Упражнение №8

УКАЗАТЕЛИ. УКАЗАТЕЛИ И МАСИВИ. ОПЕРАЦИИ С УКАЗАТЕЛИ

1. Указатели

Указателят е променлива, съдържаща адрес на друга променлива.

Деклариране на указател:

<тип> *<име_на_указател>;

Пример 1.

```
int *px, *py, x, y;
```

В горният пример имаме декларирани два указателя (**px** и **py**) и две променливи (**x** и **y**) от целочислен тип.

За да можем да използваме един указател, той трябва да е инициализиран, т.е. на него да е присвоен адресът на конкретна променлива.

Пример 2.

```
px = &x;
py = &y;
```

Забележка: Типът на указателя трябва да съответства на типа на променливата, чийто адрес съдържа.

Чрез използване на операция за косвена адресация (*****) и указател се обръщаме към съдържанието на променлива, чийто адрес се съдържа в указателя.

Пример 3.

```
*px = 5;
```

По този начин променяме стойността на променливата **x**, към която сочи указателят **px**. Това ни позволява да променим стойността на променлива, декларирана в друга функция, чрез използване на указателя към тази променлива.

Пример 4. Размяна стойностите на две променливи.

```
#include <stdio.h>
void x_y_change(int *u, int *v);
int main( )
{
   int x = 5, y = 10;
   printf(" x = %d     y = %d \n", x , y);
   /* предаваме адресите на променливите към функцията */
   x_y_change(&x, &y);
   printf(" x = %d     y = %d \n", x , y);
   return 0;
}
void x_y_change(int *u, int *v)
{
   int temp;
   temp = *u;
   *u = *v;
   *v = temp;
}
```

2. Указатели и масиви

Когато предаваме масив като параметър на функция, при извикване на функцията чрез неговото име ние предаваме адреса на първия елемент на масива. В списъка на

формални параметри при описанието на функцията като формален параметър можем да използваме както име на масив и индексиране, така и указател.

Пример 5.

```
int main ( )
{
   int a [10];
…….
   f ( a,…);
…….
}
void f (int b[ ],...) { ...}
```

или

```
void f (int *b,...) { ... }
```

Във функцията, за да се обърнем към елемент на масива, можем да използваме както индексна операция, така и операция за косвена адресация.

Пример 6.

`b[i] = 5;` или `*(b+i) = 5;`

Пример 7. Да се състави функция, намираща индексите на най-малкия и най-големия по стойност елемент на едномерен масив от целочислен тип. Функцията да се използва в програма, която отпечатва най-малкия и най-големия по стойност елемент. Въвеждането на входните данни и извеждането на резултатите да става във извикващата (главната) функция.

```
#include <stdio.h>
void minmax (int a[ ], int *imin, int *imax, int n);
int main ( )
{
   int i, a[10], imin, imax;
   for(i=0; i<10; i++)
   {
        printf("a[%d]=", i);
        scanf("%d", &a[i]);
   }
   printf("Masiv\n");
   for(i=0; i<10; i++)
   {
        printf("%d ", a[i]);
   }
   printf("\n");
   minmax( a, &imin, &imax, 10);
   printf("min = %d\n", a[imin] );
   printf("max = %d\n", a[imax] );
   return 0;
}
void minmax (int a[ ], int *imin, int *imax, int n)
{
   int i, min, max;
   min = a[0];
   *imin = 0;
   max = a[0];
   *imax = 0;
   for(i =1; i < n ; i++)
   {    if(a[i] < min)
        {    min = a[i];
             *imin = i;
        }
```

```
            if(a[i] > max)
            {     max = a[i];
                  *imax = i;
            }
      }
   }
```

Функцията **minmax()** връща два резултата, които се представят като указатели: ***imin** за индекса на минималния и ***imax** за индекса на максималния елемент на едномерен целочислен масив **a** с **n** елемента. При извикването на **minmax()** се предават адресите на променливите.

3. Операции с указатели

По отношение на указателите могат се използват следните операции:

- *присвояване*: на указателя може да се присвои адрес. Това става или чрез името на масив, или чрез операцията за извличане на адрес **&**;

Пример 8.

```
int x, m, *px, a[10], *pa;
px = &x;
pa = a; /* на указателя pa се присвоява адресът на първия елемент на
           масива a */
```

- *индиректно извличане на съдържание*: чрез операция ***** можем да извлечем или променим съдържанието на променливата, чийто адрес се съдържа в указателя.

Пример 9.

```
int x, m, *px;
px = &x;
*px = 5;
m = *px; /* на променливата m присвояваме стойност 5 */
```

- *обръщение към адреса на указател*;
- *увеличаване стойността на указател с константа*: **+** или **++;**
- *намаляване стойността на указател с константа*: **--;**
- *разлика между два указателя*.

Пример 10. Какво точно ще изведе следният програмен фрагмент на C?

```
int x[]={10,11,12,13,14,15}, *y, *p;
for(p = x; p < x+6; p+=2)
   printf("%3d", *p);
printf("\n");
y = x + 3;
printf("%d %d %d", *(y-2), *y, *(y+2));
*y = 100;
for(p = x; p < x+6; p++)
   printf("%4d", *p);
printf("\n");
```