Лекція 3. Вказівники, масиви, рядки

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

Оголошення вказівника

тип_данних *ім'я_вказівника;

Приклад:

int *p; - вказівник p на змінну типу int

double *pnt; - вказівник pnt на змінну типу double

char *s; - вказівник s на змінну типу char

Оператори:

* - значення, записане за адресою, що міститься у вказівнику

& - адреса змінної

Приклад 3.1. Створення вказівника

```
int main(){
int *q,n,*p;                    // вказівники q, p та змінна n
n=100;                          //  n=100
p=&n;     // вказівник  p містить адресу змінної n
q=p;      // вказівник  q містить адресу змінної n
(*p)++;
cout<<*q<<"\n";       // значення змінної n
cout<<n<<"\n";        // значення змінної n
cout<<p<<"\n";        // адреса змінної n
return 0;}
```

Адресна арифметика

Допустимі операції з вказівниками:

1. Присвоєння одному вказівнику в якості значення іншого вказівника. Результат – копіювання значення (адреси)

2. Вказівники можна порівнювати. Результат – логічне значення

3. До вказівника можна добавляти ціле число. Результат – адреса комірки, що відстоїть від вихідної на відповідну кількість комірок “вправо”

4. Від вказівника можна віднімати ціле число. Результат – адреса комірки, що відстоїть від вихідної на відповідну кількість комірок “вліво”

5. Можна розраховувати різницю вказівників. Результат – кількість комірок між відповідними адресами

6. Вказівники можна індексувати (див. далі про масиви). Результат – адреса комірки, що відстоїть від даної на відповідну кількість комірок

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 в С++

1. Індексація елементів масиву завжди починається з нуля

2. В С++ всі елементи масиву в пам’яті розміщуються підряд, один за одним

3. В С++ немає перевірки на предмет виходу за межі масиву

4. Ім’я масиву є вказівником на його перший елемент (елемент з нульовим індексом)

Приклад 3.2. Створення одномірного масиву

```
#include<iostream>
#include<cstdlib>
using namespace std;
int main(){
int n[10];   // масив n цілих чисел розміру 10
for(int i=0;i<10;i++){
n[i]=rand() % 10; //заповнення масиву
cout<<n[i]<<" ";} //виведення елементів
cout<<"\n";
return 0;}
```

Приклад 3.3. Вказівник на масив

```
#include <iostream>
using namespace std;
int main(){
int n[10],*p;  // масив n і вказівник p
p=n;           // p містить адресу ел-ту n[0]
for(int i=0;i<10;i++){
p[i]=10-i;     // p[i] - елемент n[i]
cout<<*(p+i)<<" ";}  // p+i – адреса ел-ту n[i]
cout<<"\n";
return 0;}
```

Результат: 10 9 8 7 6 5 4 3 2 1

Приклад 3.4. Вказівник на масив - 2

```
#include <iostream>
using namespace std;
int main(){
int n[10];
for(int i=0;i<10;i++){
*(n+i)=10-i;
cout<<n[i]<<"\n";}
return 0;}
```

Результат: 10 9 8 7 6 5 4 3 2 1

Приклад 3.5. Заповнення двомірного масиву випадковими числами

```
#include<iostream>
using namespace std;
int main(){
int n[4][5];
for(int i=0;i<4;i++){
for(int j=0;j<5;j++){
n[i][j]=rand() % 10;
cout<<n[i][j]<<" ";}
cout<<"\n";}
return 0;}
```

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]={2,4,6,3}; або

int n[]={2,4,6,3}; - розмір масиву визначається автоматично

```
double numbers[3][2]{{1.1,3.2},
                     {8.3,5.4},
                     {9.5,2.6}};

double numbers[][2]{{1.1,3.2},
                    {8.3,5.4},
                    {9.5,2.6}};
```

Масиви символів (текстові рядки)

В C++ існує два способи реалізації текстових рядків

1. У вигляді символьних масивів (масиви типу char)

2. Об’єкти класу string

Текстові рядки у вигляді char-масивів

1. Оголошуються як звичайні масиви: char str[80];
2. Розмір масиву не співпадає з розміром текстового рядка
3. Ознакою закінчення текстового рядка є спеціальний нуль-символ ‘\0’ (не плутати з нулем!)
4. Символьні масиви можна ініціалізувати текстовим рядком: char str[80]=“Hello, World!”;
5. Існують спрощені алгоритми роботи з символьними масивами, наприклад: cout<<str;

Приклад 3.6. Ініціалізація символьного масиву

```
#include <iostream>
using namespace std;
int main(){
char str1[20]="hello";
char str2[20]={'h','e','l','l','o','\0'};
cout<<str1<<"\n";
cout<<str2<<"\n";
return 0;}
```

Приклад 3.7. Зчитування рядка

```
#include <iostream>
#include <cstdio>
using namespace std;
int main(){
char str[100];
cout<<"Enter your text, please: ";
cin>>str;  // краще використати команду gets(str)
cout<<"Your text is: "<<str<<endl;
return 0;}
```

Приклад 3.8. Поелементне виведення рядка на екран

```
#include <iostream>
#include <cstdio>
using namespace std;
int main(){
char str[20];
cout<<"Enter a string: ";
gets(str);
for(int i=0;str[i];i++)
cout<<str[i];
cout<<endl;
return 0;}
```

ЗВЕРНІТЬ УВАГУ!!!

Приклад 3.9. Довжина рядка

```
#include <iostream>
#include <cstdio>
using namespace std;
//Функція для визначення довжини рядка:
int length(char *str){
int i=0;
while(str[i]){
i++;}
return i;}
int main(){
char str[80];
cout<<"Enter a string: ";
gets(str);
cout<<"String length is "<<length(str)<<endl;
return 0;}
```

Існує стандартна функція
strlen()
доступна після підключення заголовка
<cstring>

Функції для роботи з рядками

Функція	Заголовок	Опис
`strcpy(s1,s2)`	`<cstring>`	Копіювання рядка `s2` в рядок `s1`
`strcat(s1,s2)`	`<cstring>`	Рядок `s2` приєднується до рядка `s1`
`strcmp(s1,s2)`	`<cstring>`	Порівняння рядків `s1` та `s2`: якщо рядки рівні, повертається значення `0`
`strchr(s,ch)`	`<cstring>`	Вказівник на першу позицію символу `ch` в рядку `s`
`strstr(s1,s2)`	`<cstring>`	Вказівник на першу позицію підрядка `s2` в рядку `s1`
`atoi(s)`	`<cstdlib>`	Перетворення рядка з цифр `s` в ціле число типу `int`
`atol(s)`	`<cstdlib>`	Перетворення рядка цифр `s` в ціле число типу `long`
`atof(s)`	`<cstdlib>`	Перетворення рядка цифр `s` в дійсне число типу `double`
`tolower(ch)`	`<cctype>`	Перетворення символу `ch` до нижнього регістру
`toupper(ch)`	`<cctype>`	Перетворення символу `ch` до верхнього регістру

Текстові літерали

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

Приклад 3.10. Робота з літералами

```
#include <iostream>
using namespace std;
int main(){
char *p,*q;
p="Hello, World!"; //вказівник на перший символ рядка ‘H’
q="Hello, World!"+7; //вказівник на 8-й символ рядка ‘W’
cout<<p<<endl;
cout<<q<<endl;
cout<<*p<<endl;
p++;//вказ. на 2-й символ
cout<<*p<<endl;
return 0;}
```

Результат виконання програми:

```
Hello, World!
World!
H
e
```

Динамічне виділення мап'яті

Оператори динамічного розподілу пам'яті:

new – виділення пам'яті

delete – вивільнення пам'яті

Змінна:

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

Приклад:

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

Масив:

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

Приклад:

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

Приклад 3.11. Динамічний масив

```
#include <iostream>
using namespace std;
int main(){
int *p,n;
cout<<"Enter n = ";
cin>>n;
p=new int[n];
for(int i=0;i<n;i++){
p[i]=2*i+1;
cout<<p[i]<<" ";}
delete [] p;
cout<<endl;
return 0;}
```

ЗВЕРНІТЬ УВАГУ!!!

Результат:

```
Enter n = 10
1 3 5 7 9 11 13 15 17 19
```

ПРИКЛАДИ ВИКОРИСТАННЯ МАСИВІВ

Векторний добуток

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

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

Вихідні вектори

Векторний добуток

$$\vec{c} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} =$$

$$= \vec{i}(a_2b_3 - a_3b_2) + \vec{j}(a_3b_1 - a_1b_3) + \vec{k}(a_1b_2 - a_2b_1)$$

Приклад 3.12. Векторний добуток

```
int main(){
int i;
double a[3], b[3], c[3];
cout<<"a = ";
for(i=0;i<3;i++) cin>>a[i];
cout<<"b = ";
for(i=0;i<3;i++) cin>>b[i];
//Результат:
for(i=0;i<3;i++) c[i]=a[(i+1)%3]*b[(i+2)%3] -
a[(i+2)%3]*b[(i+1)%3];
//Відображення результату:
cout<<"[a.b] =";
for(i=0;i<3;i++)
cout<<" "<<c[i];
cout<<endl;
return 0;}
```

ЗВЕРНІТЬ УВАГУ!!!

Результат:

a = **0 1 0**

b = **1 0 0**

[a.b] = 0 0 -1

Приклад 3.13. Сортування масиву (метод бульбашки)

```
int main(){
const int m=10;
int MyArray[m],i,j,s;
cout<<"Before:\n"; // Вихідний масив:
for(i=0;i<m;i++){
MyArray[i]=rand() % 20;
cout<<MyArray[i]<<" ";} // Сортування масиву:
for(j=1;j<=(m-1);j++)
for(i=0;i<m-j;i++)
if(MyArray[i]>MyArray[i+1]){
s=MyArray[i+1];
MyArray[i+1]=MyArray[i];
MyArray[i]=s;}
cout<<"\nAfter:\n"; // Масив після сортування:
for(i=0;i<m;i++)
cout<<MyArray[i]<<" ";
cout<<"\n";
return 0;}
```

Результат:

Before:
1 7 14 0 9 4 18 18 2 4
After:
0 1 2 4 4 7 9 14 18 18

Приклад 3.14. Множення матриць

```
int main(){
const int N=3;
int i,j,k;
double A[N][N],B[N][N],C[N][N];
cout<<"Matrix A:\n";
for(i=0;i<N;i++)
for(j=0;j<N;j++) cin>>A[i][j];
cout<<"Matrix B:\n";
for(i=0;i<N;i++)
for(j=0;j<N;j++) cin>>B[i][j]; //Результат:
cout<<"Matrix C=AB:\n";
for(i=0;i<N;i++){
for(j=0;j<N;j++){ C[i][j]=0;
for(k=0;k<N;k++) C[i][j]+=A[i][k]*B[k][j];
//Вивід на екран:
cout<<C[i][j]<<" ";}
cout<<endl;}
return 0;}
```

Результат:

Matrix A:

1 1 1
1 0 1
0 0 1

Matrix B:

2 1 0
0 2 1
1 1 1

Matrix C=AB:

3 4 2
3 2 1
1 1 1

$$\hat{C} = \hat{A} \cdot \hat{B} \quad \Rightarrow \quad c_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj}$$

Резюме - 1

1. Вказівник – змінна, значенням якої є адреса пам’яті. При оголошенні вказівника використовують оператор *.

2. Для отримання адреси змінної перед її ім’ям вказують оператор &, а для доступу до значення, записаного за адресою вказівника, перед вказівником використовують оператор *.

3. До вказівників можна додавати та віднімати цілі числа, їх можна індексувати та розраховувати різницю вказівників. Також вказівники можна порівнювати та присвоювати одному вказівнику значення іншого.

Резюме - 2

4. Масив – набір змінних одного типу, об’єднаних спільним іменем.

5. При оголошенні масиву вказують тип його елементів, назву та розмір (в квадратних дужках).

6. Індексація елементів масиву починається з нуля.

7. В пам’яті елементи розміщуються послідовно.

8. Перевірка на предмет виходу за межі масиву не виконується.

9. Ім’я масиву є вказівником на його початковий елемент.

Резюме - 3

10. Текстовий рядок може бути реалізований у вигляді символьного масиву.

11. Ознакою закінчення масиву є нуль-символ.

12. Динамічний розподіл пам’яті виконується за допомогою операторів new (виділення пам’яті) і delete (звільнення пам’яті).

13.Привиділенні пам’яті оператором new повертається вказівник на виділену комірку (для змінної) або вказівник на початковий елемент масиву.

24 жовтня на другій півпарі (о 9.25)

ТЕСТ!!!

Теми:

1. Основи синтаксису (Лекція 1)
2. Керуючі інструкції (Лекція 2)
3. Вказівники, масиви, рядки (Лекція 3)

Приклади тестових завдань та pdf-файли лекційних презентацій на сторінці

www.vasilev.kiev.ua/ci.php