ЛЕКЦІЯ 8

1. ДИНАМІЧНИЙ РОЗПОДІЛ ПАМ’ЯТІ
2. МАСИВИ У ПАМ’ЯТІ, ЩО ДИНАМІЧНО РОЗПОДІЛЯЄТЬСЯ
3. ОДНОВИМІРНІ МАСИВИ ЯК ПАРАМЕТРИ ФУНКЦІЙ
4. ДВОВИМІРНІ МАСИВИ У ПАМ’ЯТІ, ЩО ДИНАМІЧНО РОЗПОДІЛЯЄТЬСЯ

На практиці часто зустрічаються ситуації, коли при написанні програми є невідомим розмір різних змінних складених типів даних, наприклад масивів. Ці розміри можуть стати відомими лише в кожному конкретному випадку, чи сеансі роботи з програмою. Як відомо, синтаксис мови C/C++ не дозволяє створювати масиви змінних розмірів у пам'яті, що статично розподіляється.

Для вирішення проблеми у мову додано механізм так званого **динамічного розподілу пам'яті** (іноді на практиці це словосполучення скорочують та вживають термін «динамічна пам'ять»).

Для розміщення змінної в оперативній пам'яті, що динамічно розподіляється, використовують зарезервоване слово *new*.

Для вилучення змінної з обігу (коли вона вже не потрібна) використовують зарезервоване слово *delete*.

Основна форма їхнього використання наступна:

pointer_var = new var_type;
delete pointer_var,

pointer_var є покажчиком типу *var_type*.

Операція *new* виділяє відповідне місце для змінної у відповідній області оперативної пам'яті та повертає адресу цього виділеного місця. Якщо з будь-яких причин пам'ять не може бути виділеною, операція *new* повертає нульовий покажчик *NULL*.

Операция *delete* звільнює відповідну частину пам'яті, на яку показує *pointer_var*.

Зручність використання операції *new* є в тому, что вона сама автоматично визначає розмір змінної var_type та повертає покажчик, вже перетворений до цього типу.

Приклад

main.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  int *p;
6  p=new int;
7  if (!p) cout<<"Insufficient memory.";
8   else {*p=10;
9   cout<<p<<"\n";
10  }
11  delete p;
12 return 0;
13 }
14
```

Ефективність використання пам’яті, що динамічно розподіляється, видно на використанні змінних складених типів великого розміру (поки масивів). При обробці великих даних з’являється можливість завантаження масиву, що займає всю доступну оперативну пам‘ять, та його обробки за заданим алгоритмом. Після закінчення роботи масив видалють з оперативної пам’яті та отримують можливість обробки інших.

Якщо необхідно виділити пам’ять під одновимірний масив змінних, це можливо зробити наступним чином:

pointer_var = new var_type[size];
delete [ ] pointer.var;

При звільненні пам’яті з-під масиву є можливим використання

delete pointer_var;

Приклад. Робота з одновимірними масивами

main.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  int *p, i;
6  int size;
7  cout<<"Enter the dimension of array:";
8  cin>>size;
9
10 p=new int [size];
11 for (i=0;i<size;i++)
12     p[i]=2*i;
13 int *q=p;
14 for (i=0;i<size;i++)
15 cout<<"Element"<<i<<"="<<*q++<<"\n";
16 delete [] p;
17 return 0;
18 }
```

```
"d:\CPP\C EDUCATION 1 sem 1 year\c82\
Enter the dimension of array:4
Element0=0
Element1=2
Element2=4
Element3=6
```

Одновимірні масиви як параметри функцій

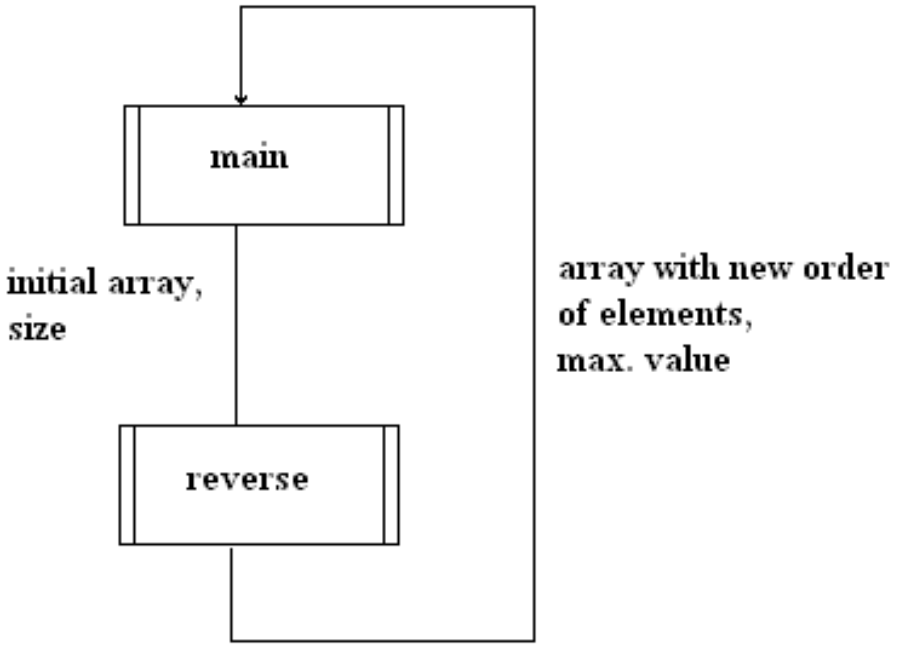
Основною ідеєю структурно-логічного програмування є побудова програм з спеціальних «будівельних» блоків, які у мові С називаються функціями. У загальному випадку функція приймає декілька параметрів, змінює деякі з них за заданими правилами та повертає у функцію, що її викликала, результати обчислень.

Для реалізації цього підходу широко використовуються покажчики та масиви, розташовані у пам’яті, яка розподіляється динамічно.

Найпростішим випадком є задача, в якій у основній функції (функції main()) задається масив, а для його обробки використовується додатково створена функціїя. Результати її роботи повертаються до функції main().

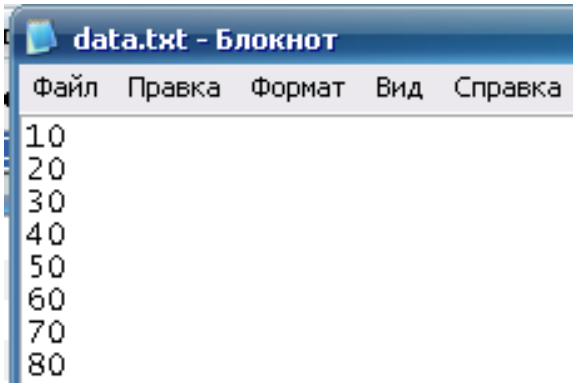
Розглянемо задачу. У функції main() задають одновимірний масив (зчитують з текстового файлу), після цього його передають до функції revrse(), в якій парні та непарні елементи міняються місцями. Змінений масив та додатково визначене у функції значення максимального елементу масиву повертають до функції main().

Блок-схема взаємодії функцій предсталена праворуч, текст програми - на наступних слайдах.



main.cpp

```
1  #include <iostream>
2  #include <fstream>
3  void revrse(int*arlocal, int*maxvaluelocal, int arsizelocal);
4  using namespace std;
5  int main()
6  { int *ar, i, arsize, maxvalue=0;
7  cout<<"Enter the number of elements:";
8  cin>>arsize;
9  ar=new int[arsize];
10 ifstream in ("data.txt");
11  for (i=0;i<arsize;i++)
12     in>>ar[i];
13  in.close();
14 revrse(ar,&maxvalue,arsize);
15  cout<<"Maximum value ="<<maxvalue<<"\n";
16  for (i=0;i<arsize;i++)
17     cout<<ar[i]<<" ";
18
19  delete[] ar;
20      return 0;
21 }
```



```
22  void revrse(int*arlocal, int*maxvaluelocal, int arsizelocal)
23  {
24       int temp,j;
25   for (j=0;j<arsizelocal;j++)
26      if (*maxvaluelocal<arlocal[j]) *maxvaluelocal=arlocal[j];
27
28    for (j=0;j<arsizelocal;j=j+2)
29   {
30      temp=arlocal[j];
31      arlocal[j]=arlocal[j+1];
32      arlocal[j+1]=temp;
33   }
34  }
```

```
"d:\CPP\C EDUCATION 1 sem 1 year\c83\bin\Debug\p20
Enter the number of elements:8
Maximum value =80
20 10 40 30 60 50 80 70
Process returned 0 (0x0)   execution time
Press any key to continue.
```

Двовимірні масиви у пам’яті, що динамічно розподіляється

Для розташування двовимірних масивів використовується механізм «покажчик на покажчик»:

Покажчик		Покажчик		Змінна
адреса	→	адреса	→	Значення

<тип> ** pointer;

Роботу з двовимірними масивами ілюструє наступний приклад

*main.cpp

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5    int rowsize, columnsize, i,j;
6     double**ar;
7     cout<<"Enter the number of rows: ";
8     cin>>rowsize;
9     cout<<"Enter the number of columns: ";
10    cin>>columnsize;
11     ar=new double*[rowsize] ;
12    for (i=0;i<rowsize;i++)
13    ar[i]= new double [columnsize];
14    for (i=0;i<rowsize;i++)
15    for (j=0;j<columnsize;j++)
16      ar[i][j]=i;
17     for (i=0;i<rowsize;i++)
18    {for (j=0;j<columnsize;j++)
19      cout<<ar[i][j]; cout<<"\n";}
20    for (i=0;i<rowsize;i++)
21      delete[] ar[i];
22     delete[]ar;
23      return 0;
24  }
```

"D:\CPP\c84\bin\Debug\p2019 1.exe"

```
Enter the number of rows: 4
Enter the number of columns: 5
00000
11111
22222
33333

Process returned 0 (0x0)   execu
Press any key to continue.
```

ДЯКУЮ ЗА УВАГУ!