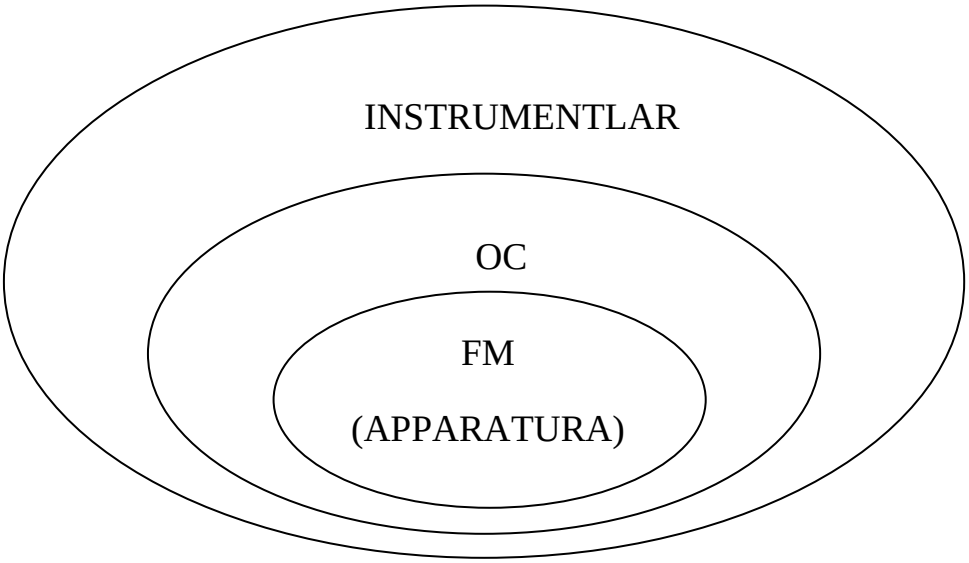
Bularning har biri o’z navbatida bir nechta bo’laklardan iborat.Masalan,

Bu yerda OS – operatsion sistema (tizim), PS – programmalash tizimi, KDP – kontrol va diagnostika (tashhis) programmalari.APP – amaliy programmalar paketi, SPK – standart programmalar ko’mplekisi, FPT – foydalanuvchi tamonidan muayyan masalalarni yechish uchun tashkil qilngan programmalar.

Har bir taminot turiga o’ziga xos usullar, programmalash texnalogiyalari mos keladi. Sistemaviy programmalash – bu umumiy programma taminotini yaratish jarayoni bo’lib OS, programmalash sistemalari, boshqaruvchi va tekshiruvchi (tashhislovchi) programmalarni tuzishga asosiy etibor beradi.Amaliy programmalash amaliy (tadbiqiy) programmalarni tuzish va taxlashga (atladka) mo’ljallangandir.

Tizimli programmalash usul, texnologiya va vositalari haqida gapirsak u holda texnalogiyalardan modulli va strukturali programmalash, klassik va hozirgi zamon vizual dasturlash paradigmalari, vositalardan esa o’ya yuqori dasturlash tillari, mashina va mashinaga mo’ljallangan tillar, assemblerlar ham ishlatiladi.

Umuman olganda OS, mashina va programma inustrumental vositalari va taminoti orasidagi aylanali diagramma yordamida tasvirlah mumkin.



Sinov savollari:

1. EHM arxitekturasi nima?
2. EHM Programma ta’minoti turlari?
3. Tizimli programma ta’minotiga nimalar kiradi?
4. Amaliy programmalar qanday xususiyatga ega?
5. Kontrollar – bu nima?
6. Platformalarning qanday turlarini bilasiz?
7. Diagnostika programmalarga misollar keltiring.

Adabiyotlar

[1], [7], [17], [21]

Ma’ruza № 2

Mavzu : Operatsion sistema turlari va tarkibi

Reja :

1. Operatsion sistema ta’rifi va xususiyatlari.
2. ShK operatsion tizimlari kompanentalari.
3. MS – DOS tizimi tarkbi.
4. Windows sistemasi qulayliklari.

Tayanch iboralar : Operatsion sistema, qism sistemalar, uzulish, bios .

Boshlang’ich yuklash bloki, FAT, yadro, qobiq, drayver, reestr, dispatcher .

Ma’ruza bayoni

Operatsion sistema (OS) sistemaviy programma taminotining asosiy turlaridan biri bo’lib apparatura (mashina) atirofida ish muhitini tashkil qiladi va mashinaga to’g’ridan – to’g’ri murojat qiladi. Malumotlarni qayta ishlash jarayonida 4 ta bir – biri bilan bog’liq kompanenta: odam (foydalanuvchi), qurilmalar (mashina), programma va ma’lumot o’zaro hamkorlik qilsa, OS hu hamkorlikni boshqaradi.

Shularni xisobga olib OS ga quydagicha tariff berish mumkin.

Tarif: Operatsion sistema sistemaviy programma bo’lib foydalanuvchi va hisoblash tizimi orasidagi muloqatni tashkil qiladi va quydagi funksiyalarni bajaradi:

- xotirani boshqarish ;
- protsessorni boshqarish ;
- jarayonlarni boshqarish ;
- qurilmalarni boshqarish ;
- fayillarni boshqarish.

OS murakkab programma bo’lib bir nechta qismlardan (modullardan) iborat.Oddiy programma o’z ishini tamomlab qandaydir ma’lumot bersa, Os esa signallarni dinamik ketma – ketligini beradi.Bu boshqaruvchi signallar ma’lumotlarni qayta ishlash jarayonini boshqaradi. Hozirgacha bir qancha turdagi OS lar ishlab chiqilgan.Ular turli mashina, arxitektura va platformalar,hisoblash tizimlari oilalari uchun mo’ljallangan.Mavjud hisoblash tizimlarni izohlarga bo’lsak har bir guruh uchun ko’plab OS ishlatiladi.

Masalan : Shaxsiy kompyuterlar uchun : MS – DOS, Windows, OS/2warp, OS MAC,

Ishchi stansiyalar uchun : Unix , Solaris ,Linux,…

Super – EHM lar uchun : UNICOS , THE , System V,…

SHEHM uchun sistemaviy programma taminotini 3 ta asosiy guruhga bo’lish mumkin : aparaturaga o’rnatilgan (bog’liq) programmalar (fermvare), aparatsion sistemalar (operating system) va programmalash sistemalari (muhitlari) – PS (programming system), fermvere doimiy xotira (PZU- ROM), aparaturada saqlanadi va juda kam holatlarda buzuladi.

ShEHM aparatsion sistemalari turli tipda bo’lishdan qatiy nazar, ularning umumiy konfiguratsiyasida **fayl qism sistemasi tashqi qurilmalar drayverlari va komandali til protsessori** mavjuddir.

Shaxsiy kompyuterlar muhitida ishlatiladigan OS lardan bazilarini ko’rib chiqamiz va strukturasini tahlil qilamiz.

MS-DOS (Microsoft Disk Operating System) 16 razriyadli ShEHEM lar uchun asosiy OS hisoblanadi.

Bu tizim quydagi asosiy modullardan iborat.

1.Kiritish / chiqarishning asosiy (bazaviy sistemasi(BIOS Basic Input/Output System, BCVV-Базавая система ввода (вывода). Bu model boshqa modullardan

shu bilan farq qiladikim u doimiy xotirada (ROM- Read Only Memory, ПЗУ-постояанное запоминающе устройства) saqlanadi. BIOS mashina qurilmalarini avtomatik ravishda tok bilan ulangandan keyin tekshiradi. Uning ikkinchi funksiyasi (asosiy ishi) DOS ning keyigi moduli- boshlang'ich yuklash blokini tashqi xotiradan operativ xotiraga yuklashdir. BIOS sistrma uzilishlarini ham qayta ishlaydi. **Uzilish** (интеррипт прерывание) – bu bajarilayotgan jarayonni unga nisbatan tashqi hodisa tasiri asosida vaqtinchalik to'xtatilishiga aytiladi. Asosan programmaviy va apparat uzilish turlari mavjud.

2. Boshlang'ich yuklash bloki (Boot Record, Блок начальной загрузки-БНЗ).БНЗ tashqi xotiradan (sistrmaviy diskdan) MS-DOSning yana ikkita modulini yuklaydi.1)biosning kengaytirish mo'duli – BIO.COM va ikkinchisi uzu-lishlarni qayta ishlash – MSDOS.SYS modullaridir.

3. Kamandali protsessor (command.com). Bu maxsus modul – programma bo'lib kloviatura yoki kamandali fayillardan kamondalarni qabul qiladi va bajaradi. DOS tashqi programmalari, amaliy programmalar va maxsus fayl – autoexec.bat faylini ham bajaradi .

4. MS – DOS ning utilitalari – yordamchi programmalar. Bu programmalar maxsus ishlar, masalan, diskni farmatlash (farmat.com), tekshirish (chudsk.com), chop etish (print.com) va boshqalarni bajaradi.

Ko'rsatilgan 1 – 3 modullar birgalikda aperativ xotiradan 60 Kb joy egallaydi.

Bu tizimda OS uchun va amaliy programmalar uchun 640 Kb xotiraga murojat qilish ushun ruxsat beriladi. DS – DOS ning oxirgi (6,6.22) versiyalarida xotiraning yuqori va kengaytirilgan sohalaridan (qismlaridan) foydalanish mexanizmlari kiritildi.

MS-DOS modullari quydagi tartibda ish boshlaydi.1)Bios 2)IO.SYS, MSDOS.SYS 3)COMPIG.SYS 4)COMMAND.COM 5)AUTOEXEC.BAT Keyinchalik .com, .exe, .bat.

Windows. Bu oilaga kiradigan tizimlarni quydagi xususiyati mavjud :

- tizim va yangi programmalar juda oson o’rnatiladi;
- foydalanuvchi interfaysini yangi turi – grafik interfays ishlatiladi ;
- fayl nomi uzum shakilda (256 simvolgacha) tashkil qilish mumkin ;
- yangi qurulmalar (plug and play – “o’rnat va ishlat” pirinsipi asosida) juda yengil kiritiladi ;
- 32 raziryatli multi masalalik (мультизадачность) bir nechta programmalarni birgalikda ishlashni taminlaydi ;
- Multimedia, telecominikatsiya, internet va masshtablanuvchi shriftlarni qollash.

Windows aparatsion tizimi (N 95,98,2000,…) quydagi asosiy qsimlardan (modullardan) iborat.

1.Windows yadirosi

2. 32 bitli foydalanuvchi interfaysini boshqaruvchi qobiq – programma (проводник –yo’l boshchi)

3.Reestr

4.Konfiguratsiya dispetcheri

5.Vertual mashina dispetcheri

6.Fayl qism sistemasi dispetchiri

7.Ilovalarni boshqarish moduli

Yadro. Boshqaruvchi (asosiy) programma bo’lib o’z navbatida uch qismdan iborat.

a) Karnel moduli. Kiritish – chiqarish, xotirani taxsimlash va jarayonlarni boshqaradi .

b) User moduli. Sichqoncha, kloviatura, portlar, taymer boshqaradi, grafik interfeys elementlari (deraza, memyu, piktogramma) ni akslantiradi.

v) GDI (Gaphik Divice Interface – Grafik prodseduralar (qurulmalar) interfeysi).Grafik prodseduralar ishini boshqaradi, manitor va pirintir drayverlari bilan hamkorlik qiladi.

Drayver maxsus sistemaviy programma bo'lib aperatsion sistema va fizik qurulma hamkorligini taminlaydi.

Drayverlardan, masalan, manitor, kloviatura, pirinter, manikulyatorlar drayverlarini misol keltirish mumkin.

Provodnik . Bu modul (EXPLORER) asosan fayllar ustida amallar (funksialar) bajarish uchun ishlatiladi . Funksialardan fayllarni koryalash, ko'chirish, maxsus belgilar (yorliq,yarlik) bilan belgilash va boshqa amallarni ijro etish mumkin

Reestr.(registry) Bu modulda qurilmalar, tizm konfiguratsiyasi va ilovalar haqida ma'lumot saqlanadi. Ma'lumotlarni o'qish va yozish uchun reestr redaktori (registry editor), yani regedit.exe ishlatiladi.Reestr asosan ikkita fayldan iborat

a)User.dat – foydalanuvchilar haqida ma'lumotni saqlaydi

b)System.dat – aniq qurilmalar va shu ShEHM haqida ma'lumot saqlaydi

Windowsning har bir marta ishga tushirilganda (a) va (b) fayllarning nusqasi (user.dao va system.dao) hosil bo'ladi.Agar tizm noto'g'ri yuklansa restrni tahlil qilish mumkin.Agar ilovalar 16 – razryadli bo'lsa ma'lumotlarning bir qismi Win.ini, system.ini va Wingile.ini fayllariga ham joylashtiriladi.

Konfiguratsiya dispetcheri. Dispetcher sistemani konfiguratsiyalash (ish muhitini tashkil qilish) va Plug and Play texnalogiasini tadbiq etib qurilmalarni ishini boshqarish uchun ishlatiladi. Bu dispecher asosiy dispetcher hisoblanadigan virtual mashina dispetcheri VMM 32 ning bir qismi hisoblanadi.

Vertual mashina dispetcheri .Bumodul tizim jarayoni va ilovalarning har biri uchun resurslar ajratadi va quydagi ishlarni bajaradi.

a) Protsestor vaqtini multi masalali rejimni tashkil qilish uchun ajratadi ;

b) svopping, yani operativ xotiradan tashqi xotiraga va teskari, tashqi xotiradan operativ xotiraga xotira sahifalarini ko'chiradi ;

v) kerakli paytda MSDOS rejimiga o'tib tizim resurslarini monopol egallashga sharoit yaratadi.

Fayl sistemasi dispetcheri. Fayl qism sistemasining kompanentalariga murojat qilish uchun maxsus dispatcher (IFS – Installable File System – диспетчер устанавливаемой файлвой системы) ishlatiladi.Bundan tashqari qism sistemaga quydagi drayverlar ham kiradi :

- 32 bitli FAT drayveri ;
- 32 bitli SD – ROM (SDFS) drayveri ;
- 32 bitli redirektor (fayllarni izlash, ochish, o'qish, yozish va o'chirish drayveri) , yani VREDIR.VXD. Bu drayver Windows NT serverga ulanish uchun ham qo'llaniladi ;

- 32 bitli NWREDIR.VXD rederektori orqali Novell Net Ware tarmoq serverlariga ulanish mumkim ; (Applications – приложения - ilovalar).

- Blokli kiritish – chiqarish qism sistemasi har bir ilova virtual mashina shaklida tashkil qilinib maxsus programmasi tamonidan boshqariladi.

Sinov savollari :

1. Operatsion sistema qanday funksiyalarni bajaradi ?
2. MS – DOS tizimi modullar to'plamini keltiring .
3. Uzilish nima ?

4. BIOS – ning asosiy vazifalari nimalardan iborat ?
5. FAT jadvallar nima uchun ishlatiladi ?
6. Konsol dirayveri qanday ishlarni bajaradi ?
7. Windows muhitining xususiyatlarini tushuntiring .

Adabiyotlar

[7], [12], [13], [17], [19]

Ma’ruza № 3

Mavzu : Tizimni konfiguratsiyalash usullari

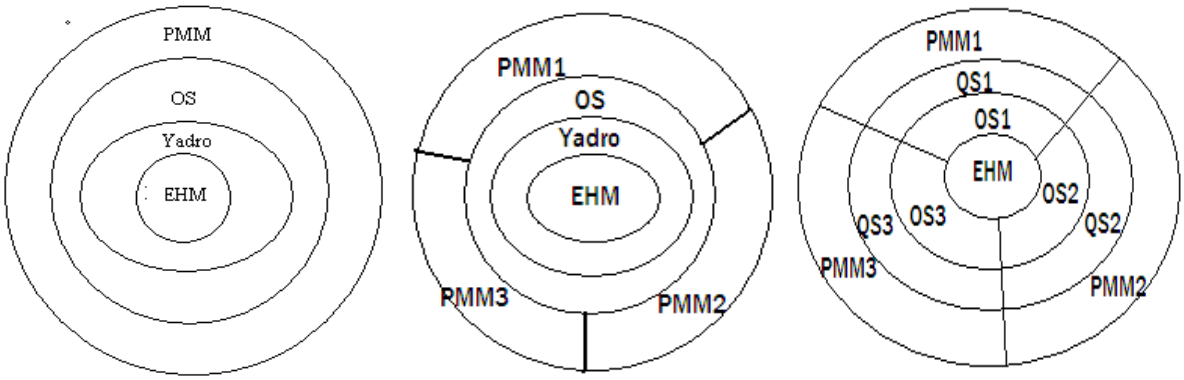
Reja :

1. Operatsion tizim muhiti
2. Muhitni tashkil qilish sxemasi .
3. Config.sys va autoexec.bat fayillari .
4. Windows sistemasi ish muhitini tashkil qilish .

Tayanch iboralar : Operatsion muhit , muhit sxemasi konfiguratsiya fayillari, boshqarish paneli , parametrlarini o’rnatish , interfeys .

Ma’ruza bayoni

Har bir operatsion sistema atrofida bir nechta operatsion muhitlar (операционная среда, operational enviroument) tuzish mumkin. Har bir muhit o’zining foydalanuvchilar guruhi va shu muhitda yechiladigan masalalar to’plamiga ega. Muhitni problema(muammo)ga mo’ljallangan muhit ham deyiladi. Operatsion qism tarkibibida bir nechta qism sistema (subsystem) ham ajratish mumkin. Operatsion muhitlar tashkil qilishda asosan quyidagi sxemalardan foydalanish mumkin.



Rasm. OS tuzish va qo’llash usullari.

Rasmda aylanalarga turli pog'onalar (EHMdan toki muammoga mo'ljallangan muhitgacha) ga to'g'ri keladi. (a) va (b) sxemalarda OS dan yadro ajratiladi. Bu yadro (Kernel) apparatura funksiyalarni davom ettiradigan maxsus funksiyalar to'plamidir. Yadro asosida turli tipdagi qism sistemalar tashkil qilinadi.

OS yadrosi va qism sistemalarning asosida monitor (boshqaruvchi programma) yotadi.

Sxemaning (b) qismida OS ning funksiyalari yana ham ko'proq maydalab taqsimlangan. Bitta EHMda bir vaqtning o'zida bir nechta qism sistemalar ishlaydi, har bir qism sistema o'zinin OS iga ega bo'lishi mumkin. Bunday mexanizmga vertual mashina (virtual machine) deyiladi. Bu holatda bir nechta OS larning parallel ishlashini **virtual mashinalar interfeysi** mexanizmi muofiqlashtiradi(koordinatsya qiladi).

SHEHM foydalanuvchisi uchun qulay ish muhitini tashkil qilish OS ga asoslangan holda konfiguratsyalash (ish muhiti) va boshlang'ich parametrlarni

o'rnatish katta rol o'ynaydi.

MS-DOS tizimida ikkita config.sys va autoexes.bat fayllari ish muhitini tashkil qiluvchi va asosiydir. Hamma vaqt shu fayllar birinchi qatorda tekshiriladi va bajariladi.

Config.sys- fayli yordamida OS qulayliklari kengaytiriladi va tashqi qurilmalar parmetrlarini o'zgartirish mumkin. Yangi drayverlar qo'shiladi va ochiladigan fayllar soni ham ko'rsatiladi.

Autoexec.bat – faylida komandalar, kerakli programmalar, OS taklif formati va boshqa paketli fayllar ko'rsatiladi. Autoexec.bat fayllarni bitta yoki bir nechta foydalanuvchi ishlash uchun ham tuzish mumkin. Bu holda har bir foydalanuvchi uchun kod va boshqa identifikatorlar o'rnatish kerak. Bu malumotlar aniqlangandan keyin foydalanuvchi uchun qulay va tushunarli ish

muhiti hosil bo’ladi.

Konfiguratsya fayllarini hosil qilish uchun maxsus komandalar to’plash ishlatiladi. Ularning soni ko’p emas, har bir komanda umumiy ko’rinish, parametr va boshqaruvchi fayllarga ega bo’lishi mumkin.

Windows tizimi muhitida bu ikkitta fayl ishlatilsa ham, ular katta rol o’ynamaydi. Bu tipdagi operatsion tizimlarni konfiguratsya qilish uchun Настройка (setting) nomli panelni bosh menyudan topib ishlatish kerak.

Boshqarish paneli(панелъ управления-control panel) yordamida kerakli piktogrammani tanlab olib, ularni bajariladigan funksyalarini o’rganib asosiy parametrlarini o’rnatish mumkin. Ish muhitini tashkil qilishda quyidagilarga etibor berish kerak:

- vaqt va tovush;
- internet, klviatura va modem;
- multimediya vositalari, parol va foydalanuvchilar uchun mustaqil ish muhiti;
- printer, shriftlar, til va standartlar va hakozolar.

Sinov savollari :

1. Operatsion muhit nima ?
2. Muhitni tashkil qilish sxemalarini tahlil qiling.
3. Kofiguratsya fayillari qanday tuziladi ?
4. Interfeys turlarini tushuntiring .
5. Windowsning ish muhiti qanday tashkil qilinadi ?

Adabiyotlar

[7], [12], [13], [17], [19]

Ma’ruza № 4

Mavzu: Programmalash tizimlari

Reja:

1. Programmalsh tizimi va uning tarkibi.
2. Integrallashgan muhitlar.
3. Transliyatorlar turlari.
4. Assemblerlar va yuklovchilar.

Tayanch iboralar: programmalash tizimi, komponentalar, integrallashgan muhitlar, transliyator, kampilyatsiya, interpretatsiya, assembler.

Ma’ruza bayoni.

Sistemaviy programma ta’minotining yana bir komponentasi programmalash sistemasidir.

Programmalash tizimi programmaviy sistema bo’lib, bitta yoki bir nechta kirish (boshlang’ich) tillari yordamida programmani tuzish va ijro etishni taminlaydi.

Tizimlar bir yoki ko’p tilli bo’lishi mumkin. Ko’p tilli tizimlar yordamida biz programmani bo’laklarini har xil til yordamida loyihalashimiz mumkin. Bunday tizimlarda yangi tillarni qo’yish imkoniyati ham beriladi.

Programma nuqtai nazardan programmalash tizimi modullar to’plamidan iborat. Har bir modul maxsus funksyalar (ishlar,vazifalar) ni bajaradi. Zamonaviy tizimlar quyidagi komponentlardan iborat:

a) **Kirish tili.** Bu til yordamida programma tuziladi.

b) **Translyator.** Kirish tilida yozilgan programmani mashina yoki boshqa tilga tarjima qiluvchi maxsus programmaviy modul(tizim).

d) **Tahrirlovchi.** Programma matnini tashkil qilish va matn ustida turli

amallarni bajaruvchi qismprogramma.

e) **Taxlovchi.** Programmani tekshirish bosqichlarini shu modul yordamida bajarish mumkin.taxlashning **statik** (matn tahlili) va **dinamik** (boshlang'ich malumotlar bilan programma ishini tekshirish) turlari mavjud.

f) **Optilizator.** (Optimallovchi). Programma tarjima qilingandan keyin hamma vaqt ixcham va qulay bo'lmaydi. Hosil hosil bo'lgan **obekt modullarni** optimallashtirish (ixchamlash, to'g'irlash, unumdorligini oshirish) shu modul yordamida bajariladi.

g) **Dispetcher.**foydalanuvchi bilan muloqat qilish va boshqa modullarni ishini boshqarish uchun ishlatiladigan modul. Dispetcherni monitor ham deyiladi.

h) **Yordamchi qismsistema.**(Help subsystem)

Tizimni kirish tili, xatolar haqida ma'lumot va foydalanuvchilar guruhlari uchun turli yordamchi malumotlarni saqlaydi.

Bulardan tashqari, har bir programmalash tizimi boshqa komponentalarni o'z ichiga olish mumkin.

Integrallashgan programmalash tizimlari.

Hozirgi paytda programmalash tizimlari asosan integrallashgan (birlashgan) shaklda tashkil qilinadi. Bunday tizimlarni **Turbo Tizimlar** ham deyiladi. Aytib o'tish joyizki SHEHM turbo rejimi va turbo tizim tushunchalari bir –biridan katta farq qiladi.

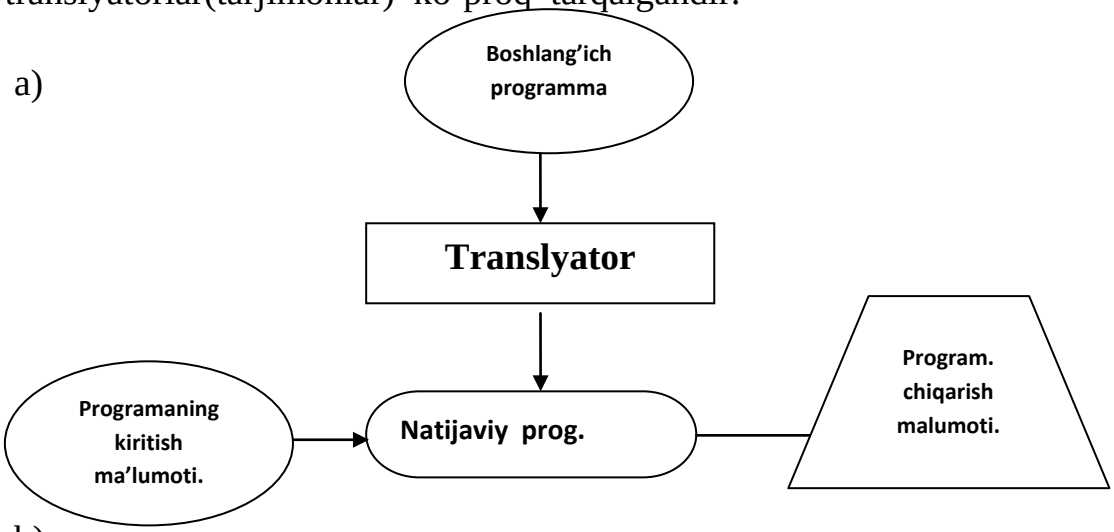
SHEHM turbo rejimi uning oddiy rejimidan farq qiladi. Bu farq takt chastotasi bilan bog'liq.(takt:prosessorning har bir komandasi bitta yoki bir nechta takt mobaynida bajariladi). Yani Turbo rejimida takt chastotasi (M Hz,G Hz) ancha ko'tariladi.

Turbotizim - integrallashgan tizim bo'lib programmani tayyorlash, taxlash va natija olish uchun tahrirlovchi,translyator, aloqalarni tashkil qiluvchi modul cheklovchi va yordamchi qism sistemalar yagona kompleks(majmua) sifatida tashkil qilingan. Tizim bilan ish menyu-muloqat shaklida olib boriladi. Turbo tizimlar 150-350 kb qo'shimcha xotirani talab qiladi. Turbo tizimlardan Turbo Pascal, Turbo Ci, Turbo assemblerni keltirish mumkin. Turbo tizimlar asosida yangi programmalash muhitlari(visual, obektga mo'ljallangan, case-texnologyalar) ishlab chiqilgan.

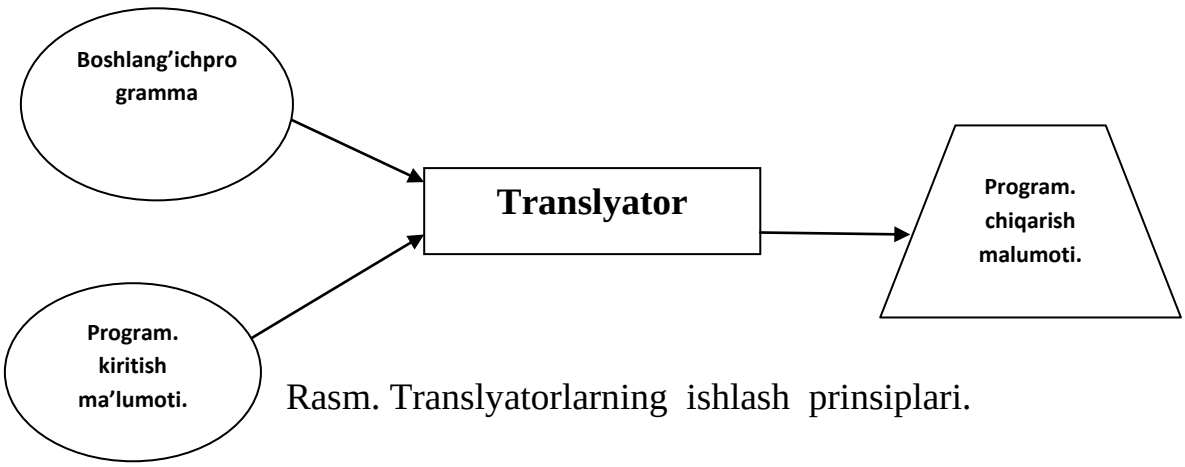
Translyatorlar turi va strukturasi.

Translyator – sistemaviy programma bo'lib programmani bir (boshlang'ich) tildan boshqa natijaviy, chiqarish) tilga tarjima qiladi. Yuqori darajali tillardan mashina yoki mashinaga mo'ljallangan tilga tarjima qiladigan translyatorlar(tarjimonlar) ko'proq tarqalgandir.

a)



b)



Rasm. Translyatorlarning ishlash prinsiplari.

Birinchi sxemada (a) programma **to’liq** tarjima qilinadi va keyin bajariladi. Ikkinchi(b) sxemada esa,programma qadamma-qadam tarjima qilinadi vabajariladi.

Birinchi sxema asosida tashkil qilingan va ishlaydigan translyatorlar kompin-syatsyalanuvchi (ya’ni komplyatorlar) translyatorlar va ikkinchi sxemadagi translyatorlar interpretatsiyalanuvchi (yani interpretatorlar) translyatorlar deymiz.

Interpretatorlar (masalan beysik, lion, refal) va komplyatorlar (PL/1, Pascal, Cu) o’ziga xos qulaylik va kamchiliklarga ham ega . Interpretatorlar sodda tashkil qilinadi va programmani tarjima qilish va bajarish jaroyonida xotirada bo’lishi zarur. Komplyatorlar murakkab bo’lsa ham programmani tarjima qilingandan keyin (.exe, .obj fayllar) operativ xotirada bo’lishi shart emas.

Translyatorlarning yana bir turi assembler yoki zagruzchik (yuklovchi) deyiladi. Ularning kirish tipi mashina yoki mashinaga mo’ljallangan tiplar geruhiga kiradi. Bu tillar yordamida programma tuzishda katta tezkorlikka (programmani mashinada bajarish nuqtai nazardan) erishiladi , ammo programmani loyihalash (ishlab chiqish) murakkablashadi, chunki biz programmani har bir kichik detalini asoslashimiz kerak.

Sinov savollari:

1. Programmalsh tizimi tarkibi nimalardan iborat?
2. Integrallashgan muhit nima?
3. Transplyatorlar turlari va qulayliklari.
4. Kampilyatsiya interpretatsiyadan qanday farq qiladi?
5. Turbo rejim va turbo-tizim nima.
6. Assemblerlar xususiyatlarini tushuntiring.
7. Turbo-tizimlarga misollar keltiring.

Adabiyotlar

[1], [5], [9], [11], [16], [17]

Ma’ruza № 5

Mavzu: Transplyatsiyalash texnikasi

Reja:

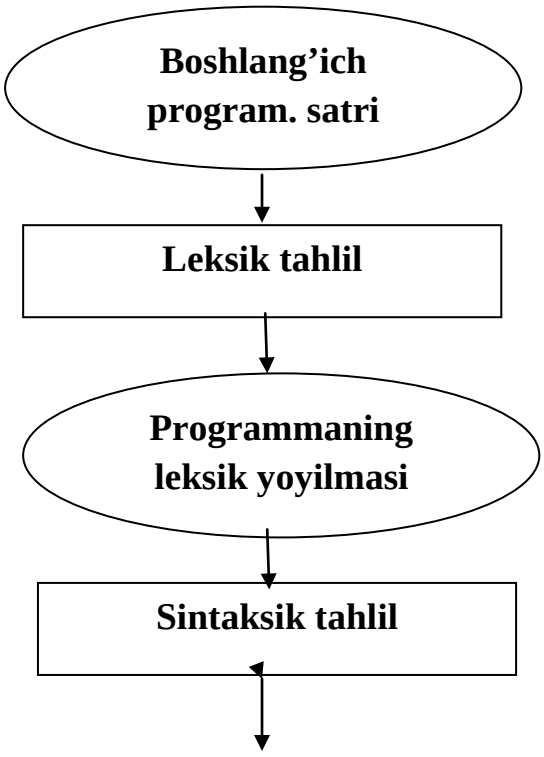
1. Transplyatsiya sxemasi.
2. Tahlil va sintez.
3. Leksik va sintaksik tahlil.
4. Generatsiya vazifalari.
5. Optimallashtirish va xatolar turlari.

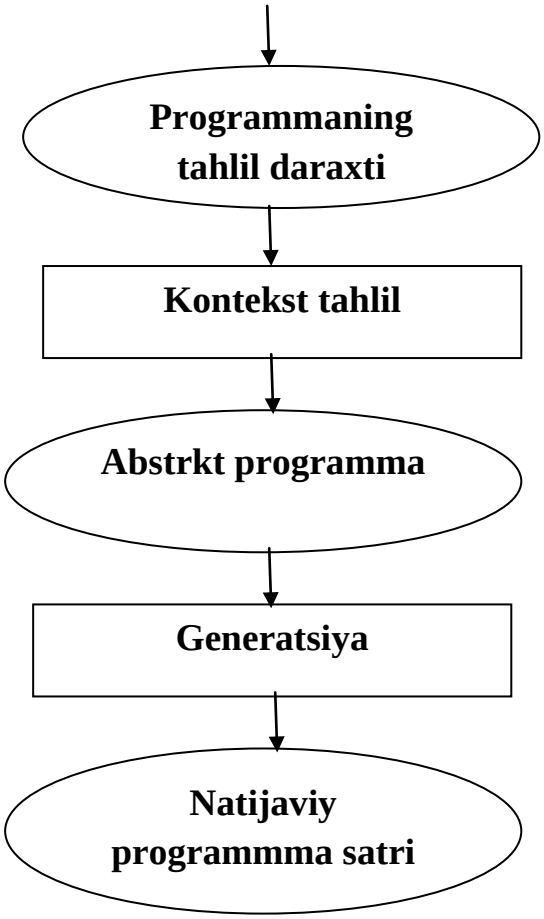
Tayanch iboralar: Sxema, analiz va sintez , leksika, sintaksis, semantika, generatsiya, optimallashtirish, xatolar turi.

Ma’ruza bayoni.

Boshlang’ich (kirish) tilida yozilgan programmani mashina tiliga tarjima qilish masalasi oddiy masala emas. Bu masala ketma-ket, bosqichlar bilan bajariladi. Tarjima qilish jarayoni quyudagi umumiy sxema yordamida tasvirlanadi.

-





Rasm . Translyasiya sxemasi.

Birinchi va ikkinchi bosqichlarni bajarishdan keyin boshlang'ich programmadan abstrakt programmani hosil qilamiz. Sxemadan ma'lumki tarjima jarayoni asosan ichki (analiz va sintez) bosqichlardan iborat. Birinchi bosqichda abstrakt programma va ikkinchi bosqichda unga ekvivalent bo'lgan natijaviy programma hosil bo'ladi.

Leksik analiz jarayonida translyatr kirish satrini litera bo'yicha o'qib leksemalarni hosil qiladi. Leksemalar jadvalda saqlanadi. Har bir leksema deskriptorga ega bo'lib, **deskriptor** leksema turi va saqlash joyini belgilaydi.

Sintaksis_ tahlil natijasida leksemani til sintaksisiga jabob berish yoki bermasligi aniqlanadi. Masalan ,a+b*c ifodasi tahlildan keyin (a+(b*c)) ko'rinishda keltiriladi. Buning natijasida programmaning ma'nosi yaqqol ko'rsatiladi. Tahlilning bu bosqichi natijasi programmani tahlil daraxtidir. Daraxt konstruksiya va leksemalarni o'z ichiga oladi. Keying bosqichda esa bu daraxt

maxsus algoritmlar bilan aylanib o’tiladi va natijaviy programma satri hosil bo’ladi.

Generatsiya bosqichida natijaviy programma satrlar to’plami ko’rinishida hosil qilinadi.

Masalan x:=b+(c-d)*(e+f) operatori berilgan bo’lsin. Bu operatorni generatsiya qilingandan keyin quyidagi matn hosil bo’ladi.

t1:=c-d ; t2:=e+f ; t3:=t1*t2 ; t4:=b+t3 ; x:=t4;

yoki shartli operator

agar x>y u holda a:=1 aks holda b:=2; generatsiyadan keyin quyudagi ko’rinishni oladi

p:=x>y;

agar p **uholda o’ting** M1;

o’ting M2;

M1:a:=1;

o’ting M3;

M2:b:=2 ;

M3:…..

Generatsiya etapining yana asosiy vazifalaridan biri – **bu xotirani taqsimlashdir.** Bu yerda statik yoki **dinamik** taqsimlash usullari ishlatiladi.

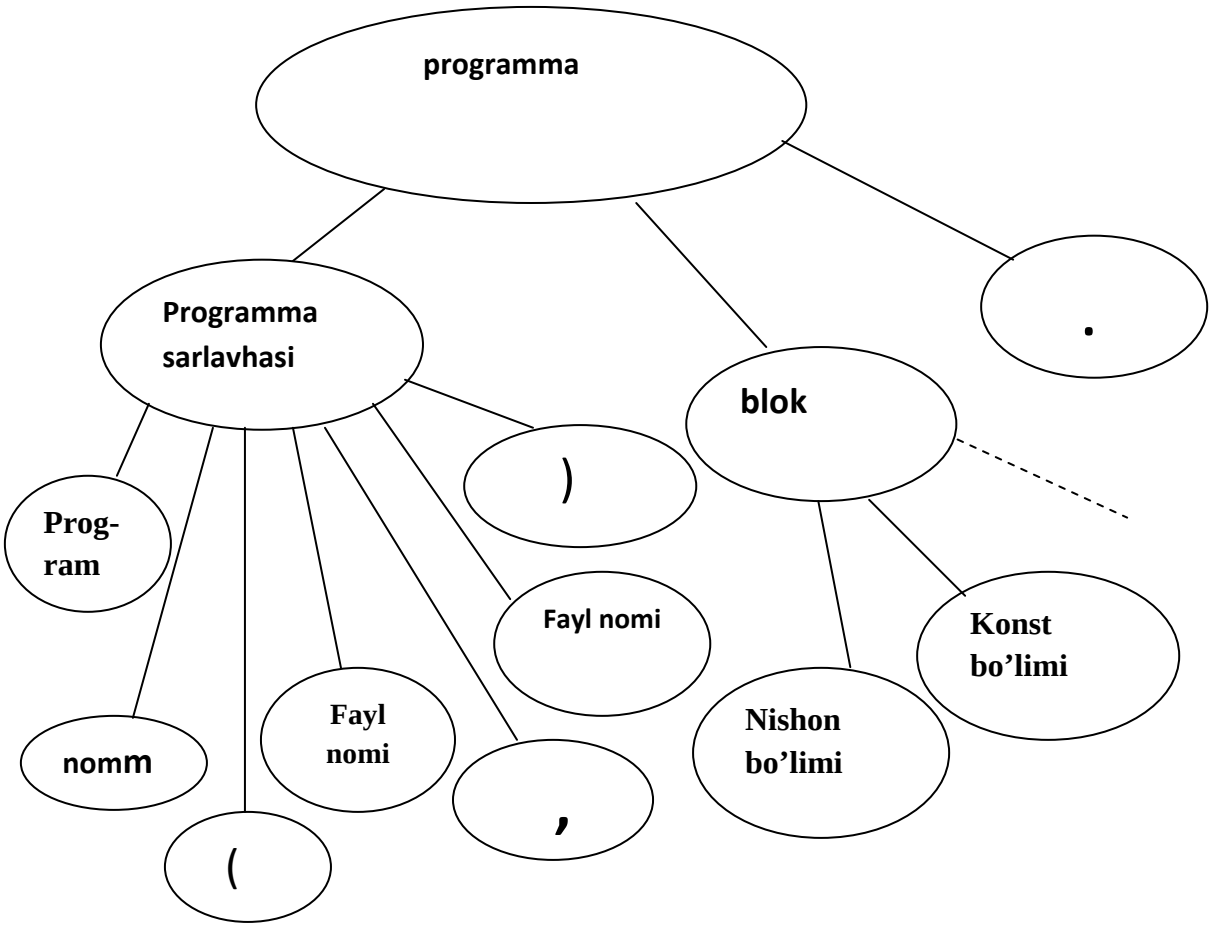
Birinchi usul xotirani translyatsiya jarayonida taqsimlashni talab qiladi. Agar programma obektlarning paydo bo’lishi va aktivlanishi ma’lum bo’lmasa dinamik taqsimlash usulidan foydalanadilar.

Faraz qilaylik ,quyudagi oddiy paskal programma berilgan bo’lsa ,uning leksik analizi belgi bilan chizilgan leksimalarni hosil qiladi.

```
Program  P;
Const m=1.5;
Var a,b:real;
Begin   read(a,b);
a:=a+b+m ; write(a)
end.
```

Sintaksik tahlil jarayonida tushunchalar strukturasi aniqlanadi va uning tashkil etuvchilari ajratiladi . Sintaksik daraxt ko'rinishi paydo bo'ladi .

Programma konstruksiyasining sintaksik daraxti quyudagichadir.



Semantik tahlil jarayonida biz programmada ishlatilgan nom va tasvirlashlar qoidalarga ishlatishi yoki mos kelmasligini aniqlaymiz .

Masalan , var a,b:real; yozuvi a va b o’zgaruvchilar, tipi haqiqiy , yoki m-konstanta, ‘+’ – qo’shish amali va u haqiqiy elementlarini qo’shishda ishlatilayotganini bildirishi kerak.

Tahlil bosqichlari bilan bog’liq ularga mos xatolar turi ham mavjud. Shuning uchun leksik ,sintaksik va semantik xatolarni ajratadi.

Leksik xatolar alfavit simvollaridan leksimalarni hosil qilishda qoidalarni buzilganligi bilan bog’liq. Masalan , const o’rniga cons yoki idintifikator 2AB shaklida yozilsa leksik xatoga yol qo’yilgan bo’ladi.

Sintaksik xatolar tilning sintaksik qoidalarini noto’g’ri ishlatilishi bilan xarakterlanadi. Masalan ,const n=5 ; var x,y:real; tasvirlash mavjud bo’lsin . Operatorlar n:=n+1; 22:=x+y; tilning qoidalariga javob bermaydi va translyator sintaksik xatolar haqida ma’lumot beradi.

Semantik xatolar obekt xususiyatini aniqlash mumkin bo’lmagan hollarda yoki obekt xususiyati obektni ishlatish jarayoniga qarshidir. Masalan , obekt tasvirlanmagan yoki tasvirlangan, ammo ishlatilishi noto’g’ri.

Xatolarning yana bir turi – bu hisoblash jarayonidagi, yani programma ishlatayotgan paytdagi xatolardir . bularni fatal xatolar deymiz. Masalan “nolga bo’lish” , massiv o’lchovlaridan chiqib ketish, stek to’lishi kvadrat ildiz ustida manfiy ifodani paydo bo’lishi va hokozolar.

Translyasiyalash texnikasida yana bir muhim masala-bu hisoblashlarni optimallshtirishdir. Foydalanuvchi asosan xotira qurilma va protsessor vaqtini ishlatadi. Ana shu resurslarni optimallshtirsak bitta emas bir nechta programmalarni qo’llash va ularga tejalgan resurslarni ishlatish sharoiti paydo bo’ladi. Umuman olganda xotira va vaqt asosan optimizatsiya qilinadi. Bu ikki komponentani optimallshtirish bir-biriga ziddiyat ko’rsatadi . Programmani optimallshtirish jarayonida quyudagi usullar ishlatiladi:

- Bazi bir amallarni translyatsiya paytida ijro etish ;

- Sikl, prosedurani chaqirish, indeksatsiyalash kabi amallarni asoslangan variantini ishlatish;
- Maxsus registrlardan foydalanish ;
- Mashinaga bog’liq va bog’liqmas optimallshtirish.

Sinov savollari:

1. Tarjima qaysi bosqichlardan iborat?
2. Leksik va sintaksik tahlil vazifalarini tushuntiring.
3. Semantik tahlil nima?
4. Generatsiyaga misollar keltiring.
5. Optimallashtirishni qanday usullarini bilasiz?
6. Sintaksik xatoni tahlil qiling.
7. Fatal xatolar haqida ma’lumot bering.

Adabiyotlar

[1], [9], [10], [12]

Ma’ruza № 6

Mavzu: Ta’minotning boshqa komponentalari

Reja:

1. Programma va mashinaga bog’liqlik.
2. Assemblerlar va disassemblerlar.
3. Assembler bajaradigan vazifalari.
4. Zagruzchiklar va makroprotsessorlar.

Tayanch iboralar: assembler, yuklovchi, makros, psevdokamanda, disassembler, direktiva, tarjima bosqichlari, makroprotsessor.

Ma’ruza bayoni.

Sistemaviy programma ta’minotining yana bir xususiyati mashinaga bog’liqligidir. Shuning uchun bu tipdagi programmalar mashina strukturasi va uning komponentalari hisobga olgan holda ishlab chiqariladi .

Masalan, assembler mnemenik kodni mashina kodiga tarjima qilishda komandalar formati, adreslash usullari va mashinaning boshqa apparat xarakteristikalarini hisobga oladi. Kompilyator esa, mashina kodiga boshlang’ich programmani tarjima qiladi, registrlar komandalar to’plamidan foydalanadi . Shularni hisobga olib sistemaviy programma ta’minoti boshqa komponentalari: assembler, yuklovchi, makroprotsessorlarni xususiyatlarini ko’rib chiqamiz.

Assembler. Assembler programmalash tizimi bo’lib assembler tili va shu tilni tarjima qiladigan translyatordan iborat. Yana assembler tilidan mashina tiliga programmani tarjima qiladi. Programmalashda **disassembler** ham ishlatiladi.

Disassemble maxsus progrmma bo’lib obyekt kodni unga ekvivalent bo’lgan assembler – programmaga o’tkazadi.

Assemblerlar asosan mashina oilasiga (platformaga) mo’ljallangan holda ishlab chiqariladi. Masalan , Intel (IBM PC) oilasiga mansub assembler, Apple

(Mac , Matorolla, Zilog) oilasi uchun ishlab chiqarilgan assembler yoki VAX arxitekturasiga mutobiq assembler va h. shularga qaramasdan, assemblerlarning bajariladigan ishining umumiy to'plamini quyudagicha deb qarash mumkin.

- 1.Amallar mnemonik kodini ularga ekvivalent bo'lgan mashina kodiga o'tkazish.
  Masalan , add ni 10 ga o'chirish.
  2. simvolik operandlarni ekvivalent bo'lgan mashina adreslarga o'tkazish.
  Masalan , alpha ni 1243 ga o'chirish.
  3. maxsus format asosida mashina komandalarini tuzish.
  Masalan ,$88/$8e/NWords ←MOVCX,bp+NWords.
  4. boshlang'ich programmadagi konstantalarni mashinaning ichki tasvirlanishiga o'tkazish.

  Masalan , edf→45F446

5. Hosil bo'lgan obekt programmani xotiraga yozish va listingni chiqarish.

Bulardan tashqari, assembler direktivalari ham programmada ishlatiladi. Bunday direktivalarni **psevdokomandalar** ham deyishadi. Direktivalar mashina tiliga tarjima qilinmaydi, assemblerni ishini boshqarish uchun ishlatiladi.

Masalan, byte, word, start, enol direktivalarini qo'llash mumkin.

Assembler programmani mumkin bo'lgan strukturalaridan birini ko'rib chiqamiz.

```
model small
stack zooh
data
  ;ma'lumotlarni e'lon qilish.
```

code

start: ; yoki bunga o’xshagan boshqa nishon programma komandalari

end start

model – xotira modelini belgilaydi. Asosan 6 xil (tiny, small (kichik), compact, medium, large, huge) modellarni ishlatish mumkin. Small modelini (ya'ni 2 ta ((NK baytli) xotira segmentlari) ishlatish mo’ljallangan. Stack – direktivasi 300h (300) hajmli stek ajratadi.

data – direktivasi ma'lumotlarni e'lon qilish sohasini va **code direktivasi** esa programma kodini sohasini aniqlaydi. Start nishoni programmani ishga yurgizish nuqtasini, end – direktivasi esa programma kodini yakunlanishini ko’rsatadi.

Assembler komandasining umumiy ko’rinishi <nishon> <komanda/ direktiva> <operandlar>

<;izoh>

Masalan,

nishon: mov AX,BX; BX ning qiymati AX ga

Bu erda nishon – nishon (yuklanadi, ko’chiriladi hamma vaqt ishlatilishi shart emas), mov – ko’chirish komandasi, AX, VX – operandlar (registrlar), ; - dan keyin izoh berilgan. Bu komanda VX ni qiymatini AX ga ko’chirish komandasidir.

Ko’pgina assemblerlarda boshlang’ich programmani tarjima qilish jarayoni ikki bosqichda bajariladi. **Birinchi bosqich.** Nomlarni aniqlash (adreslash, nishonlarni saqlash, ba'zi bir direktivalarni bajarish). **Ikkinchi bosqich.** Komandalarni translyatsiya qilish va obekt kodlarni generatsiya qilish (translyatsiya, ma'lumotlar generatsiyasi, bajarilmagan direktivalarni bajarish, obekt kodni yozish va listing chiqarish).

SHEHM muhitida asosan TURBO assembler va makroassembler (TASM va MASM) ishlatiladi. Har birining bir qancha versiyalari mavjud.

Zagruzchiklar (Yuklovchilar). Obekt programma tarjima qilingan komandalar boshlang’ich ma'lumotlar qiymatlari, xotira adreslarini saqlaydi. Bu jarayonlar uchta bosqichni o’z ichiga oladi.

1. Zagruzka (yuklash). Programmani bajarish uchun operativ xotira joylashtirish.

2. Siljitish (peremehenie). Obekt programmani adreslarni o’zgartirish natijasida modifikatsiya qilish.

3. Bog’lash (svyazyvanie). Ikkita va undan ko’p alohida tarjima qilingan obekt kodlarni bog’lash va ular orasidagi bog’lanishni tashkil qilish.

Zagruzchik (Yuklovchi) sistemaviy programma bo’lib yuklash vazifasini bajaradi. Ba'zi bir yuklovchilar ko’rsatilgan (1-3) ammalarni ham bajarishi mumkin. Bog’lovchi programmalarni bog’lanish (aloqa) muharriri ham (redaktor svyazey) deydilar.

Zagruzchik va bog’lovchi programmalar o’z funksiyalarini turli tillarda yozilgan va tarjima qilingan kodlar ustida bajaradi. Zagruzchiklarning turli bo’lishidan qat'iy nazar, ularning ikkita asosiy funksiyalari mavjud.

a) Obekt programmani operativ xotiraga yozish.

b) Boshqaruvni birinchi bajariladigan komanda adresiga uzatish.

Zagruzchiklarni programmalashda (ishlab shiqishda) overley texnologiyasidan foydalananadilar. Bu holatda xotirani tejash mumkin va ikki bosqichni ish uchun xotiraning bitta sohasi ishlatiladi.

Assembler tilida programma tuzish va uni taxlash (bajarish) uchun assembler translyatori va zagruzchik (yuklovchi) birgalikda ishlatiladi. Shu bosqichlarni quyidagi ketma-ketlikda berish mumkin.

1) asm prog. asm. Boshlang’ich fayl.

2) prog. obj. Translyatsiya qilingan obekt fayl (obekt modul)

3) link prog. Bajariladigan faylni zagruzchik (linker) yordamida hosil qilish.

4) prog. exe – Bajariladigan fayl.

Agar programma bir nechta obekt modullardan iborat bo’lsa u holda ularni zagruzchik orqali yagona programma shaklida tashkil qilish mumkin.

Masalan, link mod1.obj+mod2. obj+mod3. obj

Natijada mod. exe yagona bajariladigan programma hosil bo’ladi.

Makroprotsessorlar. Programma yaratish jarayonida kirish (asosiy) tilining ba'zi bir konstruktsiyalari to’plami juda ko’p hollarda takrorlanishi va ishlatilishi mumkin. Shu to’plamni makroinstruktsiya (makrobuyruq, makroko’rsatma, makrodastur) deydilar.

Makroprotsessor ham maxsus sistemaviy programma bo’lib makroinstruktsiyalarni asosiy til konstruktsiyalariga tarjima qiladi. Shu jarayonni makrogeneratsiya deb nomlaydilar. Makroprotsessorlar mustaqil yoki programmalash tizimi tarkibida bo’lishi mumkin. Ularning ishi jarayonida simvollar yoki satrlarning bir guruhi boshqasiga almashtiriladi. Ko’p hollarda makroprotsessorlar assembler tilida programma tuzish jarayonida ishlatiladi. Makroprotsessorlar aniq bir til bilan bog’liq yoki umumiy (bog’liqmas) bo’lishi mumkin.

Assembler tilida makrota'riflar (makroopredelenie) mexanizmi qism programmalarga (podprogrammy) o’xshashdir. Har bir makrota'rif 3 ta qismdan iborat.

a) Sarlavha – psevdooperator macro. Uning nishon maydonida makrota'rif nomi ko’rsatiladi. Makrota'rif formal parametrlarga ham ega bo’lishi mumkin.

b) Tana – assembler operatorlarning ketma-ketligi.

d) Tugash – endm – psevdooperatori.

Masalan, so’z (mashina so’zi – masalan, 16 bayt) kattaligidagi qiymatlarni yig’ish uchun ishlab chiqarilgan makrota'rif quyidagi ko’rinishda bo’ladi.

```
add_words macro term1, term2, sum
        mov ax, term1
        add ax, term2
        more sum, ax
        endm
```

Agar xotiraning ikkita yacheykasidagi ma'lumotni qo’shish kerak bo’lsa biz bu makrota'rifga add_words alpha, beta, gamma shaklida murojaat qilsak makroprotsessor (assembler) bu operatorni o’rniga programmaga quyidagi komandalarni kiritadi.

```
        mov ax, alpha
        add ax, beta
        mov gamma, AX
```

Yoki xuddi shunday, add_words bx, cx, dx shakldagi operatorni assembler ikkita registrdagi ma'lumotni qo’shish uchun ishlatiladi.

```
        mov ax, bx
        add ax, cx
        mov dx, ax
```

Makrota'riflar prosedura (funksiyalardan) quyidagi xususiyatlari bilan farq qiladi.

1) Makrota'riflar dinamik xarakterga ega. Proseduralar esa faqat ma'lumotlarni o’zgartiradi.

2) Makroprotsessor protsedurani chaqirishi va undan qaytish ishlarni bajarmaydi.

3) Makrota'riflar kutubxonasi ham katta qulayliklarga ega.

Sinov savollari:

1. Mashinaga bog’liq tillarga misol keltiring.
2. Assembler vazifalari nimadan iborat?
3. Assemblerlash algoritmini tushuntiring.
4. Tarjimaning ikki bosqichini tahlil qiling.
5. Yuklovchi nima?
6. Makroprotsessor ta’rifini bering.
7. Makroslarga misol bo’ladigan fragmentlarni yozing.

Adabiyotlar

[2], [3], [4], [6], [10], [12]

Ma’ruza № 7

Mavzu: Interfeysni tashkil qilish

Reja:

1. Interfeys tushunchasi va unga qo’yiladigan talablar
2. WIMP- interfeyslar
3. Interfeysni standartlashtirish
4. Instrumental vositalar

Tayanch iboralar: interfeys, interfeys turlari, muloqotni loyihalash, WIMP-interfeyslar, piktogramma, IBM-standarti, standartlash.

Ma’ruza bayoni

Foydalanuvchi iterfeysi va unga qo’yiladigan talablar.

Ixtiyoriy programma tizimini ikkita kriteriya asosida baholash mumkin. Birinchisi, **aniqlik,** ya’ni berilgan ma’lumotlar asosida aniq natijalar olish. Ikkinchisi, **qulaylik,** ya’ni tizim yoki programma bilian ishlash qulayligi. Bu xususiyat asosan tizim bilan foydalanuvchi orasidagi hamkorlik- foydalanuvchi interfeysi bilan xarakterlanadi [4].

Programma ta’minotini yaratishda interfeys elementlarini, ya’ni kiritilishi va chiqarilishi kerak bo’lgan ma’lumotlar to’plami va ko’rinishini alohida loyihalash maqsadga muvofiq. Muloqot ko’rinishini tabiiy, foydalanuvchiga tushunarli va ixcham tashkil qilinadi.

Agar tizim murakkab iyerarxik strukturaga ega bo’lsa, u holda foydalanuvchi ixtiyoriy paytda strukturaning qaysi bosqichida ishlayotganligini bilishi kerak.

Interfeysga qo’yiladigan talablarning yana bittasi bu **adaptatsiya konsepsiyasidir,** ya’ni muloqot strukturasini foydalanuvchi talablariga **mutobiqlanishi** hisoblanadi.

Foydalanuvchi interfeysini loyihalashda muloqotni tasvirlash katta rol o’ynaydi. Bu yerda quyidagi talablar va prinsiplar ishlatiladi.

a) Ekran maketi. Ekranning (monitorning) pozisiya va formatlar maydoni. Maket muloqotning bir fragmentini akslantiradi.
b) O’tish turlari yoki holatlar diagrammasi. Har bir holat graf uchlari bilan xarakterlanadi. Uchlar, ya’ni nuqtalar tizimining foydalanuvchiga yoki foydalanuvchining tizimga beradigan ma’lumotlarni bildiradi. Murakkab tizimlarga muloqot to’rlari qismiy to’rlarga ajratiladi.

O’tish to’rlarida ularning uchta turini ajratish mumkin:

- Ma’lumotlarning kiritilishini talab qiladigan uchlar;
- Ma’lumotlarning kiritilishini talab qilmaydigan uchlar;
- Ma’lumotlarning kiritilishi talab qilinadi, ammo boshqaruv doimo hamsoya uchlga uzatiladi.

c) O’tish shartlari asosan kirish tilida yozilgan fragmentlar yordamida tashkil qilinadi. Ana shu fragmentlarni leksik, sintaktik va semantik tahlil qilish natijasida muloqot sxemasi boshqariladi.

Leksemalarga klaviaturaning bosilgan klavishasi (tugmachasi) kodi, menyu punktlari nomerlari va hokazolar kiradi.

Semantik tahlil jarayonida gap va iboralarning ma’no jihatdan to’gri yoki to’g’ri emasligi aniqlanadi.

Sintaksis va sintaktik tahlil qoidalar to’plami va gramatik konstruksiyalarni ishlatib ularning mumkin yoki emasligi aniqlaydi

Hozirgi paytda ko’rsatilgan va qushimcha talab va prinsiplar asosida foydalanuvchu interfeysini loyihalashning abtomatlashtirilgan tizimlari foydalanuvchi interfeysini boshqarish tizimlari (FIBT) ishlab chiqarilmoqda bunday tizimlarning tarkibiga quyidagi konponentalar (instrumental vositalar) mavjud[5]:

- ekran generasiyasi;

- grafika paketi;

- malumotlar va yordamchi informasiya muharrirlari;

- kupoynali ekranni boshqrish;

- kiritiladigan malumotlarni modifikatsiyalash;

- bajariladigan kod generator.

Shunday qilibbu bulimda biz foydalanuvchi interfeysiga quyiladigan asosiy talablar, muloqatni tashkil qilish usulllari va prinsiplari, surov-javob fragmentlarining taxlil bosqichlarini aniqladik. Murakkab muloqatlarni loyihalash va boshqarish tizimlarning umumiy konponentalari tuplamini kursatdik.

Multioynali WIMP- interfeysilar elementlarini loyihalash

Bu tundagi interfeyslar asosan oddiy va malakasi uncha yo'qoribulmagan foydalanuvchilarga muljallangan. Ana shu turdagi interfeysilardan WIMP so'zi quyidagi 4 ta asosiy xususiyatlardan kelib ciqqan:

- malumot bir nechta oynalar (windows) yordamida akslantiriladi;

- obektlarni piktogrammalar yordamida ikonalar (Icons) bilan tasvirlanadi;

- maluotni tanlab olish sichqoncha (Mouse) orqali bajariladi;

- avtomatik ravishda paydo bo'ladigan (Pop-down)menyo'lar ishlatiladi.

WIMP- interfeyslarning paydo bo'lishi va ko'p ishlatilishi ham o'ziga xos sabablarga ega. Birinchidan tasvir (obekt)ekranga sig'masa ham uning qismlarini bo'laklab xotiraga saqlash mumkin. Shuning uchun katta hajmdagi tugri murojat qilinadigan xotira ishlatiladi. Ikkinchidan piktogrammalarni akslantirish uchun monitorning yoqori kuchlanish (nuqtali adrislar)turlari qullaniladi.

Uchunchidan, tizim juda katta tezkorlik bilan ishlab obektlari tasvirlash va qulayliklarini beradi.

WIMP- interfeyslar asosida juda kup operasion tizimlar, masalan,Windows, MAC OS ishlaydi. Amaliy programrammalar majmualari ham shu elementlar yordamida tashkil qilinayabdi. Mutaxassis-foydalanuvchining "ish stoli" elementlari,yani birnechta hujjatlar bilan ishlash, soat,kalkulyator qulayliklaridan foydalanish, bir hujjatdan boshqasiga tez o'tish amallari bajarish kabi vositalari bu muhitda tashkil qilinadi har bir oyinaga o'ziga xos malumotlar turini akslantirish mumkin. Masalan, piktogramma –katta bo'lmagan maxsus oyin(belgi) bo'lib u bilan bog'liq bo'lgan buferdagi malumot turini akslantiradi. Piktogrammani ochib malumotni ekranda ko'rish mumkin bo'ladi. Tayyor piktogrammalardan tashqari, foydalanuvchi o'z piktogrammalarini tashkil qilish mumkin. Piktogramma bilan programma modullarini ham (bajariladigan fayllar) bog'lashga imkoniyat birilgan.

Bu va boshqa qulayliklarni hisobga olsak; ko'poyinali texnologiya bitta ekran bilan ishlashdan ancha farq qiladi, ham tekst va ham grafika elimentlarini qo'llaydi

Hozirgi paytda foydalanuvchi interfeysini loyihalash uchun bir qancha talab va standart lar ishlab chiqilgan. Standartlashning asosiy maqsadi umumiy terminologiyani (atamani),asosiy tushunchalarni, ularni ishlab chiqish vositalarni, texnologiyalarini taklif qilishdir. Standartlarni qullash esa programmalashda yaxshi natijalar beradi va programmalovchining yo'qori malakali ekanligi haqida dalolat beradi.

Ana shunday standartlardan biri IBM formulasi tomonidan taklif qilingan SAA(sustem Application Architecture-ilovalarni ishlab chiqish muhiti arxitekturasi, tuzulishi) standartidir. Bu standart programma taminotini ishlab chiqishning kupgina tomonini, yani foydalanuvchio interfeysi, programmalash til va vositalari, kodlashtirish usuli, grafika, oyinali qo'llash va protokollarni telekommunikasiyada ishlatilishini o'z ichiga oladi. Standartning asosiy qismini CUA (Common User Acces-matn va grafik ilovarni ishlab chiqishda

foydalanuvchi interfeysini tashkil qilish) tashkil qiladi. Standartda quyidagi tushuncha, vosita va mexanizmlar ishlatiladi.

Ekran -disiliy qismini va unda foydalanuvchi uchun malumot joylashtiriladi.

Panel –malumotlarning aniq bir usul yoddamida ekranda chiqarilisi.Standar 5 xil panel turlarini (menyo', kiritish, informasion, ro'yxat va ilovalarni aniqlash) qullaydi.Panelni hosil qilishda uni 3 ta qismga ajratish mumkin: harakat meyo'si, panel tanasi va funksional kilavisha sohasi.

Muloqatning olib borilishi quyidagi qadamlardan tashkil qilinadi.

a) Bir qadam oldongi (malumotni kiritish).

b) Bir qadam orqaga (harakatni to'xtatish)

d) Ilovaning aniq bir nuqtasiga qaytish(funksiyadan chiqish)

e) Operatsion sistema muhitiga qaytish (ilovadan chiqish).

Ekran formasi -interfeysning asosiy visual (ko'rinishli) elementi. Elementlardan maydon, knopka bilan maxsus funksiya (programma matni) bog'lanadi va uni **ssienariy** deymiz.

Bu va boshqa elementlarni tashkil qilish va boshqarish uchun maxsus interfeys programmalari ishlatiladi. Asosiy programmalarni ko'rib chiqamiz [6].

Menyu menejeri menyu daraxti bo'ylab harakatni boshqaradi va menyu satridagi programmani ishga yurgizadi.

Menyu taxrirlovchisining bajaradigan ishlari to'plami:

- menyuning yangi oynasini hosil qilish va daraxtga qo'shish;
- sarlavha va uning nomini aniqlash;
- daraxtdan menyu punktini chiqarish;
- menyu punktiga programmani bog'lash;
- har bir oyna uchun yordamchi qism sistema (Help) ni tashkil qilish.

Menyu monitori. Ma’lumotlarni qayta ishlovchi programmalarga uzatadi va natijalarni chiqaradi.

Mavjud programmalash muhitlari va tizimlarini tahlil qilsak yuqorida keltirilgan usul, vosita va mexanizmlar Delphi va Visual Basic integrallashgan muhitlarda boshqalarga nisbatan ko’proq ishlatiladi.

Sinov savollari:

1. Interfeys nima va uning qaysi turlarini bilasiz ?
2. Interfeysga qo’yiladigan talablar.
3. WIMP –interfeys nimani anglatadi ?
4. “Ish stoli” metaforasi qanday tashkil qilinadi
5. Interfeysni standartlashga misollar keltiring.

Adabiyotlar

[6], [8], [12], [13], [15], [18]

Ma’ruza № 8

Mavzu: Klassik tizimli programmalash

Reja:

1. Klassik tizimli dasturlash xususiyatlari.
2. Uzulishlar va ularning turlari.
3. OS ning uzulishlarini qayta ishlash moduli.
4. Yuqori darajali tillar va uzilishlar

Tayanch iboralar: uzilishlar OS moduli, uzulish vektori, BIOS funksiyalari, quyi darajali til, programma uzulishlari, apparat uzulishlari.

Ma'ruza bayoni

Uzulishlar va assebler-programma xususiyatlari.

Tizimli programmalashda uzulishlarni tahlil qilish va qayta ishlash katta rol o`ynaydi. Uzulish-bu bajarilayotgan jarayonni unga nisbatan tashqi hodisa tasiri asosida vaqtinchalik to`xtatilishidir. Biz uzulishlar va ular bilan bog`liq bo`lgan tushunchalarni SHEHM muhitida qarab chiqamiz. Boshqa (universal superEHM) muhitlarda ham xuddi shunga o`xshgan uzulishlar turi, ularni tahlil qilish usullari va qayta ishlash mexanizmlari mavjud.

MS-DOS operatsion sistemaning asosiy modullaridan biri-bu BIOS (KCHAS-kiritishlar chiqarishning asosiy sistemasi) hisoblanadi. BIOSning asosiy vazifalaridan biri operatsion sistemaning uzulishlariga xizmat ko`rsatishdir. Bu ish uzulishlar mexanizmi yordamida bajariladi, ya`ni mashenaning joriy ishi qisqa vaqt davomida to`xtatiladi va unga sababchi bo`lgan hodisa qayta ishlanadi. Uzulishlarni uchta guruhga: aparat, mantiqiy va programmaviy bo`ladilar. Aparat uzulishlarining sabablari (manbalari): tok kuchini pasayishi, klaviaturadagi klavisha (tugmachani) bosilishi, qattiq va yumshoq disk qurilmalaridan keladigan signallar va hokazolar bo`lishi mumkin.

Mantiqiy (protsissor) uzulishlar nostandart holatlarda, masalan, mekroprotsissorning ish jarayonidagi nolga bo`lish registrlarni to`lib ketishi, “to`xtash nuqtasi” paydo bo`lganda yuzaga keladi.

Programmaviy uzulishlar uzulishlarning asosiy qismini tashkil qiladi. Bunday uzulishlar bir programma boshqa programmadan xizmat ko`rsatishni talab qilgan paytda hosil bo`ladi. Xizmat ko`rsatish turi esa yana aparat bilan bog`liq bo`ladi.

Har bir uzulish maxsus (yagona) nomerga ega va u bilan aniq bir qismprogramma bog`langandir. Bu qismprogramma paydo bo`lgan holatni tahlil qiladi va unga xizmat ko`rsatadi. Agar bu qismprogrammani boshqa uzulish qismprogrammasi to`xtatmoqchi bo`lsa ikkinchi qismprogramma “navbatga” qo`yiladi, ya`ni “maskirovka” qilinadi.

BIOS past bosqich (прерывания нижнеего уровня) uzulishlarni qayta ishlaydi. Bu turdagi uzulishlar aparatura bilan bog`liq. Past (quyi) bosqich uzulishlarining nomerlari $0\text{-}31_{10}(0\text{-}1F_{16})$. Boshqa uzulishlar, ya`ni $32\text{-}63_{10}(20\text{-}3F_{16})$ tartib raqamli uzulishlar yuqori bosqich uzulishlari deyiladi. Bu turdagi uzulishlarni DOS uzulishlarini qayta ishlash moduli (MS-DOS.COM yoki MS-DOS.SYS) boshqaradi.

BIOSning qayta ishlaydigan uzulishlaridan, masalan 0-nolga burish $22_{10}(16_{16})$-klaviaturani boshqarish, 26_{10} $(1A_{16})$-sana va vaqtni so`rash (o`rnatish)larni keltirish mumkin.

Bazi bir uzulishlar, masalan, $19_{10}(13_{16})$-qattiq disk qurilmasini boshqarish bir nechta (18 ta) funksiyalarga murojat qilish imkoniyatini beradi. Bu funksiyalar ham o`z nomeriga (kodiga, 0-17) ega.

Masalan, 0-boshlang`ich holat (o`rnatish), 4-yozish yoki o`qishdan keyin tekshirish, C-kerakli yulakni (дирожкани) izlash yoki $16_{10}(10_{16})$-displeyni boshqarish uzulish esa 24 ta funksiyalarga murojat qiladi.

MS-DOSning uzulishlarni qayta ishlash moduli.

Bu modul yuqori bosqich uzulishlari bilan ishlaydi.Anashu uzulishlarga amaliy programmalardan murojaat qilish mumkin.

Maskur guruhdagi qism programmalar fayl sistemasi ishini, komunnikatsiyalar (klaviatura, displey,printer) va maxsus holatlarni boshqarish bilan bog'liqdir.Masalan, $33_{10}=21_{16}$ nomerli uzilish bilan funksiyalarning katta to'plami birlashtirilgan.Bu funksiyalar fayl qism sistemasi va standart qurilmalar ishini boshqarish uchun xizmat qiladi.Misol uchun, $2A_{16}$-funksiyasi sana haqidagi, $2C_{16}$-vaqt,41_{16}-kursatilgan katalogdan faylni o'chirish kabi malumotlar beradi va amallarni bajaradi.

Amaliy va tizimli programmalar taminotini yaratishda biz operatsion sistema va apparatura bilan bog'liq bo'lgan uzilishlarga quydagi yullar bilan murojat qilishimiz mumkin. 2.Yo'qori darajali tillar (paskal, ci, delfi) tarkibidagi maxsus protsedura va funksiyalardan foydalanish.Ammo bular apparaturaga to'g'ridan-to'g'ri murojat qilishni to'liq taminlamaydi.

2. Opertsion sistemaning yuqori bosqich ($32\text{-}63_{10}$) uzulishlarini qo'llash. Programmalash tizimining translyatori maxsus protsedura (funksiya) yordamida OSning uzulishlari murojaat qilish imkoniyatini beradi.

Bu proseduralar parametrlari uzulish nomerini ko`rsatadi va mikroprosessor asosiy registrlarini qiymatini aniqlaydi.

Masalan, TPasda MSDOS modulining Intr prosedurasini qaraymiz. Uning umumiy ko`rinishi

procedure Intr(InrNo:Byte; var Regs: Registers);

Bu yerda IntNo-uzulish nimeri; Regs-Registers tipli o`zgaruvchi va uning umumiy ko`rinishi variantli yozuv shaklida tasvirlanadi.

Type

```
Registers=recora
        Case integer of
        0: (Ax,Bx,Cx,Dx,BP,SI,DI,DS,ES,Flags:word);
        1: (AL,AH,BL,BH,CL,CH,DL,DH:Byte)
        End;
```

Bu prosedurani ishlashini oddiy misol yordamida ko`ramiz. Faraz qilaylik, Manitor ekranidagi ma`lumotni printerga chop etish kerak bo`lsin. Bu ishni Shift+PrtScr klavishalarni birgalikda bosish yordamida bajarish mumkin yoki uzulishlardan foydalanib ham ijro etish yo`llari mavjid. Programmaning umumiy ko`rinishi quyidagicha bo`ladi.

```
Uses dos;
  Procedure PrintScreen;
    Var r:registers;
      Begin intr($05,r) end.
{Tizim uzulishi nomeri 16 lik sanoq sistemasi}
Begin
  Write(`ekranni chop etish uchun ENTERni bosing');
   Readln;
   PrintScreen
End.
```

Programma 5-uzulishni (ekranni grafik nusxasini chop etish) ishlatadi va ekrandagi ma`limotni printerdan chiqaradi.

Uzulishlardan drayverlarni ishlab chiqishda, assembler-programma tuzushda yoki operatsion tizimning funksiyalarini o`zgartirishda ko`p foydalanadilar.

3. Ikkinchi usulni qo`llab quyibosqich, ya`ni BIOS murojaat qiladigan uzulishlarni ham ishlatish mumkin. Ammo apparaturaga murojaat qilish SHEHM turi bilan bog`liq, shuning uchun programmalar umumiy emas xususiy, SHEHMning konkret turi bilan bog`liq bo`lib qolishi mumkin. Shunday qilib, birinchi usul konkret programmalash tili bilan, ikkinchi usul esa SHEHM arxitekturasi bilan bog`liq bo`lgan holda uzulishlarni ishlatishga imkon beradi.

Endi uzulishlarni assembler-programma tuzish paytida qo`llanishi va xususiyatlarini ko`rib chiqamiz.

Accembler-programma tuzish jarayonida operatsion sitema va uning qismsistemasi BIOS yordamchi funksiyalari ishlatish maxsus qoidalar bo`yicha tashkil qilinadi.

Shu uzulishlardan biri 21_{16} nomerli uzulish bo`lib birnechta funksiyalarini qo`llashimiz mumkin. Funksiya nomeri AH registrida joylashtiriladi.

Masalan, mov AH,1 komandalari programma uchun kiritilishi

Int 21h kerak bo`lgan simvolni klaviaturadan kiritishni kutib turadi. Bu yerda 1 funksiya nomeri va u $21_{16}$ nomerli uzulish tarkibiga kiradi. Shundan foydalanib programmani dialog (so`rov-javob) shaklida tashkil qilish mumkin. Ma`limotni kiritish uchun “yes” nishoniga yoki kiritmaslik uchun “no” nishoniga o`tish kerak bo`lsa u holda quyidagi fragmentni yozish mumkin.

```
Wodsim: mov AH,1    ; simvolni o`qish (kiritish)
        Int 21h     ; simvol AL da kiritiladi.
        Cmp AL,’4’ ; simvol=’yes’
        Je  YES     ; agar  ‘yes’ u holda yes nishoniga o`ting.
```

Cmp AL,‘N’ ; agar ‘NO’ bo`lsa No nishoniga o`ting.

Je NO

Jne wodsim ; ‘y’ yoki ‘N’ bo`lmasa simvolni kiritishni kutish.

YES :

NO :

Sinov savollari:

1. Tizimli dasturlash vazifalarini aniqlang.
2. Uzulish nima?
3. Uzulishlar qanday guruhlarga bo'linadi ?
4. OS, apparatura va dasturlash tili uzulishlarni ishlatishni tushuntiring
5. OS ning qaysi moduli uzulishlarni qayta ishlaydi
6. Dasturlashning qanday sohalarida uzulishlar ishlatiladi?
7. Aniq bir guruh (OS, apparatura, foydalanuvchi) uzulishlari haqida ma'lumot bering

Adabiyotlar

[1], [2], [3], [4], [12], [19]

Ma’ruza № 9

Mavzu: Modullar kutubxonasini yaratish

Reja:

1. Modulli programmalash asoslari
2. Modul interfeysi va turlari
3. Usulning qulayliklari
4. Zamonaviy tillar va modulli dasturlash

Tayanch iboralar: modul, interfeys, ob’ekt, bo’lim, abstrakt tip, modullarni boshqarish, modul spetsifikatsiyasi.

Ma’ruza bayoni

Modulli programmalash zamonaviy texnologiyalardan hisoblanadi. Katta va murakkab programma ta’minotini yaratishda avvalo uning strukturasi va komponentalari orasidagi munosabatni aniqlash maqsadga muvofiq.

Struktura (tarkib) masalasini muhokama qilsak u holda quyidagi holatlarni qarab chiqishimiz kerak:

a) Programma yagona moduldan iborat va uni ijro etish uchun operativ xotiraga to’liq yuklash kerak;
b) Programma bir nechta segmentlardan (bo’laklardan) iborat va har bir segment kerakli paytda operativ xotiraga yuklanadi va ijro etiladi. i- modul o’rniga (i+1)- modul yuklanishi mumkin.
d) Programma rezident (bosh) qism va bir nechta norezident qismlardan iborat. Rezident qism ish boshlanishida xotiraga yuklanadi. Norezident qismlar esa navbat bilan ketma-ket yuklanadilar.

Shu va boshqa xususiyatlar programmaning tezkorligi va xotirani egallashiga katta ta’sir qiladi, programma modullari, kutubxonalarini yaratish jarayonida

hisobga olinadi. Bu yerda bir nechta holatlar va usullar mavjud. Ularni qisqacha ko’rib chiqamiz.

Programmani matn moduli bo’yicha strukturalash (tarkiblash). Murakkab programma konstanta,o’zgaruvchilar, turlar, qismprogrammalardan iborat. Shuning uchun boshlang’ich modulni qismprogrammalar, ya’ni **protsedura** va **funksiyalarga** nisbatan tarkiblash (tartiblash) maqsadga muvofiq.

Faraz qilaylik Pascal oilasiga qarashli tillar berilgan. Bu tillarda programmaning umumiy strukturasi quyidagicha berish mumkin.

```
Program <nom>;
  <Global konstanta, tip va o’zgaruvchilarni aniqlash va tasvirlash >
  Procedure P1(P1 parametrlari );
  <P1 lokal ob’ektlari tasvirlanishi>;
     Begin
        <P1 tanasi>
     End;
  Procedure P2(P2 parametrlari );
  <P2 lokal ob’ektlari tasvirlanishi>;
     Begin
        <P2 tanasi>
     End;
Begin
  <asosiy programma tanasi boshlanishi>;
    P1(faktik parametrlar);
```

```
    P2(faktik parametrlar);
    ... ... ... ... ... ... ... ... ...
  <asosiy programma tanasi oxiri>
End.
```

Strukturada uchta asosiy komponentalarni ajratish mumkin

a) Programma sarlavhasi
b) Protseduralar (funksiyalar) tasvirlanishi
d) Programma tanasi. Operatorlar va qismprogrammalarga murojaatlar.

Aytish kerakkim, formal va faktik parametrlar orasidagi aloqalarni tashkil qilish programmalash tizimining translyatori zimmasiga yuklanadi.

Programma modullarini matn shaklida o'rnatish. Bu usulni **makrogeneratsiya** usuli deydilar. Boshlang'ich programma matni bir nechta qismlar (fayllar)ga bo'linadi. Asosiy programma matniga maxsus ifoda (kalit) lar kiritiladi. Bu kalitlar programma matniga boshqa fayllarda saqlanadigan modullarni kiritishni ta'minlaydi.

Faraz qilaylik, P.PAS programmasi juda katta hajmni tashkil qiladi. Uni P1.PAS va P2.PAS qism (modul) larga bo'lib alohida fayllarga saqlaymiz. Shu bo'laklarni birlashtirib yagona programma shaklida tarjima qilish va bajarish uchun quyidagi sxema yordamida ishlash maqsadga muvofiq.

```
 Program P;
    <tasvirlashlar>
    {$INCLUDE:'P1.PAS'}
    <P- ning boshqa bo'limlari>
    {$INCLUDE:'P2.PAS'}     end.
```

Bu holda translyator makroo’rnatish (makropodstonovka) usulini qo’llab bo’laklardan yagona programma hosil qilib, tarjimadan keyin bajaradi. Ammo programma matni murakkab va katta hajmga ega bo’lgan holda bu usul unchalik qulay emas. Shuning uchun keyingi metodni ko’p hollarda qo’llaydilar.

Alohida tarjima qilinadigan modullar

Programma alohida modullarga bo’linadi. Har bir modul mustaqil tarjima qilinadi. Modullar bajarish jarayonida birlashtiriladi va ijro etiladi.

UNIT tipidagi modul. Bu texnalogiyaning quyidagi qulayliklari mavjud.

a) Programma bo’lak (modul) larga bo’linadi va tarjima qilinadi.
b) Qismprogrammalar kutubxonasini yaratish mumkin.
d) Standart modullardan foydalaniladi.
e) Programma hajmini kengaytirish mumkin.

Paskal oilasidagi tizimlar muhitida modulning umumiy strukturasi quyidagicha.

Unit moduli – bu maxsus shaklda tashkil qilingan kutubxona bo’lib unda konstanta, turlar, o’zgaruvchilar, protsedura va funksiyalar saqlanadi. Bunday modulning programmadan asosiy farqi shundan iboratkim, u mustaqil ishga tayyorlanmaydi. Faqat boshqa mustaqil programmalar tarkibida ishlashga da’vat etiladi.

Unit modulining umumiy strukturasi quyidagichadir [4].

Unit <modul nomi>;

interface {e’lonlar bo’limi boshlanishi}

USES {boshqa mod1, mod2, … , modn- larni bog’lash}

Const <konstantalar>

Type <turlar>

Var <o’zgaruvchilar>

{shu modulda ishlatiladigan protseduralar (funksiyalar) sarlavhalari}

implementation {ishlab chiqarish bo’limi}

Uses {modullar ro’yxati}

Const <konstantalar>

Type <turlar>

Var <o’zgaruvchilar>

Label <nishonlar>

begin

<modulni ishga yurgizish bloki>

end.

Ko’rinib turibdikim, modul to’rtta qismga bo’linadi.

a) Modul sarlavhasi (unit nom);
b) Interfeys, ya’ni e’lonlar bo’limi (interface);
d) Ishlab chiqarish bo’limi (implementation);
e) Ishga yurgizish bo’limi (begin va end orasida);

Modulni komplyatsiya qilish natijasida, masalan, TPas muhitida .tpu kengaymali (ya’ni Turbo Pascal Unit) fayl hosil bo’ladi. Modulni programma yoki boshqa modulda ishlatish uchun uning nomini uses direktivasiga ko’rsatish kifoya.

Masalan, uses mod1, mod2;

Bundan keyin mod1 va mod2 modullardagi hamma ob’ektlar (konstanta, tur, o’zgaruvchi, nishon, protsedura va funksiyalar) qo’llashga tayyor bo’ladi.

Programmalarni modulli tashkil qilish ikkita asosiy qulaylikka olib keladi.

Birinchisi. Xususiy (shaxsiy) programmalar kutubxonasini yaratish va ularni boshqa programmalar tarkibida ishlatish.

Ikkinchisi. Modullardan tashkil topadigan programmalar hajmi juda katta bo'lishi mumkin.

Modulli texnalogiya asosida zamonaviy programmalash tizimlarining **standart modullari** kutubxonasi yaratilgan. Masalan, TPas, Borland Pascal tizimlar muhitida standart modullar Turbo.tpl (Turbo Pascal Library) kutubxonasida saqlanadi. Standart modullardan: System, Crt, Strings, Graph, Overlay juda ko`p ishlatiladi.

Shunga o'xshash programmalash tizimlarining ya'na bir qulayligi ob'ekt modullarini va yuklanadigan fayllarni hosil qilish hisoblanadi [5].

Ob'ekt modullar va yuklanadigan fayllar.

Boshlang'ich programma matnini translyatsiya qilib **ob'ekt modullarni** hosil qilamiz. Ob'ekt modul .obj kengaymali faylda saqlanadi, ya'ni

p.pas → translyator → p.obj

Keyingi qadamda bitta yoki bir nechta ob'ekt modullardan bajariladigan (yuklanadigan) .exe yoki .com fayllarni hosil qilamiz. Bu ish maxsus Link programmasi yordamida bajariladi, ya'ni

P.obj → Link → P.exe yoki

P.obj → Link → P.com

Link yordamida bir nechta ob'ekt modullarni birlashtirib yagona programmani hosil qilish ham mumkin. Masalan,

Link P1+P2+P2, P.exe.

Programmalash tizimlarining kutubxonalari elementlari, ya'ni programmalar (modullar) ustida amallar bajarish uchun maxsus **utilitalar** (yordamchi

programmalar) ham ishlatiladi. Masalan, BP muhitida tpumover.exe utilitasi qo’llaniladi. U bilan ishlash sxemasini quyidagicha berish mumkin.

Tpumover <fayl nomi> <operatsiya>

bu yerda <fayl nomi>.tpu kengaymali faylni yoki .tpl kengaymali faylni bildiradi. <operatsiya> esa aniq amalni anglatadi, ya’ni

+unit_nomi – blokni kutubxonaga qo’shish.

-unit_nomi – blokni kutubxonadan o’chirish.

*unit_nomi – blokni kutubxonadan o’chirmasdan ajratib olish.

Shunday qilib, biz bu bo’limda programma ta’limotini yaratishda qo’llaniladigan modullar va kutubxonalarni bir nechta klassik usullarini ko’rib chiqdik. Bu metodlar quyidagi qulayliklarni beradi:

- Programmalar tahlil qilish va o’zgartirishga qulay bo’ladi;
- Har bir dasturlovchi o’zining moduli (modullari) ustida ishlaydi va masala bir nechta mustaqil ishlar to’plamiga taqsimlanadi;
- Modullarga murojaat qilish, ularni yig’ish va turli to’plamlarini yaratish masalasi osonlashadi;
- Modullarni alohida komplyatsiya qilinishi ularni turli tillar va tizimlar muhitida yaratishga imkon beradi.

Sinov savollari:

1. Katta va murakkab programmalar qanday tashkil qilinadi?
2. Modulni matn shaklida o’rnatish qulayliklarini keltiring.
3. Makrogeneratsiya usulini qanday holatlarda qo’llash kerak?
4. Alohida tarjima qilinadigan modullar strukturasini tasvirlang.
5. Standart modullar nima?

Adabiyotlar

[5], [8], [11], [12], [14], [16], [17]

Ma’ruza № 10

Mavzu: Subjarayonlarni tashkil qilish

Reja:

1. Katta hajmdagi programmalarni loyihalash muammolari
2. Subjarayon va uni boshqarish
3. Xotirani boshqarish va uni taqsimlash
4. Subjarayonni boshqaruvchi dastur

Tayanch iboralar: subjarayon, kucha, xotirani taqsimlash, uzulish, yakunlash kodi, xato turi.

Ma’ruza bayoni

Amaliy informatikada katta programmaviy tizimlarni ishlab chiqish uchun tayyor programmalardan foydalanish va ularni dinamik tarzda bir- biri bilan bog’lash muhim masalalardan biridir. Bizga ma’lum bo’lgan TPas va Borland Pascal integrallashgan programmalash muhitlarida bu ishni bajarish uchun maxsus texnologiyalar ishlatiladi. Bu texnologiyalarni quyidagi xususiyatlarga ajratish mumkin:

- muhitning tizimli modullarining tayyor protsedura va funksiyalari ishlatiladi;
- chaqirilishi kerak bo’lgan tashqi programmalar ixtiyoriy programmalash muhitida tuzilishi mumkin;
- programmalar .exe va .com kengaytmali fayl sifatida bo’lishi kerak;
- programma sifatida operatsion sistemaning ixtiyoriy komandasini ham qo’llash mumkin.

Bu xususiyatlar va yuqoridagi mavzularda keltirilgan usul, texnologik yechimlar va mavjud bo’lgan instrumental vositalarining qulayliklaridan foydalanib biz quyida programmalashda subjarayonlarni tashkil qilishni ko’ramiz.

Programmalashda subjarayonlarni tashkil qilish uchun quyidagicha ish ko’radilar.

1. Qo’yilgan masalani yechish uchun programmalar ketma-ketligi hosil qilinadi.
2. Bitta programma boshqasini chaqirishini rejalashtiriladi.
3. Chaqiruvchi boshqa programmani ishga yurgizgandan keyin xotirada qolishi e’tiborga olinadi.
4. Chaqirilgan programma o’z ishini tugatgandan keyin uni ishga da’vat etgan (nisbatan yuqori darajali) programma yana o’z ishini davom ettiradi.

Ana shu texnologiya asosida integrator programmalar tashkil qilingan. Bunday programmalarda ishga da’vat etilishi kerak bo’lgan programmalar menyusi yordamida kerakli subjarayonlar (programmalar) ishga tushiriladi.

Oddiy programma o’z ishini tamomlab xotirani bo’shatsa, subjarayonlar esa ketma-ket bir-biriga boshqaruvni uzatadi va oxirgi subjarayon esa operatsion sistemaga boshqaruvni beradi.

Subjarayonlarni tashkil qilish uchun xotira hajmini ko’rsatish kerak. Subjarayonlar ozod xotirada, ya’ni operatsion sistema, rezident programmalar va ishga tushirilgan programmadan qolgan ortiqcha xotira sohasida tashkil qilinadi.

Masalan, bizga tanish bo’lgan Borland Pascal yoki Turbo Pascal muhitlarida xotirani boshqarish uchun $M turdagi kompilasiya kaliti ishlatiladi. Uni asosiy programmada birinchi satr o’rnida qo’yish kerak.

{$M Stek, Minimumkucha, Maksimumkucha}

Bu yerda 1024 bayt ≤ Stek ≤ 65520 bayt

0 ≤ Minimumkucha ≤ 640 k

0 ≤ Maksimumkucha ≤ 640 k

Ammo Minimumkucha ≤ Maksimumkucha

Dinamik o’zgaruvchilarni joylashtirish uchun ishlatiladigan xotiraning qismini **kucha** deymiz. Kuchani boshqarish uchun maxsus programma (kucha monitori) ishlatiladi.

$M direktivasiga misol tariqasida {$M 1024, 0, 2048}- ni keltirish mumkin.

Subjarayonlarni tashkil qilish uchun quyidagi DOS-modulining protseduralari ishlatiladi.

1. SwapVectors; Bu protseduraning parametrlari yo’q, tizimning yoki tashqi uzilishlar vektorlarini tiklash (qayta tiklash) uchun ishlatiladi.
2. Exec (ExeFile, ComLine: String); Bu yerda ExeFile- subjarayon, ya’ni bajariladigan (.exe, .com) fayl va ComLine- parametrlar
   Masalan, Exec(’abc.exe’, parameters);
3. DosExitCode: word; Bu funksiya subjarayonni yakunlash kodini qaytaradi. Ya’ni subjarayon qanday bajarilganligi haqida ma’lumot beradi.

Programma umumiy, ya’ni ixtiyoriy tashqi mustaqil bajariluvchi dasturni subjarayon sifatida ishlatiladigan bo’lishi uchun uning asosiy bo’laklarini algoritm ko’rinishda tasvirlaymiz.

```
{$M 1512, 0, 0} {Xotirani boshqarish}
Uses dos, crt; {Muhit modullarini ishlatish}
Function Execute(ExeFile, Parameters, Inf):boolean;
    Begin
        SwapVectors; {OC vektorlarni o’rnatish}
        Exec(ExeFile, Parameters); {subjarayonni ishga yurgizish}
        SwapVectors; {Muhit vektorlarini o’rnatish}
        Case DosError of
```

```
            0: begin … end;
            2: begin … end;
            8: begin … end;
          end;
        end;
      {DosError o’zgaruvchisi operatsion sistemaning xatolar nomerini saqlaydi}
    Begin {Programmaning ishchi qismi}
      Case Ch of
        i: begin … end;
        i+1: begin … end;
        ⋮
        i+n: begin … end;
      end;
        {Tanlash operatori yordamida turli rejalashtirilgan ishlar bajariladi}
    end.
```

Execute- funksiyasining tanasida subjarayonni tahlil qiladigan, ya’ni uning qanday yakunlaganligi haqida ma’lumot beradigan maxsus DosExitCode: word; funksiyasini kiritish maqsadga muvofiq. Funksiyaning natijasi bir bayt (word) ma’lumot beradi. Wordning yuqori bayti quyidagi qiymatlarni qabul qilishi mumkin.

$H_i = 0$, subjarayon yaxshi va to’g’ri tugallandi.

$H_i = 1$, subjarayon Ctrl+Break tugmachalarini bosish natijasida to’xtatildi.

$H_i = 2$, ExeFile topilmadi.

$H_i = 8$, subjarayon uchun xotira yetishmadi.

Wordning quyi bayti esa subjarayon sifatida ishlatilgan (ExeFile) programma yakunlanish kodi haqida ma’lumot saqlaydi.

Tahlil qilingan umumiy sxema asosida katta va boshqaruvchi programmalarni turli dasturlovchilarga taqsimlab, ishlab chiqib yagona programma shakliga keltirish osonlashadi.

Bu texnologiyaning yana bir qulayligi turli programmalash muhitlarda (Pascal, Cu, Delphi) tashkil qilingan bajariladigan (.exe, .com) fayllarni birlashtirib yagona programma tashkil qilishdir.

Subjarayonlarni boshqaruvchi dasturning umumiy va to’liq ko’rinishi quyida berligan

Programmani qo’llash uchun Exec protsedurasidagi ExeFile va Parameters formal parametrlarni haqiqiy parametrlar bilan almashtirish kifoya. Agar ExeFilening parametrlari mavjud bo’lmasa, u holda ikkinchi parametr bo’sh qoldiriladi.

Masalan, Exec(’alpha.exe’, ’ ’);

Subjarayonli programma umumiy sxemasi

```
Program Psubprocess;
{$M xotirani boshqarish parametrlari}
 Begin
    Read(k); {k- masala nomeri}
    Case k of
        1: begin
```

```
            SwapVectors;
            Exec('prog1.exe', parameters1);
            SwapVectors;
        end;
    2: begin
            SwapVectors;
            Exec('prog1.exe', parameters2);
            SwapVectors;
        end;
        ⋮
    3: begin
            SwapVectors;
            Exec('prog1.exe', parameters2);
            SwapVectors;
        end;
  end;
end.
```

Sinov savollari:

1) Subjarayon nima uchun tashkil qilinadi?
2) Kompilyatsiya kalitlari vazifasini aniqlang
3) Subjarayonni tashkil qilishda qaysi modul ishlatiladi?
4) Exec protsedurasining rolini aniqlang.

5) Uzulish vektorini qayta tiklash nima?

6) OS Vektorlarini o’rnatishdan maqsad?
7) Subjarayonlarni sxema yordamida tasvirlang.

Adabiyotlar

[1], [5], [11], [14], [16]

.

Ma’ruza № 11

Mavzu: Overley programmalar

Reja:

1. Zamonaviy muhitlar va overley programmalar
2. Overley tashkil qilish bosqichlari
3. Overleyni rasmiylashtirish
4. Unit modulining roli
5. Overley fragmentlar va bosh programma

Tayanch iboralar: Overley, modul, fragmentlar, kompilyatsiya rejimi,

Ma’ruza bayoni

Endi tizimli programmalashning yana bir kichik va qiziqarli masalasi overley programmalarni tashkil qilish haqida to’xtalib o’tamiz.

Programmalash jarayonida programma hajmi(matni) juda katta bo’lgan holda ko’pgina programmalashtirish tizimlari dastur transliyatsidan keyin “ Not enough memory – Xotira yetishmayapti ” degan ogohlantirish ma’lumotini beradi. Bu muammoning hal qilishning mumkin bo’lgan yo’llaridan biri programmani overley tashkil qilish va qo’llashdir. Shuning uchun zamonaviy programmalash muhitlarida maxsus overley modullari mavjud va ulardan foydalanib overley programmalar tuziladi. Tpas va Borland Pascal qulayliklardan foydalanib overley programma yaratish asos va bosqichlarini ko’rib chiqamiz.

1) Overley moduli yordamida, ya’ni uning talablarini hisobga olgan holda programmani alohida bo’laklarga bo’lamiz.

2) Bo’laklar hohlagancha bo’lib, ularning birgalikda talab qiladigan xotira hajmi SHEHM ning ish uchun taqsimlanadigan xotira hajmidan ancha katta bo’lishi mumkin.

3) Overley tashkil qilgan programma asosan ikkita fayldan iborat, masalan,

a) prog.exc

b) prog.ovr

Bir xil nom va har xil kengaytma. Prog.exe – fayli programmaning doimiy qismi, prog.ovr – fayli esa xotiraga kerakli paytda chaqirilishi mumkin bo'lgan bo'laklar kodini saqlaydi.

4) Har bir vaqtning davomida xotiraga faqat kerak bo'lgan bo'lakcha yuklanib ishlatiladi. Programmaning overley qismlari(masalan, i+1 qism) uchun oldingi (i - qism) ishlatilgan qism egallagan xotirani ishlatadi. Qo'shimcha xotira umuman kerak bo'lmasligi mumkin.

5) Bo'laklarni xotiraga chaqirish va undan chiqarish uchun maxsus protsedura, ya'ni overleylar administratori (boshqaruvchisi) yordamida bajariladi.

Endi Overley programmalarni tasvirlash, ya'ni rasmiylashtirish algoritmi qadamlari bilan tanishamiz.

a) Overley programmalar ko'pgina tizimlarda(nasalan, Tpas, BP, C) faqat modul shaklida , ya'ni **unit** moduli shaklida tashkil qilinadi.

b) Modullarda mavjud bo'lgan bo'laklar, ya'ni bo'limlar (interface – e'lon, implementation – tadbiq etish, begin … end - initsializatsiya – ishga, yuritish) overley programmaga kiritiladi. Ammo bu yerda quyidagi shartlar bajarilishi kerak.

b.1) Har bir overley moduli bosh qismida kompilyatsiya rejimi {$0+}, ya'ni modulni overley bo'lishiga ruxsat beruvchi durektiva mavjud bo'lishi.

b.2) Overley qismprogrammalarni chaqiruvchi hamma protsedura va funksiyalar {$F+} (far chaqirish modeli, usuli) rejimida kompilyatsiya qilinishi.

Overley programmalarni tashkil qilish.

Overley programmalarni tashkil qilishning umumiy sxemasi va bosqichlarini oddiy misolar yordamida ko'rib chiqamiz. Asosiy instrumental vosita sifatida Borland Pascal programmalash muhitini ishlatamiz.

Programmaning overley bo’lakalari modul (unit) shaklida tashkil qilinadi. Modulning umuiy ko’rinishi quyidagicha

```
Unit  < modul nnomi >;
  Interface {elonlar bo’limi}
  Uses mod1, mod2, …, modn;
  Const
 Type {konstanta, tip, va o’zgaruvchilarni aniqlanishi}
 Var
     <protsedura va funksiyalarning sarlavhasi>
Implementation {amaliylashtirish bo’limi}
  Uses mod1, mod2, …, modn;
  Const
  Type
  Var
Begin
    {Ijro etish bo’limi}
End.
```

Faraz qilaylik, quyidagi sodda masala berilgan bo’lsin. Haqiqiy elementlardan iborat massiv berilgan. Massivning eng katta(maksimal) va eng kichik(minimal) elementlarini aniqlab ularning joyini almashtirish kerak bo’lsin.

Masalani bir qancha bo’laklarga bo’lish mumkin. Masalan, ma’lumotlarni tayyorlash va kiritish, maksimal va minimal elementlarni topish, ularni joyini almashtirish, hosil bo’lgan natijani chiqarish. Har bir bo’lakni overley programma

shaklida tashkil qilib, ya’ni overley fragment ko’rinishda loyihalab asosiy programmada ketma – ket ishga da’vat etib masalani hal qilish mumkin.

Overley fragmentlar va bosh programmaning umumiy sxemalari quyidagilardan iborat.

1) Massivning maksimal va minimal elementlari o’rnini aniqlaydigan modul.

```
{$F+, 0+}
Unit mod1;
  Interface
    Uses mod2;
    Procedure mest0;
      Begin … end;
  End.
```

2) Ma’lumotlarni kiritish moduli

```
{$F+, 0+}
Unit mod2;
  Interface
    Const n=8;
    Type mass=array[1..n] of real;
    Var arr:mass;
  Implementation
    Begin … end;
```

3) Orinalmashtirish moduli

```
{$F+, 0+}
Unit mod3;
  Interface
    Procedure zamenamesto;
  Implementation
     Begin  … end;
  End.
```

4) Overley bo’laklarni ishini boshlash moduli

```
Unit modprima;
   Interface
      Uses overley;
   Implementation
      Begin
         OvrInit(‘overprog.ovr’);
       End.
```

5) Asosiy programma umumiy ko’rinishi:

```
Program asosiy;
  {$F+}
  Uses Crt, modprima, overley, mod1, mod2, mod3;
  {$0 mod1}
  {$0 mod2}
```

{$0 mod3}

Begin … end.

Sinov savollari:

1. Overley usulining vazifasi nimadan iborat?
2. Programmalarni Overley tashkil qilish umumiy sxemasini bering
3. Overley modullar qanday taskil qilinadi?
4. Qaysi kompilyatsiya kalitlari ishlatiladi?
5. Overleyni rasmiylashtirish algaritmini keltiring
6. Overley programma nechta fayldan iborat?

Adabiyotlar

[1], [5], [13], [14], [19]

Ma’ruza № 12

Mavzu: Ob’yektga – mo’ljallangan programmalash xususiyatlari

Reja:

1. Kirish. Ob’ekt va uning xossalari.
2. OMPning asosiy prinsiplari.
3. Texnologiyaning qulayliklari.
4. Texnologiyaning qo’llash bosqichlari.

Tayanch iboralar: Ob’ekt, maydon, metod, qobiqlash, vorislik, polimorfizm, virtual metod, konstruktor, destruktor.

Ma’ruza bayoni

Ob’yektga- yo’naltirilgan programmalash (OYP) zamonaviy texnologiyalardan bo’lib uning negizida ob’ekt (object) tushunchasi yotadi. Shuning uchun bu texnologiyani ob’ektga- mo’ljallangan yoki ob’ektli texnologiya ham deydilar. Ob’ekt bizga programmalash kursining kirish qismidan ma’lum bo’lgan yozuv, aralash tip (record) tipiga o’xshashdir. Ob’ektga yozuvdan farqliroq nafaqat ma’lumotlar, balki ularni qayta ishlash uchun kerakli bo’lgan algoritmlar (qismprogrammalar) saqlanadi.

Ta’rif: Ob’ekt- bu shunday strukturadirkim uning komponentalarini turli tipli o’zaro bog’liq ma’lumotlar va bu ma’lumotlarni ishlatadigan qismprogrammalar (protsedura va funksiyalar) tashkil qiladi.

Ob’ektni ma’lumotlarning yangi tipi ham deyish mumkin. Shu asosda Cumula-67 tilidan boshlab yangi OYP texnologiyasi paydo bo’ldi va keyinchalik paskal va C++ tillarida rivojlantirildi. Texnologiya strukturaga va modulli programmalash texnologiyalariga suyanadi va ularni kengaytiradi.

Ob’ekt tushunchasiga misol qilib “Kompyuter”ni olish mumkin.

Kompyuter alohida **qismlardan** (protsessor, monitor, klaviatura va hokazo) iborat, xotira hajmi, monitor kuchlanish, tezkorlik va hokazo parametrlar bilan xarakterlanadi. Bular kompyuter haqida **ma'lumotni** tashkil qiladi. Bundan tashqari kompyuter turli amallarni bajaradi yoki uning ustida har- xil amallar bajariladi, masalan, disklarni joylashtirish, ekranda figurani chiqarish, hisoblashni bajarish. Ya'ni "kompyuter" ob'ekti parametrlar va ular ustida bajariladigan amallar majmuidir. Ob'ekt o'ziga xos xususiyatlarni birlashtiradi va ularni qayta ishlaydi.

OMD texnologiyasi katta programmaviy ma'lumotlarni ishlab chiqishda quyidagi qulayliklar beradi:

- Sohaning oddiy tabiiy tushunchalarini ishlatish va yangi tushunchalarni hosil qilish ;
- Programma hajmini xossa va amallarni ixchamlashtirish asosida kamaytirish, xotirani dinamik ob'ektlarni qo'llash natijasida tejash;
- Ob'ektlar kutubxonasini yaratish, programmani o'zgartirishni osonlashtirish;
- Qismprogrammalarni turli to'plamli formal parametrlari bilan qo'llash;
- Ob'ekt haqida to'liq ma'lumotni bir joyda birlashtirish va programma taxlashni soddalashtirish.

Aytish kerakkim, OYP texnologiyasi kamchiliklarga ham ega va qator holatlarda o'z samaradorligini ko'rsata olmaydi. Masalan, agar uning **virtual metodlari** qo'llanilsa programma tezkorligi pasayishi mumkin. Sodda va kichik hajmga ega bo'lgan programmalarda bu texnologiyani ishlatish **maqsadga muvofiq emas.** OMPni ishlatish programmani **soddalashtirmaydi,** ammo programmani yaratish texnologiyasini osonlashtiradi.

Bizga ma'lum bo'lgan TPas yoki BP muhitlarda OMP-ga asoslangan Turbo Vision kutubxona (qismprogrammalar majmua) si ishlatiladi va muloqotli (interaktiv) programmalar to'plamini loyihalash uchun katta qulayliklar beradi.

OYP texnologiyasi asosiy prinsiplari (xususiyatlari). Texnologiyaning negizida uchta bosh tushunchalar yotadi.

1. **Inkapsulyatsiya** (Qobiqlash)**.** Qobiqlash ma'lumotlar to'plamini va bu to'plamni qayta ishlashga mo'ljallangan algoritmlarni (qismprogrammalarni, ya'ni protsedura va funksiyalarni) birlashtirishdir. Natijada yangi tip- ob'ekt (object) paydo bo'ladi. OMP muhitida ma'lumotlarni **ob'ekt maydonlari** va qismprogrammalarni **ob'ekt metodlari** deymiz.

   Qulayliklari:

   a) Ob'ekt tashqi muhitdan mustaqil bo'ladi.
   b) Ma'lumotlar va algoritmlarni modifikatsiya qilish (o'zgartirish) osonlashadi.
   c) Ob'ektlar kutubxonasini yaratishga qulay sharoitlar paydo bo'ladi.

2. **Vorislik** (Meroslik, irsiylik- наследование). Tayyor ob'ektlar yordamida yangi ob'ektlar ketma- ketligini yaratish. Eski (bosh) ob'ektlarga ma'lumotlar tasvirlanishi va ularni qayta ishlash metodlari yangi ob'ektlarga "merosiy" (otadan- o'g'ilga- nevaraga) o'tishi. Ya'ni bosh ob'ektdagi maydonlar va metodlar avtomatik ravshda "otadan" "o'g'ilga" o'tadi, yangilari bilan kengaytiriladi yoki o'zgartiriladi.

   Qulayliklari:

   a) Ob'ekt xususiyatlarini o'zgartirish osonlashadi.
   b) Masala qo'yilishida bosh (ota) ob'ektda mujassamlangan xususiyatlar keyinchalik yangi (o'g'il) ob'ektlarning xususiyatlari bilan kengaytiriladi.
   c) Murakkab programmani qadamba- qadam ishlab chiqish sharoiti paydo bo'ladi.

3. **Polimorfizm** (umumiylik, o'rindoshlik). Polimorfizm deb- bitta bosh (ota) ob'ektdan paydo bo'lgan qarindosh ob'ektlarning o'xshash muammolarni turlicha (o'ziga xos) usullar yordamida yechishga aytiladi. Boshqacha qilib aytganda, tasvirlangan qismprogrammalar ob'ektlar ketma- ketligida ta'sir

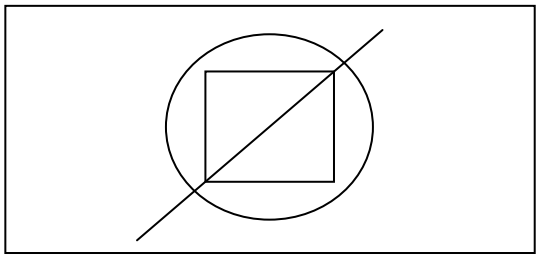
kuchiga ega bo’lib, har bir keyingi pog’onada (bosqichda) turgan ob’ekt **o’zi ustida** bu qismprogrammani **o’zicha** qo’llashi mumkin.

Qulayliklari:

a) Bosh ob’ektdagi metodlarda mavjud bo’lmagan xususiyatlarni keyingi pog’onadagi ob’ektlarda hosil qilish.
b) Yangi ob’ektga bosh ob’ektdagi metod nomi bilan metodni e’lon qilib unda yangi xossalarni kiritish.
c) Ikkita bir xil metod va turli xususiyatlarni loyihalash.
d) Virtuallashtirish (виртуализация), ya’ni nafaqat yangi (o’g’il) ob’ekt metodini bosh (ota) ob’ekt metodlariga murojaat qilinishini, balki teskarisi ham, ya’ni bosh ob’ekt metodlarini yangi ob’ekt metodlariga murojaat qilinishini tashkil qilish.

Texnologiya asoslarini qo’llash va talablarini qondirish uchun quyidagi masalaning yechimining ketma-ketligini tahlil qilamiz.

Masala: Ekranda turli grafik obrazlarni (nuqta, aylana, kvadrat, chiziq) hosil qilib ularni harakatini tashkil qiluvchi programma tuzilsin: Ekran holatini quyidagi chizmada ko’rasatish mumkin.



Programmada obrazlarni ko’chirishni namoyish etish uchun kursorni boshqaruvchi klavishalar: Home, End, PgUp, PgDn- dioganal bo’yicha harakat uchun, Tab- ko’chirish kerak bo’lgan obrazni tanlab olish uchun va Esc- programmadan chiqish uchun ishlatiladi.

a) **Obe’ktlarni tashkil qilish.** Bizga ma’lum programmalash muhitlarida ob’ektlarni yaratish uchun uchta **maxsus so’z:** object, constructor, destructor va uchta **standart direktivalar**: private, public va virtual ishlatiladi.

Object- ob’ektni tasvirlash uchun qo’llanib tiplarni aniqlash bo’limida joylashtiriladi.

```
Type MyObject = object
    {ob’ekt maydonlari}
    {ob’ekt metodlari}
End;
```

Agar ob’ekt boshqa bosh (ota) ob’ektdan hosil qilingan bo’lsa, u holda bosh ob’ekt nomi object so’zidan keyin qavslarda ko’rsatiladi.

```
Type MysecondObject = Object(MyfirstObject)
    { ob’ekt maydonlari}
    {ob’ekt metodlari}
End;
```

Ob’ektlarni tasvirlashda (e’lon qilishda) quyidagilarga rioya qilish kerak.

- Ixtiyoriy ob’ekt xohlagancha avlodga (farzandlarga) va faqat bitta bosh (ota) ob’ektga ega bo’lishi mumkin;
- Ob’ekt tipi faqat asosiy programmaning turlarini tasvirlash bo’limida (type bo’limida) yoki modullar bo’limlarida e’lon qilinishlari kerak. Qismprogrammalarda (protsedura va funksiyalarda) ob’ektlarni tasvirlash mumkin emas;
- Ob’ekt tipini tasvirlashda ma’lumotlar maydonlari metodlarni tasvirlashdan oldin bo’lishi kerak;
- Ob’ekt komponentalari fayl bo’lishi mumkin emas va o’z navbatida fayllar “ob’ekt” tipidagi komponentalarni saqlamaydi.

Masalani yechishga o’tsak, bosh (ota) ob’ektini TGraphObject bilan belgilaymiz. Qobiqlash asosida boshqa ob’ektlar (o’g’illar) uchun umumiy bo’ladigan maydon va metodlarni kiritamiz.

```
Type TgraphObj = object
        Private {ob’ekt maydonlari foydalanuvchidan yashirilinadi}
        x, y: integer; {nuqta koordinatasi}
        color: word;{figura rangi}
        public {ob’ekt metodlari foydalanuvchi uchun ochiq}
        constructor Init(ax, ay: integer; aColor: word); {ob’ekt ekzemplyarini
                                                (nusxasini) hosil qiladi}
        procedure Draw(aColor: word); virtual; {ob’ektni berilgan aColor
                                        rangi bilan chizadi}
        procedure Show; {color rangi bilan ob’ektni ko’rsatadi}
        procedure Hide; {fon rangi bilan ob’ektni yashiradi}
        procedure MoveTo (dx, dy: integer); {ob’ektni koordinatasi x+dx,
                                                y+dy bo’lgan nuqtaga ko’chiradi }
    end;
```

Keyinchalik TgraphObj – dan “o’g’il” ob’ektlarni hosil qilib nuqta, chiziq, to’rtburchak va aylananing o’ziga xos xususiyatlarini ko’rsatuvchi xarakteristikalarini tashkil qilamiz. Bu grafik obrazlarning har biri ekrandagi o’rni (x va y maydonlari) va rangi bilan (color maydoni) xarakterlanadi. Ular Draw metodi yordamida ekranda akslantiriladi, Show metodi bilan “o’zini ko’rsatadi”, Hide metodi bilan “yashirinadi” va MoveTo yordamida ekran bo’ylab harakat qiladi.

Private direktivasi yashirilgan (yopilgan) maydon va metodlar seksiyasini ochadi. Agar shu seksiyada programmalovchi TPU- modulidan foydalangan bo'lsa u holda foydalangan ob'ektlarning elementlari "ko'rmas" shakliga o'tadi. Yuqoridagi tasvirlashda reper (nishonli) nuqta hisoblanadigan (x, y) ixtiyoriy tarzda o'zgartirilishi mumkin emas. X va Y qiymatlarini o'zgartirish uchun ob'ekt tarkibiga kiradigan Init va MoveTo metodlari ishlatiladi.

Yashirilgan (ko'rmas) maydon va metodlar ob'ekt tasvirlangan programmaviy birliklarda (programma yoki modul) ochiq deb qabul qilinadi. Agar biz keyinchalik GraphObj modulini yaratsak, u holda yashirilgan elementlar faqat shu modulda "ochiq" va bu modulni ishlatuvchi asosiy programmada "yopiq" deb hisoblanadi.

Aytib o'tish kerakkim, oddiy programmalarda "yashiriladigan" elementlarni tashkil qilish shart emas. Shuning uchun yuqoridagi tasvirlashda Private…, public menizmlarni ishlatmasak quyidagi fragmentni hosil qilamiz.

```
Type TGraphObj=object
                    x, y: integer; color: word;
                    Constructor Init(ax, ay: integer; acolor: word);
                    Procedure Draw(acolor: word): virtual;
                    Procedure Show;
                    Procedure Hide;
                    Procedure MoveTo(dx, dy:integer);
                End;
```

Umuman olganda, maydonlar oddiy o'zgaruvchi kabi tasvirlanadi, ixtiyoriy ma'lumotlar strukturasi va o'z navbatida boshqa ob'ekt ham bo'lishi mumkin. Metodlar esa oddiy protsedura va funksiyalarni tasvirlashga o'xshaydi.

Ob’ekt- bu ma’lumotlar tipidir. Shuning uchun uni qandaydir “shablon” (qolib) deb qabul qilib uning ekzimplyarlarini (nusxalarini) **konstruktor** yordamida aniqlaydilar. Bunday nusxalar xohlagancha bo’lishi mumkin. Constructor so’zi procedure so’zining o’rnida ishlatiladi. Ob’ektda virtual metodlar bo’lmasa, u holda konstruktor ishlatilmaydi. Aksincha agar bitta metod virtual tasvirlangan bo’lsa, u holda hech bo’lmaganda bitta konstruktor tasvirlanadi. **Konstruktorga murojaat qilish ixtiyoriy virtual metodga murojaat qilishdan oldin kelishi kerak.**

Kostruktorning asosiy vazifasi ob’ekt maydonlarini aniq qiymatlar bilan to’ldirishdir. Bir ob’ektning har xil nusxalari (ekzimplyarlari) faqat qiymatlari bilan, ya’ni maydonlar qiymatlari bilan farq qiladi. Keltirilgan misolda Init konstruktori nusxani to’liq aniqlash uchun ma’lumotlarni ax, ay, acolor parametrlari orqali qabul qiladi.

Draw protsedurasi TGraphObj ob’ektining avlodlarida turlicha ishlatilib har xil figuralarni (nuqta- PutPixel, chiziq- Line va hokazo) tasvirlash (chiqarish) uchun qo’llaniladi.

Ob’ektning hamma xususiyatlarini tasvirlash uchun ob’ekt metodlarini ochib berish, ya’ni protsedura va funksiyalarni tasvirlash kerak. Tasvirlash (protsedura va funksiya) ob’ektni tasvirlashdan keyin joylashtirilishi talab qilinadi.

Bu qoidalarni hisobga olib tasvirlashni davom ettiramiz va quyidagilarni hosil qilamiz.

```
Type
    TGraphObj=object
    ⋮
    End;
Constructor TGraphObj.Init;
```

```
        Begin x:=ax; y:=ay; color:=acolor; end;
    Procedure TGraphObj.Draw;
        Begin;
        {bu protsedura “o’g’il” ob’ektlarda ishlatiladi}
        End;
    Procedure TGraphObj.Show;
        Begin Draw(Color) end;
    Procedure TGraphObj.Hide;
        Begin Draw(GetBKColor) end;
    Procedure TGraphObj.MoveTo;
        Begin Hide; x:=x+dx; y:=y+dy; Show end;
```

Metodlarni tasvirlash jarayonida **Metod nomidan oldin ob’ekt nomi** qo’shiladi, ya’ni metodning tarkibiy (murakkab-составное) nomi hosil bo’ladi. Bunday nomlanish aniq bir protsedurani nimaga qarashli ekanligini bildiradi. Ob’ekt maydonlarini protseduralarda **qayta tasvirlash mumkin emas.** Masalan, bu fragment

```
    Constructor TGraphObj.Init;
        Var
            x, y: integer; color:word; {Xato!}
        Begin … end;
```

Ko’rinib turibdikim, abstrakt ob’ekt TGraphObj ekranga ma’lumot chiqarish uchun ishlatilmayabdi. Shuning uchun uning metodi Draw ish bajarmaydi. Ammo Hide, Show va MoveTo metodlari Draw metodiga murojaat qilib ish bajaradi.

Sinov savollari:

1. Ob’ekt nima? Uning xususiyatlarini izohlang.
2. Texnologiyaning mohiyati nimadan iborat?
3. Bu texnologiya qaysi metodlarga asoslanadi?
4. Virtual metod nima uchun kerak?
5. Texnologiyaning noqulayliklari ham bormi?

Adabiyotlar

[5], [8], [11], [13], [16]

Ma’ruza № 13

Mavzu: Vizual programmalash muhiti

Reja:

1. Interfeysni tashkil qilish asoslari.
2. Integrallashgan muhit va interfeys.
3. Vizual texnologiya xususiyatlari.
4. Texnologiyaning instrumental vositalari.
5. Delphi muhiti qulayliklari.

Tayanch iboralar: interfeys, integrallashgan muhit, oyna, forma, menyu, komponenta, holat, vizual programmalash, ilovalar.

Ma’ruza bayoni

Programma ta’minotini yaratishda foydalanuvchi va programma (tizim) orasidagi muloqot turini, ya’ni **foydalanuvchi interfeysini** tashkil qilish muhim masalaga aylandi. Buning sabalari quyidagilardir:

- Kompyuter bilan muloqot qilish jarayoni foydalanuvchiga qo’shimcha noqulayliklar yaratmasligi kerak;
- Muloqot paytida aktivlik (iteraktivlik) programmadan foydalanuvchiga va teskarisi, foydalanuvchidan programmaga o’tishi maqsadga muvofiq;
- Interfeysni qulay va tushunarli formalarini ishlab chiqish va standartlash uchun yangi metod va vositalarni loyihalash talab qilinadi.

Zamonaviy integrallashgan tizimlar (Delphi, VBA, C++, Java) muhitida foydalanuvchi interfeysi quyidagi elementlar yordamida loyihalanadi va ishlab chiqariladi:

- Ro’yxat;
- Tahrirlash satri;
- Nishon;

- Menyu;
- Opsiya;
- Esga solish (ko’rsatma);
- Instrumentlar paneli: knopkalar, ro’yxatlar va ular bilan bog’liq bo’lgan amallar;
- Holat satri va hokazo.

Bunday tizimlarning yana bir xususiyati ilovalarni (Application) juda tez ishlab chiqarish vositalar va metodlarga ega ekanligidir. Ana shunday texnologiyalardan biri **vizual (ayonli, ko’rinarli) programmalash** hisoblanadi.

Vizual programmalashning asosida **oynali iterfeys** va **vizual dizayn** (oro berish) prinsiplari yotadi. “Sichqoncha” manipulyatori vizual (ayonli) ob’ektlar (piktogramma, rasm, panel) yordamida turli xil ilovalarni tashkil qilishga yordam qiladi. Dasturlash bu usulining yana bir qulayligi programmaviy modullardan foydalanib ilovalar yaratish va “satrma- satr” (operatorli) programmalashni qisqartirishga asoslanadi. Bular natijasida programma tuzish jarayoni vizuallash-tiriladi. Ko’rsatilgan xususiyatlarga ega bo’lgan muhitlardan biri Delphi tizimi va uning vizual programmalashga mo’ljallashgan usul va vositalarini tahlil qilamiz.

Delphi muhitida ish ob’ektga-mo’ljallangan programmalash texnologiyasi va programmalash jarayonini vizuallashtirishga asoslanadi. Bu jarayonning negizida **ob’ektlarni aniqlash** va ular ustida **amallar bajarish** yotadi. Foydalanuvchi tomonidan ishlab chiqariladigan programma windows- ilovasi hisoblanadi. Programma **proekt** deyiladi. Natijada biz kompilyatsiya (tarjima) qilingan va Delphi muhitidan tashqari muhitda ham ishlaydigan programmani hosil qilamiz. Proekt bitta yoki bir nechta formalardan tashkil topishi mumkin. **Forma** programma oynasi bo’lib interfeys vositalarini saqlaydi. Foydalanuvchi bu vositalar yordamida programma bilan Windows-ilova kabi hamkorlik qiladi. Interfeys vositalari **interfeysli komponentalar** deb nomlanadi va vizual programmalashning tarkibiy birliklaridir.

Formalar va uning har bir komponentasi Object Pascal tilining **ob’ekti** bo’lib **xossalar** to’plami va **metodlar** (harakat, amal, operatsiya) to’plamidan iborat. Xossalar ob’ekt holatini va metodlar esa shu ob’ekt ustida bajariladigan amallarni bildiradi.

Vizual (interfeysli) komponentalar xossalariga quyidagi xarakteristikalar kiradi:

- Nom (name);
- Sarlavha (Caption);
- Rang (Color);
- Matn (Text) va boshqalar.

Xossalar **dinamik** tarzda (programma ishlash paytida) yoki **statik** tarzda (loyihalash paytida) aniqlanadi.

Metodlar ham ikki guruhga bo’linadi. Birinchi guruh metodlari oldindan tizim muhitiga kiritilgan tayyor metodlardir. Ikkinchisi esa, programmalovchi tomonidan yaratiladigan metodlar hisoblanadi.

Proekt (loyiha) turli tipdagi fayllar majmuasidan tashkil topadi.

Proekt fayli (.dpr-kengaymali) boshqaruvchi programma bo’lib, Object Pascal tilida yoziladi. Katta hajmga ega emas Delphi tomonidan avtomatik ravshda tashkil qilinadi, proektning bosh formasini ishga yurgizadi. Bu faylni tahrirlash **mumkin emas.**

Proekt formalarining har biri ikkita fayl yordamida tasvirlanadi. Birinchisi .dfm kengaymali fayl bo’lib **forma xossalarining** qiymatlarini saqlaydi. Ikkinchisi esa, .pas fayli Object Pascal tilining **moduli** (unit) hisoblanadi.

Modulning tarkibiga quyidagilar kiradi:

- Uses direktivasi. Drektiva proektda standart va boshqa proekt formalardagi modullar ishlatilishi haqida ma’lumot beradi;

- Forma tipini tasvirlanishi. Formaga qarashli bo’lgan komponentalar tipini aniqlaydi;
- Modul qismprogrammalari ishlatadigan global konstanta, tip va o’zgaruvchilarni tasvirlanishi;
- Forma holatlariga aks ta’sirni (reaksiya) tasvirlovchi protseduralar.
- Proekt tarkibiga **resurs** (manba) fayllari ham kiradi.

Delphining **ishlab chiqarish** (loyihalash) muhiti **to’rtta oynalardan** iborat. Bu muhit **ko’poynali ilovalarning** bir hujjat interfeysi yordamida boshqarilishi (SDI-Single Document Interface) shaklida tashkil qilingan. Tizim bilan muloqot qilish va ish bajarish oynalar yordamida amalga oshiriladi. Oynalar turi va ular bajaradigan ishlarni ko’rib chiqamiz.

Bosh oyna. Asosiy oyna bo’lib unda **menyu, tezkor o’tish paneli** (lavhasi, SpeedBar), **komponentlar majmui** (palitra) joylashtirilgan. Menyu satri muhitning oynalarini tekshirish uchun ishlatiladi. Panel umumiy operatsiyalar (amallar) ga tezkor murojaat qilishni ta’minlaydi. Komponentalar palitrasida bir nechta guruh mavjud bo’lib ularning har birida piktogrammalar saqlanadi.

Palitradan kerakli komponentni topib “sichqoncha” yordamida forma oynasiga ko’chirish mumkin. Shu asosda **aktiv ob’ekt** hosil qilinadi va uning xossalari, holatlari belgilanadi. Bosh oyna quyidagi oynalarni boshqaradi.

a) **Ob’ektlar inspektori (nazoratchisi) oynasi.**

Bu oyna ko’p sahifali bo’lib yorliqlar bilan belgilangan xossalar (Propeties) va holatlarni (Events) saqlaydi. Xossalar sahifasida ob’ektlarga tegishli xossalar,holat-lar sahifasida esa, joriy ob’ektlar javob berishi kerak bo’lgan holatlar saqlanadi.

b) **Formalar oynasi.** Oynada komponentlar (ob’ektlar) saqlanadi. Ular yordamida foydalanuvchi ish jarayonida programma bilan muloqot qiladi, ya’ni ma’lumot kiritadi va ma’lumot oladi.

d) Tahrirlovchi oynasi. Object Pascal tilida programmani (programma moduli kodini) terish va tahrirlash uchun ishlatiladi. Oyna yordamida programma modullari orasida o’tishlarni bajarish ham mumkin.

Umuman olganda Delphi muhitida proektni (loyihani) yaratish uchun quyidagi bosqichlarni bajarish kerak.

1. Ishlab chiqarish proektni tashkil qilishdan boshlanadi. (File\New Project).
2. Agar proekt mavjud bo’lsa uni ochish kerak. (File\Open Project).
3. Proektni to’liq (File\ Save Project) yoki uning alohida fayllarini saqlash mumkin. (File\ Save).
4. Proektni kompilyatsiya qilish va bajarish uchun Run\Run yoki SpeedBardagi Run tugmachasini bosib ijro etish kifoya.
5. Proekt bilan ish tamom bo’lganda uni yopish (File\Close Project) kerak.

Asosiy forma oynasi proektni tuzish yoki ochishda avtomatik ravishda tashkil qilinadi. Foydalanuvchi tomonidan bu oynaga kerakli komponentlar ko’chiriladi, inspektor yordamida forma xossalari aniqlanadi, holatlari belgilanadi va kod tahrirlovchisi oynasida esa Object Pascal tilida tashkil qilingan protsedura hosil bo’ladi.

Sinov savollari:

1. Interfeysning qaysi turlarini bilasiz?
2. Integrallashgan muhit nima?
3. Vizual texnologiya xususiyatlari nimadan iborat?
4. Texnologiyaning instrumental vositalarini tahlil qiling.
5. Delphi muhitida vizual programmalash qanday bajariladi?

Adabiyotlar

[8], [13], [14], [15], [16], [18]

Adabiyotlar

1. Бек Л. Введение в системное программирование. М: Мир, 1988
2. Богославский А. Системное программирование на Ассемблере для IBM- совместимых персональных компьютеров М: Память, 1992
3. Лей Р. Разработка драйверов устройств для MS-DOS. –Рязань: Versus Ltd, 1992.
4. Скляров В. Программирование на языке ассемблера: Учеб. пособие. М: Высш. шк., 1999
5. Поляков Д., Круглов И. Программирование в среде Турбо Паскаль (версия 5.5). –М: МАИ, 1992.
6. Микляев А. Настольная книга пользователя IBM PC. –М: Солон-Р, 2000.
7. Поттосин И., Бежанова М. Математическое обеспечение ЭВМ: Инструментарий и обучающие средства/НГУ – Новосибирск, 1996.
8. Бобровский С. Delphi 5: Учебный курс. – СПб: Питер, 2000.
9. Касьянов В. Поттосин И. Методы построения трансляторов. – Новосибирск: Наука, 1986.
10. Вирт Н. Алгоритмы +структуры данных = программы. М: Мир, 1986.
11. Фаронов В. Турбо Паскаль 7.0. Практика программирования. М: Нолидж, 1999.
12. Брябрин В. Программное обеспечение персональных ЭВМ. М: Наука, 1989.
13. Андреев А., и др. Новые технологии Windows-2000. –СПБ: БХВ, 2000.
14. Фаронов В. Паскаль и Windows. –М: МВТУ, 1995.
15. Бежанова М., Москвина Л. Проектирование визуальных интерфейсов и приложений баз данных на основе среды Delphi: Учеб. Пос./НГУ - Новосибирск. 1999.

16. Богуславский А. Соколов С. Основы программирования на языке Си ++. Коломна: КГПИ, 2002
17. Энциклопедия Интернет. –СПб: Питер, 1999
18. Мурадян А. Визуальное программирование//Монитор. 1995. №4
19. Кейлингерт П. Элементы операционных систем. М: Мир, 1985
20. Абрамов С. И др. Задачи по программированию. – М.: Наука , 1988. 224 с.
21. Қосимов С. Ахборот технологиялари. Ўқув қўлланма. – Тошкент: Алоқачи, 2006. – 370б.
22. http://ablsoft.uz
23. http://www.litera.ru
24. http://informatikaplus.narod.ru
25. http://www.softdrom.ru
26. http://softblog.ru
27. http://www.ziyonet.uz
28. http://www.referat.uz