

# Отличия языков C и C++

У языка C++ есть ряд небольших отличий от C. Хотя каждое из этих отличий само по себе незначительно, вместе они достаточно распространены в программах C++.

❖ Во-первых, если в C функция не имеет параметров, ее прототип содержит слово **void** в списке параметров функции.

Например, если в C функция **f1()** не имеет параметров (и возвращает **char**), ее прототип будет выглядеть следующим образом:

```
char f1(void);
```

В C++ слово **void** не обязательно. Поэтому в C++ прототип пишется так:

```
char f1();
```

C++ отличается от C способом задания пустого списка параметров. Если бы предыдущий прототип (**char f1();**) имел место в программе C, то это бы означало, что о параметрах функции сказать ничего нельзя.

А в C++ это означает, что у функции нет параметров.

Поэтому в предыдущих примерах для исчерпывающего обозначения пустого списка параметров слово **void** не использовалось. (Использование **void** для обозначения пустого списка параметров не ошибочно, скорее, оно излишне. Поскольку большинство программистов C++ стремятся к эффективности, то **void** в таких случаях не употребляется).

В C++ следующие два объявления эквивалентны:

```
int f1();           int f1(void);
```

# Отличия языков С и С++

- ❖ Другим небольшим отличием между С и С++ является то, что в программах С++ все функции должны иметь прототипы. В С прототипы функций рекомендуются, но технически они не обязательны, а в С++ прототипы необходимы и обязательны.
- ❖ Третьим отличием между С и С++ является то, что если в С++ функция имеет отличный от **void** тип возвращаемого значения, то инструкция **return** внутри этой функции должна содержать значение данного типа. В языке С функции с отличным от **void** типом возвращаемого значения фактически не требуется возвращать что-либо. Если значения нет, то функция возвращает неопределенное значение.  
В С, если тип возвращаемого функцией значения явно не задан, функция по умолчанию возвращает значение целого типа. В С++ такого правила нет. Следовательно, необходимо явно объявлять тип возвращаемого значения всех функций.
- ❖ Следующим отличием между С и С++ является то, что в программах С++ вы можете выбирать место для объявления локальных переменных.  
В С локальные переменные могут объявляться только в начале блока, перед любой инструкцией "действия". В С++ локальные переменные могут объявляться в любом месте программы.  
Одним из преимуществ такого подхода является то, что локальные переменные для предотвращения нежелательных побочных эффектов можно объявлять рядом с местом их первого использования.

# Отличия языков С и С++

- ❖ И последнее. Для хранения значений булева типа (истина или ложь) в С++ определен тип данных **bool**. В С++ также определены ключевые слова **true** и **false** – единственные значения, которыми могут быть данные типа **bool**. В С++ результатом выполнения операторов отношения и логических операторов являются значения типа **bool**, и направление развития любой условной инструкции должно определяться относительно значения типа **bool**. Хотя такое отличие от С на первый взгляд кажется значительным, на самом деле это не так. Фактически оно совершенно прозрачно и вот почему: как известно, в С любое ненулевое значение является истинным, а нулевое – ложным. В С++ это положение сохраняется, поскольку при использовании таких значений в булевом выражении ненулевое значение автоматически преобразуется в **true**, а нулевое – в **false**. Правильно и обратное: **true** преобразуется в 1, а **false** в 0, если значение типа **bool** оказывается в целом выражении. Добавление в С++ данных типа **bool** усиливает контроль типа и дает возможность различать данные булева и целого типов. Естественно, что использование данных булева типа не обязательно, скорее оно просто удобно.

# Отличия языков C и C++

## Пример 1

Эта программа C++ не будет компилироваться, поскольку у функции `sum()` нет прототипа; русский текст выводится некорректно:

//Эта программа не будет компилироваться

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int a,b,c;
```

```
    cout << "Введите два числа: ";
    cin >> a >> b;
    c=sum(a, b) ;
    cout << "Сумма равна:" << c;
    return 0;
}
```

```
// Этой функции необходим прототип
sum(int a, int b)
{
    return a+b;
}
```

//В этой программе всё в порядке

```
#include <iostream>
using namespace std;
```

```
int sum(int a, int b); //Прототип функции
```

```
int main ()
{
    int a,b,c;
```

```
// функция для вывода строки с кириллицей
setlocale(LC_STYPE, «rus");
cout << "Введите два числа: ";
cin >> a >> b;
    c=sum(a, b) ;
    cout << "Сумма равна: " << c << "\n";
    return 0;
}
```

```
// Этой функции необходим прототип
int sum(int a, int b)
{
    return a+b;
}
```

# Отличия языков С и С++

## Пример 2

Эта программа иллюстрирует, что локальные переменные можно объявить в любом месте блока:

```
#include <iostream>
using namespace std;

int main()
{
    int i; // локальная переменная, объявленная в начале блока
    // функция для вывода строки с кириллицей
    setlocale(LC_STYPE, "rus");
    cout << "Введите число:";
    cin >> i;
        // расчет факториала
    int j;
    long fact=1; // переменные, объявленные перед инструкциями
    for (j=i; j>=1; j--) fact=fact * j; // действия
    cout << "Факториал равен:" << fact;
return 0;
}
```

Хотя объявление переменных **j** и **fact** рядом с местом их первого использования в этом коротком примере и не слишком впечатляет, в больших функциях такая возможность может обеспечить программе ясность и предотвратить нежелательные побочные эффекты.

# Отличия языков С и С++

## Пример 3

В этой программе создается булева переменная **outcome** и ей присваивается значение **false**. Затем эта переменная используется в инструкции **if**.

```
#include <iostream>
using namespace std;

int main()
{
    bool outcome;
    outcome = false;
    setlocale(LC_STYPE, "rus");
    if(outcome)
        cout << "истина";
    else cout << "ложь";
    return 0;
}
```

В результате выполнения программы на экране появляется слово **ложь**.

# Отличия языков С и С++

Следующая программа не будет компилироваться в качестве программы С. Почему?

```
// В этой программе есть ошибка
#include <iostream>
using namespace std;

int main()
{
    f();
    return 0;
}

void f()
{
    cout << "Программа работать не будет";
}
```