13. Указатели и операции с тях. Адресна аритметика.

Указатели

Указателят е променлива, чиято стойност е адрес от паметта.

```
име_на_тип *име_на_указателя
```

След като се дефинира даден указател, той съдържа произволна стойност. Един указател може да бъде инициализиран с нулева стойнтост. За целта се използва макрос с име **NULL**, който е дефиниран в *stdio.h*. Това означава, че указателят е свободен и не сочи нищо.

Стойностите на указателите не са цели числа.

Операции с указатели

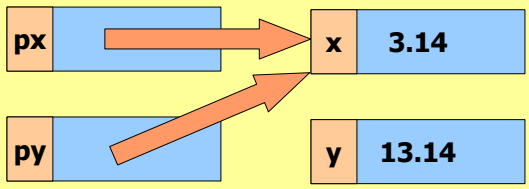
```
&операнд
```

Операндът трябва да бъде променлива, а операцията дава адреса на променливата.

```
*операнд
```

Операндът е израз, чиято стойност е адрес. Операцията * дава променливата, чиито адрес е операндът.

```
float x, y, *px, *py;
x = 3.14;
px = &x;
y = *px + 10;
py = px;
```




```
double x;
int *px;
px = (int*)(&x);          // px = &x; ще бъде грешка
```

Във втория пример не можем директно да присвоим на *px* адреса на *x*, тъй като *x* и *px* са от различен тип. Ако го направим, тъй като *px* е указател към променлива от тип int (int е 4 байта, а double – 8), то *px* ще сочи само към първите 4 байта от *x* и съответно получените данни ще бъдат некоректни.

Адресна аритметика

При адресна аритметика са валидни само операциите събиране и изваждане на указател и цяло число. При добавяне или изваждане на цяло число, всяка

единица всъщност е равна на броя байтове, отделени за дадения тип от данни.

```
double x, *px;
px = &x;
px = px + 4;              // стойността на px + 4x8, т.е. + 32
```

Много често се използват следните комбинации от операции, в случаите, когато два или повече указателя сочат към променливи, между които съществува връзка, например към различни елементи на масив. Такива указатели могат да бъдета изваждани един от друг. Например *p2 – p1* ще определи броя на елементите между елементите, към които сочат *p1* и *p2*.

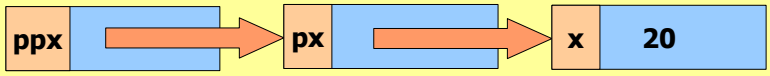
Указатели от тип void

На указател от тип void може да бъде присвоена стойността на указател от произволен тип. В ANSI C и обратното е вярно.

```
void *gp;
int *ip;
gp = ip;
//ip = gp;                // само в ANSI C
ip = (int*)gp;            // само в C++
```

Може да се използва указател към указател (в C и C++). Това разширява косвения достъп:

```
int x, *px, **ppx;
x = 20;
px = &x;
ppx = &px;
// **ppx, *px и x са еквивалентни
```



На указател не може да се присвои адресът на променлива от тип **const**. Възможно е указателят да е от тип **const**.

```
const int initial = 100;
//int *ptr = &initial;   // неправилно
const int *ptr = &initial;
```

Следваща тема: *14. Указатели и масиви. Операции с отделни битове.*