КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА

Програмування на мові C++

Лектор: проф. Васильєв О.М.

Лекція №3.

Вказівники, масиви, рядки.

Вказівники

Вказівник – змінна, значенням якої є адреса комірки пам'яті.

Оголошення вказівника:

```
тип_даних *ім'я_вказівника;
```

Наприклад:

```
int *p;          // вказівник p на
                 // int-змінну
double *q,r;     // вказівник q на
                 // double-змінну та
                 // double-змінна r
char *s;         // вказівник s на char-
                 // змінну
```

Вказівники (продовження)

Основні оператори:

- Оператор **&** дозволяє отримати адресу змінної.
- Оператор ***** дозволяє отримати значення, записане за адресую, на яку вказує вказівник.

Наприклад:

```
int *p,m,n; // вказівник p
            // та змінні m та n
m=123;      // значення змінної m
p=&m; // адреса змінної m записана в p
n=*p; // значення за адресою p записане
      // в змінну n
```

Адресна арифметика

Основні операції із вказівниками:

- **Присвоєння** одному вказівнику значення іншого вказівника. В результаті відбувається копіювання значення (адреси).
- Вказівники можна **порівнювати**. Результат – логічне значення.
- До вказівника можна **додавати ціле** (додатне) **число**. Результат – адреса комірки, що відстоїть від вихідної на відповідну кількість комірок "вправо".
- Від вказівника можна **віднімати ціле** (додатне) **число**. Результат – адреса комірки, що відстоїть від вихідної на відповідну кількість комірок "вліво".
- Можна розраховувати **різницю вказівників**. Результат – кількість комірок між відповідними адресами.
- Вказівники можна **індексувати**. Результат – адреса комірки, що відстоїть від даної на відповідну кількість комірок.

Приклад використання вказівників

Pointers.cpp

```
#include <iostream>
using namespace std;

int main(){
    int n;    // змінна
    int *p;   // вказівник
    p=&n;     // адреса змінної
    cout<<"n=";
    cin>>n;
    cout<<"address "<<p<<": "; // значення вказівника
    cout<<"value is "<<*p<<endl; // значення, записане за адресою
    n++; // змінюємо значення змінної
    cout<<"new value is "<<p[0]<<endl; // перевіряємо значення за адресою
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Масиви

● **Масив –** сукупність змінних одного типу, об’єднаних спільним іменем

● **Статичний масив –** розмір (кількість елементів) масиву відомий на момент компіляції

● **Динамічний масив –** розмір визначається під час виконання програми

● **Доступ до елементу масиву –** ім’я масиву та індекс (індекси) елементу. Кожен індекс вказується в окремих квадратних дужках

● **Розмірність масиву –** кількість індексів, необхідних для однозначного визначення елементу масиву

● **Оголошення масиву –** тип елементів масиву, назва масиву та розмір масиву по кожному з індексів. Розмір по кожному індексу вказується в окремих квадратних дужках

Властивості масивів с С++

- Індексація елементів масиву завжди починається з нуля
- В С++ всі елементи масиву в пам’яті розміщуються підряд, один за одним
- В С++ немає перевірки на предмет виходу за межі масиву
- Ім’я масиву є вказівником на його перший елемент (елемент з нульовим індексом)

Приклад створення одномірного статичного масиву

SimpleArray.cpp

```
#include <iostream>
using namespace std;

int main(){
    int i; // індексна змінна
    const int n=10; // розмір масиву - константа
    int MyArray[n]; // створення м асиву
    cout<<"This is array:\n";
    for(i=0;i<n;i++){
       MyArray[i]=2*i+1; // заповнення масиву
       cout<<MyArray[i]<<" "; // виводимо елемент масиву
    }
    cout<<endl;
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Приклад створення одномірного статичного масиву - версія 2

SimpleArray2.cpp

```
#include <iostream>
using namespace std;

int main(){
    int i; // індексна змінна
    const int n=10; // розмір масиву
    int *p; // вказівник
    int MyArray[n]; // масив
    p=MyArray; // значення вказівника - адреса першого
               // елементу масиву
    // Перевіряємо адреси:
    cout<<p<<" : "<<MyArray<<" : "<<&MyArray[0]<<endl;
    for(i=0;i<n;i++){
       p[i]=2*i+1; // заповнюємо масив
    }
    for(i=0;i<n;i++){
       // Друкуємо елементи масиву:
       cout<<MyArray[i]<<" ";
    }
    cout<<endl;
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Приклад створення одномірного статичного масиву - версія 3

SimpleArray3.cpp

```
#include <iostream>
using namespace std;

int main(){
    int i; // індексна змінна
    const int n=10; // розмір масиву
    int *p; // вказівник
    int MyArray[n]; // масив
    p=MyArray; // значення вказівника - адреса першого
               // елементу масиву
    // Перевіряємо адреси:
    cout<<p<<" : "<<MyArray<<" : "<<&MyArray[0]<<endl;
    for(i=0;i<n;i++){
       *p=2*i+1; // заповнюємо масив
       p++;
    }
    p--; // змінюємо значення вказівника
    // Перевіряємо адреси:
    cout<<p<<" : "<<MyArray+(n-1)<<" : "<<&MyArray[n-1]<<endl;
    for(i=0;i<n;i++){
       // Друкуємо елементи масиву:
       cout<<MyArray[i]<<" ";
    }
    cout<<endl;
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Приклад створення двовимірного статичного масиву

SimpleMatrix.cpp

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main(){
    int i,j; // індексні змінні
    const int m=3,n=5; // розміри масиву
    int MyMatrix[m][n]; // двомірний масив
    srand(time(NULL));
    // Заповнення масиву випадковими числами:
    for(i=0;i<m;i++){
       for(j=0;j<n;j++){
          MyMatrix[i][j]=rand()%10;
          cout<<MyMatrix[i][j]<<" ";
       }
       cout<<endl; // кінець рядка
    }
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Ініціалізація масиву

При оголошенні масиву його можна ініціалізувати – вказати для елементів значення

Синтаксис оголошення з ініціалізацією:

```
тип імя_масиву[розмір]={значення1,значення2,...};
```

Приклад:

```
int n[4]={1,2,3,4};
```
або

```
int n[]={1,2,3,4};
```
- розмір масиву визначається автоматично

```
int numbers[3][2]={{1, 2},
                   {3, 4},
                   {5, 6}};

int numbers[][2]={{1, 2},
                  {3, 4},
                  {5, 6}};
```

Приклад ініціалізації масивів

ArrayInit.cpp

```
#include <iostream>
using namespace std;

int main(){
    // ініціалізація одномірного масиву:
    int n[]={1,2,3,4,5};
    // ініціалізація двомірного масиву:
    int m[][2]={{1,2},{3,4},{5,6}};
    int i,j; // індексні змінні
    // відображаємо одномірний масив:
    cout<<"1D array:\n";
    for(i=0;i<5;i++){
       cout<<n[i]<<(i<4?" ":"\n");
    }
    // відображаємо двомірний масив:
    cout<<"\n2D array:\n";
    for(i=0;i<3;i++){
       for(j=0;j<2;j++){
          cout<<m[i][j]<<" ";
       }
       cout<<endl;
    }
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Символьні масиви

В С++ існує два способи реалізації текстових рядків:

- У вигляді символьних масивів (масиви типу **char**)
- Об'єкти класу **string**

Текстові рядки у вигляді char-масивів

- Оголошуються як звичайні масиви. Наприклад:

```
char str[80];
```

- Розмір масиву не співпадає з розміром текстового рядка
- Ознакою закінчення текстового рядка є спеціальний нуль-символ **'\0'** **(не плутати з нулем!)**
- Символьні масиви можна ініціалізувати текстовим рядком:

```
char str[80]="Hello, World!";
```

- Існують спрощені алгоритми роботи з символьними масивами, наприклад:

```
cout<<str;
```

При оголошенні масиву його можна ініціалізувати – вказати для елементів значення

Приклад роботи з символьним масивом

CharArray.cpp

```
#include <cstdio>
#include <iostream>
using namespace std;

int main(){
    // символьний масив з ініціалізацією:
    char txt[50]="Please, enter your text here: ";
    // символьний масив:
    char input[100];
    // індексна змінна:
    int i;
    cout<<txt; // відображаємо текст
    // cin>>input;
    gets(input); // зчитуємо текст
    for(i=0;input[i];i++){
       // посимвольно виводимо елементи масиву:
       cout<<input[i]<<" * ";
    }
    cout<<endl;
    // виводимо текст не з самого початку:
    cout<<txt+8;
    // зчитуємо текст:
    gets(input);
    // вказівник на символьне значення:
    char *s=input;
    for(;*s;s++){ // виводимо фрагменти тексту
       cout<<s<<endl;
    }
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Текстові літерали

В C++ текстові літерали реалізуються у вигляді символьного масиву, і на такий літерал автоматично створюється посилання

Приклад "Робота з літералами"

TextConstants.cpp

```
#include <iostream>
using namespace std;

int main(){
    // вказівники на символьні значення:
    char *p,*q;
    // текстовий літерал:
    p="In Physics We Trust";
    // ще один літерал:
    q=p+11;
    // перевіряємо результат:
    cout<<p<<endl;
    cout<<q<<endl;
    // літерал із нуль-символом:
    q="You Can Win\0 If You Want";
    // перевіряємо результат:
    cout<<q<<endl;
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Динамічне виділення пам'яті

Оператори динамічного розподілу пам’яті:

new – виділення пам’яті

delete – вивільнення пам’яті

Змінна:

```
вказівник=new тип_данних;
delete вказівник;
```

Наприклад:

```
int *p;
p=new int;
    …
delete p;
```

Масив:

```
вказівник=new тип_данних[розмір];
delete [] вказівник;
```

Наприклад:

```
int *q;
q=new int[10];
    …
delete [] q;
```

Приклад створення динамічного масиву

DArray.cpp

```
#include <iostream>
using namespace std;

int main(){
    // розмір масиву, вказівник та
    // індексна змінна:
    int n,*DArray,i;
    // визначаємо кількість елементів масиву:
    cout<<"enter n=";
    cin>>n;
    // створюємо масив:
    DArray=new int[n];
    // заповнюємо масив и виводимо елементи:
    for(i=0;i<n;i++){
       DArray[i]=i+1;
       cout<<DArray[i]<<" ";
    }
    cout<<endl;
    system("PAUSE");
    return 0;
}
```

Показати результат

Показати програмний код

Деякі задачі

Векторний добуток

$$\vec{c} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = \vec{i}(a_2b_3 - a_3b_2) + \vec{j}(a_3b_1 - a_1b_3) + \vec{k}(a_1b_2 - a_2b_1)$$

VMult.cpp

```
#include <iostream>
using namespace std;

int main(){
   int i; // індексна змінна
   // масиви для запису векторів:
   double a[3], b[3], c[3];
   // вводимо перший вектор:
   cout<<"a = ";
   for(i=0;i<3;i++){
      cin>>a[i];
   }
   // вводимо другий вектор:
   cout<<"b = ";
   for(i=0;i<3;i++){
      cin>>b[i];
   }
   // розраховуємо векторний добуток:
   for(i=0;i<3;i++){
      c[i]=a[(i+1)%3]*b[(i+2)%3] -a[(i+2)%3]*b[(i+1)%3];
   }
   // результат:
   cout<<"[a,b] = (";
   for(i=0;i<3;i++){
      cout<<c[i]<<(i<2?" ":")\n");
   }
   system("PAUSE");
   return 0;
}
```

$\vec{a} = (a_1, a_2, a_3)$

$\vec{b} = (b_1, b_2, b_3)$

$\vec{c} = \vec{a} \times \vec{b} = [\vec{a}, \vec{b}]$

Показати результат

Показати програмний код

Сортування масиву (метод бульбашки)

Bubbles.cpp

```
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;
int main(){
   const int length=20; // розмір масиву
   int nums[length];    // створюємо масив
   int i,j;             // індексні змінні
   int a;               // технічна змінна
   // ініціалізація генератора випадкових чисел:
   srand(time(NULL));
   cout<<"We start with:\n";
   // заповнюємо масив випадковими числами:
      for(i=0;i<length;i++){
         nums[i]=rand()%100;
         cout<<nums[i]<<" ";
      }
   // сортуємо масив:
   for(j=1;j<=(length-1);j++){
      for(i=0;i<length-j;i++){
         if(nums[i]>nums[i+1]){
            a=nums[i+1];
            nums[i+1]=nums[i];
            nums[i]=a;
         }
      }
   }
   cout<<"\nAnd we have:\n";
   // масив після сортування:
   for(i=0;i<length;i++){
      cout<<nums[i]<<" ";
   }
   cout<<endl;
   system("PAUSE");
   return 0;
}
```

Показати результат

Показати програмний код

Задача:

Треба впорядкувати числовий масив в порядку зростання елементів.

Загальний алгоритм:

Перебираємо всі елементи масиву і порівнюємо два сусідніх. Якщо той що зліва, більше за того, що справа - міняємо їх місцями.

За один повний "прохід" масиву самий великий елемент буде вкінці масиву.

Перебираємо ще раз - "правильне" місце займе другий за величиною елемент, і так далі.

MatrixProduct.cpp

```
int main(){
    const int n=3;
    int i,j,k;
    double A[n][n],B[n][n],C[n][n];
    srand(time(NULL));
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            A[i][j]=rand()%6;
            B[i][j]=rand()%6;
            C[i][j]=0;
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            for(k=0;k<n;k++){
                C[i][j]+=A[i][k]*B[k][j];
            }
        }
    }
    cout<<"Matrix A is:"<<endl;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<A[i][j]<<"\t";
        }
        cout<<endl;
    }
    cout<<endl;
    cout<<"Matrix B is:"<<endl;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<B[i][j]<<"\t";
        }
        cout<<endl;
    }
    cout<<endl;
    cout<<"Matrix C=AB is:"<<endl;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<C[i][j]<<"\t";
```

Журнал компіляції | Налаштування | Результати пошуку

Множення матриць

$$\hat{A} = \begin{vmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{vmatrix} \qquad \hat{B} = \begin{vmatrix} b_{11} & \dots & b_{1k} \\ \dots & \dots & \dots \\ b_{n1} & \dots & b_{nk} \end{vmatrix}$$

$$\hat{C} = \hat{A} \cdot \hat{B}$$

$$\hat{C} = \begin{vmatrix} c_{11} & \dots & c_{1k} \\ \dots & \dots & \dots \\ c_{m1} & \dots & c_{mk} \end{vmatrix} \qquad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Показати результат

Показати програмний код

```
    cout<<"Matrix C=AB is:"<<endl;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<C[i][j]<<"\t";
        }
        cout<<endl;
    }
    cout<<endl;
    system("PAUSE");
    return 0;
}
```