14. Указатели и масиви. Операции с отделни битове.

Указатели и масиви

Указател към масив се дефинира по общите правила, като се спазват изискванията за еднаквост на типа на укацателя и на масива.

```
double array[20];
double *pa;

pa = array;
pa = &array[0];
```

```
array + i;
&array[i];

*(array + i);
array[i];
```

Всеки два израза от горепосочените са еквивалентни. Те демонстрират двата начина за обръщение към елементите на масива:

- чрез индекси
- чрез указатели

Използването на указатели е по-бърз начин за достъп до елементите на масив

Повечето функции в C/C++, които обработват низове, използват указатели и адресна аритметика.

```
int strcmp(char*, char*);
```

Функцията **strcmp()** сравнява два низа. Ако те са равни, функцията връща **0**, в случай, че първият е „по-голям" връща **1**, а ако вторият е „по-голям" - **-1**.

```
int strcmp(char *s1, char *s2) {
     while(*s1)
          if(*s1++ != *s2++)
               return *--s1;
     if(*s2)
          return s2;
     return 0;
}
```

В езика C/C++ няма вградени операции, обработващи низове. Вместо тях се използват стандартните функции от библиотеките. Те осъществяват достъпа до всеки низ от програмата с помощта на указател към този низ.

```
char *ps;
ps = "Това е низ.";
```

Тук низът не се копира, а се инициализира указателя *ps* с адреса на първия елемент на низа. Тъй като указателите са променливи е възможно да се обединият в масиви от указатели.

```
int x1, x2, x3, x4;
int *px[] = {&x1, &x2, &x3, &x4};
```

```
char *error[] = {
     "грешка",
     "предупреждение",
     "фатална грешка - край",
     "няма грешка"
}
```

Всеки елемент в масива *error* е указател към съответния низ, а отделните низове могат да бъдат съхранявани на различни места в паметта.

Името на двумерния масив е указател константа към указателя на първия елемент н амасива.

```
#define DIM1 4
#define DIM2 3

double matrix[DIM1][DIM2];
```

Всеки подмасив може да се укаже чрез използването на името на масива и левите индекси. Например, следващите два записа са еквивалентни:

```
matrix[i]
&matrix[i][0]
```

При достъп до елементите на масив може да се използва както указател към началото на масива и после той да се увеличава, така и и указатели към редовете на масива.

Да се върнем към примера с двумерния масив по-горе!

```
double *pa;

//pa = matrix;                  // грешен запис
pa = *matrix;
pa = (double*)matrix;
```

Последните два реда са еквивалентни. Първият вариант е валиден само за двумерен масив. При повече измерения ще се увеличава броя на звездите (по една за всяко). Вторият вариант е по-добрият, тъй като действа за всякакви масиви, без да се налага да променяме броя на звездите.

А сега и пример с двумерни масиви (показаните записи също са еквивалентни):

```
matrix[i][j]
*(matrix[i] + j)
*(*matrix + i*DIM2 + j)
```

В действителност всеки двумерен масив може да се представи като едномерен масив от указатели към едномерни масиви с еднаква дължина.

Операции с отделни битове

Езикът C/C++ разполага с пълен набор от операции за обработване на отедлни битове от целочислени стойности. Тези операции позволяват проверяване, установяване или преместване на отделни битове в стойности от тип *int* (*char*). Тези операции често се използват за създаване на драйверни програми за създаване на връзки с външни устройства. По действие се основават на известните логичести операции, прилагани за отделни битове.

Поразрядно логическо **И**:

```
операнд1&операнд2
```

Операцията се използва за анулиране на отделни битове в числовата стойност на резултата. Ако някой бит в един от операндите е **0**, съответния по позиция бит в операнда също има стойност 0.

Поразрядно логическо **ИЛИ**:

```
операнд1|операнд2
```

Операцията се използва за задаване на стойност **1** на отделни битове в резултата. Всеки бит, чиято стойност е **1** в операндите, определя стойност **1** на съответния бит в резултата.

Поразрядно логическо **ИЗКЛЮЧВАЩО ИЛИ**:

```
операнд1^операнд2
```

При тази операция, определени битове в резулатат се установяват равни на 1, само ако съответните битове в операндите им са различни.

Следваща тема: *15. Функции и указатели. Указатели към функции.*

Структури от данни и обектно ориентирано програмиране
По лекции на доц. Стоян Бъчваров (СУ, ФМИ)