O’ZBEKISTON RESPUBLIKASI OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI

BUXORO DAVLAT UNIVERSITETI

Fizika – matematika fakulteti

“Amaliy matematika va axborot texnologiyalari” *kafedrasi*

5480100-“Amaliy matematika va informatika” ta’lim yo’nalishi bo’yicha bakalavr darajasini olish uchun

Bobomurodov Javlon Boboqulovichning

“Ob’yektga mo’ljallangan dasturlashda keraksiz ob’yektlarni yo’qotish strategiyalari” mavzusida

BITIRUV MALAKAVIY ISHI

“Ish ko’rildi va himoyaga ruxsat berildi”

Ilmiy rahbar: dots.T.B Boltayev

“_____”_____2012y.

Kafedra mudiri:
_____dots.I.G.Rasulov

“_____” _____ 2012y.

Taqrizchi: dots.Sh.S.Yo’ldoshev

“_____” _____ 2012y.

“Himoya qilishga ruxsat berildi”
Fakultet dekani **prof.Sh.M.Mirzayev**
“_____” _____ 2012y.

Buxoro-2012y.

MUNDARIJA.

Kirish.

O’zbekistonda ta’lim tizimining axborotlashtirishi xalqaro hamjamiyatda ham tan olinadi. Masofaviy ta’limni rivojlantirish bo’yicha bir qator dasturlar ishlab chiqilmoqda. Iqtisodiyot va jamiyatda islohotlarning o’tkazilishi o’quv jarayonining zahira hajmini keskin oshirish bo’yicha yangi talablar qo’yildi. Hozirgi kunda axborot eng asosiy ishlab chiqaruvchi resurslardan biriga iqtisodiyot va umuman jamiyatning rivojlanish poydevoriga aylanmoqda.

O’zbekistonda o’qitish texnologiyalari zamonaviylashtirish jadallashtirish rivojlangan iqtisodiyotli mamlakatlarga qaraganda yanada dolzarb ahamiyatga ega. Chunki hozirgi kunda milliy ta’lim tiziminnig salohiyatli tizimli rivojlanishinig yanada yuqori pog’onasiga ko’tarildi

Prezidentimiz I.A.Karimov 2001-yil Oliy Majlisining 5-sessiyasida so’zlagan nutqida axborot texnologiyalarni va kompyuterlarni jamiyat hayotiga kishilarning turmush tarziga, o’rta maktab, o’rta maxsus va oily ta’lim muassasalariga jadallik bilan kirish g’oyasi ilgari surildi.

Hozirgi zamon mutaxisslardan faoliyat doirasi qanday bo’lishidan qanday bo’lishidan qat’iy nazar, informatika bo’yisha keng ko’lamdagi bilimlarga zamonaviy hisoblash texnikasi va informatsion aloqa va kommunikatsiya tizimlari, orgtexnika vositalari va ulardan foydalanish borasida yetarli malakaga ega bo’lishi, hamda yangi informatsion texnika va texnologiya asoslarini uning ertangi kuni, rivoji to’g’risidagi bilimlarni o’zida mujassamlashtirgan bo’lishi shart. Zamonaviy hisoblash texnikalari va informatsion texnologiyalarining kun sayin rivojlanib, jamiyatning esa, tobora informatsiyalashib borishi sababli, uzluksiz ta’lim tizimining o’rta va yuqori bosqishlariga informatika, ishlab chiqarish va boshqarish jarayonining kompyuterlashtirish bo’yicha bir qator o’quv fanlari kiritilgan.

XX asr o’rtalariga kelib, tezkor mashina mexanizmlaridan foydalanila boshlandi, murakkab texnika va texnologiyalardan ko’p funksiyalilari o’ylab topildi. Ko’pgina masalalarni hal qilish jarayonida axborot hajmi behisob bir majmuaga aylandi, hamda bu axborotlarni yig’ish va uzatish vositalarini yaratish,

ularni vaqtida qayta ishlab, boshqarish uchun zarur bo’lgan choralarni belgilab chiqish kerak bo’ladi.

Respublikamizda olib borilayotgan islohotlarning tarkibida yuqori malakali mutaxasislarning roli benihoya kattadir. Prezidentimiz ta’kidlaganidek, “Ertangi kun yangicha fikrlay oladigan zamonaviy bilimga ega bo’lgan yuksak malakali mutaxasislarni talab etadi”. Shu sababli xalqimizning boy intelektual merosi va umumbashariy qadriyatlari, zamonaviy madaniyat, iqtisodiyot, fan, texnika va texnologiyalar asosida yuksak mutaxasislar tayyorlash tizimi ishlab chiqildi va jadal sur’atlar bilan hayotga tadbiq etilmoqda.

Ta’lim tizimi keng ko’lamli va chuqur islohotlarni amalga oshirish muddatlari O’zbekiston Respublikasining “Ta’lim to’g’risida” gi qonun va “Kadrlar tayyorlash milliy dasturi” da ta’kidlanganidek, “Kadrlar tayyorlash tizimi va mazmunini mamlakatning ijtimoiy va iqtisodiy taraqqiyoti istiqbollaridan, jamiyat ehtiyojlaridan fan, madaniyat texnika va texnologiyalarning zamonaviy yutuqlaridan kelib chiqqan holda qayta qurish“ lozim.

Mavzuning dolzarbligi. Ko`rsatkichlarni osilib qolishi dasturchilarning keng tarqalgan xatoliklaridan biridir. Agarda ko`rsatkich murojaat qilgan ob`ektni `delete` operatori orqali o`chirsak va bu ko`rsatkichga 0 qiymati o`zlashtirilmasa, bu holda joriy ko`rsatgich osilib qoladi. Ko`rsatkichlar ustida `delete` operatorini ishlatishda ehtiyot bo`lish lozim. Bu holda, agarda ko`rsatkichga 0 qiymat o`zlashtirilmasa u oldingi qiymatini, ya`ni o`chirilgan ob`ekt adresini o`zida saqlaydi. Dasturni ishlashi davomida aynan shu adresga boshqa ob`ekt aniqlanishi mumkin, chunki oldingi ob`ekt `delete` operatori orqali o`chirilganidan so`ng, dinamik xotirani bu qismidan bemalol foydalanish mumkin bo`lib qoladi. Natijada biz joriy ko`rsatkichda biz umuman kutmagan ob`ektga murojaat hosil bo`lib qoladi va u dasturni ishlash jarayonida noto`g’ri hodisalarni keltirib chiqarishi mumkin. Ko`rsatkichlar bilan e`tiborsiz ishlash natijasida xotiraning sirqib ketishiga yo`l qo`yish mumkin. Bu ko`rsatkich murojaat qilib turgan xotira bo`shatilmasdan, shu ko`rsatkichga yangi qiymat o`zlashtirilgan vaqtda ro`y

beradi.Agarda o`zgaruvchi uchun ajratilgan xotira kerak bo`lmasa uni bo`shatish zarur.

Muammoning o'rganilganlik darajasi. Javada o'rganilgan C++, Pascalda o'rganilmagan.

BMI maqsadi. Xotirani dinamik boshqarish apparatini C++ da amalga oshirish. Dinamik hotira bu dastur bajarilishi jarayonida ajratiladigan hotiradir. Dinamik hotira ajratilgandan so'ng to free() funktsiyasi tomonidan bo'shatilmaguncha saqlanadi.

Agar dinamik hotira dastur bajarilishi tugaguncha bo'shatilmagan bo'lsa, avtomatik ravishda dastur tugaganda bo'shatiladi. Shunga qaramay ajratilgan hotirani dasturda mahsus bo'shatish dasturlashning sifatini oshiradi.Dastur bajarilish davomida ajratilgan hotira qismiga murojaat imkoniyati shu qismga adreslovchi ko'rsatkichga bog'likdir. Shunday qilib, biror blokda ajratilayotgan dinamik hotira bilan ishlashning quyidagi uchta varianti mavjuddir:

-Ko'rsatkich avtomatik hotira turiga kiruvchi lokal ob'ekt. Bu holda ajratildgan hotiraga blokdan tashqarida murojaat qilib bo'lmaydi, shuning uchun blokdan chiqishda bu hotirani bo'shatish lozimdir.

-Ko'rsatkich avtomatik hotira turiga kiruvchi lokal statitk ob'ekt. Blokda bir marta ajratilgan dinamik hotiraga, blokka har bir qayta kirilganda ko'rsatkich orqali murojaat qilish mumkindir. Hotirani blokdan foydalanib bo'lgandan so'ng bo'shatish lozimdir.

-Ko'rsatkich blokka nisbatan global ob'ektdir. Dinamik hotiraga ko'rsatkich ko'rinuvchi hamma bloklardan murojaat qilish mumkin.Hotirani fakat undan foydalanib bo'lgandan so'ng b'oshatish lozimdir.

BMI ob'yekti. Kompyuter xotirasi nomerlangan yacheykalar ketma-ketligidan iborat. Har bir o'zgaruvchining qiymati uning adresi bilan ataluvchi alohida xotira yacheykasida saqlanadi. Har bir yacheyka bir bayt o`lchovga ega. Agar o`zgaruvchi uchun ko`rsatilgan tip 4 baytni talab qilsa, uning uchun to`rtta yacheyka ajratiladi. Aynan o`zgaruvchini tipiga muvofiq ravishda kompilyator bu o`zgaruvchi uchun qancha joy ajratish kerakligini aniqlaydi. Xotira bir necha

sohalarga ajraladi,bular Global o’zgaruvchilar sohasi, Bo’sh yoki ob’yektlar o’rtasida dinamik taqsimlanuvchi xotira, Registrli xotira, Dastur sigmenti, Stekli xotira.

BMI predmeti. O'zgaruvchilar hotirada ikki uslubda saqlanishi mumkin. Birinchisi dinamik, ikkinchisi statik yo'ldir. Dinamik bo'lgan birlik u e'lon qilingan blok bajarilishi boshlanganda tuziladi, va ushbu blok tugaganda buziladi, u hotirada egallagan joy esa bo'shatiladi.

Hotirada boshqa tur saqlanish yo'li bu statik saqlanishdir. Statik sifatini o'zgaruvchi va funksiyalar olishlari mumkin. Bunday birliklar dastur boshlanish nuqtasida hotirada quriladilar va dastur tugashiga qadar saqlanib turadilar.

BMI metodologik asoslari. Tuzilma bu tarkibli ob’ekt bo‘lib, unga har qanday turdagi elementlar kiradi. Bir turdagi ob’ekt bo‘lgan massivdan farqli o‘laroq, tuzilma bir turda bo‘lmasligi mumkin. Tuzimaning turi quyidagi ko‘rinishdagi yozuv bilan aniqlanadi:

struct {aniqlovchilar ro‘yxati}

Eng soda dinamik informatsion tuzilma elementlari qo’yidagi tuzilmali tip orqali ta'riflangan ob'ektlardan iborat ro’yhatdir.

```
Struct tuzilmali tip nomi
{
tuzilma elementlari ;
Struct tuzilmali tip nomi*kursatkich;
};
```

BMI metodlari. Dasturda ikki asosiy tur ma'lumot tuzilmalari mavjuddir. Birinchisi statik, ikkinchisi dinamikdir. Statik deganimizda hotirada egallagan joyi o'zgarmas, dastur boshida beriladigan tuzilmalarni nazarda tutamiz. Dinamik ma'lumot tiplari dastur davomida o'z hajmini, egallagan hotirasini o'zgartirishi mumkin. Agar tuzilma bir hil kattalikdagi tiplardan tuzilgan bo'lsa, uning nomi massiv (array) deyiladi. Massivlar dasturlashda eng ko'p qo'laniladigan ma'lumot tiplaridir. Bundan tashqari tuzilmalar bir necha farqli tipdagi o'zgaruvchilardan tashkil topgan bo'lishi mumkin.

I.BOB.DINAMIK TUZILMALAR HAQIDA UMUMIY MA’LUMOT.

1.1.Ob’yekt tushunchasi.

Ob’ekt - mavhum (abstrakt) mohiyat bo‘lib, u bizni o‘rab turgan haqiqiy olamning tavsiflariga ega. Ob’ektlarni yaratish va ular ustida manipulyatsiyalar olib borish - C++tilining qandaydir alohida imtiyozi emas, balki ob’ektlarning tavsifi va ular ustida o‘tkaziladigan operatsiyalarni kodli konstruktsiyalarda o‘zida mujassamlantiradigan dasturlash metodologiyasi (uslubiyoti) ning natijasidir. Dasturning har bir ob’ekti, har qanday haqiqiy ob’ekt kabi, o‘z atributlari va o‘ziga xos xulq-atvori bilan ajralib turadi. Sinfning ekzemplyarlariga ob`ekt deb murojaat qilishimiz mumkin. Bu ob`ektlarga, va ob`ektga mo`ljallangan – bajarilish vaqtini hisoblash modeliga diqqatimizni qaratishimiz kerak. Endi realizatsiya aspektlariga e`tibor qilishimiz zarur. Xususan xotiradan foydalanish savolini ko`rib chiqamiz. Bir necha marotaba belgilab o`tdikki, dasturiy ta`minot ishlab chiqishda ob`ektli texnologiyalarni yutug’i shundaki qismlarini ishlab chiqganda to`liq hajmda boshqariladi. Shuning uchun bu sohada sayohat foydali bo`ladi, hattoki sizning qiziqtiradigan analizlash va proyektlash sohasiga bog’langan bo`lsa. Tuzilmalarini bajarilish vaqtiga ta`sirini qaramasdan, metodni tushunish imkonsiz. Juda ham yaxshi misolda "yuqori" darajali muammolarni realizatsiyalash qanchalik savolni xato ajratish ob`ektlarning tuzilmasini o`rganish hozirgi bo`limimizda xizmat qiladi. Realizatsiyalash savoliga bog’liq, yangi texnik jarayonlarni ko`rish paytida yana ham chuqurroq tushuncha abstrakt tushunchasiga kelamiz. Ko`rsatkichli va ochiq qiymatli va tipik misol sifatida xizmat qiladi. Haqiqattan ham bu to`liq va qisman munosabatni, Ob`ektga mo`ljallangan diskursiyaga doimo muhokama qilinadigan savolimizni javobidir. Komp’yuterning ayrim adabiyotlarida realizatsiya qiymatini kamsitib va tahlil qilishni eng asosiy deb hisoblaydiki. Dasturiy ta`minotni ishlab chiqish – bu modelni ishlab chiqishdir. Yaxshi ishlab chiqish texnikasi ko`pincha yaxshi modellash muhitini birgalikda yaratishdan iborat bo`ladi. Dasturiy tizimdan tashqari buni boshqa sohalarda ham ko`p foydalansa bo`ladi. Bu bo`limda ko`pincha modellashga mo`ljallangan, realizatsiya

bu shunchaki ibora. Bu erda turli xil tizimlarda haqiqiy va kerakli buyruqlar ro`yxatini qurish uchun qanday ob`ektlar tuzilmasidan foydalanish ko`rsatilgan.

Ob`ektga mo`ljallangan dasturlashni bajarilish jarayonida bir qancha ob`ektlar yaratiladi. Bu ob`ektlarni tashkil etish va ular orasidagi munosabatlar, bajarilish vaqtidagi konstruktsiyasida aniqlanadi. Ob`ektlar xususiyatini ko`rib chiqamiz.Hammasidan avval, "ob`ekt" iborasini ma`nosini esga tushiramiz.

Ob`ekt – bu ba`zi bir sinfning ekzemplyari.

Dastur tizimi bajarilish paytida, berilgan C sinfini ekzemplyarini yaratish yoki klonlash potseduralari, C sinfini ma`lumotlar tuzilmasini mos ravishda turli nuqtalarda saqlashi mumkin. Masalan POIN sinfini ekzemplyarining ma`lumotlar tuzilmasi ikki maydondan iboratligini tasavvur etsak bo`ladi, mos ravishda sinfning x va y atributlari sifatida. Mumkin bo`lgan barcha sinflarning ekzemplyarlari ob`ektlar to`plamini tashkil etadi.

Ob`ektga mo`ljallangan –dasturiy ta`minot dunyoda rasman aniqlangan. Lekin tilda doimo "ob`ekt" iborasi ancha keng ma`noni anglatadi. Ixtiyoriy dasturiy tizim, aniq bir tashqi tizimga bog’langan bo`lib u "ob`ekt" larni o`zi ichiga saqlashi mumkin: nuqta, chiziq, grafikali tizimda jism va yuzasi; to`lov hisob – kitob ni hisoblash tizimida xodimlar va ularni okladlari, ishlagan maoshlarini hisoblash va h.k. Bunday hollarda, qoida sifatida haqiqiy ob`ektlar mos ravishda dastur ob`ektlariga o`tkaziladi. EMPLOYEE sinfi oylik maoshni hisoblash tizimida misol bo`la oladi, uning ekzemplyarlari xodimlarning komp’yuterdagi modeli hisoblanadi.

Haqiqiy tizimlarning modellari maqsadi uchun, "ob`ekt" so`zining haqiqiy ikkilangan ma`nosi ob`ektga mo`ljallangan - metodning natijaviyligi va kuchidir. Bu to`g’ridan – to`g’ri aks etirish (direct mapping) printsipini ko`rganimizdek, modulli loyihalash printsiplarni talabi sifatida hisoblanadi. Ajablanadigan joyi yo`qki, ayrim sinflar tashqi muammoli tiplarning sohasidagi modeli hisoblanib, sifning ekzemplyari - haqiqiy ob`ektning modelidir.Lekin "реальность" so`zi bilan "ob`ekt" so`zini aralashtirmaslik kerak. Haqiqiy tildan so`zni qarzga olish va maxsus maqsadlarga foydalanish, ilm va texnika uchun katta xavf tug’diradi.

"Ob`ekt" iborasining har kunlik ma`nosi to`lib ketganki, undan texnik ma`noda foydalanish tushunmovchilik manba`siga aylanishi mumkin. Tavsilotlar:

-Muammoli soha tipiga hamma sinflar ham munosib emas. Ko`p sinflar, loyihalash va ishlab chiqish uchun kiritilgan, modellash tizimiga o`xshashlari yo`q. Amaliyotda aynan shu sinflar katta bo`lmagan qiymatlarga ega va ularni loyihalash qiyinroq bo`ladi.

-Ayrim kontseptsiyalar muammoli sohalarda sinflarga olib keladi, Garchi muammoli sohada haqiqiy ob`ektlar mavjud emas, qaysiki sinfning ekzemplyarlarini mos ravishda qo`yish mumkin bo`lsa. STATE sinfiga, tavsiflovchi holat tizimiga yoki COMMAND sinfiga misol bo’ladi.

"Ob`ekt" so`zi bu bo`limning kontekstida foydalanilishi, umumiy va texnik ma`noda termin ifodalanayapti. Bu holatlarda, dasturiy ob`ekt yoki tashqi ob`ektlar uchun aniqlikdan foydalaniladi qachonki bu farqlarni belgilash kerak.

1.2. Statik va dinamik tuzilmalar.

Dasturda ikki asosiy tur ma'lumot tuzilmalari mavjuddir. Birinchisi statik, ikkinchisi dinamikdir. Statik deganimizda hotirada egallagan joyi o'zgarmas, dastur boshida beriladigan tuzilmalarni nazarda tutamiz. Dinamik ma'lumot tiplari dastur davomida o'z hajmini, egallagan hotirasini o'zgartirishi mumkin. Agar tuzilma bir hil kattalikdagi tiplardan tuzilgan bo'lsa, uning nomi massiv (array) deyiladi. Massivlar dasturlashda eng ko'p qo'laniladigan ma'lumot tiplaridir. Bundan tashqari tuzilmalar bir necha farqli tipdagi o'zgaruvchilardan tashkil topgan bo'lishi mumkin.

Tuzilma – bu ma`lumotlarni bir butun nomlangan elementlar to`plamiga birlashtirish. Tuzilma elementlari har xil tipda bo`lishi mumkin va ular har xil nomlarga ega bo`lishi kerak.

Tuzilmali tip quyidagicha aniqlanadi:

struct { <ta`riflar ro`yxati > }

Tuzilmada albatta bitta komponenta bo`lishi kerak. Tuzilma tipidagi o`zgaruvchi quyidagicha ta`riflanadi:

<tuzilma_nomi > <o`zgaruvchi>;

Tuzilma tipidagi o`zgaruvchi ta`riflanganda initsializatsiya qilinishi mumkin:

<tuzilma_nomi > <o`zgaruvchi>=<initsializator>;

Tuzilmani initsializatsiyalash uchun uning elementlar qiymatlarini figurali qavslarda tavsiflanadi.

Misollar:

-struct Student

{

char name[20];

int kurs;

float rating;

};

Student s={”Qurbonov”,1,3.5};

-struct

{

char name[20];

char title[30];

float rate;

}employee={“Ashurov", “direktor”,10000};

**Tuzilmalarni o`zlashtirish.** Bitta tuzilma tipdagi o`zgaruvchilar uchun o`zlashtirish operatsiyasi aniqlangan. Bunda har bir elementdan nusxa olinadi. Masalan:

Student ss=s;

Tuzilma elementlariga murojaat. Tuzilma elementlariga murojaat aniqlangan ismlar yordamida bajariladi:

<Tuzilma_nomi>.<element_nomi>

Primeri:

employee.name – «Ashurov» satriga ko`rsatkich;

employee.rate – 10000 qiymatga ega bo`lgan butun tipdagi o`zgaruvchi

**Tuzilmaga ko`rsatkichlar.** Tuzilmaga ko`rsatkichlar oddiy ko`rsatkichlar kabi tasvirlanadi:

Student*ps;

Tuzilmaga ko`rsatkich ta`riflanganda initsializatsiya kilinishi mumkin:

Student *ps=&mas[0];

Ko`rsatkich orkali tuzilma elementlariga ikki usulda murojaat kilish mumkin. Birinchi usul adres bo'yicha qiymat olish amaliga asoslangan bo`lib quyidagi shaklda qo`llaniladi:

(* tuzilmaga ko`rsatkich).element nomi;

Ikkinchi usul maxsus strelka (->) amaliga asoslangan bo`lib quyidagi ko`rinishga ega:

tuzilmaga ko`rsatkich->element nomi

Tuzilma elementlariga quyidagi murojaatlar o'zaro tengdir:

cin>>(*ps).name;

cin>>ps->title;

1.3.Dinamik massivlar.

C++ tilida o‘zgaruvchilar yo statik tarzda - kompilyatsiya paytida, yoki standart kutubxonadan funktsiyalarni chaqirib olish yo‘li bilan dinamik tarzda - dasturni bajarish paytida joylashtirilishi mumkin. Asosiy farq ushbu usullarni qo‘llashda ko‘rinadi - ularning samaradorligi va moslashuvchanligida. Statik joylashtirish samaraliroq, chunki bunda xotirani ajratish dastur bajarilishidan oldin sodir bo‘ladi. Biroq bu usulning moslashuvchanligi ancha past, chunki bunda biz joylashtirilayotgan ob'ektning turi va o‘lchamlarini avvaldan bilishimiz kerak bo‘ladi. Masalan, matniy faylning ichidagisini satrlarning statik massivida joylashtirish qiyin: avvaldan uning o‘lchamlarini bilish kerak bo‘ladi. Noma'lum sonli elementlarni oldindan saqlash va ishlov berish kerak bo‘lgan masalalar odatda xotiraning dinamik ajratilishini talab qiladi.

Xotirani dinamik va statik ajratish o‘rtasidagi asosiy farqlar quyidagicha:

-statik ob'ektlar nomlangan o‘zgaruvchilar bilan belgilanadi, hamda ushbu ob'ektlar o‘rtasidagi amallar to‘g‘ridan-to‘g‘ri, ularning nomlaridan foydalangan

holda, amalga oshiriladi. Dinamik ob’ektlar o‘z shaxsiy otlariga ega bo‘lmaydi, va ular ustidagi amallar bilvosita, ko‘rsatkichlar yordamida, amalga oshiriladi;

-statik ob’ektlar uchun xotirani ajratish va bo‘shatish kompilyator tomonidan avtomatik tarzda amalga oshiriladi. Dasturchi bu haqda o‘zi qayg‘urishi kerak emas. Statik ob’ektlar uchun xotirani ajratish va bo‘shatish to‘laligicha dasturchi zimmasiga yuklatiladi. Bu anchayin qiyin masala va uni echishda xatoga yo‘l qo‘yish oson.

Dinamik tarzda ajratilayotgan xotira ustida turli xatti-harakatlarni amalga oshirish uchun new va delete poeratorlari xizmat qiladi.

Shu paytga qadar barcha misollarda statik xotira ajratish qo‘llanadi. Masalan, i o‘zgaruvchisini aniqlash:

int i=1024;

Bu komanda xotirada shunday sohani ajratib beradiki, u int turidagi o‘zgaruvchini saqlash, ushbu soha bilan i nomini bog‘lash hamda u erga 1024 qiymatini joylashtirish uchun etarli bo‘ladi. Bularning hammasi dastur bajarilishidan oldin kompilyatsiya bosqichida amalga oshiriladi.

Biroq o‘zgaruvchiga xotirani ajratib berish uchun yana bir usul mavjud bo‘lib, u new operatorini qo‘llashdan iborat.

New operatori ikkita shaklga ega. Birinchi shakl ma’lum bir turdagi yakka ob’ektga xotirani ajratib beradi;

int*pint=new int(1024)

Bu erda new operatori int turidagi nomsiz ob’ektga xotirani ajratib beradi, uni 1024 qiymati bilan nomlantiradi (initsiallashtiradi) hamda yaratilgan ob’ekt adresini qaytarib beradi. Bu adres pint ko‘rsatkichiga joylashtiriladi. Ushbu nomsiz ob’ekt ustidagi barcha xatti-harakatlar shu ko‘rsatkich bilan ishlash orqali amalga oshiriladi, chunki dinamik ob’ekt bilan to‘g‘ridan-to‘g‘ri ish olib borish (manipulyatsiyalar o‘tkazish) mumkin emas.

New operatorining ikkinchi shakli ma’lum bir turdagi elementlardan tashkil topgan berilgan o‘lchamlardagi massivga xotira ajratib beradi:

int *pia=new int[4];

Bu misolda xotira int turidagi to‘rtta elementdan iborat massivga xotira ajratiladi. Afsuski, new operatorining bu shakli massiv elementlarini nomlantirish (initsiallashtirish) imkonini bermaydi.

New operatorining har ikkala shakli ham bir xil ko‘rsatkichni qaytarishi (keltirilgan misolda bu butun sonning ko‘rsatkichi) ayrim chalkashliklarga olib keladi. pint ham pia ham aynan bir xil e’lon qilingan Biroq pint operatori int turidagi bitta ob’ektni ko‘rsatadi, pia esa int turidagi to‘rtta ob’ektdan iborat massivning birinchi elementini ko‘rsatadi.

Dinamik ob’ekt kerak bo‘lmay qolganda, unga ajratilgan xotirani to‘g‘ridan-to‘g‘ri bo‘shatish kerak. Bu ish delete operatori yordamida amalga oshiriladi:

delete pint;

Ob’ektning bo‘shatilishi, new kabi, ikkita shaklga ega - yakka ob’ekt uchun va massiv uchun:

delete[] pia;

Agar ajratilgan xotirani bo‘shatish esdan chiqqudek bo‘lsa, bu xotira bekordan-bekorga sarflana boshlaydi, foydalanilmay qoladi, biroq, agar uning ko‘rsatkichi o‘z qiymatini o‘zgartirgan bo‘lsa, uni tizimga qaytarish mumkin emas. Bu hodisa xotiraning yo‘qotilishi degan maxsus nom bilan ataladi. Pirovard natijada dastur xotira etishmagani tufayli avariya holatida tugallanadi (agar u ancha vaqt ishlayversa).

Belgili axborot va satrlar.

C++ da belgili ma’lumotlar uchun char turi qabul qilingan. Beligili axborotni taqdim etishda belgilar, simvolli o‘zgaruvchilar va matniy konstantalar qabul qilingan.

Misollar:

const char c=’c’;//belgi - bir baytni egallaydi, uning qiymati o‘zgarmaydi

char a,b;//belgili o‘zgaruvchilar, bir baytdan joy egallaydi, qiymatlari o‘zgaradi.

const char *s= ‘‘\n satrining misoli’’;//matniy konstanta

C++ dagi satr - bu nul-belgi - ‘\0’ (nul-terminator)- bilan tugallanuvchi beliglar massivi. Nul-terminatorning holatiga qarab satrning amaldagi uzunligi aniqlanadi. Bunday massivdagi elementlar soni, satr tasviriga qaraganda, bittaga ko‘p.

Qiymat berish operatori yordamida satrga qiymat berish mumkin emas. Satrni massivga yoki kiritish paytida yoki nomlantirish yordamida joylashtirish mumkin.

Misol:

```
void main()
{
  char s1[10]=’’string1’’;
  int k=sizeof (s1);
  cout<<s1<<’’\t’’<<k<<endl;
  char s2[]=’’string2’’;
  k=sizeof(s2);
  cout<<s2<<’’\t’’<<k<<endl;
  char s3[]={‘s’,’t’,’r’,’i’,’n’,’g’,’3’};
  k=sizeof(s3);
  cout<<s3<<’’\t’’<<k<<endl;
  char *s4=’’string4’’;//satr ko‘rsatkichi, uni o‘zgartirib bo‘lmaydi
  k=sizeof(s4);
  cout<<s4<<’’\t’’<<k<<endl;
}
```

Natijalar:

string1 10 - 10 bayt ajratilgan, shu jumladan \0 ga

string2 8 - 8 bayt ajratilgan (7+1 bayt /0 ga)

string3 8 - 8 bayt ajratilgan (7+1 bayt /0 ga)

string4 4 - ko‘rsatkichning o‘lchamlari

1.4.Bog’lamli ro’yhatlar.

Eng sodda dinamik informatsion tuzilma elementlari qo’yidagi tuzilmali tip orqali ta'riflangan ob'ektlardan iborat ro’yhatdir.

```
Struct tuzilmali tip nomi
{
tuzilma elementlari ;
Struct tuzilmali tip nomi*ko’rsatkich;
};
```

Qo’yidagi misolni ko’rib chiqamiz, Klaviatura orqali ihtiyoriy sondagi tuzilmalarni bir bog’lamli ro’yhatga birlashgan holda kiritish, so’ngra ro’yhat elementlarini kiritilgan tartibda ekranga chiqarish. Ro’yhat bilan ishlash uchun uchta ko’rsatkichdan foydalaniladi: beg ro’yhat boshiga ko’rsatkich, end ro’yhat ohiriga ko’rsatkich, rex ro’yhatni boshidan qarab chiqish uchun ishlatiladigan ko’rsatkich.

Qo’yidagi dastur qo’yilgan vazifani bajaradi:

```
#Include <stdlib>
#include <stdio.h>
struct cell {
char sign[10];
int weight;
struct cell * pc;
};
void main()
{
struct cell * rex;
 struct cell * beg=NULL;
struct cell * end=NULL;
do
{
rex=(struct cell*malloc(sizeof(struct cell));
printf(“sign=”);
scanf(“%s, & rex->sign);
printf(“weight=”);
```

```
scanf(“%d”,&rex->weight);
if (rex->weight==0)
{
free(rex);
break;
}
if (beg==NULL&&end==NULL)
beg=rex;
else
end->pc=rex;
end=rex;
end->pc=NULL;
}
while(1);
printf(“\nRo’yhatni chiqarish:”);
rex=beg;
while(rex!=NULL);
{
printf(“\nsign=%s\tweight=%d”,rex->sign,rex->weight);
rex=rex->pc;
}
}
```

Dastur bajarilishiga misol:

Tuzilma haqidagi ma'lumotni kiriting:

Sign=sigma

Weight=16

Sign=omega

Weight=44

Sign=alfa

Weight=0

Ro’yhatni chiqarish

Sign=sigma weight=16

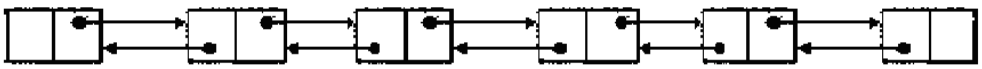
Sign=omega weight=44

Dasturda ma'lumotlarni kiritish sikl orqali bajariladi.Sikl tugatilishi sharti navbatdagi tuzilmaning int weight elementiga kiritilgan nol qiymatdir.

Tuzilmalar ro’yhati dinamik tashkil etiladi. Beg va end ko’rsatkichlar nol qiymati orqali initsializatsiya qilingan. Dasturda (struct cell*) tip o’zgartirishidan foydalanilgan chunki malloc funktsiyasi har doim void tipidagi ko’rsatkich qaytaradi.

Ikki bog’lamli ro’yhat.

Ro’yhat bo’g’inlar ketma-ketligidan tashkil topgan bo’lib chapdan o’nga tomon hosil qilinadi, har bir bo’g’in ikki qismdan iborat bo’ladi. Birinchi qismda ma’lumot, ikkinchi qismda keyingi bo’g’inga ko’rsatgich bo’ladi. Ohirgi bo’g’inning ikkinchi qismi nolga teng bo’lsa ro’yhat tugaydi.



1.4.1-chizma.Oddiy ikki bog’lamli ro’yhat ko’rinishi.

Bir qancha ilovalarda dasturchi uchun ro’yhatga kirish teskari tartibda bo’ladi. Masalan, bezbol o’yinchilarining ro’yhatini ma’lum bir tartibda olib borib uning o’rtacha qiymatini eng quyi va yuqori qaytargan miqdorlaridan olib boradi. O’yinchilarning o’yinini baholash uchun va eng yaxshi o’yinchini ajratishda ro’yhatni teskarisidan ko’rib chiqadi. Ikki bog’lamli ro’yhat ixtiyoriy bo’g’inga murojaat qilish uchun qulay hisoblanadi. Ikki bog’lamli ro’yhatlarda ro’yhatga qayta ishlov berish oson bajariladi.

Ayrim hollarda ro’yhatni teskaridan chiqarish zarur bo’ladi. Bunda ikki bog’lamli ro’hatdan foydalaniladi. Quyidagi dastur klaviatura orqali kiritilgan sonlarni teskari tartibda chiqaradi.

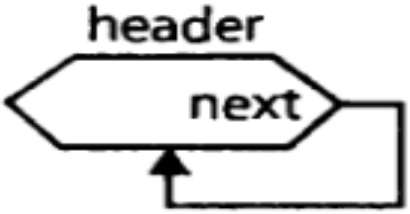
```
#include <vcl.h>
#pragma hdrstop
```

```
#include <stdio>
#include <stdlib.h>
#include <iostream.h>
//---------------------------------------------------------------------------
#pragma argsused
struct mrec {
int value;
mrec* next;
};
void main() {
struct mrec* rex;
struct mrec* cur;
struct mrec* beg=0;
struct mrec* end=0;
int i;
cin >> i;
beg = (mrec*)malloc(sizeof(struct mrec));
beg->value = i;
beg->next = 0;
rex =beg;
while (i>=0){
 cin >> i;
 cur = (mrec*)malloc(sizeof(struct mrec));
 cur->value = i;
 cur->next = rex;
 beg=cur;
 rex = beg;  }
 rex =beg;
 cout << "kiritilgan sonlar teskarisidan \n";
 while (rex != 0){
```

```
cout << rex->value <<"\n";
cur=rex;
rex=rex->next;
free(cur);
}
cin >> i;
}
```

Takrorlanuvchi ro’yhat.

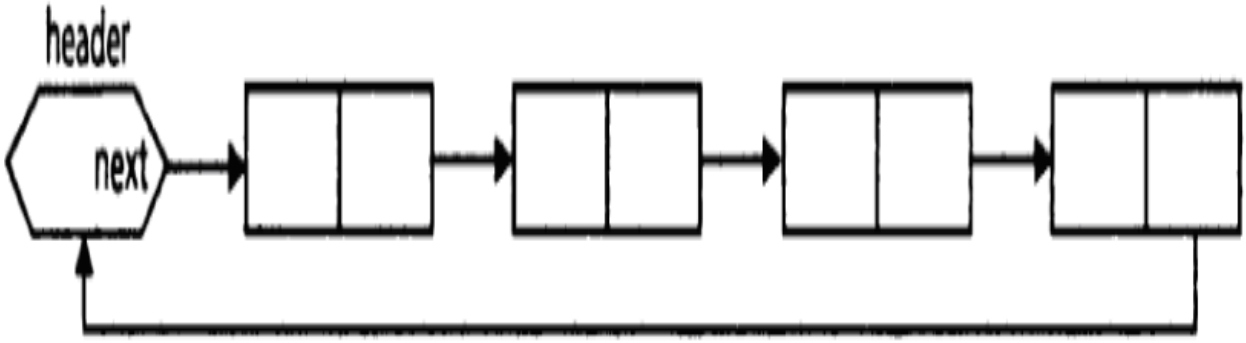
Bosh bo’g’undan boshlanib ohirgi bo’g’inini ikkinchi qismi nol bilan tugab ko’rsatgich bosh bo’g’inga yuborilgan bo’ladi. Ko’p dasturchilar o’zaro bog’langan ro’yxatlarni amalga oshirish uchun davriy modelni qo’llaydilar. Bo’sh davriy ro’yhat o’z tarkibida initsiyalizatsiya qilinmaydigan maydon qiymatlari bo’g’inidan tashkil topgan.



.4.1-chizma.Bo’sh takrorlanuvchi ro’yhat.

Bosh bo’g’in header so’zidan boshlanadi. Header ro’yhatdagi birinchi aniq bo’g’inni belgilaydi. Davriy model ro’yhat bilan bog’liq, bo’sh ro’yhat bitta bo’g’inga ega. Biz bo’g’inning bosh sixemasini burchak chizig’ini bo’g’inning tamoni sifatida keltiramiz. Bo’g’inlar ketma-ket bog’lanib borib oxiri bosh bo’g’inga keltrib ko’rsatiladi. Davriy ro’yhat juda optemal va nusxalashni hohlagan holatidan boshlashga imkon beradi. Hamda uning boshlang’ich pozitsiyasigacha davom etadi. Ro’yhat chegaralangan chunki u dasturgacha

o’tilgan yo’lni va nusxalashni teskari yo’nalishda imkon bermaydi.



1.4.2-chizma.Bo’sh takrorlanuvchi ro’yhat.

Takrorlanuvchi ro’yhatning afzallik tomoni uning ixtiyoriy bo’g’inidan boshlab chiqarishning mumkinligi. Masalan, ro’yhatning beshinchi elementidan chiqarish kerak bo’lsa shu elementdan boshlanib yana shu elementgacha chiqaradi.

Quyidagi misol kiritilgan elementlarni ixtiyoriy bo’g’inidan boshlab chiqaradi.

```
#include <vcl.h>
#pragma hdrstop
#include <stdio>
#include <stdlib.h>
#include <iostream.h>
//---------------------------------------------------------------------------
#pragma argsused
struct mrec {
int value;
mrec* next;
};
void main() {
struct mrec* rex;
struct mrec* cur;
struct mrec* beg=0;
struct mrec* end=0;
int i;
```

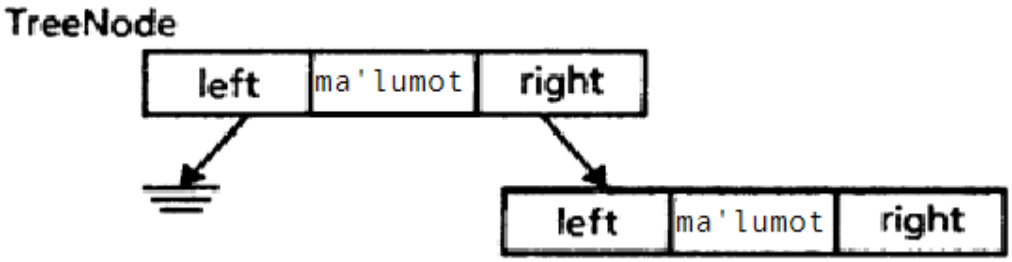
```
cin >> i;
beg = (mrec*)malloc(sizeof(struct mrec));
beg->value = i;
beg->next = 0;
rex =beg;
while (i>=0){
 cin >> i;
 cur = (mrec*)malloc(sizeof(struct mrec));
 cur->value = i;
 cur->next = rex;
 beg=cur;
 rex = beg;  }
 rex =beg;
 cout << "kiritilgan sonlar teskarisidan \n";
 while (rex != 0){
 cout << rex->value <<"\n";
 cur=rex;
 rex=rex->next;
 free(cur);
 }
 cin >> i;
 }
```

1.5.Daraxtsimon tuzilma.

Binar daraxt tuzilmasi bo’g’inlardan tashkil topadi.Bu tuzilma bo’g’inida malumotlar,boshqa bo’g’inlarga ko’rsatkichlar va boshqa bo’g’inlar yig’indisidan iborat.Bu taqsimot daraxt bo’g’inlarini aniqlash va bo’g’indan-bo’g’inga o’tishni ta’minlaydi.

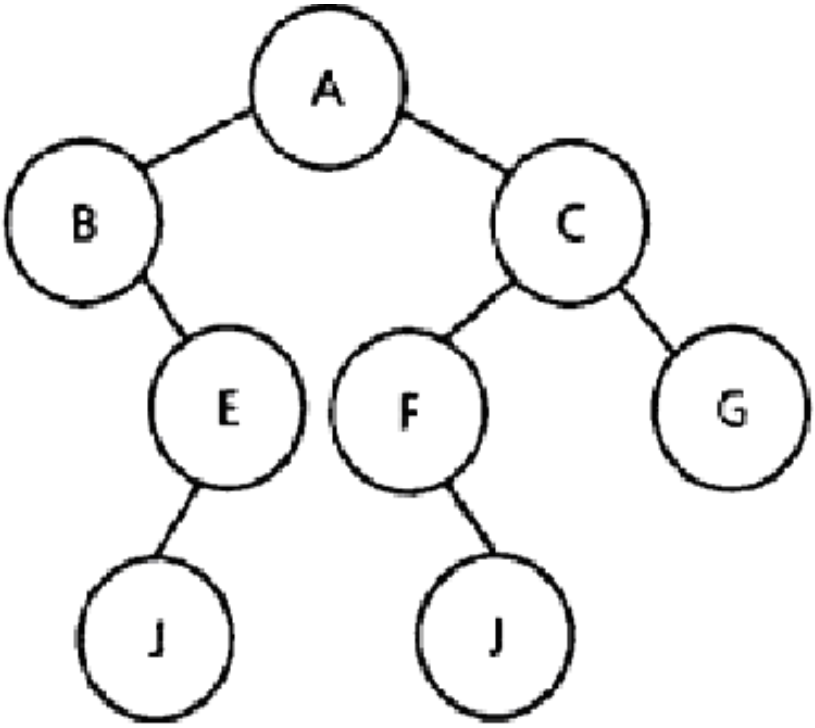
Daraxt tuzilmasining har bir bo’g’ini uch qismdan tashkil topgan bo’lib, ularning bittasida ma’lumot saqlanib, qolgan ikkitasi boshqa bo’g’inlarga ko’rsatkich bo’ladi.Bu qismlardagi ko’rsatkichlar chap ko’rsatkich(left) va o’ng

ko’rsatkich(right) deb ataladi, chunki daraxt qismlariga chap va o’ng tomondan ko’rsatkich bo’ladi.



1.5.1-chizma. Chap tomoni bo’sh bo’lgan daraxtsimon tuzilma.

Bosh bo’g’in daraxtning kirish nuqtasini aniqlaydi, ko’rsatkich bo’g’ini esa keyingi bo’g’inga o’tadi.

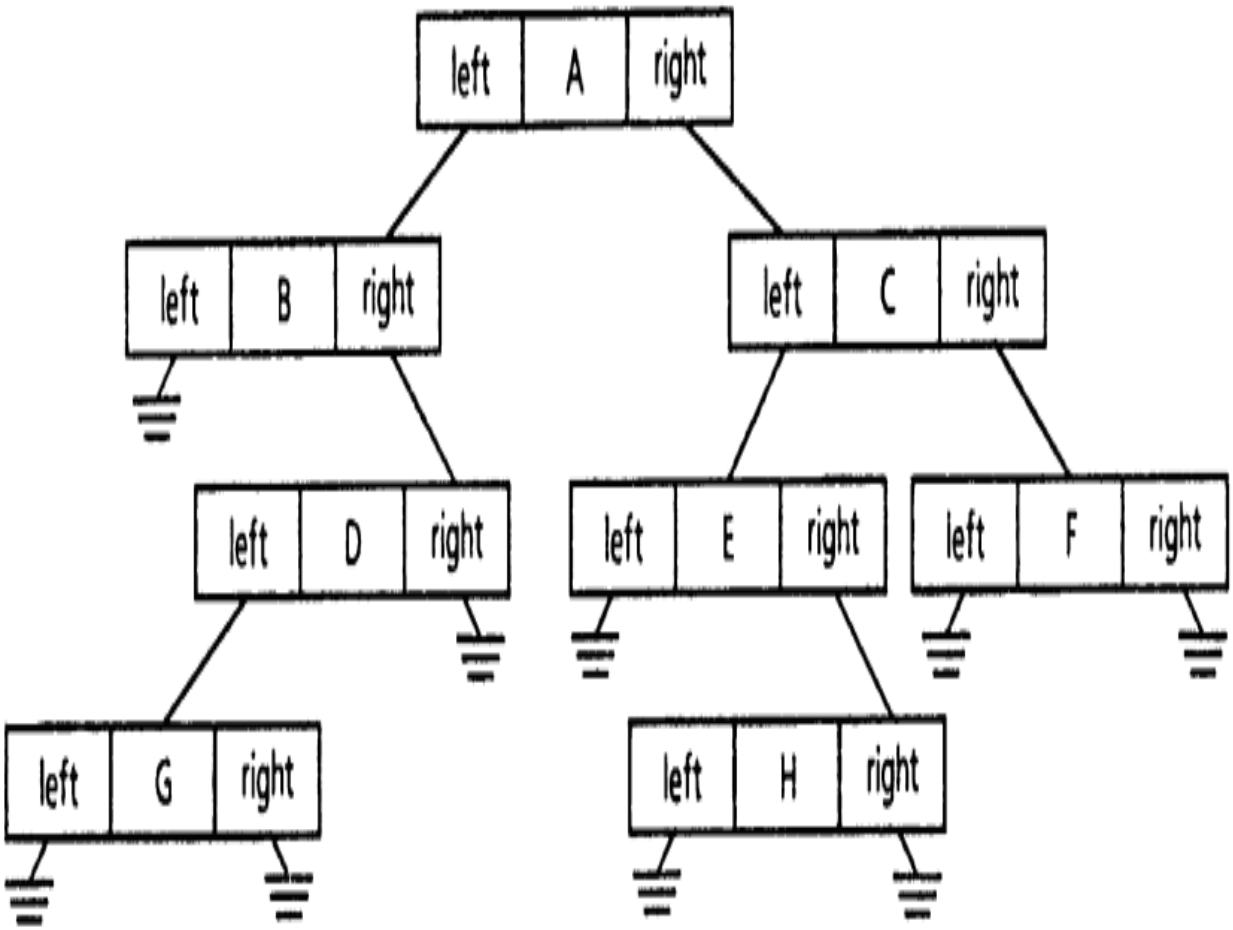


1.5.2-chizma. Daraxt.

TreeNode sinfini yaratish.

Bu qismda biz TreeNode sinfini yaratish, bu sinfga daraxt bo’g’ini obyektlarini joylashtirishni ko’rib o’tamiz.Bo’g’in tuzilishi malumotlar maydoni, ochiq elementlardan (public) iborat bo’lganligi uchun, foydalanuvchi bevosita barcha elementlarga murojaat qilib biladi. Bu foydalanish mijozga ma’lumotlarni o’qish yoki o’gartirish hattoki ma’lumotlarga ko’rsatgich orqali qaytishga ruxsat beradi.Oxirida foydalanuvchi juda murakkab ma’lumotlar tuzilmasiga ega bo’ladi. Ikkita ko’rsatkich yopiq (private) elementlardan iborat, bu elementlarga murojaat

left() va right() funksiyalari orqali amalga oshiriladi. TreeNode sinfini aniqlash va o’zgartirishni **treenode.h** fayli bajaradi.



TreeNode tuzilmasi.
1.5.3-chizma. Daraxtsimon tuzilma.

Birinchi bobning qisqacha xulosasi.

Dinamik tuzilmalar haqida umumiy ma’lumot. Bu bobda ob’yekt haqida ma’lumot berilgan. Ob’ektlarni yaratish va ular ustida manipulyatsiyalar olib borish - C++tilining qandaydir alohida imtiyozi emas, balki ob’ektlarning tavsifi va ular ustida o‘tkaziladigan operatsiyalarni kodli konstruktsiyalarda o‘zida mujassamlantiradigan dasturlash metodologiyasi (uslubiyoti) ning natijasi ekanligi ko’rsatilgan. Bundan tashqari statik va dinamik tuzilmalar haqida ma’lumot berilgan. Bir-biridan qanday farq qilish aytib o’tilgan.

Bu bobda oddiy dinamik tuzilmalar haqida ma’lumot berilgan. Ya’ni dinamik massiv, bir bog’lamli ro’yhat, ikki bog’lamli ro’yhat, takrorlanuvchi ro’yhat va daraxtsimon tuzilmalar tuzish ro’yhatlarga element qo’shish, o’zgartirish, axtarish,

yo’qotish kabi ishlar qilingan. Daraxtsimon tuzilma yaratishda treenode tuzilmasimasini tuzish va qanday qismlardan tuzilgani aytib o’tilgan.

II Bob. Xotirani boshqarish.

2.1.Ob’yektlarni boshqarish rejimlari.

To`g’risini aytganda, xotirani esdan chiqarsak yaxshi bo`lar edi. Ehtiyojlarga qarab ob`ektlarni dasturlar yaratardi. Ishlatilmaydigan ob`ektlar mavhumlikka g’oyib bo`lardi, keraklilari esa sekin yuqoriga ko`tarilardi. Bu jarayon katta tashkilotning xodimlari foydalaniladigan zinapoyaga o`xshaydi, eng oxiriga (erishilgan mavqelar) boshqaruv rahbariyat tizimi. Lekin bunday emas. Xotira cheksiz emas va bir qator uzluksiz qatlamlar tezlikni kamayib borishni tashkil qila olmaydi. Shuning uchun biz ishga layoqatsiz ishchilarni ishdan bo`shatishimiz kerak, agar biz buni barvaqt nafaqada chiqish deb atasak bo`ladi, umumiy iqtisodiy holat uchun yozdirilgan. Bu ma`ruza haqiqattan ham kimning ishdan ketishi, kim tomonidan va qandayligini o`rganadi.

Ob`ektamlar bilan nima sodir bo`ladi. Ob`ektga mo`ljallangan dasturlar ob`ektlarni yaratadi. Yuqoridagi ma`ruza shuni ko`rsatadiki, tizimni ehtiyojiga ko`ra, gibkiy ob`ektlar strukturasi uchun dinamik yaratish foydali.

Biz yangi ob`ektlarni joylashtirish bazali jarayon operatsiyalarini ko`rib chiqdik. Oddiy joylashtirish usuli quyidagicha yoziladi. create x va uning effektivligi uchlikdayda aniqlangan: yangi ob`ekt yaratish; x ni ko`rsatkichiday uni bog’lash; va uning maydonlarining boshlang’ich qiymatini berish.Bu variant instruktsiyasi boshlang’ich qiymatini berish protsedurasini chaqiradi; clone va deep_clone qism dastur yordamida yangi ob`ektlarni yaratish mumkin. Shu sababli bu joylashtirish shakllarining hammasi bazali instruktsiyaga asoslanib yaratadi, hech qanday yo`qotishsiz **create** x ko`rib chiqish chegaralash mumkin.

Yaratilgan xotirani boshqarishni instruktsiyalari effektlarini ko`rib chiqamiz.

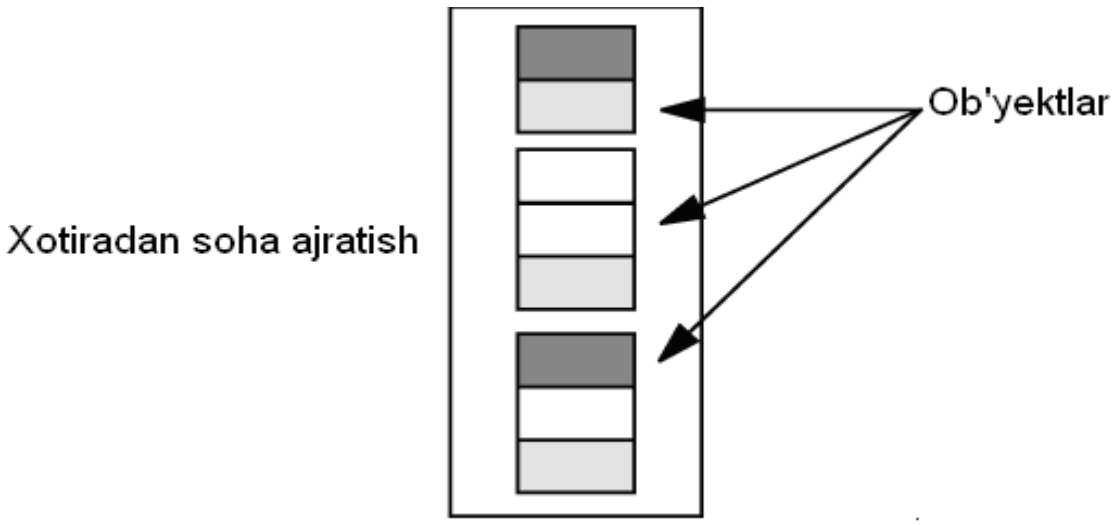
Ob`yektlarni boshqarishni uchta rejimi mavjud.

Birinchidan munozara chegaralarini kengaytirish foydali bo`ladi. Ob`ektga mo`ljallangan hisoblash uchun ob`ektlarni boshqarish shakli mo`ljallangan, uchta uchraydigan oddiy rejimdan birini qo`llab – quvvatlash mumkin: **statik**, **stekli** va

dinamik taqsimlash. Tanlash rejimi - Ob`ektni mavjudlikka qo`shilishini aniqlaydi.

Eslatamiz, mavjudlik bu – matnli dastur ismi bo`lib, ayrim qiymatlarni yoki bajarish vaqtidagi hamma qiymatlarni ifodalaydi. Bunaday qiymatlar ob`ekt yoki (aniqlanmagan qiymatli bo`lishi mumkin) ob`ektga ko`rsatkich bo`ladi. Atributlarning bo`lishi mavjud, dastur osti argumentlarning formalligi, lokal o`zgaruvchilari va Result. Ob`ektlar va mavjudlikni orasidagi bog'lanishni tavsiflash atamasi: Dasturning aniq bir bosqichda bajarilishi x ning mavjudligi O ob`ektiga qo`shiladi, agar x ning qiymati O (x uchun развернутого tipi), yoki O ga ko`rsatkich (x uchun ko`rsatkichli tipi) bo`lsa. Ko`pincha aytiladiki, agar x O ga birlashtirgan bo`lsa, O x ga birlashtirilgan deb aytiladi. Ko`rsatkichni bitta ob`ektdan ortiq bog'lash mumkin emas, ob`ektni esa ikkita va undan ortiq ko`rsatkichlarga bog'lash mumkin.Oldingi ma`ruzada dinamik muammolarni ko`rilgan edi.

Dastur bajarilish jarayonida, statik rejimda mavjudlik maksimum bitta ob`ektga bog'lanishi mumkin. Bu sxema, Fortran tili kabilarda barcha ob`ektlar uchun joy rezervlanadi va shu ob`ektlarni nomlariga bog'lanadi va doimiy dasturni yuklash yoki uni bajarilishi boshida qo`llaniladi.



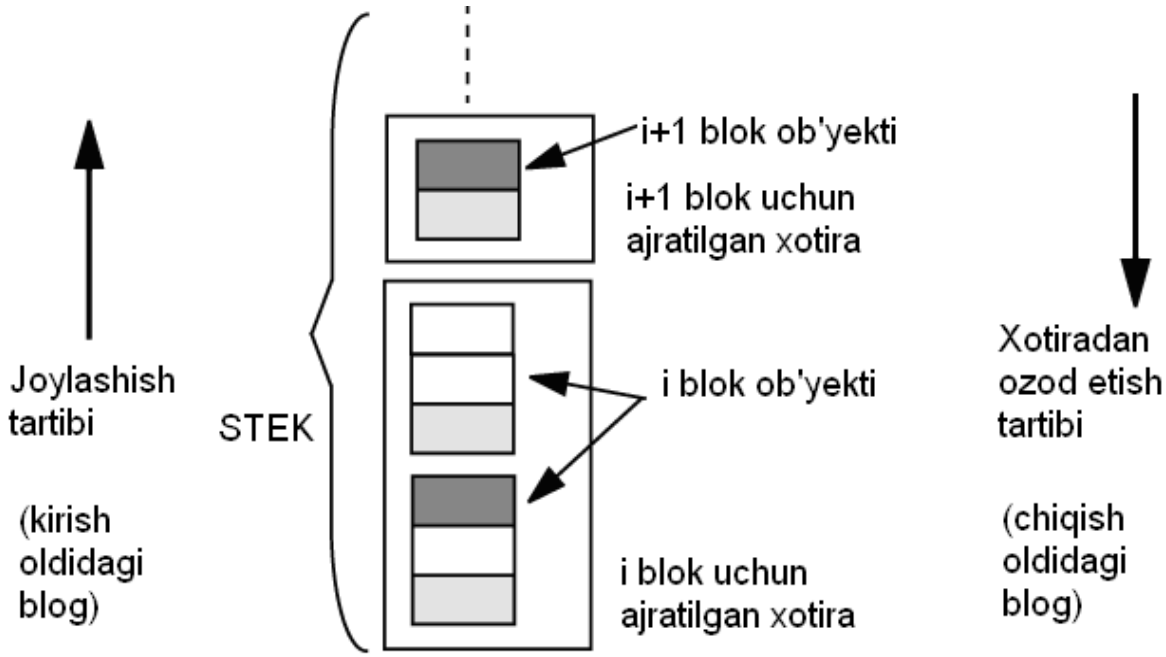
2.1.1-chizma.Statik rejimi.

Statik rejimda sodda va effektiv komp’yuterni arxitekturasini realizitsiya qilamiz. Lekin u jiddiy cheklanishlarga ega:

-Rekursiyadan foydalanib bo`lmaydi. Rekursiyali dasturga bir vaqtni o`zida bir nechta faol nusxalariga ega bo`lishi zarur, har bir ekzemplyari birta mavjudlikdan iborat.

-Ma`lumotlar strukturasini dinamik yaratishga g’ov bo`ladi. Dasturning matnidan kompilyator har bir ma`lumot strukturasining aniq o`lchamini aniqlay bilishi kerak. Masalan, bu holatga, har qanday massiv statik va aniq o`lchamini e`lon qilinishi kerak. Bu tilning quvvatini jiddiy chegaralaydi: strukturani tuzilishini, dasturning bajarilishi hodisasi paytida o`zgartirib bo`lmaydi (strukturani o`sishini). Har bir struktura uchun maksimal imkoniyatdan iborat xotirani rezervlash to`g’ri keladi – bu nafaqat samarasiz, balki xavfli hamdir. Agar bironta struktura o`lchamini e`tiborga olmasak, sistemani bajarilish davrida xatolikni keltirib chiqaradi.

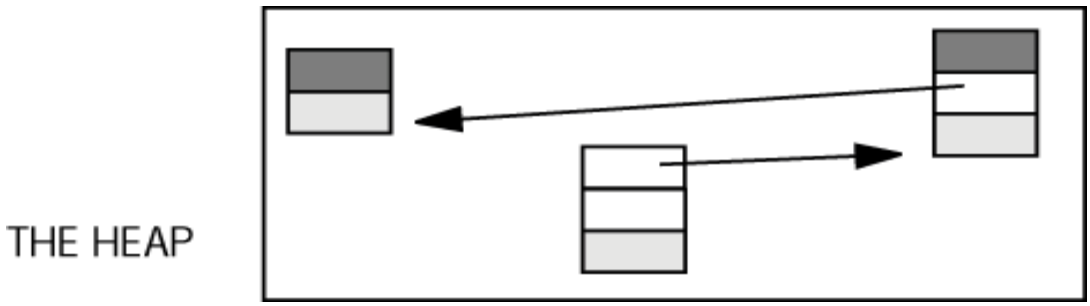
Ob`ektlarni joylashtirishni ikkinchi rejimi - stek rejimi. Bu erda haqiqiy bir vaqtda mavjudlik bir nechta ob`ektga ketma-ket bog’lanishi mumkin. Bajarilish mexanizmi bu ob`ektlarni quyidagi “oxirgi kelgan, birinchi ketada” tartibida joylashtiradi va o`chiradi. Qachon ob`ekt o`chirilsa, unga ta`luqli mavjudlik oldingi bog’langan ob`ektga qaytadan bog’lanadi, yangi ob`ekt paydo bo`lgunicha, agar, bunaqa ob`ekt mavjud bo`lsa.



3.1.2-chizma.Stekka asoslangan tartib.

Stekka asoslangan ob`ektlarni boshqaruvi Algol 60 ni taniqli qildi va hozirgacha ko`pgina dasturlash tillarini qo`llab-quvvatlamoqda (ko`pgina boshqa ikki rejimda birgalikda bo`ladi). Bunday usul rekursiya va dinamik massivlarni qo`llab - quvvatlaydi, qaysiki jarayonni bajarilish paytida massiv chegaralari aniqlanadi. **Pascal** va **C** da bu mexanizm massivlarda qo`llanilmaydi, ya'ni **Algol** ga bajariladiganidek. Ammo ishlab chiqadiganlar xohlaydilarki massivlarni shu usul bilan taqimlashni. Shunday qilib, agar bu mexanizmni massivga qo`llashlari mumkin bo`lsa ham, ko`pgina murakkab strukturali ma`lumotlarni stekga joylashtirib bo`lmaydi.

Murakkab strukturali ma`lumotlar uchun bizlarga uchinchi va oxirga rejim kerak bo`ladi: dinamik xotira "kucha" deb nomlanadi va shu usuldan foydalaniladi. Xotirada so`rov asosida ob`ektlar dinamik yaratiladi. Mohiyati shundan iboratki, turli xil ob`ektlarni dinamik bog'lash mumkin. Qanday ob`ektlar yaratish va ularni bog'lashni kompilyatsiya vaqtiga aytib bo`lmaydi. Bundan tashqari, ob`ektlarni o`zi boshqa ob`ektlarga murojaatni saqlashi ham mumkin.



2.1.3-chizma.Dinamik rejim.

Dasturiy ta`minot, modellashning metodlarini to`liq kuchini talab qiladi oldingi bo`limga ko`rganimizdek. Murakkab dinamik strukturali ma`lumotlarni yaratishda dinamik xotira imkon beradi.

2.2.Dinamik rejimdan foydalanish.

Shubhasiz ob`ektga mo`ljallangan dasturlash, dinamik rejimni asosiy umumiy qismidir. Uni ko`pgina ob`ektga mo`ljallanmagan dasturlash tillarida ham foydalaniladi. Xususan:

-C tili Pascal ga o`xshash bo`lib massiv bo`lmagan statik o`zgaruvchilar va dinamik massivlar qo`shimcha kiritilgan. S tili ko`rsatkich va massiv tipli o`zgaruvchilarni, bibliotekadagi `malloc` funktsiyasidan foydalanib, dinamik joylashtiradi.

-PL/I barcha modullarni qo`llab - quvvatlaydi.

-Lisp tizimi yuqori dinamikli bo`lib, va u dinamik rejimli xotirani taqsimlash uchun yaratilgan. Lisp ni asosiy buyruqlaridan biri, ro`yxatni ko`p martali tasvirlash uchun foydalaniladi, - CONS, orqali ikki maydonli struktura yaratadi. Birinchi maydonida qiymatlar, ikkinchi maydoniga – keyingi elementiga ko`rsatkich saqlanadi. Bu erda `CONS`, ularni yaratish instruktsiyasidan ko`ra, ob`ektlar manbasi hisoblanadi.

Uch rejimda xotirani qayta ishlatish.

Stek rejimida, hamda dinamik rejimda, yaratilgan ob`ektlar uchun va uni ishlatilmaydigan ob`ektlarini nima qilishimiz kerak degan savol tug’iladi. Shunaqa ob`ektlarni xotiraga saqlagandan keyin, u xotiradan keyinchalik qayta foydalanish mumkin bo`larmikan?

Statik rejimlarga va statik modellarga muammo tug’ilmaydi: Har bir ob`ekt uchun doimo bitta xotiraga bog’langan mavjudlik bor. Dasturning bajarilish paytida ob`ekt bilan mavjudlik o`rtasidagi bog’liqlik doim faol bo`ladi. Shuning uchun xotradan qayta ishlashdan foydalanish mumkin emas. Biroq xotira etmaydigan paytda bu texnologiyadan foydalansak bo`ladi. Agar ikkita ob`ektning qiymatlari dasturni ketma - ket bajarilish paytida to tugagunicha bir xil bo`lsa, u holda bu ob`ektlar uchun ikkta xotira kerak bo`lmaydi. Bu texnika, **qoplash (overlay)** nomi bilan taniqli bo`lib, har bir amalda bajarilish paytida qo`l bilan birma bir boshqarilishi yomon.

Agar qoplashdan (overlay) foydalansak, u holda hammasini avtomatik bajariladi, maxsus uskunalardan foydalanish xatolik bo`lmaslikka olib keladi. Asosiy muammo tuzatish imkoni qoladi: ikkita o`zgaruvchini qoplash, dasturni bajarilish hayotida konkret echimi bo`lishi mumkin Noxosdan uning qiymatini o`zgarishi to`g’ri bo`lmagan echimga olib kelishi mumkin. Biz bunga o`xshash muammo bilan pastki qismimizda uchrashdik, Axlatlarni yig’ish texnologiyasida. Stek asoslangan rejimida, ob`ektlar mavjudlik bilan bog’lanadi, va ular stekka joylashishi mumkin. Strukturali blokli dasturlash tillarida vaziyat engilashadi: Blokli qismda bir vaqtni o`zida barcha ob`ektlarni joylashtirish, dasturni barcha sohasi uchun, birta stekdan foydalanishda ruxsat beradi. Sxema haqiqatdan ham elegant, chunki ikkita hamroh bo`lgan to`plamlar hodisasidan foydalanilayapti:

2.1 jadval. Strukturali, blokli dasturlash tillarida Ob`ektlarni joylashtirish va o`chirish

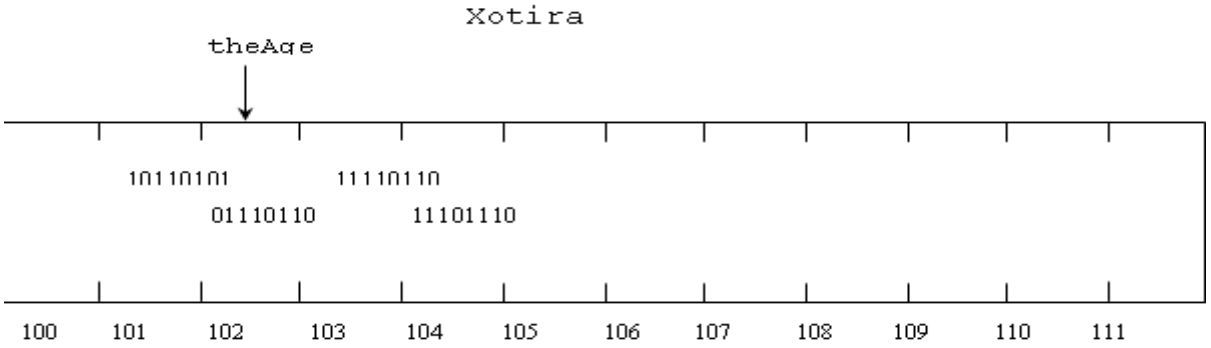
Dinamik xususiyat (bajarilish vaqtidagi hodisa)	**Statik xususiyat (dastur matni dagi holatida)**	**Realizatsiyalash texnikasi**
Ob`ektni joylashishi	Blok boshi	Stekda ob`ektlarni kiritmoq (mahalliy mavjudlik bloki uchun bitta)

Ob`ektni o`chirilishi	Blok oxiri	Stekdan ob`ektlarni chiqarmoq

(2.2.1-jadval)

2.3.Ma’lumotlarni xotirada joylashishi.

Mashina xotirasi nomerlangan yacheykalar ketma-ketligidan iboratdir. Har bir o`zgaruvchining qiymati uning adresi deb ataluvchi alohida xotira yacheykasida saqlanadi. Quyidagi chizmada theAge nomidagi, xotiradan to`rt bayt joy egallaydigan butun qiymatli o`zgaruvchining xotirada ifodalanishi ko`rsatilgan.



2.2.1-chizma.TheAge o`zgaruvchining xotirada saqlanishi

Har bir yacheyka = 1 bayt.unsigned long int theAge = 4 bayt= 32 bit. theAge o’zgaruvchilar birinchi baytni ko’rsatadi. theAge o’zgaruvchilarning adresi 102.

Turli kompyuterlarda xotirani adreslash turlicha qoida asosida tashkil etiladi. Ko`p hollarda dasturchilar uchun biror bir o`zgaruvchini aniq adresini bilish zarur emas. Zarurat tug’ilganda bunday axborotni adres operatori (&) yordamida olish mumkin. Bu operatorning qo`llanilishiga misol keltirilgan.

Misol: Adres operatori .

```
#inslude < iostheam.h >
int main()
{
unsigned short shortVar = 5;
unsigned long longVar = 65535;
long sVar = - 65535;
cout << “shortVar :\t” << shortVar ;
cout << “Address of shortVar:\t” ;
cout<<&shortVar<< “\n”;
```

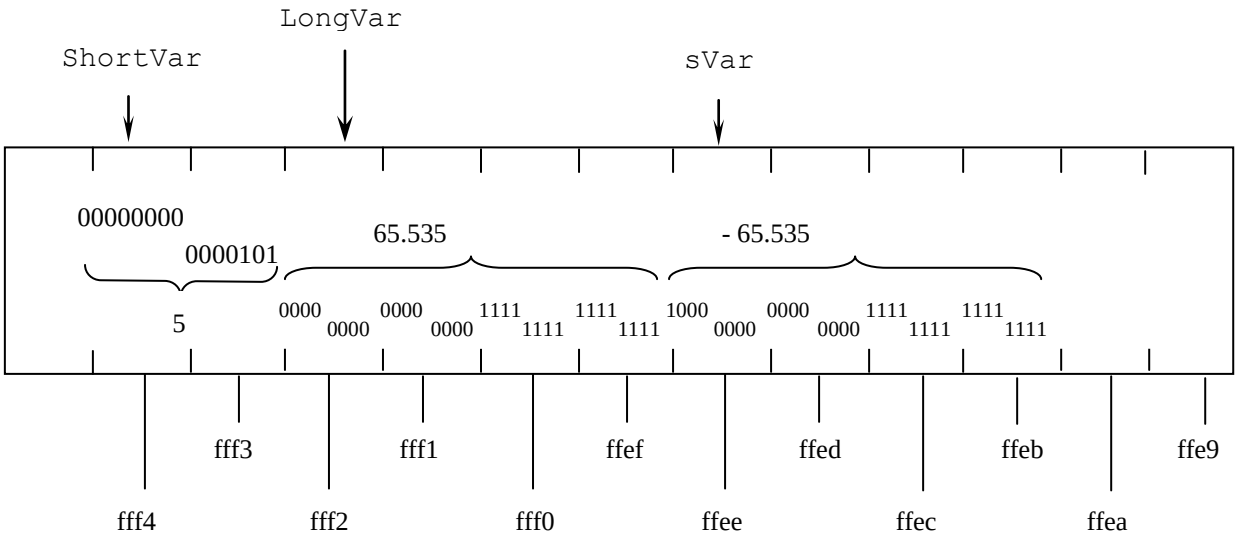
```
cout << “longVar :\ t” << longVar ;
cout << “ Address of LongVar: \ t” ;
cout<< &longVar<< “\n”;
cout << “ sVar :\t” << sVar;
cout << “Address of sVar:\t”;
cout << &sVar <<“\n” ;
return  0;
}
```

NATIJA:

shortVar : 5 Address of shortVar : 0x8fc9:fff4

longVar:65535 Address of longVar: 0x8fc9:fff2

sVar: -65535 Address of sVar : 0x8fc9:ffee.



2.2.2-chizma.O`zgaruvchining xotirada saqlanish sxemasi

Ko`rsatkichning adreslarni saqlash vositasi sifatida qo`llanilishi.

Dasturning har bir o`zgaruvchisi o`zining adresiga egadir. Bu adresni saqlash uchun esa o`zgaruvchiga ko`rsatkich e`lon qilish kerak. Adresning o`zining qiymatini bilish esa unchalik shart emas.

Masalan, howOld o`zgaruvchisi int tipiga ega. Bu o`zgaruvchini adresini saqlovchi pAge ko`rsatkichini e`lon qilish uchun quyidagi dastur fragmentini yozish lozim.

int *pAge = 0;

Bu satrda pAge o`zgaruvchisi int tipiga ko`rsatkich sifatida e`lon qilinayapti. pAge o`zgaruvchisi boshqa o`zgaruvchilardan umuman farq qilmaydi. Butun tipli o`zgaruvchini e`lon qilishda biz unda butun son saqlanishini ko`rsatamiz. Qachonki o`zgaruvchi biror bir tipga ko`rsatkich deb e`lon qilinsa, bu ko`rsatkich o`sha tipdagi o`zgaruvchining adresini o`zida saqlaydi. Demak, ko`rsatkichlar ham o`zgaruvchilarning alohida turi ekan.

Bu misolda pAge o`zgaruvchisiga nol qiymat o`zlashtirilayapti. Qiymati nolga teng bo`lgan ko`rsatkichlar bo`sh ko`rsatkichlar deyiladi. Ko`rsatkich e`lon qilinganda unga albatta biror bir qiymat o`zlashtirilishi lozim. Agarda bu oldindan aniq bo`lmasa unga 0 qiymat berish zarur. Qiymat o`zlashtirilmagan ko`rsatkichlar keyinchalik ko`plab noxush hodisalarga sabab bo`lishi mumkin. Biz ular bilan keyinroq batafsil tanishamiz.

pAge ko`rsatkichi e`lon qilinganda unga 0 qiymat berilib, keyinchalik esa unga boshqa qiymat, masalan howOld o`zgaruvchisi adresini o`zlashtirish mumkin. Quyida buni amalga oshirilishi ko`rsatilgan.

unsigned short int howOld=50; //o`zgaruvchini e`lon qilamiz.

unsigned short int *pAge=0; //ko`rsatkichni e`lon qilamiz.

pAge= &howOld; //pAge ko`rsatkichga howOld o`zgaruvchi adresini o`zlashtiramiz.

Birinchi satrda unsigned short int tipidagi o`zgaruvchi e`lon qilingan va unga 50 qiymat o`zlashtirilgan. Ikkinchi satrda esa unsigned short int tipidagi pAge tipiga ko`rsatkich e`lon qilingan va unga 0 qiymat o`zlashtirilgan. Tip nomi va o`zgaruvchi orasidagi yulduzcha (*) belgisi undan keyin kelgan o`zgaruvchini ko`rsatkich ekanligini anglatadi.

Oxirgi satrda pAge ko`rsatkichiga howOld o`zgaruvchisi adresi o`zlashtirilayapti. Bunda adres operatori (&) howOld o`zgaruvchisini adresini olayapti va u o`zlashtirish amali orqali pAge ko`rsatkichiga o`zlashtirilayapti.

Bu holatda pAge o`zgaruvchisining qiymati howOld o`zgaruvchining adresiga tengdir. Qarab chiqilgan dastur kodidagi oxirgi ikki satrni bitta satrga birlashtirish mumkin.

```
unsigned short int = 50;
unsigned short int * pAge = & howOld;
```

Bilvosita murojaat operatori.

Bilvosita murojaat operatori ko`rsatkichda adresi saqlanayotgan qiymatni ko`rsatkich nomidan foydalanib olish imkonini beradi.

Ko`rsatkichlardan farqli ravishda oddiy o`zgaruvchilarga murojaat qilishda ularning to`g'ridan – to`g'ri qiymatiga murojaat qilinadi. Masalan, unsigned short int tipidagi yangi o`zgaruvchini e`lon qildik, keyin esa unga boshqa o`zgaruvchining qiymatini o`zlashtirdik. Buni quyidagicha yozish mumkin.

```
unsigned short int yourAge;
yourAge = &howOld;
```

Bilvosita murojaat operatori orqali esa ko`rsatkichda adresi saqlangan o`zgaruvchining qiymati olinadi. YourAge yangi o`zgaruvchiga adresi pAge ko`rsatkichida saqlanayotgan howOld o`zgaruvchisining qiymatini berish uchun quyidagi dastur fragmenti yoziladi.

```
unsigned short int yourAge;
yourAge = * pAge;
```

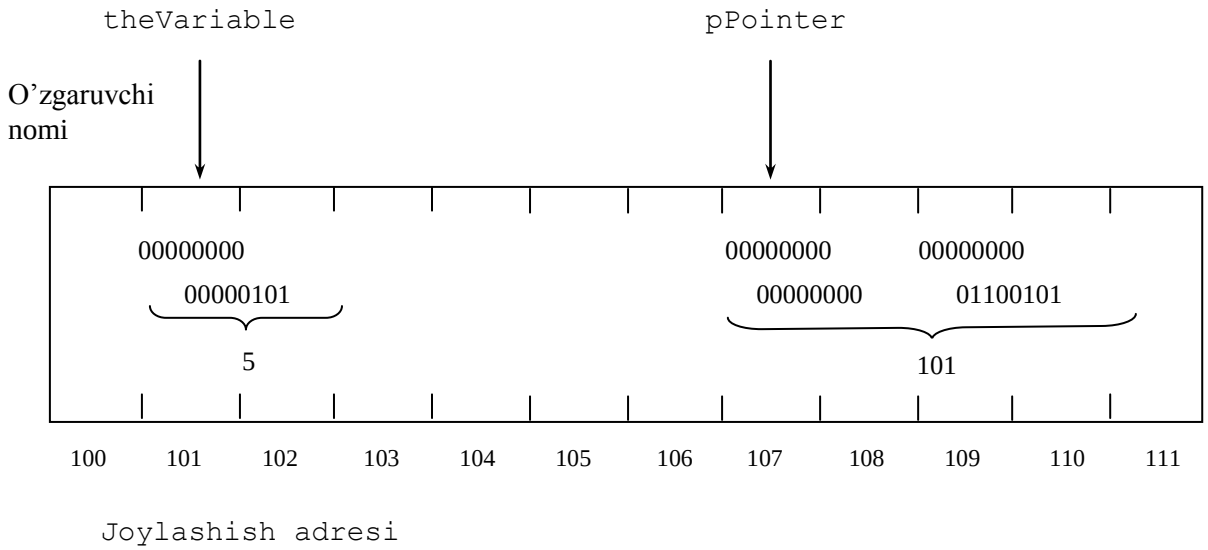
Bilvosita murojaat operatori pAge o`zgaruvchisining «adresida saqlanayotgan qiymatni» olish uchun ishlatiladi. Demak, yuqoridagi dastur fragmenti «pAge ko`rsatkichida adresi saqlanayotgan qiymatni olib, uni yourAge o`zgaruvchisiga o`zlashtirilsin» kabi o`qiladi.

Ko`rsatkichlar, adreslar va o`zgaruvchilar.

C++ dasturlash tili imkoniyatlarini o`rganishda, birinchi navbatda ko`rsatkichlar, ko`rsatkichda saqlanayotgan adreslar, ko`rsatkichda saqlanayotgan adresda yozilgan qiymatlar o`rtasidagi farqlarni bilishingiz kerak bo`ladi.

Yana bir dastur fragmentini qaraymiz.

```
int theVariable = 5;
int * pPointer = &theVariable;
```



2.2.3-chizma.Xotirani taqsimlanish sxemasi

Birinchi satrda butun tipli bo`lgan theVariable o`zgaruvchisi e`lon qilindi va unga 5 qiymat o`zlashtirildi. Keyingi satrda int tipiga ko`rsatkich e`lon qilindi va unga theVariable o`zgaruvchisining adresi o`zlashtirildi. pPointer ko`rsatkichi theVariable o`zgaruvchisi adresini o`zida saqlaydi. pPointer da yozilgan adresda saqlanuvchi qiymat 5 ga teng. rasmda bu o`zgaruvchilar strukturasi sxematik tarzda ko`rsatilgan.

Lokal o`zgaruvchilar va Funksiya parametrlari stekli xotirada joylashtiriladi. Dastur kodlari esa dastur segmentida joylashadi. Global o`zgaruvchilar xotiraning global o`zgaruvchilar sohasida joylashadi. Registrli xotira dasturning ichki xizmatchi ma`lumotlarini saqlash uchun xizmat qiladi. Xotiraning qolgan qismi esa turli ob`ektlar o`rtasida dinamik tarzda taqsimlanuvchi xotira sohasi, ya`ni bo`sh xotiradan iboratdir.

Lokal o`zgaruvchilarning o`ziga xos xususiyati shundan iboratki, dastur boshqaruvi ular tegishli Funksiyadan chiqishi bilan ular lokal o`zgaruvchilar uchun ajratilgan xotira sohasi bo`shatiladi, ya`ni bu o`zgaruvchilar o`chiriladi.

Global o`zgaruvchilar esa dastur ishga tushirilgandan boshlab toki o`z ishini yakunlaguncha xotiradan o`chirilmaydi, dasturning ixtiyoriy joyidan ularning qiymatidan foydalanishimiz mumkin bo`ladi. Lekin bu dastur matni tushunarliligini ancha murakkablashtiradi. Lokal va global o`zgaruvchilar o`ziga xos kamchiliklarga ega. Lokal o`zgaruvchilardan faqatgina u e`lon qilingan Funksiya ishlab turganda foydalanish mumkin bo`lsa, global o`zgaruvchilar dastur ishini boshlagandan toki oxirigacha global o`zgaruvchilar sohasidan joy olib turadi. Dinamik xotiradan foydalanish bu muammolarni butunlay hal qiladi.

Dinamik xotirani axborotlar yozilgan yacheykalarning nomerlangan to`plami sifatida qarash mumkin. Stek o`zgaruvchilaridan farqli ravishda bu xotira yacheykalarini nomlash mumkin emas. Ularga murojaat kerakli yacheyka adresini o`zida saqlagan ko`rsatkichlar orqali amalga oshiriladi.

Ma`lumki, stek o`zgaruvchilari Funksiya ishini tugatishi bilan tozalanadi. Natijada, barcha lokal o`zgaruvchilar qiymati bilan birgalikda xotiradan o`chiriladi. Stekdan farqli ravishda dinamik xotira dastur ishini tugatgunga qadar bo`shatilmaydi, bu xotirani bo`shatish bilan dasturchilarning o`zlari shug’ullanishlari lozim.

Dinamik xotiraning yana bir muhim imkoniyati shundan iboratki, uning uchun ajratilgan xotira sohasi bo`shatilmaguncha bu joydan foydalanish mumkin emas. Ya`ni, biror bir qiymatni dinamik xotiraga yozsak, toki uni u erdan o`chirmagunimizcha bu o`zgaruvchi uchun ajratilgan joy bo`shatilmaydi. Shuning uchun, dinamik xotira sohasi Funksiya bilan ishlash jarayonida ajratilgan bo`lsa, Funksiya ishini tugatgandan keyin ham biz undan bemalol foydalanishimiz mumkin bo`ladi. Xotirani dinamik tarzda belgilashni global o`zgaruvchilarni qo`llashdan yana bir ustunligi undagi ma`lumotlarga faqatgina uning ko`rsatkichiga murojaat qilish imkoniga ega bo`lgan Funksiyalar orqaligina murojaat qilish mumkin.

Kirish imkonining bunday tarzda tashkil etilishi joriy ma`lumotlarni o`zgartirilishini qat`iy nazorat qilish imkoniyatini beradi.

Bunday uslubda ishlash uchun birinchi navbatda dinamik xotira sohasi yacheykalariga ko`rsatkich tuzish lozim. Bu qanday amalga oshirilishini navbatdagi qismlarda ko`rib chiqamiz.

new operatori.

Xotiraning ob`ektlar o`rtasidan dinamik taqsimlanuvchi sohasidan joy ajratish uchun new operatori ishlatiladi. new operatoridan keyin xotiraga joylashtiriladigan ob`ekt tipini ko`rsatish lozim. Bu ob`ektni saqlash uchun talab etiladigan xotira sohasi o`lchovini aniqlash uchun kerak bo`ladi. Masalan, new unsigned short int deb yozish orqali biz dinamik taqsimlanuvchi xotiradan ikki bayt joy ajratamiz. Xuddi shuningdek, new long satri orqali to`rt bayt joy ob`ektlar o`trasida dinamik taqsimlanuvchi sohadan ajratiladi.

new operatori natija sifatida belgilangan xotira yacheykasining adresini qaytaradi. Bu adres ko`rsatkichga o`zlashtirilishi lozim. Masalan, unsigned short tipidagi o`zgaruvchi uchun dinamik sohadan joy ajratish uchun quyidagi dastur kodi yoziladi:

```
unsigned short int *pPointer;
pPointer = new unsigned short int;
```

Yoki xuddi shu amalni bitta satrda ham yozish mumkin.

```
unsigned short int * pPoiner = new unsigned short int;
```

Ikkala holatda ham pPointer ko`rsatkichi unsigned short int tipidagi qiymatni saqlovchi dinamik soha xotirasining yacheykasini ko`rsatib turadi. enda pPointer ko`rsatkichini shu tipdagi ixtiyoriy o`zgaruvchiga ko`rsatkich sifatida qo`llash mumkin. Ajratilgan xotira sohasiga biror bir qiymat joylashtirish uchun quyidagicha yozuv yoziladi:

```
* pPointer = 72 ;
```

Bu satr quyidagi ma`noni anglatadi: «pPointer ko`rsatkichida adresi saqlanayotgan xotiraga 72 sonini yozing». Dinamik xotira sohasi albatta

chegaralangan bo`ladi. U to`lib qolganda new operatori orqali xotiradan joy ajratishga urinsak xatolik yuz beradi. Bu haqda keyinroq to`xtalib o`tamiz.

delete operatori.

Agarda o`zgaruvchi uchun ajratilgan xotira kerak bo`lmasa uni bo`shatish zarur. Bu o`zidan keyin ko`rsatkich nomi yoziladigan delete operatori yordamida amalga oshiriladi. delete operatori ko`rsatkich orqali aniqlangan xotira sohasini bo`shatadi. SHuni esda saqlash lozimki, dinamik xotira sohasidagi adresni o`zida saqlovchi ko`rsatkich lokal o`zgaruvchi bo`lishi mumkin. SHuning uchun bu ko`rsatkich e`lon qilingan Funksiyadan chiqishimiz bilan ko`rsatkich ham xotiradan o`chiriladi. Lekin new operatori orqali bu ko`rsatkichga dinamik xotiradan ajratilgan joy bo`shatilmaydi. Natijada xotiraning bu qismi kirishga imkonsiz bo`lib qoladi. Dasturchilar bu holatni xotiraning sirqib ketishi, yoki yo`qolishi (utechka pamyati) deb tavsiflaydilar. Bu tavsif haqiqatga butunlay mos keladi, chunki dastur ishini yakunlaguncha xotirani bu qismidan foydalanib bo`lmaydi.

Xotirani ajratilgan qismini bo`shatish uchun delete kalitli so`zidan foydalaniladi. Masalan:

delete pPointer;

Bunda ko`rsatkich o`chirilmaydi, balki unda saqlanayotgan adresdagi xotira sohasi bo`shatiladi. Belgilangan xotirani bo`shatilishi ko`rsatkichga ta`sir qilmaydi, unga boshqa adresni o`zlashtirish ham mumkin. Dinamik o`zgaruvchi uchun qanday xotira ajratilishi, uni ishlatish va ajratilgan xotirani bo`shatishga oid misol keltirilgan.

Misol: Dinamik xotirani ajratish, undan foydalanish va uni bo`shatish.

```
 # include < iostream.h>
int main()
{
int   local variable = 5;
int * pLocal = & local variable;
```

```
pHeap = 7;
cout << “local variable:” << local variable <<”\n”;
cout << “ *pLocal: ”<< *pLocal  << “\n”;
cout << “ *pHeap: ” << *pHeap  << “\n”;
delete pHeap;
pHeap = new int;
*pHeap = 9;
cout <<“*pHeap:”<<*pHeap << “\n”;
delete pHeap;
return 0;
}
```

NATIJA:

local variable: 5

*pLocal: 5

*pHeap: 7

*pHeap: 9

Xotiraning sirqib ketishi nima?

Ko`rsatkichlar bilan e`tiborsiz ishlash natijasida xotiraning sirqib ketishiga yo`l qo`yish mumkin. Bu ko`rsatkich murojaat qilib turgan xotira bo`shatilmasdan, shu ko`rsatkichga yangi qiymat o`zlashtirilgan vaqtda ro`y beradi. Bunday holatga quyida misol keltirilgan:

-unsigned short int*pPointer=new unsigned short int;

-*pPointer = 72;

-pPointer = new unsigned short int;

- *pPointer = 84;

Birinchi satrda ko`rsatkich e`lon qilinyapti va unsigned short int tipidagi o`zgaruvchini saqlash uchun xotira ajratilayapti. Navbatdagi satrda ajratilgan sohaga 72 qiymat yozildi.Uchinchi satrda esa  ko`rsatkichga xotira sohasining boshqa adresi o`zlashtirildi va u adresdagi sohaga 84 qiymat o`zlashtirildi. Bu

Cat * pCat= new Cat.

Bu holatda new operatori sinfni boshlang’ich konstruktorini, ya`ni parametrsiz konstruktorini chaqiradi. Ob`ektni tuzishda u stekda yoki dinamik xotira sohasida joylashtirilishidan qat`iy nazar doimo u tegishli sinf konstruktori chaqiriladi.

Ob`ektni dinamik taqsimlanuvchi xotiradan o`chirish.

delete operatori ishlatilganda avtomatik tarzda undan keyin yozilgan ko`rsatkichda adresi saqlanuvchi ob`ekt tegishli sinf destruktori chaqiriladi. Qoida bo`yicha sinf destruktori ob`ektning dinamik xotira sohasida egallagan barcha xotira sohasini bo`shatadi. Ob`ektni dinamik xotiraga joylashtirish va o`chirishga oid misol ko`rsatilgan.

Misol: Dinamik xotira sohasiga ob`ektlarni joylashtirish va ularni o`chirishga oid misol.

```
// Dinamik taqsimlanuvchi sohada ob`ektlarni
//joylashtirish va o`chirish
# inslude <iostream.h>

class SimpleCat
{
public:
SimpleCat();
~SimpleCat();
private:
int itsAge;
}
SimpleCat:: SimpleCat( )
{
cout<< “Contructor called .\n ”;
itsYosh= 1;
}
```

```
SimpleCat:: ~ SimpleCat( )
{
cout <<”Destructor called .\n”;
}
int main()
 {
 cout << “Simple Cat Fricky… \n”;
 SimpleCat Frisky;
 cout << “SimpleCat *pRags = new SimpleCat… \n”;
 SimpleCat* pRags = new SimpleCat;
 cout<< “delete pRags… \n”;
 delete pRags
 cout<< “Exiting, watch Fricky go … \n”;
 return 0;
 }
```

NATIJA

SimpleCat Frisky…
Constructor called.
Simple Cat*pRags = new Simple Cat…
Construstor called
delete pRags…
Destructor called
Exiting, wath Frisky go …
Destructor called.

Sinfning ekzemplyarining (ob`ektining) lokal o`zgaruvchi bo`lgan a`zolariga murojaat to`g’ri murojaat(.) operatori yordamida amalga oshiriladi. Dinamik taqsimlanuvchi sohada hosil qilingan sinf ekzemplyarlarining a`zolariga esa quyidagi tarzda murojaat qilinadi: oldin sinf ekzemplyariga uni adresini o`zida

saqlagan ko`rsatkich orqali murojaat qilinadi (bilvosita murojaat operatori orqali), keyin esa to`g'ri murojaat operatori orqali uning a`zolariga murojaat qilinadi. Masalan, GetAge() Funksiya a`zosini chaqirish uchun quyidagicha yozuv yozish lozim:

(*pRags).GetAge();

Bu erda qavslar bilvosita murojaat operatori (*) GetAge() Funksiyasi chaqirilishidan oldin bajarilishini anglatadi.

Bunday konstruktsiya yozish uchun biroz noqulayroqdir. Bu muammo ctrelkani eslatuvchi, sinf a`zosiga bilvosita murojaat operatori(–>) orqali hal qilinadi. Bu operatorni yozish uchun uzluksiz ravishda ketma–ket ikki belgi, tire va katta ishorasini terish lozim. S++ da bu belgilar bitta operator sifatida karaladi. Misolda dinamik sohada joylashgan sinf ekzemplyarining a`zolari, uning maydonlari va metodlariga murojaat qilish namoyish etilgan.

Misol: Dinamik taqsimlanuvchi sohadagi ob`ektning a`zolariga murojaat qilishga oid misol.

```
// Dinamik taqsimlanuvchi sohaga ob`ektlarni
//  joylashtirish va ularni o`chirish
# include < iostream.h>
class SimpleCat
{
public:
SimpleCat() { itsYosh= 2; }
~SimpleCat() { }
int GetAge() const   { return itsAge; }
void SetAge(int age) {itsYosh= age; }
private:
int itsAge;
}
int main()
{
```

```
SimpleCat *Frisky = new SimpleCat;
cout<<“Frisky”<<Frisky–>GetAge()<<“years old\n”;
Frisky–>SetAge(5);
cout<<“Frisky”<<Frisky–>GetAge()<<“years old\n”;
cout << “years old\n”;
delete Frisky:
return 0;
}
```

NATIJA:

Frisky 2 years old

Frisky 5 years old

Sinf a`zolarini dinamik taqsimlanuvchi sohada joylashtirilishi.

Sinf a`zolari dinamik taqsimlanuvchi sohada joylashtirilgan ob`ektlarga ko`rsatkichlar ham bo`lishi mumkin. Bunday holatlarda bu ob`ektlar uchun xotiradan joy yoki konstruktor, yoki sinfning boshqa bir metodlari orqali ajratiladi. Qoida bo`yicha ob`ekt band etgan xotira sohasini bo`shatish shu ob`ektning destruktorini chaqirish orqali amalga oshiriladi.

Misol. Ko`rsatkichlar sinf a`zolari sifatida.

```
//8.7. – misol.
//Sinf a`zolariga ko`rsatkichlar
# include <iostream.h>
class SimpleCat
{
public:
SimpleCat( );
~SimpleCat( );
int GetAge( ) const { return *itsAge;}
void SetAge(int age) { * itsYosh= age }
```

```
int GetWeight ( int weight){ return * itsWeight; }
void SetWeight (int weight){ *itsOgirlik= weight }
private:
int * itsAge;
int * itsWeight;
}
SimpleCat::SimpleCat( )
{
itsYosh= new int(2);
itsOgirlik= new int(5);
}
SimpleCat::~SimpleCat( )
{
delete itsAge;
delete itsWeight;
}
int main( )
{
SimpleCat* Frisky = new SimpleCat;
cout<<“Frisky”<<Frisky->GetAge()<<“years old\n”;
Frisky -> SetAge(5);
cout<<“Frisky”<<Frisky->GetAge()<<“years old\n”;
delete Frisky;
return 0;
}
```

NATIJA:

```
Frisky 2 years old
Frisky 5 years old
```

This ko`rsatkichi.

Sinfning har bir metodi yashirin parametrga – this ko`rsatkichiga egadir. Bu ko`rsatkich joriy ob`ektning adresini o`zida saqlaydi. Oldingi qismlarda qarab chiqilgan GetAge() va SetAge() Funksiyalari ham o`zlarida bu parametrni saqlar edilar.this ko`rsatkichini oshkor ko`rinishda qo`llanilishiga misol keltirilgan.

Misol: this ko`rsatkichi.

```
// this ko`rsatkichi
# inslude <iostream.h>
class Turtburchak
{
public:
Turtburchak( );
~Turtburchak( );
void SetLength(int Length){this->itsLength=length }
int GetLength() const {return this ->itsLength; }
void SetWidth(int width) {itsWidth= width;}
int GetWidth( ) const { return itsWidth;}
private:
int itsLength;
int itsWidth;
}
Turtburchak::Turtburchak()
{
itsWidth = 5;
itsLength = 10;
}
Turtburchak:: ~Turtburchak()
{ }
int main()
{
```

```
Turtburchak theRect;
cout<<“theRect is”<<theRect.GetLength()<<“meters long.\n”;
cout<<“theRect is”<<theRect.GetWidth ( ) << “meters wide.\n”;
TheRect.SetLength(20);
TheRect.SetWidth(10);
cout<<“theRect is”<<theRect.GetLength()<<“meters  long.\n”;
cout<<“theRect is”<<TheRect.GetWidth()<< “meters wide.\n”;
return 0;
```

NATIJA:

The Rect is 10 meters long

The Rect is 5 meters wide

The Rect is 20 meters long

The Rect is 10 meters wide

Xotiraning taqsimlanishi.

Siz o`zingizni dasturingiz bilan ishlay boshlashingiz bilan operatsion sistema (Dos yoki Microsoft Windows) kompilyatorning talabiga muvofiq xotira sohasidan joy ajratadi. C++ dasturchisi sifatida siz global nomlar fazosi, erkin taqsimlanuvchi xotira, registr, segmentli xotira va stek tushunchalarini bilishingiz lozim.

Global nomlar fazosida global o`zgaruvchilar saqlanadi. Global nomlar fazosi va erkin taqsimlanuvchi xotira haqida keyingi Ma`ruzalarda batafsilroq tanishib chiqamiz. Hozir esa registr, dastur segmenti va stek haqida to`xtalib o`tamiz.

Registr xotiraning maxsus sohasi bo`lib, uning asosiy vazifasi ichki yordamchi Funksiyalarni ishlashini tashkil etishdir.

Dasturning o`zi dastur operatorlari ikkilik formatda saqlanishi uchun ajratilgan komp’yuter xotirasida saqlanadi.

Stek – bu sizning dasturingizda Funksiya chaqirilganda undagi ma`lumotlar uchun talab qilinadigan xotiraning maxsus sohasidir. Uning stek deb atalishiga sabab «oxirgi kelgan – birinchi ketadi» printsipi asosida ishlashidir. Haqiqatan ham, Funksiyalarni chaqirilishi xuddi shu printsip asosida amalga oshiriladi.

Agarda bir Funksiya ikkinchisini chaqirsa, ikkinchi Funksiya o`z ishini tugatgandan so`ng birinchi Funksiya o`z ishini yakunlaydi, ya`ni oxirgi chaqirilgan Funksiya birinchi ishini tugatadi.

Ikkinchi bobning qisqacha xulosasi. Bu bobda xotirani boshqarish tushuntirilgan. Xotirani boshqarishda nimalarga e’tibor berish kerakligi, dastur bajarilish jarayonida xotirani tejash qanchalik muhim ekanligi, xotiraning bo’linish sohalariga ta’riflar berilgan.

Xotima.

Ushbu bitiruv malakaviy ishimning mavzusi “Ob’yektga mo’ljallangan dasturlashda keraksiz ob’yektlarni yo’qotish strategiyalari” deb nomlanib, u kirish qismi, uchta bob, boblarning qisqacha xulosalari, xotima va foydalanilgan adabiyotlardan iborat.

Birinchi bob besh qismdan iborat birinchisida, ob’yekt haqida tushuncha berilgan. Bunda asosan Ob`ektga mo`ljallangan dasturlashni bajarilish jarayonida bir qancha ob`ektlar yaratilishi, bu ob`ektlarni tashkil etish va ular orasidagi munosabatlar, bajarilish vaqtidagi konstruktsiyasida aniqlanishi, "ob`ekt" iborasini ma`nosini keltirilgan, ya’ni ob`ekt – bu ba`zi bir sinfning ekzemplyari deb qaralgan.

Ikkinchi qismda statik va dinamik tuzilmalar haqida yozilgan. Bir-biriga o’hshash va farqli tomonlari yoritib berilgan. Dasturda asosiy tur ma'lumot tuzilmalari ikki xil bo’lishligi, birinchisi statik, ikkinchisi dinamik ekanligi, statik deganimizda hotirada egallagan joyi o'zgarmas, dastur boshida beriladigan tuzilmalarni nazarda tutilishi, dinamik ma'lumot tiplari dastur davomida o'z hajmini, egallagan hotirasini o'zgartirishi mumkinligi nazariy hamda misollarda ko’rsatilgan.

Uchinchi qism dinamik massivlar. Agar tuzilma bir xil kattalikdagi tiplardan tuzilgan bo'lsa, uning nomi massiv (array) deyiladi. Massivlar dasturlashda eng ko'p qo'laniladigan ma'lumot tiplari bo’lishligi, ikkinchi qism va keyingi qismlarda bir bog’lamli, ikki bog’lamli, takrorlanuvchi ro’yhat va daraxtsimon tuzilmalar haqida ma’lumot berilgan, hamda har birining qanday tuzilishga ega ekanligi, ro’yhatlarni chiqarishda kiritilgan tartibda chiqarish,ro’yhatni teskaridan chiqarish, ma’lum bir elementidan boshlab chiqarish misollarda ko’rsatilgan.

Ikkinchi bob uchta qismdan tashkil topgan. Ob’yektlarni boshqarish rejimlari, dinamik rejimdan foydalanish, ma’lumotlarni xotirada joylashishi. Ob`yektlarni boshqarishni uchta rejimi mavjud: statik, stekli va dinamik taqsimlash. Bu uchta taqsimlanish haqida tushunarli javob berilgan. Va mukammal taqsimlanishga alohida e’tibor berilgan.Mashina xotirasi nomerlangan yacheykalar ketma-

ketligidan iboratligi. Har bir o`zgaruvchining qiymati uning adresi deb ataluvchi alohida xotira yacheykasida saqlanishi. Dinamik xotirani axborotlar yozilgan yacheykalarning nomerlangan to`plami sifatida qaralgan. Stek o`zgaruvchilaridan farqli ravishda bu xotira yacheykalarini nomlash mumkin emas. Ularga murojaat kerakli yacheyka adresini o`zida saqlagan ko`rsatkichlar orqali amalga oshirish mumkin.

Foydalanilgan adabiyotlar

1. Karimov I.A. “O’zbekiston buyuk kelajak sari” – Toshkent. O’zbekiston 1998 y.
2. Barkamolavlod–O’zbekiston taraqqiyotining poydevori (O’zbekiston Respublikasining “Ta’lim to’g’risida” gi va “Kadrlar tayyorlash milliy dasturi to’g’risidagi” qonunlar) – T. Sharq, 1998y.
3. Уильям Топп, Уильям Форд.Структуры данных в C++: Пер. с англ. — М.: ЗАО «Издательство БИНОМ», 1999. - 816 с. : ил
4. Фридман А., Кландер Т., Михаэлис М., Шильдт Х.С/C++. Архив программ ~ М.: ЗАО «Издательство БИНОМ», 2001 г. 640 с: ил.
5. Sh.A.Nazirov, R.V.Qobulov “Ob’ektga mo‘ljallangan dasturlash” T.2006y. 179 b.
6. Жесс Либерти, “Освой самостоятельно C++ за 21 день”, Санкт Петербург 2000, 815 с.
7. Coplien, J. 1992. Advanced C++ Programming Styles and Idioms.
8. Bergin, J., and Greenfield, S. March 1988. What Does Modula-2 Need to Fully Support Object-oriented Programming? SIGPLAN Notices vol.23(3).
9. Bhaskar, K. October 1983. How Object-oriented Is Your System? SIGPLAN Notices vol.1
10. Althoff, J. August 1981. Building Data Structures in the Smalltalk-80 System. Byte vol.6(8).
11. Agha, G. October 1986. An Overview of Actor Languages. SIGPLAN Notices vol.21(10).
12. Chang, S. 1990. Visual Languages and Visual Programming. New York, New York: Plenum Press.