ЎЗБЕКИСТОН РЕСПУБЛИКАСИ АЛОҚА , АХБОРОТЛАШТИРИШ ВА ТЕЛЕКОММУНИКАЦИЯ ТЕХНОЛОГИЯЛАРИ ДАВЛАТ ҚЎМИТАСИ

ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ

«Компьютер» инжиниринг" факультети
«Информатика асослари» кафедраси

«C/C++ тилида дастурлаш» фани бўйича маърузалар матни

3-қисм

Тошкент -2012

Маъруза -17, 18. (8 с)

Мавзу: КАТТАЛИКЛАРНИНГ ТАРТИБЛАНГАН ТУЗИЛМАСИ. МАССИВЛАР, УЛАРНИНГ ЎЛЧАМЛАРИ, КЎП ЎЛЧАМЛИ МАССИВЛАР.

Мақсад: ***Талабаларда катталикларнинг тартибланган тузилмаси, массивлар, уларнинг ўлчамлари, массивлар билан ишлаш кўникмасини шакллантириш.***

Калит сўзлар: ***Массив, базавий тоифа, индекс, бир ўлчамли массив, кўп ўлчамли массив.***

Режа:

1. **Бир ўлчамли массивларни тавсифлаш. Ьир ўлчамли массивларга ишлов бериш.**
2. **Массив элементларини киритиш.**
3. **Кўп ўлчамли массивларга ишлов бериш.**
4. **Массивларни саралаш. Мисоллар**

Умумий номга эга ва тартибланган катталиклар тўпламига массив дейилади. Массив – индексли ўзгарувчи деган маънони билдиради. Массив элементларининг тоифаси базавий тоифа деб юритилади. Масалан: $a = (a_1, a_2, \ldots, a_{10})$;
бу ерда: a – массив номи, a_1, a_2, …. , a_{10} - массив элементлари, 1,2,..10 – элемент индекслари. Индекс ўзгарувчиларнинг тартибланган ўрнини билдиради ва у C/C++ тилида [ ] қавсларда келтирилади. Массив таърифланганда унинг тоифаси, номи ва индекслари кўрсатилади. Мисол:
int a[5]; char ww[20]; double f [100];
Массив элементларининг индекслари доим 0 дан бошланади, демак a массивда a[0],a[1],a[2],a[3],a[4] элемент, ww массивида ww[0], ww[1], … ww[19] элемент, f массивида f[0],f[1],f[2]…f[99] элемент бор.
Массив элементларига сон қийматларини бериш усуллари.
1. Массивлар таърифланганда уларни бевосита инициализация қилиниши мумкин. Масалан: float c[4] = {1, 0.1, -45, 7.23}; Бу ёзувни қуйидагича ҳам ёзса бўлади: float c[] = {1, 0.1, -45, 7.23}; демак агар массив чегараси кўрсатиламаган бўлса, сон қийматларга қараб аниқланиши ҳам мумкин. Массив чегараси кўрсатилган, лекин унга бериладиган сон қийматлар кам бўлиши ҳам мумкин. У ҳолда қолган қийматлар аниқланмаган деб қаралади. Масалан: float c[4] = {1.56, 7.23}; , яъни c[1]=1.56, c[2]=7.23 , яъни қолган 2таси аниқланмаган дейилади. Лекин сон қийматлари кўп бўлиши мумкин эмас.
Масалан: float c[4] = {1, 0.1, -45, 7.23, -8.96, 7.78};
2. Киритиш операторн ёрдамида сон қийматларни аниқлаштириш. Бунда cin оператори for ёрдамида берилади. Масалан:

```
# include <ioctream.h>
void main ( )
{  int  a[10];
for (int i=0; i<10; i++)
cin >> a[i];     }
```

бу усулда дастур тузилганда массив элементлари клавиатурадан киритилади.
3. Массив элементларининг сон қийматларини const орқали ҳам кўрсатилиши мумкин, бу ҳолда уларнинг сон қийматларини кейин ўзгартириб бўлмайди.

4. Массив элементларининг сон қийматларини чиқариш ҳам for оператори ёрдамида бўлади.

Мисоллар кўриб ўтамиз.
1-мисол. Массив элементларидан мусбатларининг сони ва суммасини ҳисоблаш дастури.

```
# include <iostream.h>
# include <conio.h>
void  main ( )
{  int  x[ ] = {1,2,56,78,-7,-45,34,12,9,-1};
int  s=0, n=0;  clrscr ( );
for  (int i = 0; i < 10; i++)
{ if  (x[i] < 0)  continue;
s = s + x[i];    n++;}
cout << “s=”<< s << “n=”<<n<<endl;
getch ( );
}
```

ёки

```
int  x[10], s=0, i, n=0;
for  (i = 0;  i < 10;  i++)
cin >> x[i];
for  (i=0; i<10; i++)
{  if (x[i] < 0) continue;
s + = x[i];  n++ ;
}
cout << “s=”<< s;  }
```

2-мисол. 10та элементдан иборат массивнинг энг катта, энг кичик элементларини ва уларнинг ўрта қийматини ҳисоблаш дастури.

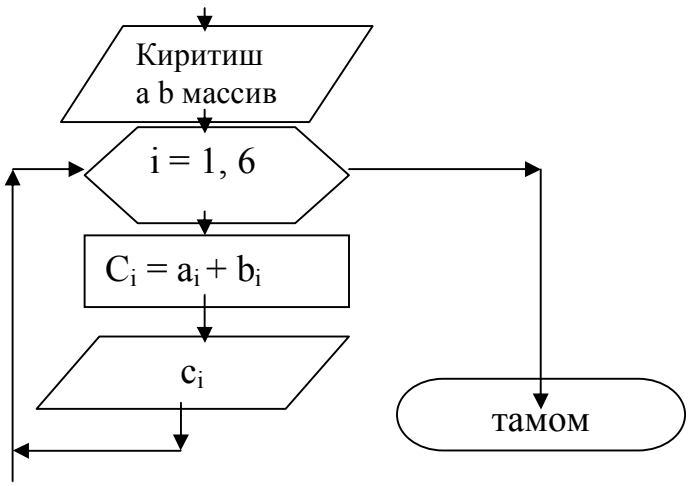
```
# include <iostream.h>
# include <conio.h>
 void main ( )
{
float   x[] = {1,2.23,5.6,-78,-7,-45.12,34.0,12,9,-1};
float s, min, max; int i; clrscr ( );
min = x[0];   max=x[0];
for  (i=0;  i<10;  i++)
{  if  (min > x[i])   min = x[i];
   if  (max < x[i])   max = x[i]; }
s = (min + max) / 2;
cout << “min=”<<min<<endl;
cout <<”max=”<<max<<endl;
cout <<”o’rta qiymat=”<<s<<endl;
getch ( );
 }
```

Изоҳ: for операторида i=1 деб олинса ҳам бўлади, у ҳолда n нечта бўлса, шунча элемент олинади. Агар i=0 деб олинса, n-1 элемент олинади.

<u>1-мисол</u>. 6та элементдан иборат a ва b массивлари берилган. Уларни ўзаро қўшиб янги c массивини ҳосил қилинг.

Бошланиш


```
# include <iostream.h>
# include <conio.h>
void  main ( )
{ int a[6] = {1,2,3,4,5,6},
```



2-мисол. Векторларнинг скаляр кўпайтмасини ҳисоблаш алгоритмини тузинг.

$S = \sum_{i=1}^{n} x_i y_i$; n=5;

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ int x[5]={1,2,3,4,5}, y[5]={6,7,8,9,10}, s=0;
for (int i=0; i < 5; i++)
s = s + x[i] * y[i];
cout << ”s=” << s << endl;
getch ( ); }
```

3-мисол. Базавий тоифаси ҳақиқий бўлган 10та элементли А массиви берилган. Жуфт индексли элементлардан алоҳида, тоқ индексли элементлардан алоҳида массив ҳосил қилинг.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ float a[10], b[5], c[5];
for (int i=0; i<10; i++)
cin >> a[i];
for (int i=0; i<5; i++)
{ c[i] = a[2*i +1];
  b[i] = a[2*i];
cout << ”c=” << c[i];
cout << ”b=” << b[i] <<endl; }
getch ( );
}
```

4-мисол. А ва В массивлари берилган. Янги С массивини қуйидагича ҳосил қилинг: А массивининг элементлари янги массивнинг тоқ элементлари, В массивининг элементлари эса янги массивнинг жуфт элементларини ташкил этади.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ float a[5], b[5], c[10]; int i;
for ( i=0; i<5; i++)
```

```
cin >> a[i] >> b[i];
for  ( i=0; i< 5; i++)
{ c[2*i+1] = b[i];
  c[2*i] = a[i];  }
for (i = 0; i < 10; i++)
  cout << ”c[” << i << ”]= ” << c[i] <<endl ;
getch ( );
}
```

Назорат саволлари:

1. Массив. Массив тушунчаси.
2. Индексли ўзгарувчилар. Масивларни киритиш ва чиқариш.
3. Икки ўлчамли массивлар.
4. Массивнинг энг катта ёки энг кичик элементини топиш алгоритми.
5. Турли ўлчамли вектор узунликларини ҳисоблаш алгоритми.

Тестлар:

1. Массив элементларидан фойдаланиш нима орқали бажарилади:
   A) FIFO йўналиш
   B) LIFO йўналиш
   C) нуқта операцияси
   D) элемент исми
   +E) элемент индекси

2. Массивнинг нотўғри таърифини кўрсатинг:
   A) int a[20];
   B) int a[]={1,2,3,4};
   C) int a[4]={1,2,3,4};
   +D) int a[2]={1,2,3,4};
   E) ҳаммаси тўғри

Маъруза-19 (6 с)

Мавзу: C++ BUILDER МУҲИТИ. УНИНГ КОМПОНЕНТЛАРИ ВА ВАЗИФАЛАРИ. STANDART, ADDITIONAL, WIN 32, SYSTEM, DATA ACCESS, DATA CONTROL, BDE, ADO САҲИФАЛАРИ.

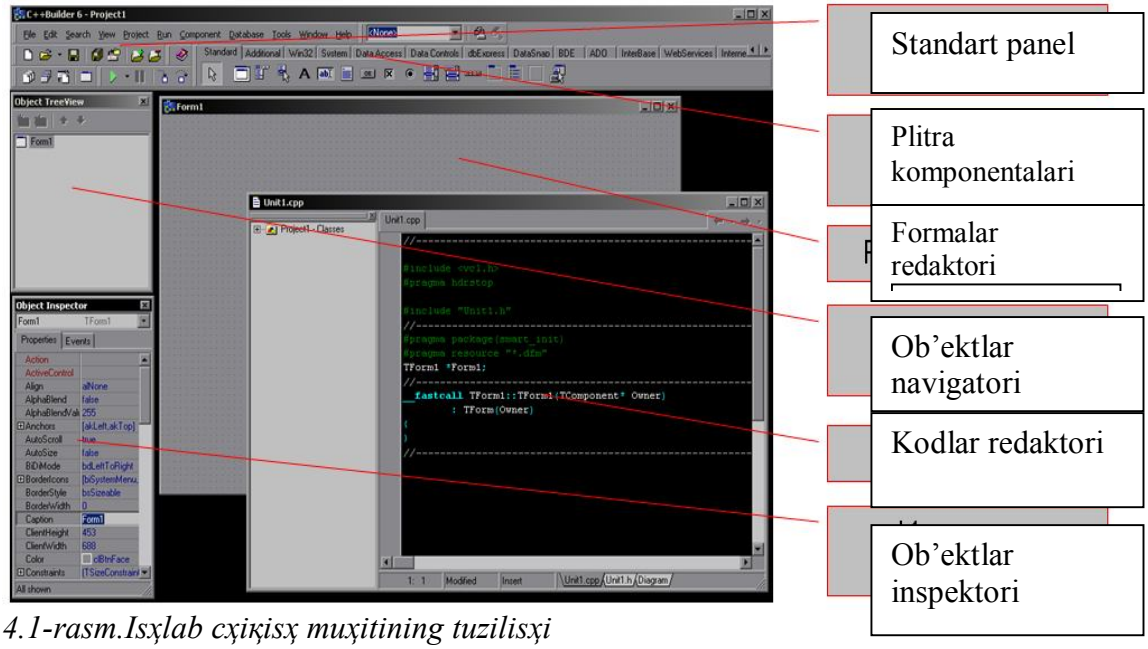
Мақсад: ***Талабаларда Borland C++ Builder 6 мухити, унинг компонентаалари ва вазифалари, Standart, Additional, Win32, System, Data Access, Data Controls, BDE, ADO саҳифалари билан ишлаш кўникмаларини ҳосил қилиш.***

Kalit sўzlar: ***Komponentlar Palitrasi, Sҳakllar Muҳarriri, Kod Muҳarriri, Obhektlar Noziri, Obhektlar Xazinasi, Vizual loyiҳalasҳ, Ikki yўnalisҳli isҳlanma texnologiyasi, Loyiҳalasҳ sҳablonlari, Two Way Tools texnologiyasi, komponenta, saҳifa, xususiytlar, metodlar, voқealar, sҳablon, yўnalisҳ.***

Reja:

1. **Vizual loyiҳalasҳ**
2. **Xususiyatlar, metodlar va voқealar**
3. **Ikki yўnalisҳli isҳlanma texnologiyasi**
4. **Loyiҳaviy sҳablonlarni қўllasҳ**

Isҳlab cҳiқisҳning integratsiyalasҳgan muҳiti Komponentalar palitrasini birlasҳtiradi. Sҳakllar Muҳarriri, Kod Muҳarriri, Obhektlar Noziri, Obhektlar Xazinasi - bular ҳammasi kod va zaҳiralar ustidan tўliқ nazoratni tahminlovcҳi dasturiy ilovalarni tez isҳlab cҳiқisҳ instrumetlari (4.1-rasm).



4.1-rasm.Isҳlab cҳiқisҳ muҳitining tuzilisҳi

■ ***Komponentlar Palitrasi*** ilovalarni қurisҳda taklif қilinadigan 100 dan ortiқ takroran қўllanadigan komponentlardan iborat.

■ ***Sҳakllar Muҳarriri*** dasturning foydalanuvcҳi bilan interfeysini yaratisҳ ucҳun mўljallangan.

■ ***Kod Muҳarriri*** dastur matnini, xususan, voқealarga isҳlov berisҳ funktsiyalarini yozisҳ ucҳun mўljallangan.

■ ***Obhektlar Noziri*** қotib қolgan cҳigal dasturlasҳ zaruratisiz obhektlar xususiyatlarini vizual ўrnatisҳ imkonini beradi ҳamda sҳunday voқealarni ўz icҳiga oladiki, bu voқealarni ularning paydo bўlisҳiga nisbatan obhektlar reaksiyasi kodlari bilan boғlasҳ mumkin bўladi.

■ ***Obhektlar Xazinasi*** mahlumotlarning sҳakl va modullari kabi obhektlarga ega bўlib, ular isҳlab cҳiқisҳda muvaққat sarflarni kamaytirisҳ maқsadida kўplab ilovalar bilan bўlinadi.

C++Builder ilovalarni қurisҳning vizual metodikasini Komponentlar Palitrasidan kerakli bosҳқarisҳ elementlarini tanlab olisҳ vositasida joriy etadi. Ҳar bir komponenta (masalan, tugmacҳa) bilan usҳbu komponenta turini va xulқ-atvorini ўzgartiradigan xususiyatlar boғliқ bўladi. Ҳar қanday komponenta usҳbu komponentaning turli xildagi tahsirlarga reaksiyasini (munosabatini) aniқlab beradigan voқealar seriyasini keltirib cҳiқarisҳi mumkin. Bunday keyin => belgilari siz C++Builder muҳitida amalga osҳiradigan xatti-ҳarakatlarni bildiradi.

=>C++Builder ni cҳaқiring va bosҳ menyudagi Fale | New Arrlication komandasi bўyicҳa yangi ilovalar ustida isҳlasҳni bosҳlang.

=>sicҳққoncҳani Komponentalar Palitrasining қўsҳimcҳa ilovalari ustida bosib, foydalanuvcҳi isҳ қўradigan dastur interfeysi elementlarining mavjud assortimentini қўrib cҳiқing.

Palitraning bir қўsximcxa ilovasidan ikkincxisiga ўtib, kirisx mumkin bўlgan komponentlar tўplami ўzgarayotganining guvoxi bўlisximiz mumkin. Sicxқoncxa kursori komponentlar belgisi ustida tўxtaganda, aytib turisx nomi paydo bўladi. Agar F1 klavisxasini bossak, tizimning mahlumotnomalar xizmati tanlab olingan komponenta xaқida tўlik mahlumot cxikarib beradi.

Vizual loyixalasx: [4(85-92)] Bizning birincxi ilovamiz bolalarning «Ўnta negr bolasi» sanoқ sxehrini generatsiya қiladi. Dastlabki versiyada faқat ucxta obhekt kerak bўladi: rўyxat, taxrir қilisx maydoni va tugmacxa. Komponentalarni loyixalasx sxakliga olib ўtamiz xamda ilovani asta-sekin rivojlantira bosxlaymiz. Tasxib olib ўtisx metodi (drag-and-drop) қuyidagilardan iborat: sicxқoncxa tugmacxasini tanlab olingan komponenta ustida bosing, kursorni sxaklning tўғri kelgan yeriga ўtkazing, keyin esa sicxқoncxa tugmacxasini yana bosing. Bosxida faқat «standart» Palitra Komponentlari bilan cxeklanamiz:

=> Standard қўsximcxa ilovani tanlab oling.

=> Rўyxat komponentasini ListBox sxakliga olib ўting.

=> Taxrir қilinatgan kiritisx maydoni EditBox ni olib ўting.

=> Button tugmacxasi komponentasini olib ўting.

=>Komponentalarni ўzingizning ilovangizdagi darcxada қanday kўrmoқcxi bўlsangiz, sxunday joylasxtiring va ўlcxamlarini sxunday ўzgartiring.

Obhekt Noziri yordamida komponentalar xususiyatlarining bosxlanғicx қiymatlarini aniқlang. Items rўyxatining xususiyatlar қiymatlari katagida tugmacxani bosing, ocxilgan muxarrir darcxasida sxehrning dastlabki 7 satrini kiriting. SXakl va tugmacxaning Caption xususiyatida ularning mahnoli nomlarini kўrsating (mos ravisxda, «Ўnta negr bolasi» va «Natija»). Taxri қilisx maxdonining Text xususiyatida natijani aytib berisx satrini kiriting («Tўққizta negr bolasi»).

Endi Kod Muxarririga ulanisx xamda, avval қabul қilinganidek, C++tilidagi xar қanday dasturni yozisx mumkin, sxu jumladan, ANSI/ISO standartining sўnggi kengaytmalarini xam. Biroқ, avval ilovalarni tez isxlab cxikisxning yangi vositalari xamda C++Builder da mavjud bўlgan қўsximcxa komponentalar atributlaridan foydalanisxga xarakat қilib kўramiz.

Ikki yўnalisxli isxlanma texnologiyasi

Loyixaviy sxablonlarni қўllasx

Xususiyatlar, metodlar va voқealar:[4(85-92)]

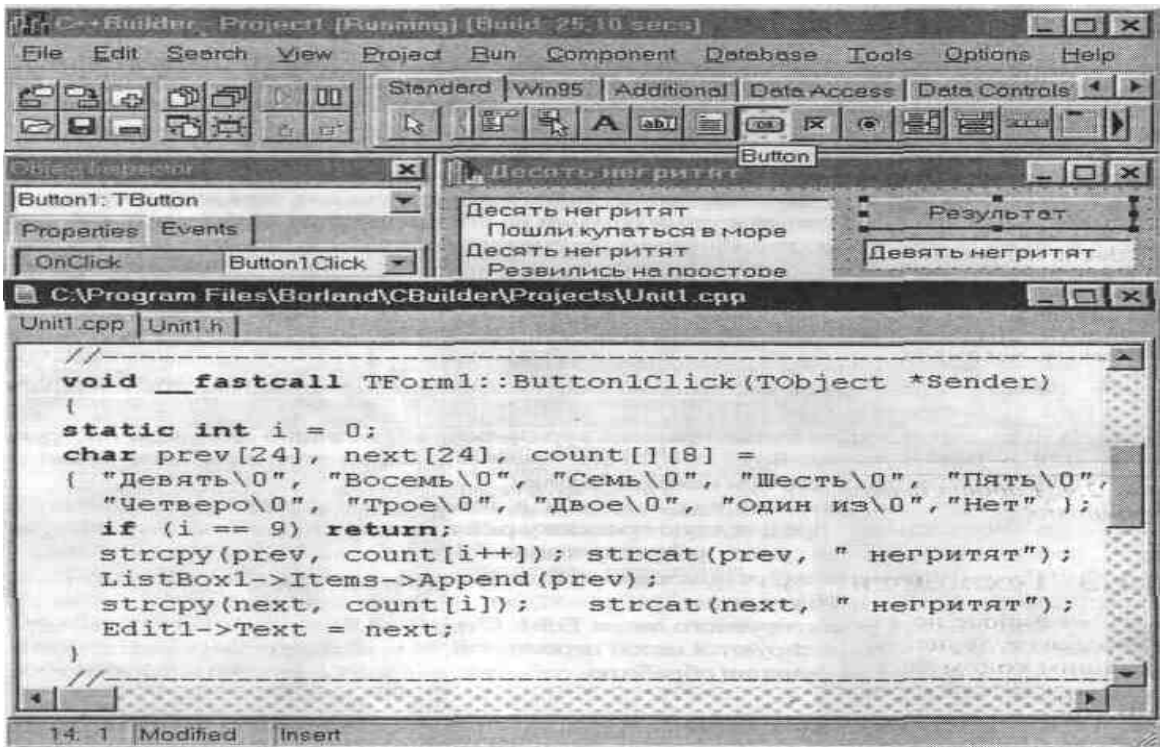
Ilovalarning tez isxlab cxikilisxi obhektli mўljallangan dasturlasx doirasida xususiyatlar, metodlar va voқealarning қўllab-қuvvatlanisxini bildiradi. *Xususiyatlar* komponentalarning nomlar, matniy aytib berisxlar yoki mahlumotlar manbalari kabi turli xildagi tavsiflarini osongina ўrnatisx imkonini beradi. *Metodlar* (ahzo-funksiyalar) komponentadagi obhekt ustida mahlum operatsiyalarni amalga osxiradi. Bunday operatsiyalar jumlasida қayta tiklasx yoki multimedia қurilmasini қayta ўrasx kabi murakkab operatsiyalarni xam kўrsatisx mumkin. *Voқealar* komponentalarga foydalanuvcxi kўrsatayotgan faollasxtirisx (aktivizatsiya), tugmalarni bosisx yoki taxrir қilinadigan kiritisx kabi tahsirlarni usxbu tahsirlarga sizning munosabat kodlaringiz bilan boғlaydi. Bundan tasxқari voқealar komponentalar xolatlarida sodir bўladigan ayrim ўziga xos ўzgarisxlar paytida xam yuzaga kelisxi mumkin. Bunday ўziga xos ўzgarisxlar қatorida mahlumotlar bazasiga kirisxning interfeysli elementlarida mahlumotlarni yangilasxni kўrsatib ўtisx kifoya. Xususiyatlar, metodlar va voқealar, birgalikda isx olib borar ekan, ular Windows ucxun isxoncxli ilovalarni intuitiv tarzda dasturlasx muxiti - RAD ni xosil қiladi.

=>Tanlangan obhekt bilan assotsiatsiyalanadigan (birgalikda yodga olinadigan) voқealarni kўrisx ucxun, Obhektlar Nozirida Voқealar (Events) қўsximcxa ilovasini kўrsating.

=>Ўzingiz sxaklga joylasxtirgan tugma komponentasini sicxқoncxa bilan ikki marta uring

=>Ocxilgan Kod Muxarriri darcxasida kursor ButtonIClick funksiyasi tanasiga instruktsiyalarni kiritisx ucxun pozitsiyani kўrsatadi. Bu funksiya esa tugmacxani bosisxda yuzaga keladigan OnClick vokeasiga isxlov berisx ucxun mўljallangan.

4.2-rasmda oddiy kod kўrsatilgan bўlib, u «Natija» tugmasini yana bir bor bosilisxiga javoban avval turgan plev aytib berisxini rўyxat oxiriga, xamda navbatdagi next aytib berisxini taxrir kilisx maydoniga kўsxadi. ListBoxI->Items->Append(prev) yўriknomasi, Append metodi yordamida, rrev satrini ListBoxI rўyxati obhektining Items xususiyatiga kўsxadi. EditI->Text=next yўriknomasi taxrir kilinayotgan EditI kiritisx obhektining Text xususiyatiga next satrini takdim etadi. Aytib berisx satrlari ikki ўlcxamli count massivida saklanadi va static turdagi butun ўzgaruvcxi tomonidan indekslanadi. Bu ўzgaruvcxi esa ButtonI tugmasini bosisx bilan yuzaga keladigan vokeaga isxlov berisx funksiyasining cxakirilisxlari ўrtasida ўzining joriy kiymatini saklaydi.



4.2-rasm. Kod muxarriri bajarilayotgan modul matnining Unit1.cpp faylida kiritilisxi va taxrir kilinisxini tahminlaydi.

Birincxi versiyali ilovani loyixalasx boskicxi sxuning bilan tugallanadi va isxcxi dasturni yaratisxga kirisxisx mumkin bўladi.

=>Run | Run bosx menyusi komandasi bilan ilovani kompilyatsiya kilisx (kўcxirisx) va yiғisx jarayonini isxga tusxirib yuboring

=>Dastur cxakirilgacx, bir necxa marta «Natija» («Rezultat») tugmasini bosing.

Ikki yўnalisxli isxlanma texnologiyasi:[4]

C++Builder dasturcxi va uning kodi ўrtasida xecx kanday tўsiklarni kўymaydi. Two-Way Tools ikki yўnalisxli isxlanma texnologiyasi vizual loyixalasx instrumentlari va Kod Muxarriri ўrtasida moslasxuvcxan, integrallasxgan va sinxronlasxtirilgan ўzaro aloka vositasida sizning kodingiz ustidan nazoratni taminlaydi. Ikki yўnalisxli isxlanma instrumentlari kanday amal kilisxini kuzatib borisx ucxun, kuyidagi operatsiyalarni bajaring:

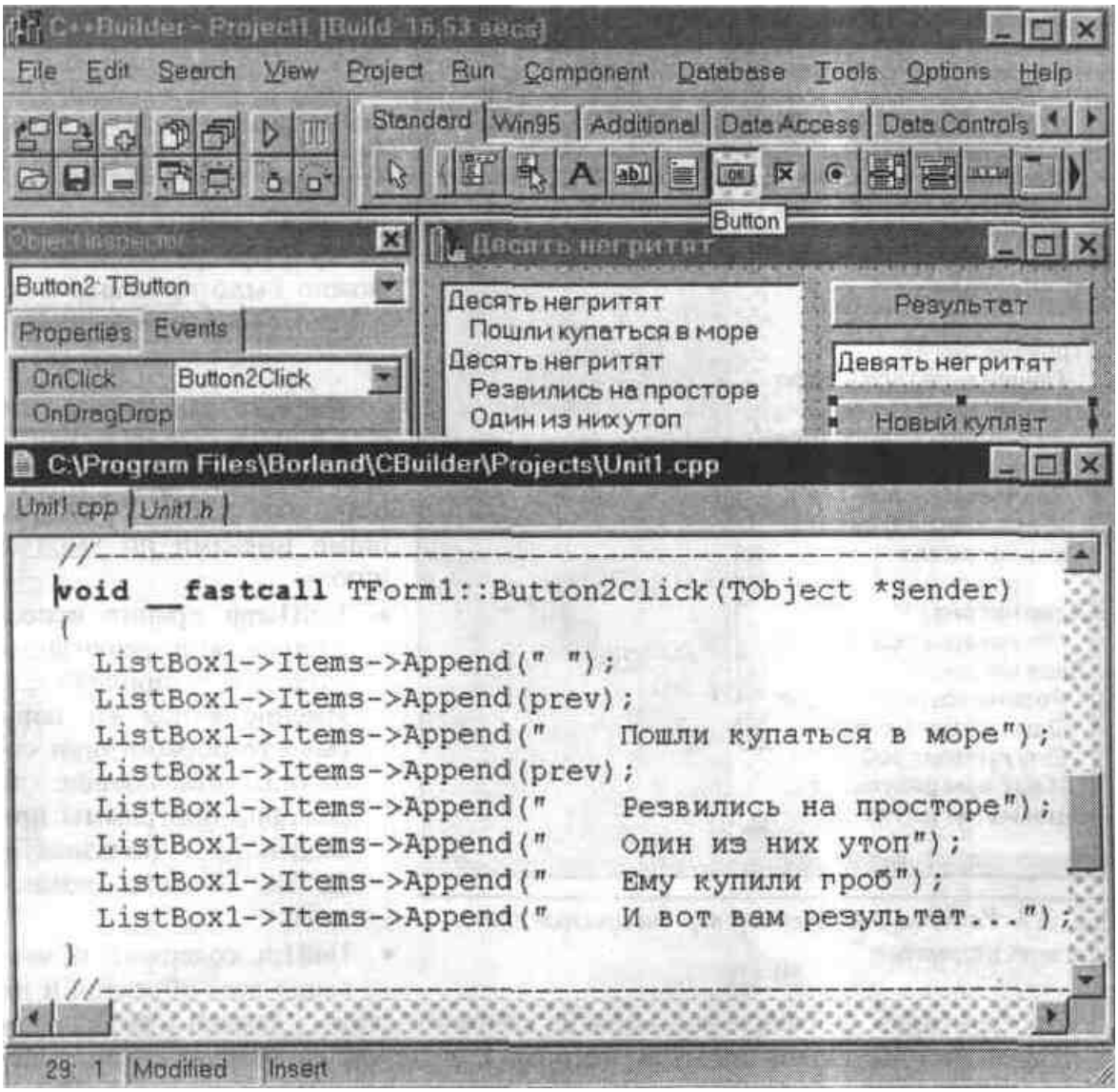
=>Sicxkoncxaning ўng tugmasini bosib, Kod Muxarririning kontekstli menyusini ocxing, keyin Swap Cpp/Xdr Files operatsiyasi yordamida Unit1.x. ehlonlar fayliga ulaning.

=> Instrumentlarning ekrandagi aksini sҳunday tasҳkil қilingki, bunda Kod Muҳarriri darcҳasida bir paytning ўzida loyiҳalanayotgan sҳakl va Unit.x fayli кўrinsin.

=>OK Button tugmasining yana bitta komponentasini sҳaklga olib ўting. Tugmaning Caption xususiyatida uning mahnoli nomi «Ягgi band» deb кўrsating.

Қuyidagilarni kuzatib boring: siz tugmani sҳaklga olib ўtisҳingiz bilan, sҳu ondayoқ Unit1.x faylida Button2 obhektining ehloni paydo bўlisҳi kerak, OnClick voқeaning aniқlanisҳi esa usҳbu vorқeaning қayta isҳlovcҳisi bўlgan Button2Click metodining ehlon қilinisҳini generatsiyalaydi. sҳaklni loyiҳalasҳning va kodni avtomatik generatsiyalasҳ jarayonlarining mana sҳunday sinxronlasҳtirilisҳi C++ilovaning vizual isҳlanmasini ҳaқiқatan ҳam tezlasҳtiradi va sҳuning bilan birga dasturning dastlabki matni ustidan nazoratni tўla saқlab қoladi.

Ўzimizning birincҳi ilovamizni isҳlasҳda yana bir қadam қўyamiz - uni sҳehr bandini avtomatik tarzda generatsiyalasҳga majbur қilamiz. Buning ucҳun OnClick voқeasi isҳlanmasining funktsiyasini «Ягgi band» tugmasini bosib, mazmun bilan tўldirisҳga tўғri keladi.



4.3-rasm. Unit1.cpp faylida voқeaning yangi қayta isҳlatgicҳi

4.3-rasmda oddiy kod кўrsatilgan bўlib, u «Ягgi band» tugmasining navbatdagi bosilisҳiga javoban yangi bandning ketma-ket yettita satrini cҳiқarib beradi, bunda birincҳi va ucҳincҳi satrlar prev ўzgaruvcҳisidan olinadi. Bu ўzgaruvcҳi қiymatini «Natija» tugmasi voқeasining қayta isҳlatgicҳi ўzlasҳtirib olisҳi tufayli, bu қiymatni sҳakl sinfining foydalanuvcҳilar

ehlonlarida public seksiyasida қayta aniқlasxga tўғri keldi (2.5-rasmdagi ajratib кўrsatilgan satr). Bu isx ikkala tugma voкealarining қayta isxlatgicxlariga bu қiymatga kirisx ucxun imkon yaratisx maқsadida қilindi.

Sxehrni butunicxa кўrib cxiқisx imkonini yaratisx maқsadida rўyxat vertikal aylantirisx cxiziғiga ega bўldi.

C++Builder xar bir ilova bilan yasxirin nomlari қuyidagicxa bўlgan ucxta dastlabki faylni eslatisxini yodda saқlab қolisx kerak:

■ UnitI.cpp ilovangizning bajarilayotgan isxga tusxirisx kodini saқlaydi. Aynan sxu yerda siz foydalanuvcxining komponentalar obhektlariga tahsiri paytidagi dastur reaksiyasiga javob beradigan voкealarning қayta isxlatgicxlarini yozib қўyasiz.

■ UnitI.x barcxa obhektlar va ularning konstruktorlarining ehlonlariga ega. Voкealarni қayta isxlasx funktsiyalari ehlonlaridagi _fastcall kalit-sўzga ehtibor bering (C++Builder bu funktsiyalarni avtomatik tarzda generatsiya қiladi). _fastcall tufayliparametrlar stek orқali emas, balki markaziy protsessor registrlari orқali uzatiladi. Voкealarni қayta isxlatgicxlarning cxaқirisxlari tez-tez rўy berib turadi, SXuning ucxun stek xotirasidan parametrlarni tanlab olisxga sraflanadigan vaқtning tejalisxi ancxa sezilarli natijalarni beradi. C++Builder komplyatsiya қiladigan va tўplaydigan ilovalarning yuқori darajada tez xarakatlanisxining sabablaridan biri xam sxu yerda yasxiringan.

■ ProjectI.cpp ilovada mujassamlangan barcxa obhektlarga xizmat кўrsatadi. Xar қanday yangi sxakl, dasturiy modul yoki mahlumotlar moduli avtomatik tarzda loyixaviy faylga kiritiladi. Siz bosx menyu komandasi - View | Project Source yordamida yoki Loyixa Administratorining kontekstli menyusidan sxu nomdagi opsiyani tanlab olib, Kod Muxarriri darcxasida loyixaviy fayl dastlabki matnining mazmunini кўrib cxiқisxingiz mumkin. Xecx қacxon loyixaviy faylni қўlda taxrir қilmang!

Balki siz, birincxi ilova isxlanmasini tugatib, dastlabki fayllarni keyingi seans ucxun saқlab қolisxni xoxlarsiz. Buning ucxun қuyidagi xatti-xarakatlardan birini bajarisx kerak:

=>File | Save All komandasi ilovaning xamma dastlabki fayllarini saқlaydi.

File | Save komandasi dasturiy modulning ikkala komandasini saқlaydi, File | Save As komandasi esa ularga yangi nom berisxga ruxsat etadi.

File | Save Project As komandasi, fayllarning joriy nomlaridan foydalanib, loyixaviy fayl tarkibiy қismlarining xammasidagi ўzgarisxlarni saқlaydi.

Ikki yўnalisxli isxlanma texnologiyasi

Loyixaviy sxablonlarni қўllasx: [1(683-696), 2(55)]

Obhektlar Xazinasidagi tayyor loyixaviy sxablonlardan foydalanar ekansiz, siz dasturni isxlab cxiқisxda kўpcxilik ilovalar ucxun tipik bўlgan operatsiyalarni cxetlab ўtisx imkoniyatiga ega bўlasiz. Bu қanday operatsiyalar dersiz. Bular, masalan, menyu va tez cxaқirib olisx tugmalari panelini tuzisx, standart cxaқirisxlar dialogi va fayllarni tuzisxni tasxkil etisx bilan boғliқ operatsiyalardir. Siz sxablonga kiritgan ўzgartirisxlar xuddi sxu loyixaviy sxablondan bosxқa isxlab cxiқuvcxilarning foydalanisxiga tahsir қilmaydi.

Kўp xujjatli interfeys (MDI)rejimida isxlasx ucxun loyixaviy sxablon asosida ilova prototipini yaratisx ucxun қuyidagi xatti-xarakatlarni amalga osxiring:

Filtrlar muxarriri darcxasida TOpenDialog komponentasining Filter xususiyati қiymatlari ustunida matniy xujjatlar fayllarining nomlari va kengayisxlarini кўrsating.

Agar siz sxunday ilovani kompilyatsiya қilib, tўplay olsangiz, bu xolda u faқat MDI rejimida darcxalar bilan amallar bajara olisxini xamda, darcxani tanlab olingan fayllarning matniy mazmuni bilan tўldirmay turib, fayllarni ocxisx dialogini cxaқirib olisxni «bilisxi»ni кўrisxingiz mumkin. Яhni prototip nofunksional va amaliy jixatdan befoyda bўlib қoldi. Ilova қandaydir ongli xulқ-atvorga ega bўlisxi ucxun, қuyidagi xatti-xarakatlarni bajaring:

Bosx menyudan **View | Forms** komandasini bering va rўyxatdan MDICxild nomli sxaklni tanlab oling.

Memo таҳрир қилисҳ maydonining кўр satrli komponentasini Palitraning **Standart** кўсҳimcҳa ilovasidan sҳўhba sҳaklga olib ўting.

Lines xususiyatli satriy muҳarrirni tugmani bosisҳ bilan cҳaқirib olib, TMemo komponentasining таҳрир maydonini tozalang. Таҳрир maydoni sҳўhba darcҳasining ҳammasini egallasҳi ucҳun, Align xususiyatli alClient қiymatini ўrnating. Uzun matniy fayllarni кўrib cҳiқisҳni osonlasҳtirisҳ maқsadida, ScrolBars xususiyatli Ssbotҳ қiymatini ўrnating.

=>Bosҳ sҳaklni sicҳқoncҳa yordamida faollasҳtirib, yana unga қayting ҳamda ilovalar menyusidan **File|Open** komandasini tanlab oling.

=>Kod Muҳarriri darcҳasida kursor menyuning tegisҳli elementini tanlasҳda yuzaga keladigan OnClick voқeasining қayta isҳlatgicҳiga yўriқnomani kiritisҳ ucҳun kerakli pozitsiyani кўrsatib beradi. C++Builder TOpenDialog bosҳ sҳakli (komponentalar Palitrasining Dialogs қistirmasidan) komponentasi ucҳun usҳbu funksiyaning ehlonini avtomatik tarzda generatsiya қiladi.

4.4-rasmda sҳu voқeaning қayta isҳlatgicҳi bўlgan FileOpenItemClick funksiyasi tanasini tasҳkil қiluvcҳi zarur yўriқnomalar кўrsatilgan.

4.4-rasm. Main.cpp faylida sҳўhba darcҳa yuklanisҳining amalga osҳirilisҳi

Ajratib olingan yўriқnoma Cҳild sҳўhba darcҳasi Memo1 obhektining Lines satrlarini OpenDialog->FileName nomli ocҳiқ matniy faylning icҳidagilari bilan yuklatadi.

Bu faylning isҳlanisҳi ҳali tugallanganicҳa yўқ, albatta. Siz uni kompilyatsiya қilib, tўplab bўlsangiz, bir paytning ўzida bir necҳa darcҳalardagi matniy fayllarni таҳрир қila olasiz. Biroқ natija beruvcҳi fayllarning saқlanisҳi ҳozircҳa кўzda tutilgan emas - ўқuvcҳining ўzi File [Save va File | Save As menyulari komandalari ucҳun osongina kod yozib oladi.

Ilovani mantiқan eng sodda matniy muҳarrirga aylantirisҳ ucҳun bu Edit nomli bosҳ menyu elementining tusҳib қoluvcҳi rўyxatiga қidirisҳ va almasҳtirisҳ komandalarini қўsҳisҳ kerak.

Nazorat savollari:

1. Komponentlar Palitrasi ҳaқida tusҳuncҳa.
2. Sҳakllar Muҳarriri ҳaқida tusҳuncҳa,
3. Kod Muҳarriri ҳaқida tusҳuncҳa
4. Obhektlar Noziri ҳaқida tusҳuncҳa
5. Obhektlar Xazinasi ҳaқida tusҳuncҳa
6. Vizual loyiҳalasҳ ҳaқida tusҳuncҳa
7. Ikki yўnalisҳli isҳlanma texnologiyasi ҳaқida tusҳuncҳa
8. Loyiҳalasҳ sҳablonlari ҳaқida tusҳuncҳa
9. Two Way Tools texnologiyasi ҳaқida tusҳuncҳa.

Testlar:

1. **Komponentlar Palitrasi** nечta komponentadan iborat?

A. Takrorlanmaydigan 50 ta komponenta.

B. Takrorlanadigan 10 ta komponenta

V. Takrorlanmaydigan 100 ta komponenta

S. Ilovalarni қurisҳda taklif қilinadigan 100 dan ortiқ takroran қўllanadigan komponentlardan iborat.

2. **Sҳakllar Muҳarriri** ning vazifasi?

A. Dasturning foydalanuvcҳi bilan interfeysini yaratisҳ ucҳun mўljallangan.

B. Dasturning interfeysini yaratisҳ ucҳun mўljallangan.

C. Foydalanuvcҳi interfeysini yaratisҳ ucҳun mўljallangan.

3. **Kod Muҳarririning vazifasi?**

A. Dastur matniga isҳlov berisҳ funktsiyalarini yozisҳ ucҳun mўljallangan.

B. Dastur matnini, xususan, voқealarga isҳlov berisҳ funktsiyalarini yozisҳ ucҳun mўljallangan.

C. Dastur voқealarga isҳlov berisҳ funktsiyalarini yozisҳ ucҳun mўljallangan.

4. **Obhektlar Noziri** imkoniyatlari?

A. Қotib қolgan cҳigal dasturlasҳ zaruratisiz obhektlar xususiyatlarini vizual ўrnatisҳ imkonini beradi ҳamda sҳunday voқealarni ўz icҳiga oladiki, bu voқealarni ularning paydo bўlisҳiga nisbatan obhektlar reaksiyasi kodlari bilan boғlasҳ mumkin bўladi.

Маъруза -20. (4 с)

Мавзу: КАТТАЛИКЛАРНИНГ МУРАККАБ ТОИФАЛАРИ, КЎП ЎЛЧАМЛИ МАССИВЛАР.

Мақсад: ***Талабаларда катталикларнинг мураккаб тоифалари: массивлар, кўп ўлчамли массивлар билан ишлаш кўникмасини шаклллантириш.***

Калит сўзлар: ***Массив, сатр, устун, кўп ўлчамли массив.***

Режа:

1. **Кўп ўлчамли массивларни тавсифлаш, уларга ишлов бериш.**
2. **Массив элементларини киритиш ва чиқариш.**
3. **Кўп ўлчамли массивларни саралаш. Мисоллар**

С/С++ алгоритмик тилида фақат бир ўлчамли массивлар билан эмас, балки кўп ўлчамли массивлар билан ҳам ишлаш мумкин. Агар массив ўз навбатида яна массивдан иборат бўлса, демак икки ўлчамли массив, яъни матрица дейилади. Массивларнинг ўлчови компьютерда ишлашга тўсқинлик қилмайди, чунки улар хотирада чизиқли кетма-кет элементлар сифатида сақланади. Кўп ўлчамли массивларни худди 1 ўлчамли массивга ўхшаб эълон қилинади, фақат индекс тоифаси сифатида массивнинг сатрлари (қаторлари) ва устунлари тоифаси кўрсатилади ва улар алоҳида [ ][ ] қавсларда кўрсатилади. Масалан: А номли бутун сонлардан иборат 2 ўлчамли массив берилган бўлса ва сатрлар сони **n** та, устунлар сони **m** та бўлса: int a[n][m]

Икки улчовли массив элементларини киритиш-чиқариш, улар устида амаллар бажариш ичма-ич жойлашган параметрли цикллар ичида бўлади, яъни 1-цикл сатрлар учун, 2-цикл устунлар учун. Масалан:

```
for  ( i=0; i<=3; i++)
for ( j=0; j<=3; j++)
cin >>a[i][j];
```

Агар уларни клавиатурадан киритиш керак бўлса, яъни cin оператори ёрдамида ташкил этилса, қуйидагича киритилади:

```
1 2 3
4 5 6
7 8 9
```

Бундан ташқари массив элементларини эълон қилиш билан бирга уларни инициализация ҳам қилиш мумкин:

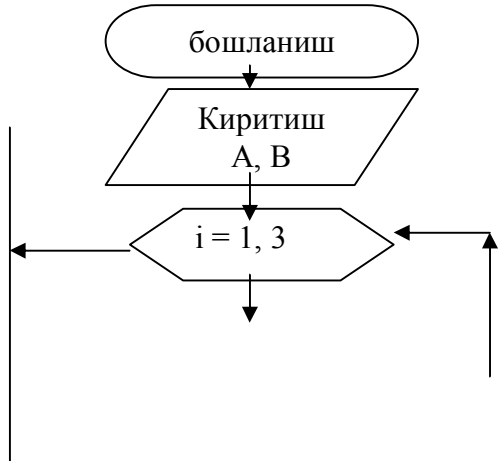
```
int   a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
```

Натижалар чиройли кўринишда бўлиши учун чиқариш операторини қуйидагича қилиб ташкил этиш керак:

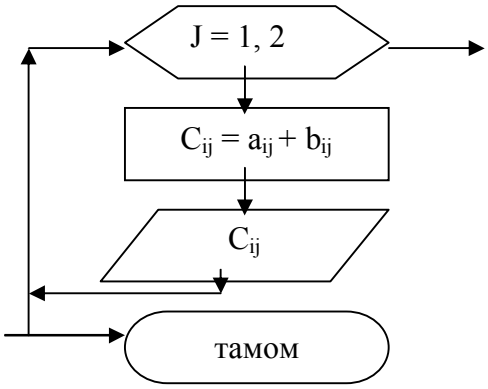
```
for  (int i=0; i<3; i++)
{ for (int j=0; j<3; j++)
 cout <<"a["<<i<<","<<j<<"]="<<a[i][j];
cout <<endl; }
getch ( );
}
```

1-мисол. А ва В матрицалари берилган. Қуйидаги формула орқали янги С матрицасини ҳосил қилинг: $C_{ij} = A_{ij} + B_{ij}$; бу ерда i=1,3; j=1,2;

$$A = \begin{Bmatrix} 24,3 & -4,15 \\ 0 & 18,4 \\ -8,8 & -15,75 \end{Bmatrix} \qquad B = \begin{pmatrix} 0,1 & -4,8 \\ 6,8 & 7,1 \\ -2,8 & 0,40 \end{pmatrix}$$




```
# include <iostream.h>
# include <conio.h>
void main ( )
{ float a[3][2]={{24.3,-4.15},{0,18.4},{-
8.8,-15.75}},
b[3][2]={{0.1,-4.8},{6.8,7.1},{-
2.8,0.40}};
float c[3][2];
```



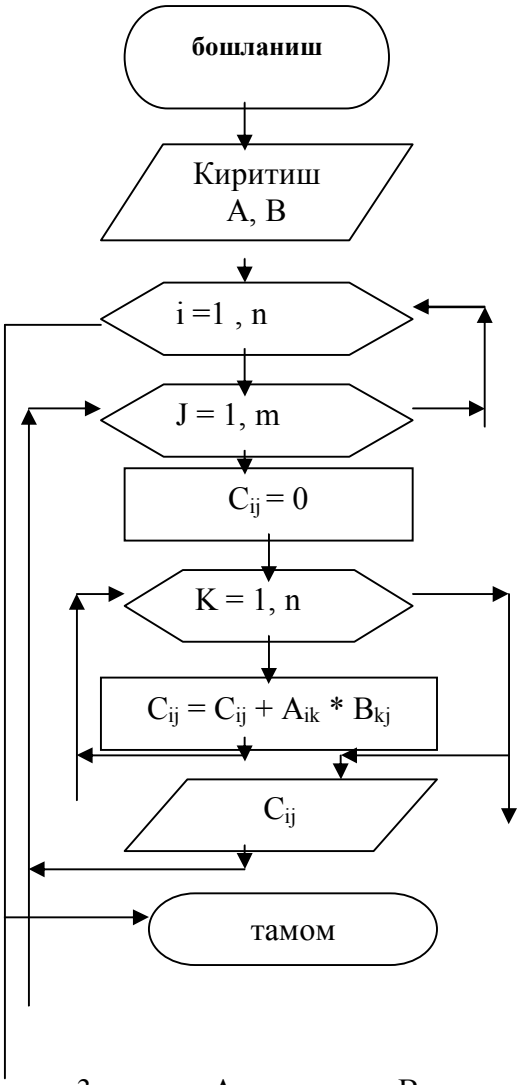
Массив элементларига сон қиймат беришда компьютер хотирасидаги тасодифий бутун сонлардан фойдаланиш ҳам мумкин. Бунинг учун стандарт кутубхонанинг rand ( ) функциясини ишга тушириш керак. rand ( ) функцияси ёрдамида 0 ÷ 32767 оралиқдаги иҳтиёрий сонларни олиш мумкин. Бу қийматлар умуман тасодифийдир. (псевдо – тасодифий дегани).

Агар дастур қайта-қайта ишлатилса, айни тасодифий қийматлар такрорланаверади. Уларни янги тасодифий қийматлар қилиш учун srand ( ) функциясини дастурда бир марта эълон қилиш керак. Дастур ишлаши жараёнида эҳтиёжга қараб rand ( ) функцияси чақирилаверади. Тасодифий қийматлар билан ишлаш учун <stdlib.h> файлини эълон қилиш зарур. srand ( ) функциясидаги қийматни автоматик равишда ўзгарадиган ҳолатга келтириш учун srand (time (NULL)) ёзиш маъқул, шунда компьютер ичидаги соатнинг қиймати time ( ) функцияси ёрдамида ўрнатилади ва srand га параметр сифатида берилади. NULL ёки 0 деб ёзилса, қиймат секундлар кўринишида берилади. Вақт билан ишлаш учун <time.h> ни эълон қилиш керак. Мисол:

```
# include <iostream.h>
# include <conio.h>
# include <stdlib.h>
# include <time.h>
void main ( )
{ srand ( time (0));
int a[5], b[5], i;
for (i = 0; i < 5;  i++)     a[i] = rand ( );
for (i = 0; i < 5;  i++)
{  b[i] = a[i] + 64;
cout << “b=”<<b[i]<<endl; }     getch ( ); }
```

Изоҳ: тасодифий сонлар ичида манфий сонларнинг ҳам қатнашишини иҳтиёр этсак,
a[i] = 1050 - rand ( ); ёки a[i] = rand ( )-1000; деб ёзиш ҳам мумкин.

2-мисол. 2та матрица берилган. Уларни ўзаро кўпайтириб янги матрица ҳосил қилинг. Бу ерда 1-матрицанинг устунлар сони 2-матрицанинг сатрлар сонига тенг бўлиши керак.




```
# include <iostream.h>
# include <conio.h>
# include <stdlib.h>
# include <time.h>
void main ( )
{
{ srand ( time (0));
int a[3][3], b[3][3],c[3][3],i,j,k;
for (i=0; i<3; i++)
for (j=0; j<3; j++)
a[i][j] = rand ( );
for (i=0; i<3; i++)
for (j=0; j<3; j++)
b[i][j] = rand ( );
for (i=0; i<3; i++)
{ for (j=0; j<3; j++)
{  c[i][j] = 0;
for (k=0; k<3; k++)
c[i][j] = c[i][j] + a[i][k]*b[k][j];
cout <<"c="<<c[i][j]<<"\t"; }
 cout << endl; }
getch ( );}
```

Изох:
i – 1-мартицанинг устунлари сони
j – 2-матрицанинг сатрлари сони
k – кўпайтиришлар сони

3-мисол. А матрицани В векторга

Изоҳ: матрицанинг сатрлари сони векторнинг сатрларига тенг бўлиши керак.

Масалан:
$$A = \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{Bmatrix} \quad B = \begin{Bmatrix} 1 \\ 3 \\ 6 \end{Bmatrix}$$

$C_1 = 1*1+2*3+3*6 = 25$
$C_2 = 4*1+5*3+6*6 = 55$
$C_3 = 7*1+8*3+9*6 = 85$

```
# include <iostream.h>
# include <conio.h>
void main ( )
{
int  a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}, b[3] = {1,3,6}, c[3], i, j;
for (i=0; i<3; i++)
{  c[i] = 0;
   for (j=0; j<3; j++)
```

```
   c[i] = c[i] + a[i][j] * b[j];
   cout <<"c="<<c[i]<<endl; }
getch ( );
}
```

4-мисол. Матрицани транспонерлаш алгоритмини тузинг. Матрицани транспонерлаш деб, устун ва сатр элементларини ўзаро ўрин алмаштиришга айтилади, яъни $A_{ij} = B_{ji}$

```
# include <iostream.h>
# include <conio.h>
void main ( )
{
int  a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}, b[3][3],
 i, j;
for ( i=0; i<3; i++)
{  for  ( j=0; j<3; j++)
 {   b[i][j] = a[j][i];
   cout <<"b["<<i<<","<<j<<"]="<<b[i][j];   }
cout << endl;   }
getch ( );
}
```

Берилган матрица:
1 2 3
4 5 6
7 8 9
Ҳосил бўлган матрица:
1 4 7
2 5 8
3 6 9

5-мисол. 3та қатор ва 4та устунга эга А матрица берилган. Ундаги энг кичик элементни ва унинг индексини топиш, ҳамда ўша қаторни массив шаклида чиқариш дастурини тузинг.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{
int  a[3][4] = {{1,2,3,4},{4,5,6,7},{7,8,9,10}}, i, j, k, h,  min;
int b[4];
min = a[0][0];
for (i=0; i<3; i++)
for (j=0; j<4; j++)
{ if ( a[i][j] > min)  { min = a[i][j];  k = i;  h = j; }  }
cout << "min="<<min<<"  k="<<k<<"    h="<<h<<endl;
for  ( j=0; j<4; j++)
{ b[j] = a[k][j];
  cout <<"b="<<b[j];  }
getch ( );
}
```

6-мисол. Саралаш масаласи. Массив элементларини ўсиб бориш тартибида саралаш алгоритмини тузинг. (пузырковый метод)

Аввал 1 ўлчовли массив элементларини саралашни кўриб ўтамиз.

```
# include <iostream.h>
# include <conio.h>
# include <stdlib.h>
# include <time.h>
void main ( )
{
srand (time (0));
```

```
float   a[10] , b;  int  i,  j;
for (i = 0; i<10; i ++)
a[i] = rand( ) /33.;
for( j = 0; j<10; j++)
for ( i = 0; i < 0;  i ++)
{
 if  (a[i] < a[i+1])
{  b = a[i];
a[i] = a[i+1];
a[i+1] = b; }
}
cout. precision (3);
for (i = 0; i  < 10; i++)
cout << a[i]<<endl;
getch ( );
}
```

Энди 2 ўлчамли массив элементларини саралашни кўрамиз:

```
# include <iostream.h>
# include <conio.h>
void main ( )
{
float   a[3][3] = {{.....},{.....},{.....}}, b;
int i, j, k;
for ( k=0; k<3; k++)
for ( i=0; i<3; i++)
for ( j=0; j<2; j++)
{  if  (a[i][j] > a[i][j+1] )
{  b = a[i][j];  a[i][j] = a[i][j+1];  a[i][j+1] = b;  }  }
for ( i=0; i<3; i++)
{  for ( j=0; j<3; j++)
cout <<"a="<<a[i][j];  cout << endl;  }
getch ( );
}
```

Юқоридаги дастур саралашни қатор бўйича олиб бориш учун. Агар саралашни устун бўйича қилиш керак бўлса, қуйидагича ёзиш керак бўлади:

```
for  ( i=0; i< 2; i++)
for  ( j=0; j<3; j++)
{ if  ( a[i,j] > a[i+1, j] ) {   b:= a[i, j];  a[i, j]:= a[i+1, j];  a[i+1, j]:= b;  }
```

Агар саралашни ўсиб бориш тартибида қилиш керак бўлса, if операторидаги солиштириш белгиси > бўлиши керак, агар камайиш тартибида керак бўлса, солиштириш белгиси < кўринишида бўлиши керак.

7-мисол. Матрицанинг изини ҳисоблаш дастурини тузинг. Матрицанинг изи деб бош диагонал элементларининг йиғиндисига айтилади. Шу дастурда тескари (қўшимча) диагонал элементларининг йиғиндисини ҳам ҳисоблашни кўриб ўтинг.

```
# include <iostream.h>
# include <conio.h>
```

```
void main ( )
{
float   a[3][3] = {{.....},{.....},{.....}}, s1=0, s2=0;
int i, j;
for  ( i=0; i<3; i++)
s1 = s1 + a[i][i];
for ( i=0; i<3; i++)
for ( j=0; j<3; j++)
if  ( i+j = = 2)    s2 = s2 + a[i][j];
cout <<"s1="<<s1<<"   s2="<< s2<< endl;
getch ( );  }
```

8-мисол. Ҳар бир ҳади $a_n = \frac{n!}{(2n)!}$ формуласи орқали ҳисобланадиган сатр йиғиндисини 0,0001 аниқликда ҳисоблаш дастурини тузинг.

```
# include <iostream.h>
# include <conio.h>
void  main ( )
{  int  n =1;  float  s1 = 0, s2 = 0;
float  p1 =1,  p2 = 1;
while  (s2 > 0.0001)
{ p1 = p1 * n;                    // p1*=n;
p2 = p2 * 2*n*(2*n-1);          // p2*= 2*n*(2*n-1);
s2 = p1 / p2;
s1 = s1 + s2;                     // s1+ = s2;
n ++; }
cout. precision (3);
cout << "s1="<<s1 <<" n=" << n << endl;
}
```

Назорат саволлари:

1. Икки ўлчамли массивлар.
2. Саралаш усуллари. Тўғри танлаш усули.
3. Максимал элемент жойлашган сатр ёки устунни ўчириш алгоритми.
4. Матрицани матрицага кўпайтириш алгоритми.

Тестлар:

1. Массив элементларидан фойдаланиш нима орқали бажарилади:
   A) FIFO йўналиш
   B) LIFO йўналиш
   C) нуқта операцияси
   D) элемент исми
   +E) элемент индекси

2. Массивнинг нотўғри таърифини кўрсатинг:
   A) int a[20];
   B) int a[]={1,2,3,4};
   C) int a[4]={1,2,3,4};
   +D) int a[2]={1,2,3,4};
   E) хаммаси тугри

Маъруза-21. (4 с)

Мавзу: ИЧКИ САРАЛАШ АЛГОРИТМЛАРИ. ШЕЛЛ, ШЕЙКЕР ВА ШАРСИМОН САРАЛАШ.

Мақсад: ***Талабаларда ички саралаш алгоритмлари, шелл, шейкер ва шарсимон, бинар ва қайта териш саралаш алгоритмларни қўллаш бўйича ишлаш кўникмаларини ҳосил қилиш.***

Калит сўзлар: ***саралаш, қўшиш, танлаш алгоритмлари*****.**

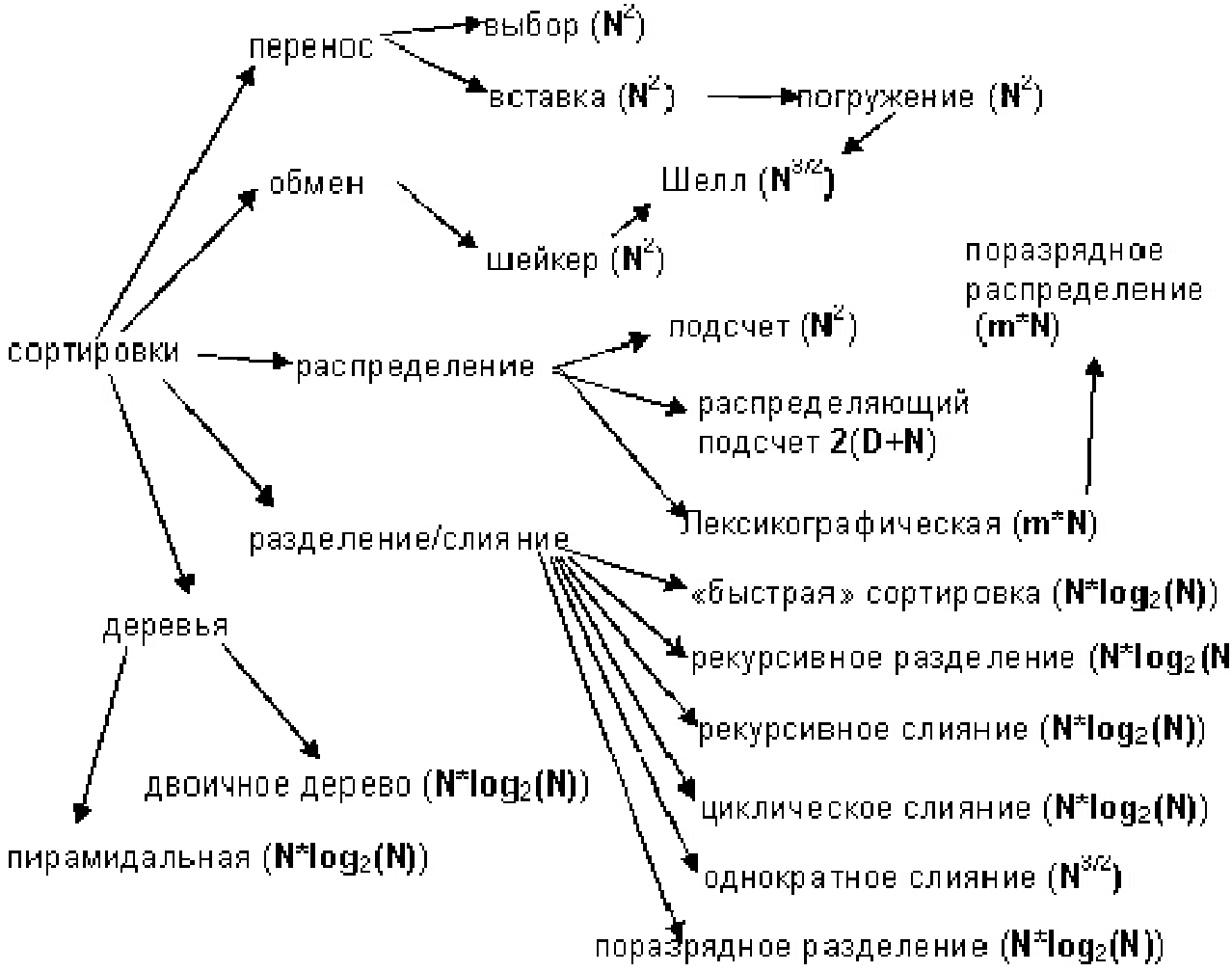
Режа:

1. **Кетма-кетликни саралаш.**
2. **Ички саралаш.**
3. **Бинар ва қайта териш усуллари.**
4. **Шелл, шейкер ва шарсимон саралаш алгоритмлари.**

Умуман олганда саралашнинг мақсади берилган объектлар тўпламини аниқ бир тартибда гуруҳлаб чиқиш жараёни таърифланади.

a_0, a_1... a_n кетма – кетлик берилган бўлсин.

Кетма – кетликнинг элементларини саралаш (масалан: $a_i <= a_{i+1}$, i = 0 дан n-1 гача) масаласи қўйилган бўлсин. Бу масалани ишлаш алгоритмини танлаганда қуйидагиларни баҳолаш зарур: *Саралаш вақти* – алгоритмни баҳолайдиган асасий параметр ҳисобланади. *Ҳотира* – алгоритм ишлаши учун кетадиган қўшимча ҳотира ҳажми. Бунда берилганлар ва дастур коди учун кетадиган ҳотира ҳажми ҳисобга олинмайди. *Турғунлик* – дастурни кетма – кетликнинг бошқа қийматларда ҳам турғун ишлаши тушунилади. Саралаш алгоритмлари классификацияси. Берилган кетма – кетликни саралашда кетма – кетликнинг характеристикаси мос равишда у ёки бу саралаш алгоритми олинади. Акс ҳолда алгоритмлар керакли натижани бермайди.



Расм 1. Саралаш алгоритми классификацияси.

1. “тез” саралаш алгоритми:

“Тез” саралаш алгоритми босқичлари:

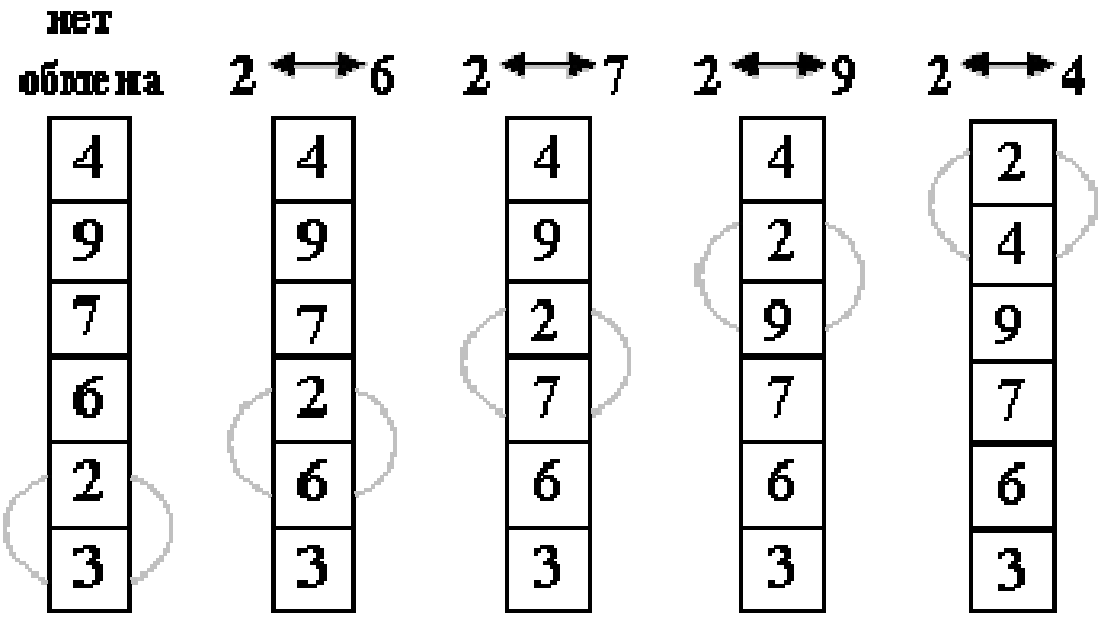
2. Массивнинг ўрта элементини танлаб оламиз.
3. Ўрта элемент чап томонига ўрта элементдан кичик элементларни жойлаштирамиз, ўрта элементнинг ўнг томонига эса ўрта элементдан катта элементларни жойлаштирамиз.

```
int i=quyi;
 int j=yuqori;
 int x=A[(quyi+yuqori)/2];
```

```
do {
   while(A[i]<x) ++i;
   while(A[j]>x) --j;
   if(i<=j){
      int temp=A[i];
      A[i]=A[j];
      A[j]=temp;
      i++; j--;
   }
} while(i<=j);
```

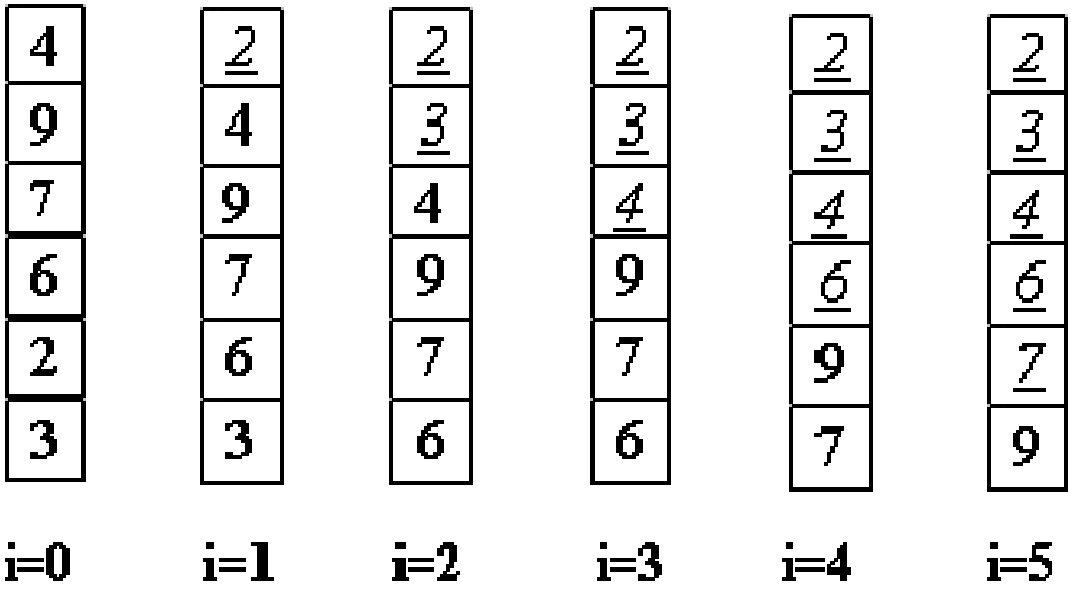
2. Шарсимон саралаш алгоритми

Массив элементларини тепадан пастга қараб саралаймиз. Бунда фақат жуфт элементлар $a_i <= a_{i+1}$ (i = 0 дан n-1 гача) шарт билан текширилади, агар шарт бажарилмаса улар ўзаро ўрин алмаштирилади.



Бу жараён оҳирги элемент қолгунча бажарилади. Натижада массив элементлари ўсиш тартибида сараланади.

Дастурдаги босқичлар:




```
long i, j,
float x, a[];
for( i=0; i < size; i++)
        for( j = size-1; j > i; j-- )
        {  if ( a[j-1] > a[j] )
        {       x=a[j-1]; a[j-1]=a[j]; a[j]=x;  } }
```

Саралашнинг мақсади кейинчалик, саралашган тўпламни қидирилаётган элементини топишдан иборат. Бу қарийб универсал, фундаментал жараён. Биз бу жараён билан ҳар куни учрашамиз – телефон дафтаридаги саралаш, китоблар сарлавҳасида, кутубхоналарда, луғатларда, почтада ва ҳ.к. Хатто ёш болалар ҳам ўз нарсаларини тартиблашга ўрганади. Саралашнинг жуда кўп усуллари мавжуд. Улар турли тўпламлар учун турлича бўлиши мумкин. Массивларни саралаш учун ишлатиладиган усул унга берилган хотирани ихчам ҳолда ишлатиш лозим. Бошқача қилиб айтганда, сараланаётган массив худди шу массивни ўзида амалга оширилиши лозим. Сараланаётган *a* массивни элементларини киритиб, унда бошқа бир d массивда сараланган ҳолда ташкил топган бизга ҳеч қандай қизиқиш уйғотмайди. тартибидаги саралашларни талаб этади. Биз қуйидаги саралаш бўйича бир нечта содда ва маълум усулларни қараймиз. Улар тўғри усуллар деб айтилади. Саралаш усуллари тўғрисида қуйидаги фикрларни билдириш мумкин:

1. Тўғри усуллар кўплаб саралашнинг асосий тамойилларининг характерини очиб бериши учун қулай.

2. Бу усулларни дастурлар осон тушунилади ва улар қисқа. Эслатиб ўтамиз, дастурнинг ўзи ҳам хотира эгаллайди.

3. Мураккаб усуллар кўп сондаги амалларни талаб қилади, лекин бу амалларнинг ўзлари етарлича мураккаб бўлганлари учун, кичик n ларда тез ва катта n ларда секин ишлайди. Аммо уларни катта n ларда ишлаб бўлмайди.

Битта массивни ўзида саралашни уларни мос аниқланган тамойиллари билан уч категорияга ажратиш мумкин:

1. Қўшиш орқали саралаш (by insertion);

2. Айириш орқали саралаш (by selection);

3. Алмашиш орқали саралаш (by exchange).

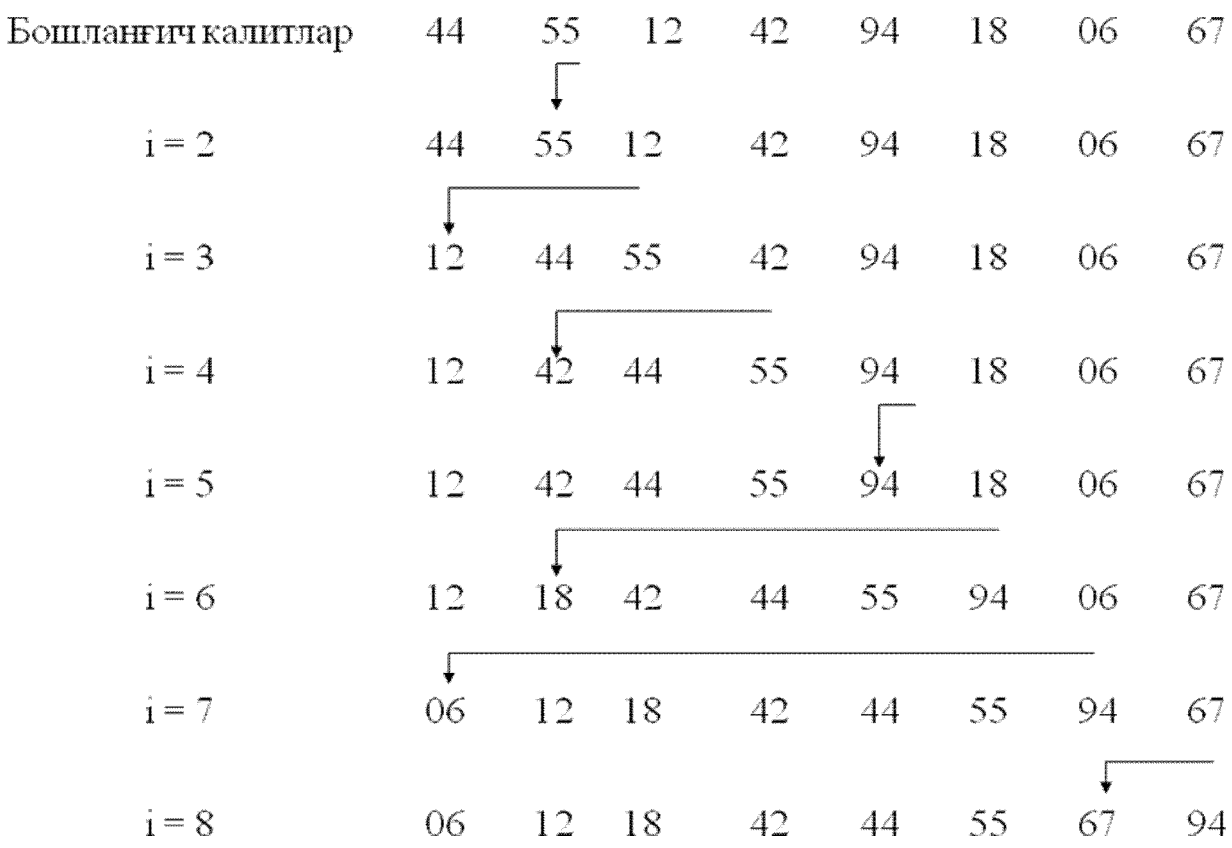
Тўғридан-тўғри қўшиш орқали саралаш

Бу усул карта ўйинида кўп қўлланилади.

Картанинг элементлари фикран тайёр ҳолдаги кетма-кетлик қисмларга бўлинади.

Ҳар қадамда i=2 дан бошлаб i та элемент кетма-кетликдан чиқарилади ва тайёр кетма-кетликка қўйилади. Бунда у ҳар доим керакли жойга қўйилади.

i нинг қиймати ҳар доим биттага ошириб борилади.



Тўғридан тўғри танлаш ёрдамида саралаш

- Энг кичик калитли элемент танланади.
- Уни биринчи элемент a_1 билан ўринлари алмаштирилади.

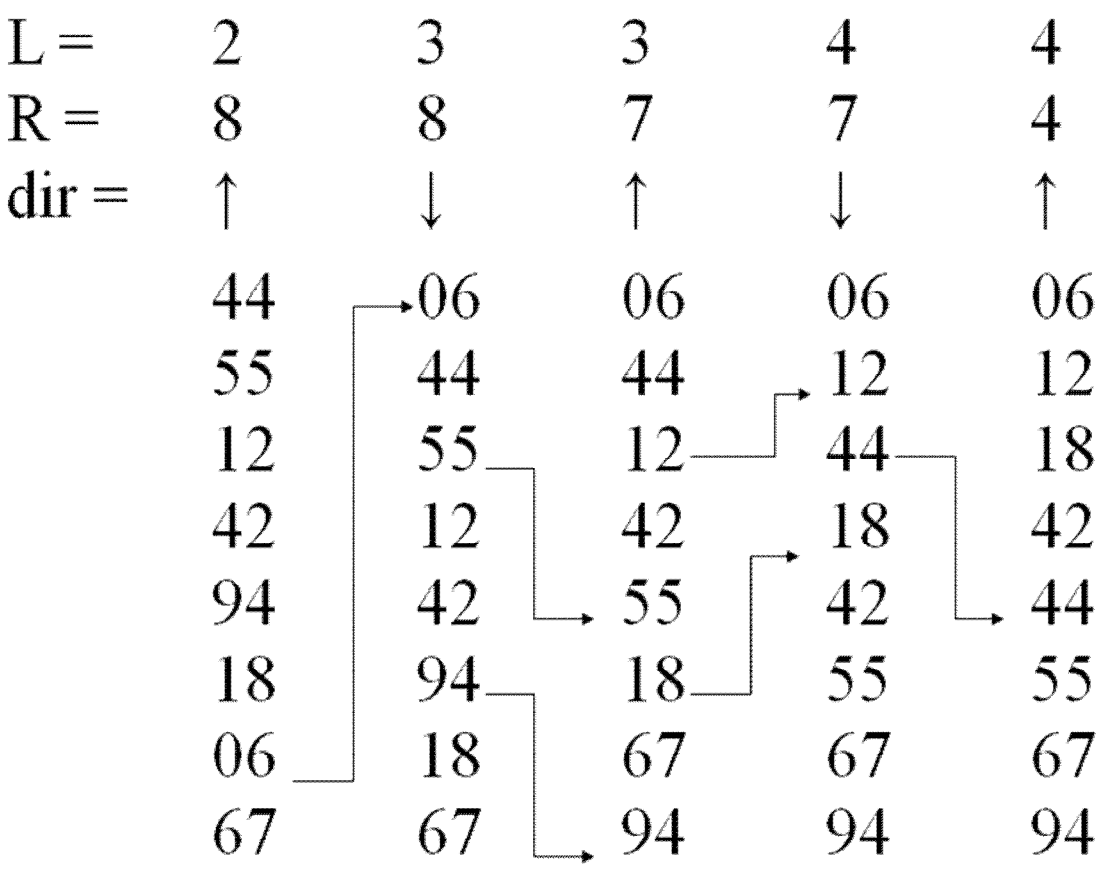
Сўнг бу жараён қолган n-1 элемент билан, сўнгра n-2 элемент билан ва ҳ.к. битта энг катта элемент қолмагунча давом эттирилади.

Бошланғич калитлар	44	55	12	42	94	18	06	67
i = 2	06	55	12	42	94	18	44	67
i = 3	06	12	55	42	94	18	44	67
i = 4	06	12	18	42	94	55	44	67
i = 5	06	12	18	42	94	55	44	67
i = 6	06	12	18	42	44	55	44	67
i = 7	06	12	18	42	44	55	94	67
i = 8	06	12	18	42	44	55	67	94

Пуфаксимон саралаш:

i =	1	2	3	4	5	6	7	8
	44	06	06	06	06	06	06	06
	55	44	12	12	12	12	12	12
	12	55	44	18	18	18	18	18
	42	12	55	44	42	42	42	42
	94	42	18	55	44	44	44	44
	18	94	42	42	55	55	55	55
	06	18	94	67	67	67	67	67
	67	67	67	94	94	94	94	94

ШЕЙКЕР саралаш усули



Назорат саволлари :

1. Танлаш йўли билан саралаш алгоритмлари.
2. Алмашув йўли билан саралаш алгоритмлари.
3. Ҳисоблаш йўли билан саралаш алгоритмлари.
4. Шелл усули билан саралаш алгоритми.
5. Шейкер усули билан саралаш алгоритми
6. Шарсимон саралаш усули.
7. Бинар ва қайта териш билан саралаш усуллари.

Тестлар :

1.Қуйидаги программа бўлаги натижасини аниқланг
C:=1; a:=3; b:=frac(7.9)-9;
Repeat a:=a+3;c:=c*a; until b<b*7; write(c);. . .
~A)6
~B)4
*~C)*7
~D)71

2. Қуйидаги операторлар бажарилиши натижасида s ўзгарувчининг қиймати нечага тенг?
S:=0;i:=1; while i>=1 begin s:=s+1/I; i:=i-1 end; . .. ;
~A)-1
~B)-0.5
~C)1.0
~D) 2
~E)7

3. Қуйидаги программа бўлаги натижасини аниқланг
C:=1; a:=3; b:=frac(7.9)-9;
WHILE b>b*7; begin a:=a+3;c:=c*a; end; write(c);. . .
~A)26
~B)4
~C)7
~D)6

Маъруза-22 (4 с)

Мавзу: МАССИВЛАР ВА ФУНКЦИЯЛАР. ҲАҚИҚИЙ ВА СОХТА ПАРАМЕТРЛАР.ФУНКЦИЯ ПРОТОТИПЛАРИ. ФУНКЦИЯ САРЛАВҲАСИ. ФУНКЦИЯ-ПРОЦЕДУРАЛАР. КЎРСАТКИЧЛАР.

Мақсад: ***Талабаларда массивлар ва функциялар, ҳақиқий ва сохта параметрлар, функциялар, расмий параметрлар, функция прототиплари, функция сарлавҳаси, функция-процедуралар, кўрсаткичлар, функциянинг аргументи сифатида массивларнинг ишлатилиши билан таништириш.***

Калит сўзлар: ***Функция, функция-процедура, расмий параметр, ҳақиқий параметр, тавсифлаш, прототип, параметрлар, объект, адрес, кўрсаткич.***

Режа:

1. **Функцияларни ташкил этиш.**
2. **Функцияларда қиймат айирбошлаш.**
3. **Рекурсияни ташкил этиш**
5. **Кўрсаткичларни ташкил этиш**
6. **Дастурда кўрсаткичлардан фойдаланиш.**
7. **Кўрсаткичлар билан ишлаш.**

Функция - бу мантиқан тўғри тугатилган дастурий қисмдир. Улар ёрдамида катта ва мураккаб ҳисоблашларни қайта - қайта ёзиш машаққатидан халос бўлинади ва дастур бажарилиши енгиллашади. Уни бир марта ташкил этиб ёзиб кўйилади ва унга дастурнинг исталган еридан мурожаат қилиш мумкин бўлади. Функцияни ташкил қилишда функциянинг тоифаси, унинг номи ва ташкил этувчи параметрлари ҳақида ахборот келтирилади. Бу параметрлар расмий параметрлар деб юритилади. Функция қуйидаги умумий тузилишга эга:

функция *тоифаси* функция *номи* (расмий параметрлар)
{ функция *танаси* ; }

Функция номи - идентификатор, иҳтиёрий лотинча сўз бўлиши мумкин.

Функция танасидан чиқиш **return** оператори орқали бўлади. Бу операторда функция натижаси бўлмиш ўзгарувчи ёки ҳисобланаётган ифода бўлиши ҳам мумкин. Функция аниқланаётганда (келтирилаётганда) ; белгиси кўйилмайди.

Функцияга мурожаат қилишдан олдин расмий параметрлар ўрнига келадиган ҳақиқий параметрлар аниқланиши лозим. Унга мурожаат қилиш қуйидагича бўлади:

ўзгарувчи = функция номи (ҳақиқий параметрлар);

Диккат! Расмий ва ҳақикий параметрлар сони, уларнинг тоифаси ва келиш ўрни албатта бир бирига мос бўлиши шарт! Расмий ва ҳақиқий параметрлар номлари бир хил бўлиши мумкин. Функцияни бош функция ичида эълон қилинганида ҳақиқий парметрлар номларини кўрсатмасдан, фақат уларнинг тоифаларини келтириш ҳам мумкин.

Функциялар main ( ) функциясидан аввал ҳам, кейин ҳам аниқланиши мумкин. Агар бош функциядан аввал аниқланган бўлса, уни main ( ) функцияси ичида алоҳида эълон қилиш шарт эмас, агар бош функциядан кейин келадиган бўлса, уни main ( ) функцияси ичида албатта эълон қилиш керак. Масалан: соннинг кубини ҳисоблаш учун функция ташкил этинг ва ундан фойдаланинг.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ int k, n, kw (int n); // kw - функция номи (иҳтиёрий)
cin>>n;                 // n - берилаётган сон
k=kw(n);                // kw функциясига мурожаат қилиняпти
cout << "k="<<k<<endl;
getch( );
}
int kw (int a) // функция аниқланаяпти. Бу ерда а расмий параметр
{   int c;        // локал ўзгарувчи
  c=a*a*a;     // ҳисоблаш
  return c;  }   // функцияга натижани қайтариш
```

Юқоридаги с локал ўзгарувчисини ишлатмасдан, тўғридан-тўғри

```
return  a*a*a;
```

деб ёзса ҳам бўлади.

Бу ерда функция бош функциядан кейин аниқланди, шунинг учун уни бош функция ичида эълон қилдик. Дастурни яна қуйидагича ёзса ҳам бўлади:

```
# include <iostream.h>
# include <conio.h>
int kw (int a)
{  return a*a*a;   }
void  main ( )
{ int k, n ;
cin>>n;
k=kw(n);
cout << "k="<<k<<endl;
getch( );
}
```

2-мисол. Иккита сондан энг каттасини топиш учун функция ташкил қилинг ва ундан фойдаланинг.

```
# include <iostream.h>
# include <conio.h>
void  main( )
{ float a=7, b=9, c, max(float , float );
c = max(a, b);
cout << "c="<<c<<endl;
getch( );
}
float max ( float x, float y)
{   if   (x > y)   return x;   else   return y; }
```

Функцияга яна қуйидагича ҳам мурожаат қилиш мумкин:
c = max(7.23, 9.145);
c = max(a, 9.145);

3-мисол. Учбурчак учларининг координаталари берилган. Шу координаталар ёрдамида учбурчак қурса бўладими? Агар мумкин бўлса шу учбурчакнинг юзини ҳисоблаш дастурини тузинг.
Демак, берилган координаталари ёрдамида учбурчак томонини кўриш функциясини, шу тамонлар асосида учбурчак кўриш мумкинми ёки йўқлигини ва унинг юзини ҳисоблаш функцияларини тузинг.

```
# include <iostream.h>
# include <math.h>
# include <conio.h>
// учбурчак томонини топиш функцияси
float  line (float x1, float x2, float y1, float y2)
{ (float) p = sqrt ((x1-x2)*(x1-x2)+ (y1-y2)*(y1-y2));
  return  p;  }
// учбурчак қуриб бўладими? функцияси
int  uch ( float a, float b, float c)
{  if ( a+b>c  && b+c>a  && c+a>b )  return  1;
else  return  0; }
//  учбурчакнинг юзини топиш функцияси
float  s (float a, float b, float c)
{ float  p, s ;
p = ( a + b + c ) / 2;  s = sqrt (p*(p-a)*(p-b)*(p-c));
return  s;  }
void main ( )
{ float  x1, x2, x3, y1, y2, y3, p1, p2, p3;  clrscr ( );
cin >> x1>> x2>> x3>> y1>> y2>> y3;
p1 = line (x1, x2, y1, y2);
p2 = line (x1, x3, y1, y3);
p3 = line (x2, x3, y2, y3);
t = uch (p1, p2, p3);
if ( t = = 1)
{  yuza = s ( p1, p2, p3);  cout << “yuza = ”<< yuza << endl;
else  cout <<”uchburchak qurib bo’lmaydi !!!”<< endl;
}  getch ( );  }
```

Бир функция ичида бошқа функция аниқланиши мумкин эмас, лекин функция ичида ўзини-ўзи чақириши мумкин. бундай ҳолатни рекурсия ҳолати дейилади. Рекурсия 2 хил бўлади: тўғри рекурсия ва билвосита рекурсия. Агар функция ўзини-ўзи чақирса, бу тўғри рекурсия дейилади. Тўғри рекурсияда функциянинг нусхаси чақирилади. Агарда функция бошқа бир функцияни чақирса ва у функция ўз навбатида 1-сини чақирса, у ҳолда билвосита рекурсия дейилади. Рекурсия 2 хил натижа билан якунланади: бирор натижа қайтаради ёки ҳеч качон тугалланмайди ва хатолик юз беради. Бундай ҳолатларда рекурсив функциялар учун рекурсияни тўхтатиш шартини бериш зарур, чунки рекурсияда хотира етишмаслиги хавфи бор.

4-мисол. F = n! ни хисоблаш учун функция ташкил этинг ва ундан фойдаланинг.

```
# include <iostream.h>
# include <conio.h>
void  main( )
```

```
{ int n, f, fac(int);
cout <<"sonni kiriting:";  cin >> n;
f = fac(n);  cout <<"sonning factoriali="<<f<<endl;
getch( );
}
int  fac(int i)
{  return  i <=1 ? 1 : i * fac( i - 1);  }
```

5-мисол. Фибоначчи сонларини хосил килиш дастурини тузинг. Фибоначчи сонлари қуйидагича топилади:

$f_0 = 1;\ f_1 = 1;\ f_2 = f_1 + f_0;$ ……

$f_n = f_{n-1} + f_{n-2};$

Рекурсив жараённи тўхтатиш шарти n < 2 деб олинади. Масалан 9-ўриндаги Фибоначчи сонини топиш керак.

```
# include <iostream.h>
void main ( )
{ int n, f ;  int  fib ( int );
cout  << “Nomerni kiriting =”;
cin >> n;
f = fib (n);
cout << “Fibonachi soni=”<< f << endl;
}
int  fib ( int n )
{   if  ( n < 2) return  1;  else  return ( fib (n-2) + fib (n-1));  }
```

6-мисол. $Z = \dfrac{a^5 + a^{-4}}{2a^n}$ ҳисоблаш дастури тузилсин. Бу ердаги даражани ҳисоблаш функция сифатида ташкил этилсин. $y = x^n$ ни функция деб ташкил этамиз, бу ерда x, n - расмий параметрлар

```
 # include <iostream.h>
float dar (float x, int n)
{  float y=1;
for (int  i=0; i<=n; i++)
   y = y*x;
   return  y; }
void  main( )
{
int n=3 ;   float  a, z;
cin>>a;
z = ( dar(a, 5) + dar(1/a, 4))/ (2* dar(a, n)) ;
 cout << "z="<<z<<endl;    }
```

Бир хил номдаги функцияларни ҳар хил тоифали ўзгарувчилар рўйхати билан мурожаат қилиб чақириш мумкин. Параметрлар сони кам ҳар хил бўлиши мумкин. Бундай ҳолатда параметрлар рўйхати ва қийматларга қараб компилятор ўзи қайси функцияни чақириш кераклигини аниқлайди. Масалан:

```
1. double  multi (float x)
   {return  x*x*x; }
2. double  multi (float x, float y)
    { return  x*y*y;  }
3.        double  multi (float x, float y, float z)
```

```
{  return  x*y*z;   }
```

ва қуйидаги мурожаатларнинг ҳаммаси тўғри ёзилган:

```
multi  (0.5);
multi (1.45, 7);
multi (10, 39, 54);
```

Функцияларнинг бир хил ном билан аталиши полиморфизм деб аталади. Поли – кўп, морфе – шакл деган маънони билдиради.

Масалан:

```
# include <iostream.h>
int  max (int a, int b)
{  if  (a>b)  return  a;  else  return  b;}
float  max (float a, float b)
{  if  (a>b)  return  a;  else  return  b;}
void  main ( )
{
int  a1, b1; float a2, b2;
cin >> a1>>b1;
cout << "butun max="<<max(a1, b1)<<endl;
cin >>a2>>b2;
cout <<"haqiqiy max= "<< max(a2, b2)<<endl;
}
```

1-мисол. В ва С векторларининг узунликларини ҳисоблаш дастурини тузинг. Вектор узунлигини ҳисоблаш учун функциядан фойдаланинг.

```
# include  < iostream. h>
float  vector (int d[ ], int k)
{  float  s=0;  int  i ;
for  (i=0;  i<k;  i++)
s = s + d[i] * d[i];
s = sqrt (s);
return  s;  }
void  main ( )
{
int  b[3] = {10,20,30},  c[4] = {14,15,16,17};
float   s1,  s2;
s1 = vector (b, 3);
s2 = vector (c, 4);
cout <<"s1=" << s1 <<"  s2=" << s2 << endl;
}
```

2-Мисол. Бутун сонли 4х5 матрицаси берилган. Аниқ бир сондан кичик бўлган хадларининг йиғиндисини топиш дастурини тузинг. Матрица элементларини киритиш (тасодифий сонлар ёрдамида), чиқариш ва йиғиндини ҳисоблаш жараёнларини функция сифатида ташкил этинг.

Функция ичида 2 ўлчовли массивлардан фойдаланилганда унинг 1-параметрини, яъни сатрлар сонини кўрсатмаслик ҳам мумкин, лекин 2-параметрини, яъни устунлар сонини албатта кўрсатиш шарт.

```
# include <iostream.h>
# include <conio.h>
```

```
# include <stdlib.h>
# include <time.h>
void  kir(int m[ ][5], int k);
void chiq(int m[ ][5], int k);
int  summa(int m[ ][5], int k, int x);
int i, j ;
void  main ( )
{ int matr[4][5]; int a, s;  int b[ ][3];
cout<<"sonni kiriting=";  cin>>a;
kir(matr, 4);   chiq(matr, 4);
s = summa(matr, 4, a);
cout<<"s="<<s<< endl;
getch( );    }

void  kir(int m[ ][5], int k)
{ srand(time(0));
for (i=0;i<k;i++)
for (j=0;j<5;j++)
m[i][j]=rand( ) - 200; }

void chiq(int m[ ][5], int k)
{ for (i=0;i<k;i++)
for (j=0;j<5;j++)
cout <<m[i][j]<<endl; }

int  summa(int m[ ][5], int k, int x)
{ int s1 = 0;
for (i=0; i<k; i++)
for (j=0; j<5; j++)
if (m[i][j] < x)  s1 = s1 + m[i][j];
return s1; }
```

Функцияларга мурожаат қилиш қуйидаги босқичлардан иборат бўлади:

1. Функция бажарилаётганда расмий параметрлар учун хотирадан жой ажратилади, яъни улар функциянинг ички параметрларига айлантирилади. Бунда параметр тоифаси float тоифаси double тоифасига, char ва shortint тоифалари int тоифасига айлантирилади.
2. Ҳақиқий параметрлар қийматлари қабул қилинади ёки ҳисобланади.
3. Ҳақиқий параметрлар расмий параметрлар учун ажратилган хотира қисмига ёзилади.
4. Функция танаси ички параметрлар ёрдамида бажарилади ва қиймат қайтариш жойига юборилади.
5. Функциядан чиқишда расмий параметрлар учун ажратилган хотира қисми бўшатилади.

Дастурдаги ҳар бир ўзгарувчи - объект ҳисобланади. Унинг номи ва қиймати бўлади. Ҳар бир объект хотирадан маълум жой эгаллайди ва улар маълум адресга эга бўлади. Дастурлашнинг маълум этапларида ўзгарувчининг ўзига эмас, балки унинг адресига мурожаат қилишга тўғри келади. Бундай пайтларда кўрсаткичлардан фойдаланилади. Кўрсаткич - бу бирор ўзгарувчининг адресини ўзида сақловчи ўзгарувчидир. Адрес - бу хотира ячейкасининг тартиб номери. Умуман олганда адрес 4 байт жой олади.

Кўрсаткичларни эълон қилишда унинг тоифаидан кейин * белгиси ва ўзгарувчи номи келтирилади.
Масалан: int a; char *d; int *p;
Кўрсаткичлар ҳам инициализация қилиниши мумкин. *p = 6; *d = '$';
cout <<*p билан cout <<p нинг фарқи бор. *p да шу ердаги қиймат чиқади, p нинг ўзини ёзсак, шу ернинг адрес номери чиқади. Масалан:
int a=10, b=5, e, *m;
e = a + b;
*m = e;
cout <<*m; деб ёзилса, m = 15 чиқади;
cout <<m; деб ёзилса, m = 0xfff2 чиқади, яъни шу ернинг адрес номери чиқади.
Уларнинг кийматларини "адресини ол!" (&) операцияси орқали амалга оширса ҳам бўлади, яъни m=&e; cout <<*m; деб ёзиш хам мумкин, у ҳолда m=15 чиқади, яъни e нинг адресидаги сон қиймат чиқади. Буни билвосита мурожаат оператори ҳам дейилади.
Масалан:
int h;
int *p=35;
h = &p;
Натижа: h = 35;
Адреси олиш амали (&) сон ёки ифодаларга қўлланилмайди, яъни &3.14 ва &(a+b) ёзувлари хатодир.
Кўрсаткичлар устида қуйидаги амалларни бажариш мумкин:
Кўрсаткичлар устида арифметик амаллар бажариш:
*p1-*p2; *p1+*p2
Кўрсаткичларга бирор сонни қўшиш ёки айириш:
*p1 - 25; *p1+3.45
Кўрсаткичларни биттага ошириш ёки камайтириш:
*p1++ ёки --*p1

Мисол.
```
# include <iostream.h>
# include <conio.h>
void main ( )
{
int x=10, y=10; int *xp, *yp;
*xp = &x; *yp = &y;
if (xp = = yp) cout <<"ular teng!"<<endl;
else cout <<"ular teng emas!"<<endl;
if (*xp = = *yp) cout <<"ular teng!"<<endl;
else cout <<"ular teng emas!"<<endl;
getch( ); }
```
1- if да улар тенг эмас чиқади, чунки уларнинг адрес қийматлари ҳар хил.
2- if да улар тенг чиқади, чунки уларнинг адресларидаги сон қийматлари бир хил

2-мисол.
```
# include <iostream.h>
void main ( )
{ int m = 5, *p = 0;
p = &m;
cout << m << endl;
cout << *p<<endl;
*p = 7;
```

Натижа:
m = 5
*p = 5
m = 7
*p = 7
m = 9
*p = 9

```
cout << m << endl;
cout << *p << endl;
m = 9;
cout << m << endl;
cout << *p << endl;
}
```

Баъзи масалаларда функция билан ишлаганда функция танаси ичида ҳақиқий параметрлар қийматларини ўзгартириш зарурияти туғилади, яъни натижа бир эмас, балки бир нечта ҳосил бўлиши керак бўлади. Бундай жараённи процедуралар ҳосил қилиш дейилади ва бу муаммони хал қилиш учун кўрсаткичлардан фойдаланилади. Функцияни аниқлаштиришда расмий параметрлар билан бир сатрда натижалар номлари ҳам кўрсатилади. Шунинг учун процедуралар билан ишлаганда функция тоифасини бўш (void) деб олиш мақсадга мувофиқдир. return шарт бўлмай қолади.

Масалан: томонлари берилган тўртбурчакнинг периметрини ва юзини ҳисоблаш учун функцияни қуйидагича аниқлаштирилади:

```
void tt (float a,float b, float* p, float* s)
{  *p = 2*(a+b);  *s = a*b;  }
```

Бу ерда float a, float b бериладиган катталик ҳисобланади, float* p, float* s лар эса натижалар ҳисобланади.

Бу функцияга мурожаат қилиш қуйидагича бўлади:

```
tt (2.3, 4, &p, &s); яъни  2*(a+b);  ва  a*b  нинг қийматлари адреслар бўйича олинади.
{Процедураларни бериладиган катталикларсиз ҳам ташкил этиш мумкин. Бунда функция танаси ичида ишлатилган барча катталиклар бериладиганлар ҳисобига ўтади. }
```

Ҳосил қилинган процедураларга мурожаат қилиш адрес (&) операцияси орқали амалга оширилади.

Масалан: $Z = \frac{a^5 + a^{-4}}{2a^n}$ ҳисоблаш дастури тузилсин. Бу ердаги даражани ҳисоблаш процедура сифатида ташкил этилсин. $y = x^n$ ни процедура деб ташкил этамиз, бу ерда x, n - расмий параметрлар

```
 # include <iostream.h>
void  dar1 (float x, int n, float *y)
{  *y=1;
for (int  i=0; i<=n; i++)
   *y = y*x;  }
void  main( )
{
int n=3 ;   float  a, z, z1, z2, z3;
cin>>a;
dar1(a, 5,&z1); dar2( 1/a, 4, &z2); dar1(a, n, &z3);
z = ( z1+z2) / z3 ;
 cout << "z="<<z<<endl;
}
```

2-мисол. 2та вектор берилган. Векторлар орасидаги бурчак қуйидаги формула билан ҳисобланади:

$$\varphi = \arccos \frac{(x, y)}{\sqrt{(x, x)(y, y)}}$$

бу ерда (x,y), (x,x), (y,y) - векторларнинг скаляр кўпайтмаси. Векторларнинг скаляр кўпайтмасини дастурда процедура сифатида ташкил этинг.

```
# include <iostream.h>
# include <math.h>
typedef  float  mm[4];
void  vec(mm a, mm b, float* s)
{*s=0;
  for (int i=0; i<4; i++)
  *s=*s+a[i]*b[i];     }
void main ( )
{float fi, f1, f2, f3;  int i;
mm   x, y;  // mm x={1,2,3,4}, y={5,6,7,8};
for  (i=0; i<4; i++)
cin >>x[i] >> y[i];
vec (x, y, &f1);  vec (x, x, &f2);  vec (y,y,&f3);
fi = f1 / sqrt( f2*f3);  fi = atan(sqrt (1-fi*fi) / fi);
cout << "fi="<<fi*180/3.1415<<endl;   // natija gradusda chiqadi
getch( );  }
```

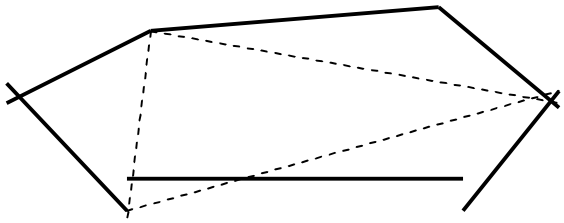
Процедураларни ташкил этишда кўрсаткичлардан ташқари яна иловалардан ҳам фойдаланилади. Бу усул янада қулай ҳисобланади. Унда (*) амалининг ўрнига тўғридан-тўғри адрес олиш (&) амали ишлатилади ва процедурага мурожаат қилиш осонлашади. Масалан:

```
void tt (float a,float b, float& p, float& s)
{  p = 2*(a+b);  s = a*b;  }
```

Бу функцияга мурожжат қилиш қуйидагича бўлади:

```
tt (2.3, 4, p, s);
```

Мисол. Бир фермернинг ер юзасини ва шу ерга тўлайдиган ер солиғини ҳисоблаш дастурини тузинг. Ер майдони қуйидаги кўринишда:



```
# include <iostream.h>
# include <math.h>
# include <conio.h>
# define pi 3.1415
void yuza (int a, int b, int al, float& c, float& s)
{ c = sqrt(a*a+b*b-2*a*b*cos(al*pi/180));
  s = a*b*sin(al*pi/180)/2;    }
void  main ( )
{ int  a1=10, b1=30, a2=40, b2=40, a3=30,b3=50,al1=85, al2=145, al3=125;
float c1, c2, c3, s1, s2, s3, s4, s, sol, p;
yuza (a1, b1, al1, c1, s1);  // yuza (10, 30, 85, c1, s1) deb yozsa ham bo'ladi
yuza (a2, b2, al2, c2, s2);
yuza (a3, b3, al3, c3, s3);
p = (c1+ c2 + c3) / 2;
```

```
s4 = sqrt ( p*(p - c1)*(p - c2)*(p - c3));
s = s1+s2+s3+s4;
s = s/100;  sol = s * 8560;  //       (som)
cout <<"er yuzasi="<< s << endl;
cout << "soliq="<<sol<<endl;
getch( );
}
```

2-мисол. Квадрат тенгламанинг ҳақикий ечимларини топиш дастурини тузинг.

```
# include <iostream . h>
# include <math . h>
int  kvad (float a, float b, float c, float &x1, float &x2)
{ float  d ;
d = b * b – 4*a*c;
if ( d < 0 )  return 0;
x1 = (-b + sqrt (d)) / (2*a);
x2 = (-b - sqrt (d)) / (2*a);
if ( x1 = = x2) return 1;  else  return 2;
}
void  main ( )
{ float a, b, c, x1, x2;   int  k;
cin >> a >> b >> c;
k = kvad (a, b, c, x1, x2);
switch ( k )
{
case 0 : cout << “echimi yo’q”<< endl;  break;
case  1 : cout << “x=”<< x1 << endl;  break;
case  2 : cout << “x1=”<< x1 << “  x2=” << x2 << endl;  break;
}
}
```

3-мисол. 4x4 ва 4x5 ўлчамли матрицалар берилган. Улардаги жуфт устунлари хадлари йғиндисини топиш дастурини тузинг. (натижа вектор кўринишида чиқади)

```
# include <iostream. h>
typedef   float  mmm[10][10];
typedef  float  mm[10];  int  i,  j;
void  nodir  (mmm  a,  int  n,  mm  b)
{
for  (j=0;  j<4;  j+=2)
{  b[j] = 0;
for  (i=0;  i<n;  i++)
b[j] = b[j] + a[i][j]; } }
void  main ( )
{  mmm  d = {{1,2,3,4},{1,2,3,4},{1,2,3,4},{1,2,3,4}};
mmm  d1 = {{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5}};
mm  c, c1;
nodir  (d, 4, 4, c);
nodir (d1, 4, 5, c1);
for  (i=0;  i<4;  i++)
{ cout <<”c=”<< c[i];
```

```
cout  << “c1=”<< c1[i] << endl;
}
```

Назорат саволлари:

1. Функцияларни ташкил этиш.
2. Функция номи, тавфсилотчиси, параметрлари.
3. Функцияга мурожаат.
4. Кўрсаткичларни эълон қилиш.
5. Кўрсаткичлар қийматини аниқлаш.
6. Кўрсаткичлар устида амаллар.
7. Вектор узунлигини ҳисобловчи функция ташкил этиш.
8. Функцияларни полиморфизм хусусияти.
9. Функцияларга мурожаат қилиш босқичлари.
10. return операторининг вазифаси.

Тестлар:

1. Қайси символ ёрдамида кўрсаткич аниқлайдиган манзил қийматини олиш мумкин?
   A) '&'
   B) '^'
   +C) '*'
   D) '%'
   E) '@'
2. Қайси символ ёрдамида ўзгарувчи адресини олиш мумкин?
   +A) '&'
   B) '^'
   C) '*'
   D) '%'
   E) '@'

3. Илова ҳақидаги нотўғри ибора аниқлансин
   A) илова –шартли ном
   +B) илова –ўзгарувчи
   C) илова қайта белгиланмайди
   D) илова нолга тенг бўлмайди
   E) илова учун адрес бўйича қиймат олиш автоматик бажарилади

Маъруза-23. (2 с)

Мавзу: РЕКУРСИЯЛАР, УЛАРНИ ТАШКИЛ ЭТИШ АСОСЛАРИ.

Мақсад: ***Талабаларга рекурсиялар, уларни ташкил этиш асослари устида амалларни бажариш асосларини ўргатиш.***

Калит сўзлар: Рекурсия, ўзига мурожаат, параметр, ҳақиқий, сохта.

Режа:

1. **Рекурсив функциялар.**
2. **Рекурсияда қиймат айирбошлаш..**

3.

C/C++ тилида функциялар.

Умуман олганда C/C++ тилида барча ёзувлар функциядан иборат деб қаралади. Функция бу маъносига кўра бажарилувчи модуль бўлиб ҳисобланади. Функцияни бошқа дастурлаш тилларида қисм дастур, процедура, процедура функция деб юритилади. C/C++ тилида функция стандарт формага асосан қуйидагияа ифодаланади:

функция тоифаси функция номи (сохта параметрлар рўйхати)
{ функция танаси }

Функция тоифаси исталган тоифа ёки void (бўш) тоифа бўлиши мумкин.
Функция номи исталган лотин ҳарфи ёки ҳарфларидан иборат бўлиб, хизматчи сўзлар билан бир хил бўлмаслиги лозим.
Сохта параметрлар рўйхатида ишлатиладиган параметрларга мос тоифали ўзгарувчилар тоифалари билан алоҳида-алоҳида келтирилади ёки бу соха бўш бўлиши ҳам мумкин. Эслатиб ўтиш лозимки, функция аниқлаштирилаётганда нуқта вергул белгиси қўйилмайди.
Функция танаси ўзининг фигурали қавсларига эга бўлиб, ўзида шу функцияни ташкил этувчи операторлар ёки операторлар блокини мужассамлаштиради. Бир функция танаси ичида бошқа функция аниқланиши мумкин эмас.
Функция танасидан чиқиш
return;
ёки
return ифода;
кўринишида бўлади. Агар функция ҳеч қандай қиймат қайтармайдиган, яъни тоифаси void бўлса, биринчи кўринишдаги чиқиш ишлатилади. Агар функция унинг тоифасига мос бирор қиймат қайтарадиган бўлса, иккинчи кўринишдаги чиқиш ишлатилади. Си тилида қуйидаги кўринишлар эквивалент хисобланади, лекин биринчи кўриниш кўпроқ ишлатилади:

```
double  f (int n, float x)
{
   функция танаси;
  }
```

```
double  f ( n, x)
int n;   float x;
{
   функция танаси;
   }
```

Дастурда функция ишлатиладиган бўлса, уни албатта эълон қилиш шарт. Функцияни эълон қилишда унинг тоифаси, номи ва қайтарадиган параметрлари ҳақида хабар берилади. Дастурда бирор функцияни олдиндан эълон қилмасдан туриб уни чақириш мумкин эмас. Функцияни асосий функция main( ) дан олдин ва кейин аниқланиши мумкин. Агар функция асосий функциядан олдин аниқланса, у аниқланиши билан бирга эълон қилинган деб ҳисобланади ва уни алоҳида main( ) ичида эълон қилиш шарт бўлмай қолади. Агар функция асосий функциядан кейин аниқланаётган бўлса, уни main( ) ичида албатта эълон қилиш шарт бўлади. Функцияни main( ) ичида эълон қилинадиган бўлса, унинг номи билан бирга ишлатиладиган параметрларининг фақатгина тоифалари кўрсатилиши ҳам мумкин. Масалан:

```
int  myFuncsion ( int, float);
double Area (float, float);
```

Функцияга мурожаат қилишдан унинг сохта параметрлари аниқланган бўлиши, яъни ҳақиқий параметрлар берилган бўлиши лозим. Функцияга мурожаат қилиш қуйидагича амалга оширилади:

функция_тоифаси функция_номи (ҳақиқий параметрлар рўйхати);

Масалан:

myFuncsion (78, 3.0+m);
Area (a, b);
g (6.4e-2, 5, 70);
Функциянинг сохта ва ҳақиқий параметрларининг тоифаси, параметрлар сони ва уларнинг келиш ўринлари албатта бир бирига мос келиши шарт!
Функцияга мурожаат қилинганидан сўнг аниқланган функция танаси бажарилади ва мос тоифали қиймат чақирилган жойга қайтиб келади.
Масалан: қуйидаги функция чақирилганида float тоифали натижа қайтаради:

```
float  ft (double x, int n)
{
 if (x < n) return  x;
 else  return  n;
 }
```

Функцияларга мурожаат қилинганида унинг узатиладиган параметрларига алоҳида эътибор бериш керак. Параметрларнинг узатилиши қуйидаги босқичлардан иборат:

- Функцияни ташкил этадиган сохта параметрлар учун хотирадан жой ажратилади. Агар параметрлар ҳақиқий тоифага эга бўлса, улар double тоифага, агар char ва short int тоифали бўлсалар, улар int тоифаси сифатида ташкил этиладилар. Агар параметрлар массив шаклида бўлсалар, массив бошига кўрсаткич қўйилади ва у функция танаси ичида массив параметр бўлиб хизмат қилади.
- Функция чақирилганидан керак бўладиган ифодалар ёки ҳақиқий параметрлар аниқланади ва улар сохта параметрлар учун ажратилган жойга ёзилади;
- Функция чақирилади ва аниқланган ҳақиқий параметрлар ёрдамида ҳисобланади. Бу ерда ҳам агар параметрлар ҳақиқий тоифага эга бўлса, улар double тоифага, агар char ва short int тоифали бўлсалар, улар int тоифаси сифатида ташкил этиладилар.
- Натижа функция чақирилган жойга қайтарилади.
- Функциядан чиқишда сохта параметрлар учун ажратилган хотира қисми бўшатилади.

Функцияга мурожаат қилиш ифодани ташкил этади, лекин агар функциянинг қайтарадиган қиймати бўш (void) бўлса, у ифода бўлмаслиги ҳам мумкин. Унда бундай функцияларга мурожаат қилиш қуйидагича бўлади:
функция номи (ҳақиқий параметрлар);
Масалан:

```
void print (int gg, int mm, int dd)
{
  cout<< ”\n yil:”<< gg;
  cout << “ \n oy: ” << mm;
  cout << “ \n kun: ” << dd;
}
```

кўринишидаги функцияга **print (1966, 11, 22);** деб мурожаат қилинса, қуйидаги натижа чиқади:

```
yil: 1966
oy: 11
kun: 22
```

Баъзан умуман ҳеч қандай параметрсиз функциялар ҳам ишлатилади. Масалан:

```
void  Real_Time (void)
{
  cout << “ Hozirgi vaqt: “ << TIME “(soat: min: sek)”;
}
```

функциясига **Real_Time ( );** деб мурожаат қилинса, экранга
Hozirgi vaqt: 14: 16: 25 (soat: min: sek) деган ахборот чиқади.

5.2. Функция параметрларида кўрсаткичлар.

...
1 –мисол.

```
# include <iostream.h>
void main( )
{
  float  x, y;
  void  aa( float *, float *);
  cout <<" x="<< x <<endl;  cin >> x;
  cout <<" y=" << y << endl;    cin >> y;
   aa ( &x, &y);
   cout << "\nNatija: \n";
  cout <<"x="<<x<<"y="<<y;
  }
void aa (float *b, float *c)
 { float e;
   e = *b;
  *b = *c;
  *c = e;
  }
```

Асосий дастурда x ва y ўзгарувчиларининг қийматлари клавиатурадан киритилади. Масалада иккита сон ўзаро ўрин алмашиши сўралмоқда. aa ( ) функциянинг сохта параметрлари сифатида float * тавсия этилган. aa () функциясига мурожаат қилинганида x ва y ларнинг сон қийматлари ҳақиқий параметрлар сифатида қабул қилинади. Бу дастурнинг ишлаши жараёнида қуйидаги натижалар олинади:

```
x=33.3 y=66.6  қийматлар киритилса
Natija:
x=66.600000  y=33.300000
```

2-мисол.
Учбурчакнинг периметри ва унинг юзасини ҳисоблаш учун дастур.

```
# include <iostream.h>
# include <math.h>
void main ( )
{
  float x, y, z, pp, ss;
  int tria (float, float, float, float *, float *);
  cout <<" x=";  cin >> x;
  cout <<" y="; cin >> y;
  cout <<" z="; cin >> z;
  if (tria (x, y, z, &pp, &ss)==1)
  cout <<"Uchburchak yuzasi="<< ss <<"  va perimetri="<< pp <<endl;
   else
  cout << "Ma'lumotlar noto'g'ri kiritilgan!" << endl;
}
```

```
 int tria ( float a, float b, float c, float *pp, float *ss)
{
  float e;
  if (a+b<=c || a+c<=b || b+c<=a) return 0;
  else
   { *pp = a+b+c;
     e=*pp/2;
    *s=sqrt(e*(e-a)*(e-b)*(e-c));
     return 1;
}
```

Дастурнинг бажарилишига мисол:

x=3
y=4
z=5
pp=12.00000
ss=6.00000

5.3. Функция параметрларида массивлар ва қаторлар

Массив параметрлар. Агар функциянинг параметри сифатида массивлар ишлатилса, функция ичида массив бошланишининг адреси узатилади. Бунга мисол тариқасида векторларнинг скаляр кўпайтмасини ҳисобловчи функция сарлавхасини кўриб чиқамиз:

```
float skalyar( int n, float a[ ], b[ ])   ёки
float skalyar (int n, float *a, float *b)
```

Бу ерда float a[ ] ва float *a ёзувлари параметр сифатида бир хил маънони англатади.

Қаторлар функция параметри сифатида.

Қаторлар функция параметри сифатида ишлатиладиган бўлса, char [ ] ёки char* тоифали кўрсаткичлардан иборат бўлади. Оддий массив параметридан фарқли ўлароқ, қаторнинг узунлигини кўрсатиш шарт эмас. Бунда \0 белгиси қатор охирини автоматик равишда кўрсатади. Мисол тариқасида қаторларни қайта ишловчи бир нечта дастурларни кўриб чиқамиз. Бу дастурларнинг ўхшаши стандарт кутубхоналарда сақланади ва уларни ишга тушириш учун string.h stdlib.h файлларини улаш керак бўлади.

1. Қатор тоифали ўзгарувчининг узунлигини аниқлаш учун функция:

```
int len(char e[ ])
{ int m;
  for(m=0; e[m]!='\0'; m++)
  return m;  }
```

Ёки бу дастурдаги массивни кўрсаткичлар орқали қуйидагича ифода этиш ҳам мумкин:

```
int len(char *s)
{ int m;
  for(m=0; *s!='\0'; m++)
  return m;  }
```

2. Қатор тоифали массив элементларини тескари ифода этиш учун функция:

```
void invert (char e[ ])
{  char s;  int  i, j, m;
   for (m=0; e[m]!='\0'; m++)
   for(j=0, j=m-i; i<j; i++, j- -)
  { s=e[i];  e[i] = e[j]; e[j] =s; }  }
```

Дастурдаги void тоифадан маълумки, бу функция ҳеч қандай қиймат қайтармайди.

Масалан:

```
# include <iostream.h>
void main( )
{ char ct[ ] =”0123456789”;
 void invert (char [ ]);
  invert(ct);  cout << ct;  }
```

Натижа: 9876543210

3. Қаторнинг чап томонидан киритилган бошқа қаторни қидириш функцияси:

```
int  index(char *ct1, char *ct2)
{ int i, j, m1, m2;
  for(m1=0; ct1[m1]!=’\0’; m1++)
  for(m2=0; ct2[m2]!=’\0’; m2++)
  if (m2>m1) return -1;
  for(i=0; i<m1-m2; i++)
  {  for(j=0; j<m2; j++)
    if (ct2[j] !=ct1[i+j] ) break;
    if (j==m2) return 1; }
    return  -1; }
```

Функциянинг ишлашига мисол:

```
# include <iostream.h>
void main ( )
{ char  c1[ ] =”og`irlik_yig`indisi”;
  int index(char[ ], char[ ]);
  char c2[ ] = “non”;
  char c3[ ] = “olma”;
  cout<< index(c1,c2);
  cout<< index(c1,c3);
 }
```

Назорат саволлари:

11. Функцияларни ташкил этиш.
12. Функция номи, тавфсилотчиси, параметрлари.
13. Функцияга мурожаат.
14. Кўрсаткичларни эълон қилиш.
15. Кўрсаткичлар қийматини аниқлаш.
16. Кўрсаткичлар устида амаллар.
17. Вектор узунлигини ҳисобловчи функция ташкил этиш.
18. Функцияларни полиморфизм хусусияти.
19. Функцияларга мурожаат қилиш босқичлари.
20. return операторининг вазифаси.

Тестлар:

2. Қайси символ ёрдамида кўрсаткич аниқлайдиган манзил қийматини олиш мумкин?
   A) '&'
   B) '^'
   +C) '*'
   D) '%'
   E) '@'
2. Қайси символ ёрдамида ўзгарувчи адресини олиш мумкин?
   +A) '&'
   B) '^'

C) '*'
D) '%'
E) '@'

3. Илова ҳақидаги нотўғри ибора аниқлансин
A) илова –шартли ном
+B) илова –ўзгарувчи
C) илова қайта белгиланмайди
D) илова нолга тенг бўлмайди
E) илова учун адрес бўйича қиймат олиш автоматик бажарилади

Маъруза-24. (2 с)

Мавзу: ҚИДИРИШ АЛГОРИТМИ: БИНАР, КНУТТ, ҚАЙТА ТЕРИШ, МОРИС-ПРАТТ УСУЛЛАРИ.

Мақсад: ***Талабаларга қидириш алгоритми: бинар, Кнутт, қайта териш, Морис-Пратт усуллари асосларини ўргатиш.***

Калит сўзлар: ***қидириш, бинар, қайта териш.***

Режа:

1. **Элементларга қидириш орқали ишлов бериш.**
2. **Кнутт алгоритми.**
3. **Қайта териш алгоритми.**
4. **Морис-Пратт усули.**

1.12. Қидирув

Дастурлашда жуда кўп маротаба дуч келадиган ҳаракатлардан бири бу – қидирувдир. У ўзи мукаммал масала бўлиб сифатида намоён бўлиб, унда турли тузилмалар(структуравий берилганлар)ни синовдан ўтказиш мумкин. Бир нечта асосий “бу мавзунинг вариациялари” мавжуд ва улар учун турли хил кўринишдаги алгоритмлар яратилган. Кейинги муҳокамаларда, биз шунга ўхшаш принципиал йўл қўйишларни қараб ўтамиз: фиксирланган берилган элементни берилганлар синфи ичидан излаш зарур. N та элементдан ташкил топган тўплам берилган деб ҳисоблаймиз, ва қуйидаги массив кўринишида келтириб ўтамиз:
a: ARRAY [0….N-1] OF item;

Одатда item типи (тури) (ключ) калит рўлини бажарадиган айрим поле (майдонлар)лардан ёзиб олишларни ифодалайди. Қидирувга қўйилган масала шундан иборат бўладики, калит “қидирув аргументи” *x* га деб олинади. *a[i].key =x* шартни қаноатлантирувчи і индексли олинган натижа, бошқа майдонларга ўша қатнашган элемент мурожаат қилишини таъминлайди. Бизни қизиқтираётган масала биринчи навбатда, топиш берилганлар эмас, қидирув жараёнининг ўзи қизиқтиради, item типи фақат у калит вазифасини ўзига олади (*key*), деб ҳисоблаймиз.

1.12.1. Чизиқли қидирув

Агар қидирилаётган “берилган”лар ҳақида ҳар қандай қўшимча маълумотлар йўқ бўлса, унда излаш оддий ёндошув – содда кетма-кет массивини қадамларнинг катталашиб борилиши қадамба қадам у ёки бу қисмини ... исталган элементини кўриб

чиқишдан иборат бўлади. Бундай усул *чизиқли излаш* дейилади. Қидирувни якунлаш шарти қуйидагича:

1. Элемент топилди, яъни $a_i = x$.

2. Барча массивлар кўриб чиқилди ва учтма-уст тушишлар кузатилмади. Бу эса бизга чизиқли алгоритмни беради:

$$i := 0 \qquad (1.36)$$

WHILE (i< N) & (a[i] # x) DO I := i+1 END;

Мантиқий ифодаларда элементлар тартиби муҳим аҳамиятга эгалигига эътибор беринг. Цикл (такрорланиш) инварианти, яъни шартнинг I индекснинг ҳар бир ортиб бориши олдидан бажарилиши, қуйидиги кўринишга эга бўлади:

$$(0 \le i < N) \ \& \ (Ak: 0 \le k < i: a_k \ne x). \qquad (1.37)$$

i нинг қанча кам қийматларида k барча қийматлари учун устма-уст тушишлар кузатилмади. Бу ерда бундай асосдан, қидирув якунланади фақат шу ҳолатда такрорланиш(цикл) сарлавҳасида фақат шартнинг мураккаблиги ҳолатида қидирув якунланади, ундан кейин узил-кесил шартни чиқариш мумкин:

$((i = N) OR (a_i = x)) \& (Ak: 0 \le ki: a_k \ne x)$.

Бу шарт нафақат қутилган натижани тасдиқлайди, балки бундан келиб чиқадики, агар элемент топилган бўлса, у ҳолда индекснинг минимал имконияти билан топилган, яъни бундай элементлардан биринчиси. $i = N$ тенглик, устма-уст тушишлар мавжуд эмаслигидан далолат беради.

Ҳақиқатдан ҳам, цикл (такрорланиш) тугаши кафолатланган, модомики, ҳар бир қадамда I нинг қиймати ортиб боради, ва у албатта N чегарасигача чекли қадамлар сони етади, ҳақиқатдан агар устма-уст тушишлар кузатилмаса, бу N қадамдан кейин бажарилиши кузатилади.

Албатта, ҳар бир қадамда индекснинг ортиб бориши ва мантиқий ифодаларни ҳисоблаш талаб этилади. Бу ишни соддалаштириш ва шу йўл билан қидирувни жадаллаштириш мумкинми? Ягона имконият - мантиқий ифоданинг ўзини соддалаштиришга ҳаракат қилиш, у иккита ҳаддан иборат эканлиги маълум. Бизнинг мураккаб масаламизга эквивалент бўлган содда шартни ифодалаш - оддий ечимга йўналтирилган ягона имконият эканлиги келиб чиқади. Агар биз устма-уст тушишлар доимо юз беришини таъминласак, буни бажариш мумкин.

Бунинг учун массив охирида қўшимча элемент x қиймати билан эълон қилинади. Бундай ёрдамчи элементни *барьер тўсиқ* деб, у массив чегарасидан ўтиб кетишидан сақлайди. Энди массив қуйидагича ифодаланади:

A : ARRAY [0…N] OF INTEGER

Натижада қидирувнинг чизиқли алгоритми тўсиқ билан қуйидаги кўринишда ёзилади:

```
a [N] : = x;  I :=0;
WHILE  a[i]  # x DO  i : =i+1  END ;
```
(1.38)

Натижавий шарт, киритилган шу инвариантларнинг ўзидан, ва дастлаб:

$(a_i = x) \& (Ak: 0 \le k < i: a_k \ne x)$.

Шак-шубхасиз, i=N тенглик шундан гувоҳлик берадики, устма-уст тушишлар (агар тўсиқ билан устма-уст тушишини ҳисобга олмаганда) бўлмаган.

1.12.2. Иккига бўлиб қидирмок (иккиламчи қидирув)

Шуниси аниққи, агар қидирилаётган маълумотлар ичида биронтаси ҳақида маълумот бўлмаса, қидирувни тезлаштирувчи бошқа усуллар мавжуд эмас. Яхши маълумки, агар маълумотлар тартибли бўлса, қидирувимизни янада осонлаштириш мумкин. Тасаввур қилиб кўринга, агар ишлатаётган телефон маълумотномасида барча фамилиялар бетартиб жойлашган бўлса. Бу гўёки, мутлақо фойдасиз жисм! Шунинг учун биз алгоритмдан, массив а тартибга солинган дея, фойдаланамиз, яъни қуйидаги шартга бажарилади:

$$Ak: 1 \le k < N : a_{k-1} \le a_k . \qquad (1.39)$$

Асосий ғоя – тасодифий равишда биронта элементни танлаш, масалан *ам*, ва уни қидирилаётган *х* аргумент билан солиштирамиз, шунда биз индекси *м* га тенг ёки ундан кичик бўлган барча элементлар кейинги қидирув жараёнидан мустасно қилинади: агарда у *х* дан катта белса, индекси м га тенг ва катта бўлган барча элементлар чиқариб ташланади, деган хулосага келамиз, бу ўз навбатида бизни (иккига бўлиб қидирмок) алгоритмга олиб келади. Бу ердаги L ва R икки ўзгарувчи мос равишда *а* массиви секциясининг чап ва ўнг тарафларини билдиради, қаердаки керакли элементни учратишимиз мумкин.

```
L :=0;   R :=N-1;   found  :=FALSE;
WHILE  ( l < = R ) & ~ found DO
m :=любое значение между L и R:
IF a[m]=x THEN found := TRUE                                    (1.40)
ELSIF   a[m] < x THEN L :=m+1
ELSE  R  :=m-1
END
END;
```

Циклнинг бошқача варианти, яни тактларнинг ҳар бир қадамида бажариладиган шарти:

$$(L \le R) \ \& \ (Ak: 0 \le k < L : ak < x) \qquad (1.41)$$
$$\& \ (Ak: R \le k < L : ak < x)$$

Шундан қуйидаги натижага келамиз:

$$found \ OR \ ((L > R) \ \& \ (Ak: 0 \le k < L: ak < x)$$
$$\& \ (Ak : R \le k < L: ak < x)$$

бундан келиб чиқадики:

$$(am = x) \ OR \ (Ak : 0 \le k < N : ak \ne x).$$

m ни танлашнинг асосий хусусияти шундаки, унинг алгоритмга боғлиқ бўламаслигидир. Бироқ, унинг эффективлигига m нинг танланиши таъсир қилади. Бундан маълум бўладики, бизнинг вазифамиз бу-танлашнинг келгуси ҳар бир босқичида, натижа қандай бўлишидан қатъий назар, иложи борича кўпроқ элементни чиқариб ташлаш ҳисобланади. Ўртача элементни танлаш эса энг тўғри ечим бўлади, чунки бунда барибир массивнинг ярми чиқариб ташланади. Натижада, максимал солиштиришлар сони log N энг яқин бир бутун яхлитлашгача бўлади. Шундай қилиб, келтирилган алгоритм, чизиқли излаш дан устун туради, чунки у ерда кутилаётган солиштиришлар сони –N/2 дир.

$$(Ak: 0 \le k < L : ak < x) \ \& \ (Ak: R \le k < N : ak \ge x), \qquad (1.42)$$

```
L : =0;   R :=N;
WHILE L<R DO
  m:=(L+R) DIV  2:                                              (1.43)
IF a[m]< x THEN  L:=m+1   ELSE   R  :=m  END
END:
```

Шартли операторларнининг бошланғич қисмларининг ўринларини ўзгартириш билан эффективликни бироз яхшилаш мумкин. Тенгликни текширишни 2- навбатда қилиш мумкин., чунки у бир маротабагина қаноатлантирилади ва ишнинг якунланишига олиб келади. Лекин, қуйидагича фикрлаш ҳақиқатга яқинроқдир: чизиқли излашда якунлаш шартларини янада соддалаштирадиган ечимни топиш мукинми. Шунда биз ҳақиқатдан ҳам тўғри келиш фиксациясида излашни якунлашмоқчи бўлганимизда, шундай тез алгоритмни топамиз. Биринчи кўринишда бу жуда ғалати туюлиши мумкин, лекин синчковлик билан кузатганда, эффективликнинг ҳар бир қадамидаги ютуғи бир неча қўшимча элементларни солиштиришдаги йўқотишлардан устундир. Эслатиб ўтамиз, қадамлар сони энг ёмон вазиятда ҳам- log N дир. Тез алгоритм қуйидаги инвариантдга асосланади:

(Ak:0≤k<L:ak<x)&(Ak:R≤k<N:ak≥x),

Излашларнинг қабул қилиши иккала секция массивни тўлиқ “босмагунча” давом этади..

```
L : =0;    R :=N;
WHILE L<R DO
  m:=(L+R) DIV  2:                                                    (1.43)
IF a[m]< x THEN  L:=m+1    ELSE   R  :=m  END
END:
```

Якунлашнинг шартлари L ≥ R дир, аммо бунга эришиш мумкинми? Буни исботлаш учун ҳар бир қадамда R-Lнинг турлилиги барча шароитларида ҳам камайиб боришини кўрсатишимиз зарур. Ҳар бир қадамнинг бошида L<R. Мнинг ўртача арифметигига L≤m<R шарти ўринлидир. Демак, турлилик ҳақиқатдан ҳам камайиб боради, ахир ёки L, m+1 да ўз ҳусусиятини юқорилатади, ёки R, m да ҳусуиятини камайтиради. L=R бўлганда циклнинг такрорланиши тугайди. Лекин, бизнинг инвариант ва L=R шарти бир-бирига мос келганидан далолат эмасдир. Албатта, R=N да ҳеч қандай мосликлар йўқ. Бошқа ҳолларда, биз a[R] элементи тенглаштиришларда ҳеч қачон иштирок этмаслигини ҳисобга олишимиз керак. Кейин эса, a[R] =x тенглигига қўшимча текширув зарур бўлади. Биринчи ечимимиздан фарқли ўлароқ (1.40), келтирилган алгоритм, чизиқли излашда бўланлиги каби, энг кичик тўғри келувчи индексли элементни топади.

$$(x = y) = (Aj : 0 \le j < M : x_j = y_j),$$

$$(x < y) = Ei : 0 \le i < N : ((Aj : 0 \le j < i : x_j) \,\&\, (x_i < y_i)).$$

1.12.3. Жадвалда излаш

Массивда излашни баъзида жадвалда излаш, деб ҳам юритилади, айниқса сонлар ва белгилар массивининг таркибий объектининг калити бўлганда кўпроқ айтилади. Белгилар массивини сатр ёки сўз деб юритилганда, охиргиси кўпроқ учрайди. Сатрли *тип* қуйидагича ифодаланади.:

String = ARRAY [0. .M-1] OF CHAR: (1.44)

Сатрларга тартибнинг муносабати ҳам аниқлаштирилади.

$$(x = y) = (Aj : 0 \le j < M : x_j = y_j),$$

$$(x < y) = Ei : 0 \le i < N : ((Aj : 0 \le j < i : x_j) \,\&\, (x_i < y_i)).$$

Мос келиш фактини ўрнатиш учун, биз тенглаштирилувчи сатрларнинг ҳамма белгилари бир-бирига тенг эканлигига ишонч ҳосил қилишимиз керак. Шунинг учун, таркибий операндларининг тенглаштирилиши, унинг мос келмайдиган қисмларининг изланишига олиб боради, яъни “тенгсизликни” излашга олиб боради. Агар тенг эмас қисмлар бўлмаса, унда тенглик ҳақида гапириш мумкин. Масалан, сўз ўлчами жуда кичик, яъни 30 дан кичик. Бундай ҳолда, биз чизиқли излашни қўллаймиз ва шундай қиламиз.

Кўпчилик тажрибавий қўлланмаларда сатрла ўзгарувчан ўлчамлардан иборат эканлигидан келиб чиқиш керак. Бунда таҳмин қилинишича, ўлчам ҳар бир алоҳида сатрда кўрсатилади.. агарда илгари таърифланган кўринишдан келиб чиқадиган бўлсак, бунда ўлчам M нинг максимал ўлчамидан ошмаслиги керак бўлади. Бундай схема жуда қулай бўлиб, барча ҳолларга тўғри келади ҳамда бир пайтнинг ўзида ҳотиранинг динамик тақсимланишининг қийинчиликларини олдини ола олади. Кўпинча сатрлар ўлчамининг қуйидагича иккита кўринишда тасаввур қилинади:

1. Якуний белгининг қўшилиши йўли билан ўлчамни ҳираоқ кўрсатиш, бу белги бошқа ҳеч қаерда ишлатилмайди. Одатда бунинг учун 0C ҳусусиятли “босмаланмайдиган” белги ишлатилади.(Келгусида бу барча белгилар ичида минимал белги эканлиги муҳимдир).

2.ўлчам биринчи элемент массиви сифатида сақланади, яъни s сатри қуйидагича кўринишга эга бўлади: $s = s_0, s_1, s_2, ..., s_{N-1}$. Бунда, $s_1, ..., s_{N-1}$ →сатрларнинг фактик белгилари, а s0 = CHR(N). Бундай ёндашувнинг ижобий томони шуки, ўлчамнинг аниқ очиқлигида, салбий томони эса ўлчам кўпчилик белгиларнинг ўлчамлари билан чегараланади(256).

Навбатдаги излаш алгоритмда биз биринчи чизмага урғу берамиз. Бу ҳолатда сатрларни солиштириш қуйидагича амалга оширилади:

```
i :=0;                                                    (1.45)
WHILE (x[i] = y[i]) & (x[i] # 0C) DO i := i+1 END:
```

Якуний белги асос сифатида ишлайди, циклнинг инварианти эса қуйидагича:
$Aj : 0 \le j < i : x_j = y_j \neq 0C$.

Натижаловчи шартлар эса қуйидаги шаклда бўлади:
$((x_i \neq y_i) OR (x_i = 0C)) \& (A_j : 0 \le j < i; x_j = y_j \neq 0C)$.

xi=yi шартида x ва *y мос тушади деб ҳисобланади, агарда xi<yi*, унда *x=y* бўлади.

Энди биз жадвалда излаш масаласига қайтишга тайёрмиз. У “киритилган” излашларни талаб этади, айнан: жадвал сатрлари бўйича излаш, ҳар бир сатр учун эса компонентлар учун кетма-кет солиштирилишидир. Масалан, Т-жадвали ва х-излаш аргументи қуйидагича кўринишда аниқланади.

```
T : ARRAY [0..N-1] OF String;
x: String;
```

Фараз қиламиз, N етарлича катта, жадвал эса алфавит тартибида тартибланган. Баравар бўлиш йўли билан излашдан фойдаланамиз. Юқорида айтиб ўтилган мос келувчи алгоритмларни (1.43) ва сатрларни тенглаштиришни қўллаб (1.45), программанинг қуйидагича фрагментини ҳосил қиламиз.

```
L := 0; R := N;
WHILE L<R DO
m := (L+R) DIV 2; i=0;
WHILE (T[m,i] = x[i]) & (x[i] #0C) DO i:= i+1 END;
IF T [m,i] < x[i] THEN L:=m+1 ELSE R :=m END                    (1.46)
END;
IF R< N THEN i:=0;
WHILE (T[R,i] = x[i]) & (x [i] # 0C) DO i:= i+1 END
END;
(*(R<N) & (T[R,i] =x[i] ) фиксирует совпадение *)
```

1.12.4. Сатрни тўғридан-тўғри излаш.

Кўпинча излашнинг сатр излаш деб номланувчи специфик излашга дуч келамиз. Уни қуйидагича йўл билан аниқлаш мумкин. N элементлардан s массиви берилган бўлсин, M элементлардан p массиви берилган бўлсин, бунда $0 < M \le N$ бўлади. Бундай кўринишда бўлади:

```
s: ARRAY [0..N-1] OF item;
p: ARRAY [0..M-1] OF item;
```

Сатр излаш биринчи навбатда p нинг s ичига кирганлигини аниқлайди. Одатда item- бу белги, яъни s ни айрим матн деб ҳисоблаш мумкин, p ни эса шакл ёки сўз деб ҳисобланиши мумкин ва биз берилган матнга биринчи киришни топишни ҳохлаймиз. Бу ҳаракат матнларни қайта ишлаш системаларининг барчасига тааллуқли бўлади. Бунда бу

масала билан эффектив алгоритм билан боғлиқлик билинади. Бироқ, эффективликка эътибор қаратишдан олдин келинг "тўғричизиқли" излаш алгоритмини таҳлил қиламиз. Биз уни тўғридан тўғри сатр излаш деб атаймиз.

Алгоритмни аниқлаштиришдан олдин, биз ундан қандай натижа олишни ҳохлашимизни аниқлаштирамиз. Сатрнинг бошига йўналтирувчи ва шаклга мос келувчи натижани *i деб ҳисоблаймиз. Ушбу мақсадда предикат P(i,j) киритилган:*

$$P(i,j) = Ak : 0 \le k < j : s_{i+k} = p_k . \quad (1.47)$$

Бизнинг натижаловчи индексимиз бу P(i,M) предикатимизни қаноатлантириш керак. Лекин бу етарли эмас. Излаш биринчи шаклнинг киришни топиши керак, P(k,M) $k<i$ ларнинг барчасига ёлғон бўлиши керак. Бу шартни *Q(i)* орқали ифодалаймиз.

$$Q(i) = Ak : 0 \le k < i :\sim P(k,M) . \quad (1.48)$$

Қўйилган масала такрорланувчи излаш расмийлаштирилиши кераклиги ҳақида ўйлантиради, ва биз қуйидагича вариантни таклиф этамиз.

```
i :=-1;
REPEAT i:=i+1 (*Q(i)*)
found :=P(i,M)
UNTIL  found OR (i=N-M);
```

Р ни ҳисоблаш, яна алоҳида белгиларнинг такрорланувчи тенглаштирилувчиларига олиб келади. Агарда Р га Морган теоремасини қўлласак, унда итерациялар шакл билан сатр белгиларининг мос келмаслигини "излаши" керак бўлади.

$$P(i,j) = (Ak : 0 \le k < j : s_{i+k} = p_k) = (\sim Ek : 0 \le k < j : s_{i+k} \ne p_k) .$$

Бундай аниқлаштириш натижасида биз такрорланиш ичидаги такрорланишга дуч келамиз. Р ва Q предикатлари программаларнинг тегишли жойларига изоҳ сифатида киритилади. Улар ўзида цикллар инвариантларининг эслатиб ўтилган итератив жараёнларида намоён бўлади.

```
i :=-1;
REPEAT i:=i+1; j:=0; (*Q(i)*)                                   (1.49)
WHILE (j < M) & (s[i+j] = p[j]) DO (*P(i,j+1)*) j := j+1 END
(*Q(i) & P(i,j)) & ((j = M) OR (s[i+j] # p[j]))*)
UNTIL (j=M) OR (i=N-M);
```

Аслида j=M аъзоси якунловчи шартда found шартига тўғри келади, чунки бундан *P(i,M)* келиб чиқади. *i=N-M* аъзоси *Q (N-M)* ни келтириб чиқаради ва сатрнинг ҳеч қаерида мослик йўқлигидан далолат беради. Агар, $j<M$ да итерациялар такрорланса, (1.47) дан *~P(i,j), кейин* (1.48) дан *Q(i+1)* келиб чиқиб, *I* нинг навбатдаги оширилишидан сўнг *Q(i)* нинг ҳаққонийлигини тасдиқлайди.

Сатрда тўғридан -тўғри излашнинг таҳлили. Агар биз ички цикллардаги бир неча мосликлардан кейин бир жуфт белгиларнинг мос келмаслигига йўл қўйганимизда бу алгоритм етарлича тўғри ишлайди. Item туринг юқори кучида бу кўп учрайдиган ҳолатдир. 128 белгидан иборат матнлар учун тўғри келмаслик бир ёки икки текширувдан сўнг аниқланади. Бундан ташқари салбий оқибатда ҳаракат хавфга олиб келади. Масалан, *N-1* дан иборат сатрдаги А белгилар ва ягона В, шакл эса *M–1* дан иборат А белгилар ва В. Сатрнинг оҳиридаги мосликни топиш учун *N* M* тартибидаги солиштиришни амалга ошириш керак бўлади. Бахтимизга, яқинда биз бу ноқулай вариантнинг яхшиланишига олиб келадиган усулни кўра оламиз.

Назорат саволлари

1.Қидирув ҳақида маълумот беринг.

2. Чизиқли қидирувни тушунтиринг.

3. Иккига бўлиб қидирмок (иккиламчи қидирув).

4. Жадвалда излаш

Тест саволлари.

1. Қайси тип (тури) (ключ) калит рўлини бажарадиган айрим поле (майдонлар)лардан ёзиб олишларни ифодалайди?

А) item Б) WHILE В) ELSIF Г) UNTIL

2. Қандай тўсиқ массив чегарасидан ўтиб кетишидан сақлайди?

А) *барьер Б) дравер В)кавер Д) тўгри жавоб йўқ*

3. Шартли операторларнининг бошланғич қисмларининг ўринларини ўзгартириш билан нимага эришиш мумкин?

А) эффективликка Б)Тэзкорлигига В) Оперативлигига Г) Натижавийлигига