Указатели

Определения

- Указателят е променлива, която съдържа адрес от паметта
- На този адрес обикновено се намира някакъв програмен елемент – например променлива, масив, обект, функция
- Адресът на даден елемент е положително число, указващо къде в паметта (т.е. на колко байта спрямо началото ѝ) е разположен елемента.

Деклариране на указател

*тип *име*;

тип – типът данни, към който сочи указателят

име – името на променливата от тип указател

действие:

декларира се променлива с указаното *име*, която ще сочи към данни от указания тип. Може да четете '*' "стойността на"

пример:

```
int *pa, a;  // pa е указател (който сочи) към цяло число,
                a е променлива, съдържаща цяло число
```

Получаване на адрес

&име

име – името на променливата (или елемента), на който искаме да получим адреса. '&' означава "адресът на".

особености:

- ако присвояваме адреса на указател, той трябва да е от същия тип
- за да покажем че указател не сочи към нищо, можем да му присвоим NULL (или 0)

пример:

```
int a = 5;              // a е променлива, съдържаща 5
int *pa = &a, *pb;      // pa сочи към a, pb сочи към произв.адрес
pb = 0;                 // сега вече pb не сочи към нищо
```

Достъп до стойност

**(израз)*

израз – указва адрес на програмен елемент, към чиято стойност искаме да се обърнем. '*' означава "стойността на".

действие:

- стойността на елемента, намиращ се на указания адрес, се използва или променя

пример:

```
int a = 5;          // a съдържа 5
int *pa = &a;       // pa сочи към a
(*pa)++;            // увеличаваме стойността, към която сочи, с 1
cout << *pa;        // извеждаме тази стойност
```

Псевдоними

Псевдонимът е друго име на дадена променлива.

Декларира се с

тип &име = променлива;

където

тип – типът данни, към който сочи псевдонимът

име – името на променливата от тип псевдоним

променлива – променливата, към която сочи псевдонимът

пример:

```
int a=5;        // a съдържа 5
int &b = a;     // b сочи към a
b = 0;          // сега вече a = 0
```

Пример за указатели

```
float x, y, *px, *py;
x = 3.14;       // x си присвоява 3.14
px = &x;        // px вече сочи към x
y = *px + 10;   // y си присвоява x + 10;
py = px;        // py сочи към същото като px
*py = 1;        // новата му стойност ще е 1
cout << y;      // какво ще се отпечата?
cout << x;      // какво ще се отпечата?
```

Какво се случва на всеки ред?

```
int n = 5, a;
int* pa = &n;
a = *pa;
*pa = 7;
cout << n << endl;
cout << a << endl;
```

```
int a = 5, b = 15;
int * p1, * p2;
p1 = &a;
p2 = &b;
*p1 = 10;
*p2 = *p1;
p1 = p2;
*p1 = 20;
cout << "a=" << a << endl;
cout << "b=" << b << endl;
```

```
float y, x, *px;
x = 3.14; px = &x;
y = ++*px;
```

```
float y, x, *px;
x = 3.14; px = &x;
y = (*px)--;
```

```
void func(int &a, int &b){
   a--; b++;
}
...
int a=5, b=6;
func(a, b);
```

Да обобщим:

- Указателят е променлива, която сочи към даден програмен елемент
- Ако не му е указана стойност, сочи към произволен адрес от паметта
- За да не сочи към нищо, трябва да му присвоим 0 или NULL
- Чрез указател също можем да променим стойността на елемента
- Два указателя могат да сочат към един и същ програмен елемент
- Псевдонимите са вид указатели, с които се работи като с променливи

Край