Практикалық жұмыс №2-3

Тақырыбы: Массивтер және функциялар. Сұрыптау алгоритмдері

Мақсаты: Бір өлшемді массивтерді программалау принциптерін үйреніңіз. Екі өлшемді массивті программалау принциптерін үйреніңіз. Динамикалық бағдарламалау негіздерін үйреніңіз. Бағдарламалау процедураларының негіздерімен және пайдаланушы анықтайтын функциялармен танысыңыз.

Қысқаша теориялық мәліметтер

Қалыпты айнымалы сияқты, массив пайдаланбас бұрын жариялануы керек. n өлшемді массивді жариялаудың негізгі формасы келесідей: деректер түрі жиым атауы [1 өлшем] [2 өлшем] ... [өлшем n].

Бірөлшемді массивті сипаттау кезінде жариялаушы жазба келесідей пішінге ие болады: тип массив аты [өлшем]; мұндағы тип – массив элементтерінің негізгі түрі; өлшем - оның элементтерінің саны.

Жариялау мысалдары:

int a[15]; // a деп аталатын 15 бүтін элементтердің массиві

float x[3]; // x деп аталатын нақты типтегі үш элементтің массиві

Бір өлшемді массив келесі формада көрсетілуі мүмкін (1-кесте):

1-кесте – Бір өлшемді массивтің құрылымы

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	…	A[n-2]	A[n-1]
5	21	-3	10	0	15		15	-9

Бір өлшемді массивтің жеке элементін таңдау жиым атын және оның индексін төртбұрышты жақшада көрсету арқылы жүзеге асырылады.

Мысалы, cout <<A[0] командасын орындау нәтижесінде; экранда 5 саны көрсетіледі, ол кестедегі массивтің бірінші элементінің мәні болып табылады. A[3]=5,5;

Бағдарламаның орындалу мысалы

Келесі бағдарламаның кодын теріп, оны іске қосыңыз. Оның жұмысының алгоритмін, осы бағдарламада қосымша айнымалы қосындыны қолданудың мағынасын түсіндіріңіз.

Бағдарлама коды:

```
#include <conio.h>
#include <iostream.h>
void main ()
{
int A[10], i;
for (i=0; i<10; i++)
{
cout << “input A[“<<i<<”]=”;
cin>> A[i];
};
int sum=0;
for (i=0; i<10; i++)
sum=sum+A[i];
cout << “\nSumma=”<<sum;
getch ();
}
```

С стандарты көп өлшемді массивтерді анықтайды. Көпөлшемді массивтің қарапайым түрі екі өлшемді массив болып табылады. Екі өлшемді массив - әрбір элементтің екі индексі бар массив. Екі өлшемді массив матрица деп те аталады.

Екі өлшемді массивтің ең қарапайым мысалы кесте болып табылады.

Мұндай массивтердің екі өлшемі болады: жол нөмірі, баған (баған) нөмірі.

Екі өлшемді массивті жариялау мысалы:

int numbers[10][10]; // 100 сандық элементтері бар массив.

float test[5][5]; // 25 нақты сандар массиві

Массивті инициализациялау - массив элементтеріне кейбір бастапқы мәндерді тағайындау. C++ осы мақсат үшін кейбір арнайы мүмкіндіктерді қамтамасыз етеді. Инициализациялаудың ең қарапайым жолы - массивтің белгілі бір элемент мәндерінің тізімі бар бұйра жақшаларда массивді жариялау.

Екі өлшемді алапты инициализациялау кезінде әрбір өлшем бұйра жақшаға алынуы керек.

Массивті инициализациялау мысалы:

int temp[2][3] = { 3, 5, 4, 7, 4, 8 }

Екі өлшемді массивтің элементтерімен осы элементтердің деректер түрі үшін берілген кез келген әрекеттерді орындауға болады.

Бағдарламаның орындалу мысалы

Келесі программаның кодын теріңіз, оны орындау үшін іске қосыңыз, бағдарлама кодын құрылымдаңыз және ізін салыңыз.

Бағдарламалауға арналған тапсырма. [-10,10] аралығындағы кездейсоқ сандармен толтырылған екі өлшемді b[10][10] массив берілген. Берілген k санынан үлкен массив элементтерін табыңыз және көрсетіңіз.

Бағдарлама коды:

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
int main(void)
{
int b[10][10];
int i, j, k;
srand(time(NULL));
for (i=0; i<=9; i++)
{
for (j=0; j<=9; j++)
{
b [i][j]=rand()%21-10;
cout << b[i][j] << " ";
};
cout << endl;
};
cout << "Введите число k";
cin >> k;
for (i=0; i<=9; i++)
for (j=0; j<=9; j++)
if (b[i][j]>k) cout << b[i][j] << "\t";
cout << endl;
} //.
```

Бағдарламаны орындау мысалы (функция)

Функциялардың көмегімен екі санның қосындысын есептейтін программаның кодын теріңіз, оны орындау үшін іске қосыңыз, бағдарлама кодын құрылымдаңыз және ізін сызыңыз.

Бағдарлама коды:

1. Бағдарламаның басы

```
/*Пример программы - содержит функцию
summa- вычисление суммы двух целых чисел
*/
#include <iostream>
#include <conio.h>//библиотека - ожидание нажатия клавиши
using namespace std;

//объявить функцию summa - прототип функции summa

int summa(int a,int b);// возможен варинт   int summa(int,int);
void message();//пример функции без значения
```

2. Негізгі бағдарламаның денесі

```
int main()
{
cout <<"\tSample function summa\n\n";

//----------------------------------------
//объявим 2 переменные-слагаемые и введем значения
int k=2,m=5;
int rez;//объявим переменную - результат суммы

rez=summa(k,m);//вызываем функцию summa

//вывод результатов
cout<<"First number - "<<k<<"\n";
cout<<"Second number - "<<m<<"\n";
cout <<"\nRezult (summa)- "<<rez<<"\n";
message(); //вызываем функцию message
//------------------------------------------------
cout <<"\nPress Enter to continue...";
_getch(); // Ожидание нажатия клавиши
 return 0;
}
```

3. Функция

```
//определение функции summa
int summa(int a,int b)
{
int nSum;
nSum=a+b;
return nSum;
}

//определение функции message
void message()
{
    cout<<"\n\tThis simple message\n\n";
    return;
}
```

Таңдаумен сұрыптау (Selection sort)

Жиымды өсу ретімен сұрыптау үшін әрбір итерацияда ең үлкен мәні бар элементті табу керек. Онымен бірге соңғы элементті орындармен ауыстыру қажет. Ең үлкен мәні бар келесі элемент соңғы орынға айналады. Алаптағы бірінші орындардағы элементтер тиісті тәртіпте болғанға дейін осылай болуы тиіс.

```
void SortAlgo::selectionSort(int data[], int lenD)
{
 int j = 0;
 int tmp = 0;
 for(int i=0; i<lenD; i++){
  j = i;
  for(int k = i; k<lenD; k++){
   if(data[j]>data[k]){
    j = k;
   }
  }
  tmp = data[i];
  data[i] = data[j];
  data[j] = tmp;
 }
      }
```

Көпіршікті сұрыптау (Bubble sort)

Көпіршікті сұрыптау кезінде көршілес элементтер салыстырылады және егер келесі элемент алдыңғы элементтен кіші болса, орындар өзгереді. Деректер бойынша бірнеше өту қажет. Бірінші өту кезінде алаптағы алғашқы екі элемент түзіледі. Егер олар дұрыс болмаса, олар орындармен ауыстырылады және одан кейін келесі жұптағы элементтер салыстырылады. Сол шартпен олар да орын ауыстырады. Осылайша сұрыптау жиымның соңына жеткенше әрбір циклде болады.

```
void SortAlgo::bubbleSort(int data[], int lenD)
{
 int tmp = 0;
 for(int i = 0;i<lenD;i++){
  for(int j = (lenD-1);j>=(i+1);j--){
   if(data[j]<data[j-1]){
    tmp = data[j];
    data[j]=data[j-1];
    data[j-1]=tmp;
   }
  }
 }
}
```

Кірістірулермен сұрыптау (Insertion sort)

Қойылымдармен сұрыптау кезінде жиым реттелген және реттелмеген екі аймаққа бөлінеді. Бастапқыда бүкіл алап реттелмеген сала болып табылады. Бірінші өту кезінде реттелмеген аумақтан бірінші элемент алынады және реттелген аумақта дұрыс жағдайда орналастырылады.

Әрбір өткелде реттелген облыстың мөлшері 1-ге өседі, ал реттелмеген облыстың мөлшері 1-ге қысқарады.

Негізгі цикл 1-ден N-1-ге дейінгі аралықта жұмыс істейді. j-ші итерацияда [i] элементі реттелген аумақта дұрыс орынға кірістірілген. Бұл [i] -ден үлкен, реттелген аумақтың барлық элементтерін оңға қарай бір орынға жылжыту арқылы жасалған. [i] кіші элементтер мен [i] үлкен элементтер арасындағы аралыққа қойылады.

```
void SortAlgo::insertionSort(int data[], int lenD)
{
  int key = 0;
  int i = 0;
  for(int j = 1;j<lenD;j++){
    key = data[j];
    i = j-1;
    while(i>=0 && data[i]>key){
      data[i+1] = data[i];
      i = i-1;
      data[i+1]=key;
    }
  }
}
```

Біріктірумен сұрыптау (Merge sort)

Біріктіру арқылы рекурсивті сұрыптау кезінде алап алдымен ұсақ бөлшектерге - бірінші кезеңде - бір элементтен тұратын бөлшектерге бөлінеді. Содан кейін бұл кесектер неғұрлым ірі бөліктерге біріктіріледі - екі элементтен және сонымен бірге элементтер салыстырылады, ал нәтижесінде жаңа бөлікте кіші элемент сол жақтан, ал үлкен бөлік оң жақтан орын алады. Одан кейін одан да үлкен бөлшектерге бірігу болады және алгоритмнің соңына дейін, барлық бөлшектер сұрыпталған массивке біріктірілгенде.

```
void SortAlgo::mergeSort(int data[], int lenD)
{
  if(lenD>1){
    int middle = lenD/2;
    int rem = lenD-middle;
    int* L = new int[middle];
    int* R = new int[rem];
    for(int i=0;i<lenD;i++){
      if(i<middle){
        L[i] = data[i];
      }
      else{
        R[i-middle] = data[i];
      }
    }
    mergeSort(L,middle);
    mergeSort(R,rem);
    merge(data, lenD, L, middle, R, rem);
  }
}

void SortAlgo::merge(int merged[], int lenD, int L[], int lenL, int R[], int lenR){
  int i = 0;
  int j = 0;
```

```
 while(i<lenL||j<lenR){
  if (i<lenL & j<lenR){
   if(L[i]<=R[j]){
    merged[i+j] = L[i];
    i++;
   }
   else{
    merged[i+j] = R[j];
    j++;
   }
  }
  else if(i<lenL){
   merged[i+j] = L[i];
   i++;
  }
  else if(j<lenR){
   merged[i+j] = R[j];
   j++;
  }
 }
```

Жылдам сұрыптау (Quick sort)

Жылдам сұрыптау «бөліп ал және билік ет» алгоритмін пайдаланады. Ол бастапқы жиымды екі аймаққа бөлуден басталады. Бұл бөліктер тірек деп аталатын белгіленген элементтің сол және оң жағында орналасқан. Процестің соңында бір бөлікте тіректен кіші элементтер болады, ал екінші бөлікте тіректен көп элементтер болады.

```
        void SortAlgo::quickSort(int* data, int const len)
{
 int const lenD = len;
 int pivot = 0;
 int ind = lenD/2;
 int i,j = 0,k = 0;
 if(lenD>1){
  int* L = new int[lenD];
  int* R = new int[lenD];
  pivot = data[ind];
  for(i=0;i<lenD;i++){
   if(i!=ind){
    if(data[i]<pivot){
     L[j] = data[i];
     j++;
    }
    else{
     R[k] = data[i];
     k++;
    }
   }
  }
  quickSort(L,j);
  quickSort(R,k);
  for(int cnt=0;cnt<lenD;cnt++){
```

```
    if(cnt<j){
      data[cnt] = L[cnt];;
    }
    else if(cnt==j){
      data[cnt] = pivot;
    }
    else{
      data[cnt] = R[cnt-(j+1)];
    }
   }
  }
}
```

Сұрыптау алгоритмін жүзеге асыру

Тапсырма: 10 элементтен тұратын массив берілген, индекстер 0 - 9. Оны өсу ретімен сұрыптаңыз.

Сұрыптау үшін циклдарды анықтайық:

Бірінші цикл 0 - 8 үшін (i = 0 ; i < 9 ; i + +)

Екінші цикл 1 - 9 үшін (j = i + 1 ; j < 1 0 ; j + +)

Ішкі циклде мәндер салыстырылады және салыстыру шарты орындалса, элементтер ауыстырылады.

Осы сұрыптау алгоритмімен мынаны ескеріңіз:

1. Массив өсу ретімен сұрыпталады.

2. Ішкі циклдің толық өтуі нәтижесінде ең үлкен мәні бар элемент бірінші пайда болады.

3. Таңбаны > белгісінен <ге өзгертсеңіз, массив кему ретімен сұрыпталады.

Бағдарлама коды:

1. Сұрыптау блогы

```
for(i=0;i<9;i++)
{
    for(j=i+1;j<10;j++)
    {
        if (a[j]>a[i])
        {
            temp=a[j];
            a[j]=a[i];
            a[i]=temp;
        }
    }
}
```

Дәстүрлі массив анықтамасы бойынша:

типі массив_атауы [элементтер_саны];

массив үшін бөлінген жадтың жалпы көлемі анықтамамен берілген және элементтер_санына тең * sizeof(тип).

Бірақ кейде массивтің жады белгілі бір мәселені шешу үшін бөлінуі қажет және оның өлшемдері алдын ала белгісіз және оны бекіту мүмкін емес.

Айнымалы өлшемдері бар массивтерді қалыптастыруды көрсеткіштерді және динамикалық жадты бөлуді екі жолмен ұйымдастыруға болады:

1) alloc.h және stdlib.h тақырып файлдарында сипатталған кітапхана функцияларын пайдалану (стандартты C);

2) жаңа және жою операцияларын қолдану (C++).

Кітапхана функцияларын пайдаланып динамикалық массивтерді қалыптастыру

Динамикалық жадты бөлу және тазарту үшін келесі функцияларды пайдаланыңыз

Функция	Прототипі және қысқаша сипаттамасы
malloc	void * malloc(unsigned s) s байт ұзын динамикалық жад аймағының басына көрсеткішті қайтарады, сәтсіз болғанда NULL мәнін қайтарады
calloc	void * calloc(unsigned n, unsigned m) Әрқайсысы m байттан тұратын n элементті ұстау үшін үйме аймағының басына көрсеткішті қайтарады, сәтсіз болғанда NULL мәнін қайтарады.
realloc	void * realloc(void * p, unsigned s) Бұрын бөлінген динамикалық жадтың блогының өлшемін s байт өлшеміне өзгертеді, p - өзгертілетін блоктың басының мекенжайы, сәтсіздік кезінде NULL мәнін қайтарады.
free	void *free(void p) Динамикалық жадтың бұрын бөлінген бөлімін шығарады, p - бірінші байттың адресі

Мысалы:

Бір өлшемді динамикалық массив құру функциясы:

```
int * make_mas(int n)
(
int *mas;
mas=(int*)malloc(n*sizeof(int));
for(int i=0;i<n;i++)
mas[i]=random(10);
return mas;
}
```

Жадты бөлу үшін malloc функциясы пайдаланылады, оның параметрі n*sizeof(int) тең бөлінген жады аймағының өлшемі болып табылады. malloc функциясы жазылмаған void* көрсеткішін қайтаратындықтан, нәтижесінде терілмеген көрсеткішті int* көрсеткішіне түрлендіру керек.

Бөлінген жадты бос(mas) функциясы арқылы босатуға болады.

new және delete операцияларының көмегімен динамикалық массивтерді қалыптастыру

Динамикалық жадты бөлу үшін new және delete операциялары қолданылады.. Операция

new тип_атауы

немесе

new тип_атауы инициализатор

өлшемі тип атымен көрсетілген деректер түріне сәйкес келетін бос жад аймағын бөлуге және қолжетімді етуге мүмкіндік береді. Таңдалған аумақ инициализатор анықтайтын мәнмен толтырылады, ол міндетті параметр болып табылмайды. Жадты сәтті бөлу жағдайында операция бөлінген жад аймағының басының мекенжайын қайтарады, егер аумақты бөлу мүмкін болмаса, онда NULL қайтарылады.

Мысалы:

```
1) int *i;
   i=new int(10);
2) float *f;
   f=new float;
3) int *mas=new[5];
```

1, 2-мысалдар скалярлық айнымалылар үшін жадты бөлу жолын көрсетеді, 3-мысал айнымалылар массиві үшін жадты қалай бөлу керектігін көрсетеді.

Көрсеткіштегі жою операциясы жаңа операциямен бұрын бөлінген жад аймағын босатады.

Мысалы:

Екі өлшемді динамикалық массив құру функциясы

```
int ** make_matr(int n)
{
int **matr;
int i,j;
matr=new int*[n];
for (i=0;i<n;i++)
{
matr[i]=new int[n];
for (j=0;j<n;j++)
matr[i][j]=random(10);
}
return matr;
}
```

Матрицаны құру кезінде жад алдымен бір өлшемді массивтерге көрсеткіштер массивіне бөлінеді, содан кейін параметрі бар циклде n бір өлшемді массив үшін жад бөлінеді.

Жадты босату үшін бір өлшемді массивтерді босату циклін орындау керек

```
for(int i=0;i<n;i++)
delete matr[i];
```

Осыдан кейін біз matr көрсеткіші көрсеткен жадты босатамыз

```
delete [] matr;
```

ТАПСЫРМАЛАР:

№1. Қолданушыдан N нақты санды сұрайтын және жұп сандары бар сандардың қосындысын көрсететін программа жазыңыз.

№2. Орталықтандырылған компьютерлер уақытының төрттен біріне дейін деректерді сұрыптауға арналған деп есептелді. Себебі алдын-ала сұрыпталған массивте мәнді табу әлдеқайда оңай. Әйтпесе, іздеу шөптен ине іздеуге ұқсайды.

Барлық жұмыс уақытын сұрыптау алгоритмдерін үйренуге және енгізуге жұмсайтын бағдарламашылар бар. Себебі бизнестегі бағдарламалардың басым көпшілігі мәліметтер базасын басқаруды қамтиды. Адамдар ақпаратты дерекқорлардан үнемі іздейді. Бұл іздеу алгоритмдері өте қажет дегенді білдіреді.

Бірақ бір "бірақ" бар. Іздеу алгоритмдері қазірдің өзінде сұрыпталған мәліметтер базасымен әлдеқайда жылдам жұмыс істейді. Бұл жағдайда тек Сызықтық іздеу қажет.

Компьютерлер кейбір уақытта пайдаланушыларсыз болса да, сұрыптау алгоритмдері дерекқорлармен жұмыс істеуді жалғастыруда. Іздеуді жүзеге асыратын пайдаланушылар қайтадан келеді, ал мәліметтер базасы іздеудің белгілі бір мақсатына қарай сұрыпталған.

Нұсқа (вариант) бойынша берілген тапсырмаларды орындау үшін сұрыптау алгоритмдерін іске асырудың жоғарыда келтірілген барлық мысалдарын қарастырыңыз.

№3 (вариант бойынша)

1. Бір өлшемді массив құру. Одан берілген саны бар элементті алып тастаңыз, берілген саны бар элементті қосыңыз; Массивті өсу бойынша сұрыптаңыз.

2. Бір өлшемді массив құру. Одан берілген кілті бар элементті алып тастаңыз, берілген кілті бар элементті қосыңыз; Массивті кему бойынша сұрыптаңыз.

3. Бір өлшемді массив құру. Одан K элементті жою, берілген саннан бастап, берілген кілті бар элементті қосу; Массивті өсу бойынша сұрыптаңыз.

4. Бір өлшемді массив құру. Одан берілген саны бар элементті алып тастаңыз, берілген саннан бастап K элементтерін қосыңыз; Массивті кему бойынша сұрыптаңыз.

5. Бір өлшемді массив құру. Одан берілген саннан бастап K элементтерін алып тастаңыз, берілген саннан бастап K элементтерін қосыңыз; Массивті өсу бойынша сұрыптаңыз.

6. Екі өлшемді массив құру. Одан берілген саны бар жолды өшіріңіз; Массивті кему бойынша сұрыптаңыз.

7. Екі өлшемді массив құрастыр. Одан берілген саны бар бағанды жою; Массивті өсу бойынша сұрыптаңыз.

8. Екі өлшемді массив құрастырыңыз. Оған берілген саны бар жолды қосыңыз; Массивті кему бойынша сұрыптаңыз.

9. Екі өлшемді массив құрастыр. Оған берілген саны бар баған қосыңыз; Массивті өсу бойынша сұрыптаңыз.

10. Екі өлшемді массив құрастырыңыз. Одан берілген саны бар жолды және бағанды жойыңыз. Массивті кему бойынша сұрыптаңыз.

11. Екі өлшемді массив құрастырыңыз. Оған берілген саны бар жолды және бағанды қосыңыз. Массивті өсу бойынша сұрыптаңыз.

12. Екі өлшемді массив құрастырыңыз. Одан берілген санды қамтитын барлық жолдарды жойыңыз. Массивті кему бойынша сұрыптаңыз.

13. Екі өлшемді массив құрастырыңыз. Одан берілген санды қамтитын барлық бағандарды жойыңыз. Массивті өсу бойынша сұрыптаңыз.

14. Екі өлшемді массив құрастырыңыз. Одан қиылысында минималды элемент болып табылатын жол мен бағанды алып тастаңыз. Массивті кему бойынша сұрыптаңыз.

15. Екі өлшемді массив құрастырыңыз. Одан қиылысында максималды элемент болып табылатын жол мен бағанды алып тастаңыз. Массивті өсу бойынша сұрыптаңыз.

Бақылау сұрақтары

1. Бір өлшемді массив дегеніміз не?
2. Бір өлшемді массивтер не үшін қолданылады? Олар қалай сипатталған?
3. Бірөлшемді массивтің белгілі бір элементінің мәнін программада қалай пайдаланамыз?
4. Бір өлшемді массивтің элемент нөмірі қалай аталады?
5. Бір өлшемді массивті қалай толтыруға болады?
6. Екі өлшемді массивтің анықтамасын беріңіз?
7. Екі өлшемді массивті қалай инициализациялауға болады?
8. Жиым элементтерімен жұмыс істеуге арналған қандай амалдарды білесіз?
9. Екі өлшемді массивке элементтерді енгізу/шығару қалай орындалады?
10. Екі өлшемді массивте деректердің қандай түрлерін сақтауға болады?
11. Ішкі программа дегеніміз не? Оның мақсаты қандай?
12. Ғаламдық және жергілікті айнымалылар.
13. Формальды және нақты параметрлер.
14. Құрылымдық программалаудың негізгі принциптері қандай.
15. Параметрлерді ішкі программаға беру қалай жүзеге асады, қандай қолданба-
16. Бұл үшін аппараттық және бағдарламалық құралдар қолданылады ма?
17. Динамикалық жадымен жұмыс қалай ұйымдастырылады?
18. Динамикалық жадымен жұмыс істеу үшін қандай операциялар қолданылады?
19. Динамикалық массив қалай ұйымдастырылады?
20. Динамикалық берілген массивтің элементі қалай жойылады?
21. Динамикалық түрде берілген массивке элемент қалай қосылады?