Тема 11
Програмни модули в език C/C++. Функции
(*продължение*)

5. Област на действие на променливите

Всеки *идентификатор* (име на променлива, константа, функция и др.) има определена област на действие. Обикновено, областта на действие на даден идентификатор обхваща *частите на програмата, в която е деклариран и се използва*. Областта на действие се нарича още ***област на видимост***. Според областта си на действие променливите се разделят на: ***автоматични*** и ***външни***.

5.1. Автоматични променливи

Автоматични променливи са параметрите на функциите и променливите, дефинирани в телата на функциите. Основни характеристики на тези променливи са:

- достъпни са само за функцията, в която са дефинирани; по този начин, автоматичните променливи са ***локални*** по отношение на функциите;
- създават се при всяко извикване на функцията и престават да съществуват след нейното завършване.

Следващият програмен код е пример за дефиниране и извикване на функция, която пресмята N! (N факториел).

```
#include <stdio.h>

int Factoriel(int n);          // декларация на функция

main()
{
      int n,fact;
      printf("n=");
      scanf("%d",&n);
      fact = Factoriel(n);     // извикване на функция
      printf("N Factoriel = %d",fact);

      return 0;
}

int Factoriel(int n)           // дефиниция на функция
{
      int nf=1,i;
      for(i=1;i<=n;i++)
            nf*=i;
      return nf;
}
```

Тъй като автоматичните променливи са локални, в различните функции могат да се използват променливи с еднакви имена. Въпреки съвпа-

дението на имената, едните променливи нямат нищо общо с другите. Например, променливите **n, fact** са локални за функция ***main()*** и те са недостъпни във функция ***Factoriel().*** Локални променливи за функция ***Factoriel()*** са параметърът **n** и локалните променливи **i, nf**. На практика, параметърът n и локалната променлива n, дефинирана във функцията main() са две различни променливи.

Автоматичните променливи могат да се инициализират при тяхното дефиниране. Тяхната инициализация е всеки път, когато се извиква функцията. След изпълнението и, автоматичните променливи престават да съществуват и стойностите им се губят. Паметта, необходима за автоматичните променливи се заделя в област от паметта, наречена ***стек***.

5.2. Външни променливи

Външни променливи са тези, които са дефинирани извън тялото на която и да е от функциите, съставящи дадена програма. В общия случай, такива променливи са: масиви, структури, обекти. Външните променливи се различават от автоматичните по следните особености:

- външните променливи се създават еднократно при стартиране на програмата и съществуват през цялото й изпълнение; в този смисъл те са ***глобални променливи***;
- достъпни са за множество функции; в повечето случаи, външните променливи са достъпни за всички функции;
- външните променливи не се разполагат в стека, а в друга област от паметта, наречена ***статична памет***.

Пример:

```
#include <stdio.h>

int a;                  // дефиниране на глобална променлива

void FunctionCall();

main()
{
      a=0;

      int n;
      printf("\nHow many times you need to call function? ");
      scanf("%d",&n);
      for(int i=0;i<n;i++) FunctionCall();
      printf("Function is called %d times",a);

      return 0;
}

void FunctionCall()
{
      a++;
      return;
}
```

В примера, променливата **a** е външна променлива и е достъпна за всички функции. Стойността ѝ се изменя както в главната функция, така и във функцията FunctionCall(). В случая стойността на променливата a нараства с единица при всяко извикване на функцията. Инициализацията на променливата е възможна още в нейната дефиниция:

```
int a = 0;
```

Външните променливи се използват в следните случаи:

- Определени променливи се използват от повечето от функциите в програмата. В случая, по-удобно е тези променливи да бъдат декларирани като външни.
- Съществуват функции с голям брой параметри. Всяко извикване на такива функции изисква представяне на дълъг списък с фактически параметри, от което програмите стават трудно разбираеми. Съответно, броят на параметрите може да бъде намален, като част от тях, се дефинират като външни променливи.
- Външни променливи се използват и за функции, които работят с общи данни, но нямат обръщение една към друга.

6. Предаване на параметри чрез указатели и псевдоними

Функциите взаимодействат помежду си чрез: *предаване на параметри*, *връщане на стойност* и чрез *външните (глобални)* променливи.

Предаването на параметри е по три начина:

- ***по стойност;***
- ***чрез указател;***
- ***чрез псевдоним.***

Особеност на предаването на параметри чрез стойност е, че фактическите параметри, с които се извиква функцията, се копират в стека и функцията не работи с оригиналните параметри, а с техни копия. По този начин, функцията не може да променя стойностите на оригиналните параметри.

Когато е необходимо функциите да работят с оригиналните параметри, а не с техните копия, параметрите трябва да бъдат предадени чрез указатели или чрез псевдоними. Така функцията получава достъп до оригиналните стойности и може да ги промени.

Предаването на параметрите на функциите чрез указатели се регламентира по следните правила:

- формалните параметри на функцията се декларират като указатели;

- в тялото на функцията се използва съдържанието на указателите, като формални параметри;
- при извикване на функцията, фактическите параметри също трябва на са указатели.

Пример за предаване на параметри чрез указатели е функцията за размяна на две числа ***swap().*** В случая, функцията работи с оригиналните стойности на променливите p и q, тъй като тя получава като фактически параметри, адресите на тези променливи.

```
#include <iostream.h>

void swap(int *, int *);
main()
{
      int p,q;
      cin>>p;
      cin>>q;
      swap(&p,&q);            // на x и y се предават адресите на p и q
      cout<<"p="<<p<<"q="<<q;
      return 0;
}
void swap( int *x, int *y)    // x,y са указатели
{
      int b;
      b=*x;       // разменя се съдържанието на клетките на адреси x,y
      *x=*y;
      *y=b;
      return;
}
```

При предаването на параметрите чрез указатели трябва да се има предвид, че функцията работи с оригиналните стойности на параметрите, а не с техни копия и може да промени тези стойности. Неправилното използване на предадените чрез указатели параметри, може да доведе до разрушаване на необходима информация.

Предаването на параметри чрез указатели се използва предимно когато е необходимо функцията да върне повече от един резултат. Тъй като стойностите на тези параметри могат да бъдат променяни, те се използват като ***входно-изходни*** параметри.

Това може да бъде илюстрирано чрез функция ***swap()***. На практика, функцията не връща резултат, а променя двете оригинални стойности на променливите **p** и **q** от функция ***main()***.

Предаването на параметри чрез указатели е валидно както в C, така и в C++. Но в C++ е възможно параметрите да бъдат предадени и чрез псевдоними, което няма аналог в C. Предаването на параметри чрез псевдоними е еквивалентно на предаване на параметри чрез указатели, но е по удобно от синтактична гледна точка. В този случай единствено в прототипа на функцията се посочва, че като параметър се изисква псевдоним, а в самата функция, както и при нейното извикване, с псевдонимите се

работи така, както и с обикновени променливи. Следващият пример е вариант на функцията ***swap()***, в който вместо чрез указатели, параметрите се предават чрез псевдоними:

```
#include <iostream.h>

void swap(int &, int &);

main()
{
      int p,q;
      cin>>p;
      cin>>q;
      swap(p,q);
      cout<<"p="<<p<<"q="<<q;
      return 0;
}

void swap( int &x, int &y)   // x,y са псведоними на променливи от тип int
{
      int b;
      b=x;
      x=y;
      y=b;
      return;
}
```

Относно въпроси по темата на адрес: ln_zh_st@yahoo.com