ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Урок №8

Програмиране на език C

Съдържание

1. Въвеждане в света на функциите

Необходимост от използване. Обявяване. Извикване

Много често се получава ситуация, когато в програмата някои отрезки на кода се повтарят няколко пъти и на нас ни се налага много често да набираме един и същ фрагмент от кода. Такава ситуация има своите отрицателни страни. Първо, процеса за създаване на програмата става доста нуден и дълъг, второ, увеличава се обема на крайния файл. Какво да правим в такъв случай?! Съществува ли някакъв механизъм позволяващ да се автоматизират действията на програмиста и да се съкрати кода на програмата? Да, съществува, този механизъм се нарича функция. И така:

Функция — специална конструкция с помощта на която, който и да е фрагмент от кода повтарящ се в програмата два или повече пъти се изкарва извън тялото на програмата. Този фрагмент получава собствено име и, в бъдеще за да се възползваме от изнесения код е нужно да се укаже неговото име.

Синтаксис за обявяване

Съществуват два метода за обявяване на функция:

- Функцията се обявява до функцията main.

Функцията се обявява с помощта на прототипа а след функцията main, се описва тялото на обявената функция.

Първи начин: преди функцията main.

Общ синтаксис за обявяване на функцията:

```
Възвръщаемо_значение име_на_функцята (параметри)
{
        Блок_на_повтарящсе_код (тяло);
}
```

1. Името на функцията се подчинява на същите правила, както и името на променливите и естествено се избира от програмиста.

2. Параметри – входни данни, които са необходими на функцията за да работи с кода. В качеството на параметри се използват обикновени променливи, указващи за всеки параметър неговия тип данни. Ако функцията не се нуждае от входни данни, скобите следва да се оставят празни. Второто име на параметрите-аргументи.

3. Възвращаемо значение – резултат от работата на функцията. На мястото на възвращаемото значение се представя който и да е от базовите типове. Това е този тип данни, които функцията ще постави на мястото на нейното извикване от програмата. Ако функцията нищо не връща, на мястото на възвращаемото значение се поставя void(пусто). Като цяло „осмислено“ възвращаемо значение се указва в такъв случай, ако резултата от работата на функцията е необходим за следващи изчисления.

Забележка: Не трябва да се създава една функция в друга.

Забележка: Не трябва да се извиква функция преди нейното обявяване.

Синтаксис за извикване на функция.

За да се възползваме от функциите на определен отрязък от кода, трябва да се извиква точно на този отрязък от кода. Извикването на функцията е задача на операционната система да започне да изпълнява фрагмент от кода, който се съдържа в тялото на функцията. Към завършването на изпълнението на функцията, програмата трябва да продължи работата си в основния код от това място, където е била извикана функцията. Извикването на функцията се състои от указването на името на функцията, предаване на аргументи (ако има такива) и получаване на възвращаемо значение(ако има необходимост да се получи) :

```
Име_на_променливата=име_на_функцията(параметър1, параметър2,....,
параметърN);
```

1. Типовете данни на значенията трябва да съответстват на типа данни на аргументите в определенията на функциите. Изключение са случаите, когато предаваното значение може да бъде с лекота преобразувано към нужния ни тип.
2. При извикването на функция трябва да се указва такова количество параметри, каквото е било определено при обявяването.
3. Типа на променливата в която ще се запише възвращаемото значение трябва да съвпада с този тип, който функцията съответно възвръща. Това не е задължително, но е желателно.

Ключовата фраза return

За връщане на значението от функция в програмата на това място от което е била извикана тази функция, се използва оператора return. Синтаксиса на връщането е:

```
return значение;
```

Ако функцията не връща никакви значения, то оператора return може да се използва просто за спиране на функцията, за това просто пишем:

```
return; // в дадения случай return ще отработи за функцията, като break за цикъла,
```

Запомнете важен момент при работата с return:

1. Оператори за възвръщане могат да бъдат няколко (в зависимост от ситуацията), но ще .
2. Ако е сработил returne (независимо от формата), всичко, което е разположено във функцията под него няма да работи.
3. Ако типа на възвращаемата форма не е void, тогава е трябва ВИНАГИ да се използва формата: **return значение**;

Сега, когато сме запознати с теорията, преминаваме към следващия раздел – примери за създаване на функции.

2. Примери за създаване и извикване на функции

Функция която не приема никакви параметри и не възвръща никакви значения.

```
#include <iostream>

using namespace std;

// създаване на
функцията void
Hello(){
          // показване на екрана редове от
          текст cout<<"Hello,
          World!!!\n\n";
}
void main(){
          Hello();
          Hello();
          Hello();
}
```

Резултат — три пъти израза — Hello, World!!! — на екрана.

Функция която приема един параметър, но не възвръща никакво значение.

```
#include <iostream>

using namespace std;

// рисува линия от звезди с
дължина count void Star(int
count){
        for(int i=0;i<count;i++)
                    cout<<'*';
         cout<<"\n\n";
}

void main(){
```

```
        Star(3); // показване на линия от 3 звезди
        Star(5); // показване на линия от 5 звезди
}
```

Функция, която приема два параметра, но все още не възвръща значения.

```
#include <iostream>
using namespace std;
// рисува линия от символа - symb, длиной
 count void AnyLine(char symb, int count){
        for(int i=0;i<count;i++)
                    cout<<symb;
          cout<<"\n\n";
 }
 void main(){

         AnyLine('+',3); // показване на линия от три плюса
        AnyLine('=',5); // показване на линия от 5 знака равно

 }
```

Функция, която приема два параметра и възвръща значение.

```
#include <iostream>
using namespace std;
// изчислява степен (Pow) на число
(Digit) int MyPow(int Digit, int Pow){
        int key=1;
        for(int i=0;i<Pow;i++)
                key*=Digit;
          return key;
}
void main(){
           // записване на възвращаемия резултат в
        променливата res int res=MyPow(5,3);
        cout<<"Res = "<<res<<"\n\n";

}
```

3. Предаване на аргументи. Прототипи на функции

Предаване на аргументи по значение.

Да поговорим за това което се случва в оперативната памет. Аргументите, които се указват при определение на функцията се наричат формални. Това е свързано с това, че те се създават в момента на повикване на функцията в оперативната памет. При изхода от функцията такива параметри ще бъдат унищожени. Затова, ако в друга функция на програмата ще бъдат параметри със същото име, то конфликт няма да има. Да разгледаме, един от начините за предаване на аргументи:

Пример за работа на формалните параметри при предаване на данни по значение.

```
#include <iostream>

using namespace std;

// трябва да променя значението на
променливите на местата void Change(int
One, int Two){
         cout<<One<<" "<<Two<<"\n\n";//
        1 2 int temp=One;
        One=Two;
        Two=temp;
          cout<<One<<" "<<Two<<"\n\n";// 2 1
}
void main(){

        int a=1,b=2;
          cout<<a<<" "<<b<<"\n\n"; //
          1 2 // предаване по
          значение
          Change(a,b);
          cout<<a<<" "<<b<<"\n\n"; // 1 2
}
```

1. Във функцията се предават не a и b a, техните точни копия.
2. Всички промени произлизат с копията (One и Two), при това самите a и b остават непроменими.
3. При изхода от функцията временните копия се унищожават.

Изхождайки от гореописаното – засега бъдете внимателни при обработване на значенията във функцията. В последствие ще се научим да решаваме дадени проблеми.

Забележка: Всъщност, с масивите такова не се случва. Всичките промени с масивите във функции се съхраняват при излизане от нея.

Някои неща за масивите...

Някои особености има при използването на масиви в качеството на аргументи. Тази особеност е в това, че името на масива се преобразува към указателя на неговия първи елемент, т.е. при предаването на масива произлиза предаване на указателя. По тази причина извиканата функция не може да се отличи, отнася ли се предавания й указател към началото на масива или към един единствен обект. При предаване на едномерен масив е достатъчно просто да се укажат празни квадратни скоби:

```
int summa (int array[ ], int
    size){ int res=0;
    for (int i = 0; i < size;
       i++) res += array[i];
     return res;
}
```

Ако във функцията се предава двумерен масив, то описанието на съответстващия аргумент на функцията трябва да съдържа .

```
int summa (int array[ ][5], int size_row, int
    size_col){ int res=0;
    for (int i = 0; i < size_row; i++) for
        (int j = 0; j < size_col; j++)
              res +=
     array[i][j]; return res;
}
```

Прототип на функциите или втори начин за обявяване.

При втория метод за обявяване на функция е необходимо да се съобщи на компилатора за това, че функцията съществува. За това преди main се предоставя името на функцията .

```
библиотеки възвращаемо значение
име_на_функцията(аргументи); void
main(){

         тело main;
}
Възвръщаемо_значение
         име_на_функцията(аргументи){ тяло
         на функцията;
}
```

```
#include <iostream>
using      namespace
std; // прототипи
void MyFunc();

void MyFuncNext();

void main(){
          MyFunc(); //MyFunc
          MyFuncNext(); //MyFuncNext
}
//описания
void MyFunc(){
          cout<<"MyFunc\n";
}
void MyFuncNext(){
          cout<<"MyFuncNext\n";
}
```

Смята се, че такова обявяване на функцията е най-красиво и правилно.

4. Област на видимост. Глобални и локални променливи

Област на видимост.

Всякакви фигурни скоби в програмния код образуват така наречената област на видимост, това означава, че променливите, обявени в тези скоби ще могат да се видят само в тези скоби. С други думи, ако към променливата създадена във функцията на цикъл if .

```
int a=5;
if(a==5){
        int b=3;
}
cout<<b; // ошибка! b не существует
```

Глобални и локални променливи

Съгласно правилата за областите на видимост – променливите се делят на два вида – локални и глобални.

Локалните променливи се създават в някакъв отрязък от кода, какво означава това за програмата, ние вече знаем.

Глобалните променливи се създават извън всякакви области на видимост. Такава променлива е видима от всяко място в програмата. По подразбиране глобалните променливи за разлика от локалните се инициализират с 0.

И главното, измененията, които се случват с глобалната променлива във функцията при изхода от последната се запазват.

Забележка: Запомнете ако има, например, глобална променлива с името a, а във функцията .

```
int a=23; // глобальная
a void main(){
        int a=7;// локальная a
          cout<<a; // 7, используется локальная
}
```

5. Аргументи (параметри) по подразбиране

На формалния параметър на функцията може да се зададе аргумент по подразбиране. Това означава, че в даден аргумент значението при извикване може да не се предава. В този случай ще бъде използвано значение по подразбиране. Общ синтаксис за реализация на такъв подход има следния вид:

```
Тип_на_връщаемото_значение име_на_функцията(тип_на_арг
име_на_арг=значение_по_подразбиране)
```

Тук значение_по_подразбиране е и значение, :

```
Тип_на_връщаемото_значение име_на_функцията (арг1=значение,
арг2=значение)
```

Аргументи по подразбиране могат да бъдат аргументи, започвайки от десния край на списъка параметри на функциите и така последователно от ляво надясно без прекъсване:

Например:

```
void foot (int i, int j = 7) ;   //допустимо
void foot (int i, int j = 2, int k) ; //недопустимо
void foot (int i, int j = 3, int k = 7) ; //допустимо
void foot (int i = 1, int j = 2, int k = 3); //допустимо

void foot (int i=- 3, int j);           //недопустимо
```

Да разгледаме пример за работа с параметри по подразбиране.

```
#include <iostream>

using namespace std;

// рисува линия от звезди с дължина
count void Star(int count=20){
        for(int i=0;i<count;i++)
                    cout<<'*';
         cout<<"\n\n";
}

void main(){

         Star(); // показване линия от 20 звезди
         Star(5); // показване линия от 5 звезди

}
```

Това беше всичко за днес. А сега – дерзайте!!! Тест днес няма да има, само домашна работа. Успех!!!

6. Домашна работа

1. Да се напише функция, която получава в качеството на аргумент цяло положително число и бройна система в която това число трябва да се преведе (бройни системи от 2 до 36). Например при превода на число 27 в бройна система 16 трябва да се получи 1B; 13 в 5-ю — 23; 35 в 18-ю — 1H.

2. Игра «кубчета». Условие: има две игрални кубчета със значения от 1 до 6. Играта се провежда с компютъра, кубчетата се хвърлят подред. Побеждава този, който има по голяма сума от точки след пет хвърляния. Предоставете възможност първия ход да е на човека или компютъра. Кубчетата се отразяват с помощта на символи. В края на играта е необходимо да се изкара средната сума от хвърляния на двамата участници.