Программирование на C++

Минцифры России

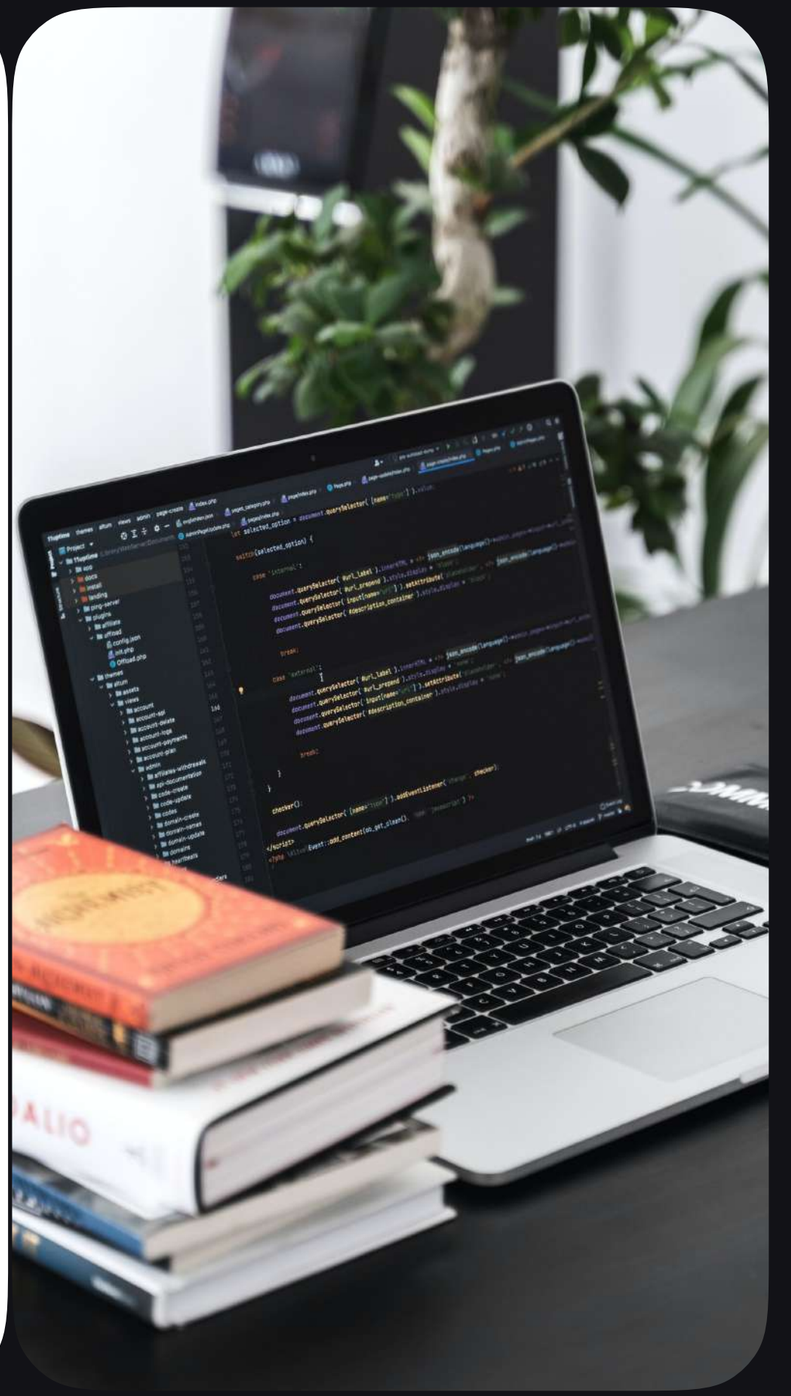
UCHi DOMA

20.35 УНИВЕРСИТЕТ

Урок 1 Модуль 2

Функции

Полезные материалы



Цели урока

- научиться создавать и вызывать функции
- отработать на практике составление алгоритмов с пользовательскими функциями на Си

Функция

Функции — это блоки кода, выполняющие определенные операции. Результат работы функции зависит от входных параметров. При необходимости функция может возвращать значение как выходное.

Функции могут многократно использоваться.

Определение функции

Каждая функция должна быть определена, то есть должны быть указаны:

- тип возвращаемого значения
- имя функции
- информация о формальных аргументах
- тело функции

Синтаксис определения функции:

```
ТипВозвращаемогоЗначения ИмяФункции(СписокФормальныхАргументов)
{
ТелоФункции;
...
return(ВозвращаемоеЗначение);
}
```

Пример

Функция сложения двух вещественных чисел

```
float function(float x, float z)
{
    float y;
    y=x+z;
    return(y);
}
```

Функция

- Если функция не возвращает значения, то тип возвращаемого значения для нее указывается как **void**.
- При этом операция return может быть опущена.
- Если функция не принимает аргументов, в круглых скобках также указывается void.

```
void function(void)
{
    printf("Hello world");
}
```

Функции

1

Встроенные

функции хранятся в стандартных библиотеках, например printf(), scanf()

2

Пользовательские

функции, написанные пользователем для решения конкретной подзадачи

Вызов функций

Общий вид вызова функции

```
Переменная = ИмяФункции(СписокФактическихАргументов);
```

Возвращаемое значение

По окончании выполнения вызываемой функции осуществляется возврат значения в точку ее вызова.

Для передачи возвращаемого значения в вызывающую функцию используется оператор return

```
return ВозвращаемоеЗначение;
```

Оператор return также завершает выполнение функции и передает управление следующему оператору в вызывающей функции.

Оператор return не обязательно должен находиться в конце тела функции.

Пример

```c
#include <stdio.h>
int sum(int x, int y)
{
    int z;
    z = x + y;
    return(z);
}
int main()
{
    int a, b, res;
    printf("a= ");
    scanf("%d", &a);
    printf("b= ");
    scanf("%d", &b);
    res = sum(a, b);
    printf("%d + %d = %d", a, b, res);
    return 0;
}
```

Задача

```c
#include <stdio.h>
int main()
{
    int sum=0,sa1,sa2,sa3;
    int m1[10]={};
    int m2[10]={};
    int m3[10]={};
    for (int i=0; i<10; i++)
        scanf("%d", &m1[i]);
    for (int i=0; i<10; i++)
        scanf("%d",&m2[i]);
    for (int i=0; i<10; i++)
        scanf("%d",&m3[i]);
    for (int i=0; i<10; i++)
        sum+=m1[i];
    sa1=sum/10;
    sum=0;
    for (int i=0; i<10; i++)
        sum+=m2[i];
    sa2=sum/10;
    sum=0;
    for (int i=0; i<10; i++)
        sum+=m3[i];
    sa3=sum/10;
    if (sa1>sa2 && sa1>sa3) printf("%d\n",sa1);
    if (sa2>sa1 && sa2>sa3) printf("%d\n",sa2);
    if (sa3>sa1 && sa3>sa2) printf("%d\n",sa3);
    return 0;
}
```

```c
#include <stdio.h>
void input(int m[10])
{
  for (int i=0; i<10; i++)
        scanf("%d",&m[i]);
}
int average(int m[10])
{
    int sum=0;
    for (int i=0; i<10; i++)
        sum+=m[i];
    return (sum/10);
}
int main()
{
    int sum=0, sa1, sa2, sa3;
    int m1[10]={};
    int m2[10]={};
    int m3[10]={};

    input(m1);
    input(m2);
    input(m3);

    sa1=average(m1);
    sa2=average(m2);
    sa3=average(m3);

    if (sa1>sa2 && sa1>sa3) printf("%d\n",sa1);
    if (sa2>sa1 && sa2>sa3) printf("%d\n",sa2);
    if (sa3>sa1 && sa3>sa2) printf("%d\n",sa3);
    return 0;
}
```

Описание функции

Нет требования, чтобы описание функции обязательно предшествовало её вызову. Функции могут определяться как до вызывающей функции, так и после нее.

- тип возвращаемого значения
- имя функции
- типы формальных аргументов в порядке их следования

Прототип необходим для того, чтобы компилятор мог осуществить проверку соответствия типов передаваемых фактических аргументов типам формальных аргументов. Имена формальных аргументов в прототипе функции могут отсутствовать.

Пример

```c
#include <stdio.h>
int sum(int, int);
int main()
{
    int a, b, res;
    printf("a= ");
    scanf("%d", &a);
    printf("b= ");
    scanf("%d", &b);
    res = sum(a, b);
    printf("%d + %d = %d", a, b, res);
    return 0;
}
int sum(int x, int y)
{
    int z;
    z = x + y;
    return 0;
}
```

Преимущества использования функций

- Функцию можно вызвать из различных мест программы, что позволяет избежать повторения программного кода.
- Одну и ту же функцию можно использовать в разных программах.
- Функции повышают уровень модульности программы и облегчают ее проектирование.
- Использование функций облегчает чтение и понимание программы и ускоряет поиск и исправление ошибок.