Программирование на C++

Минцифры России

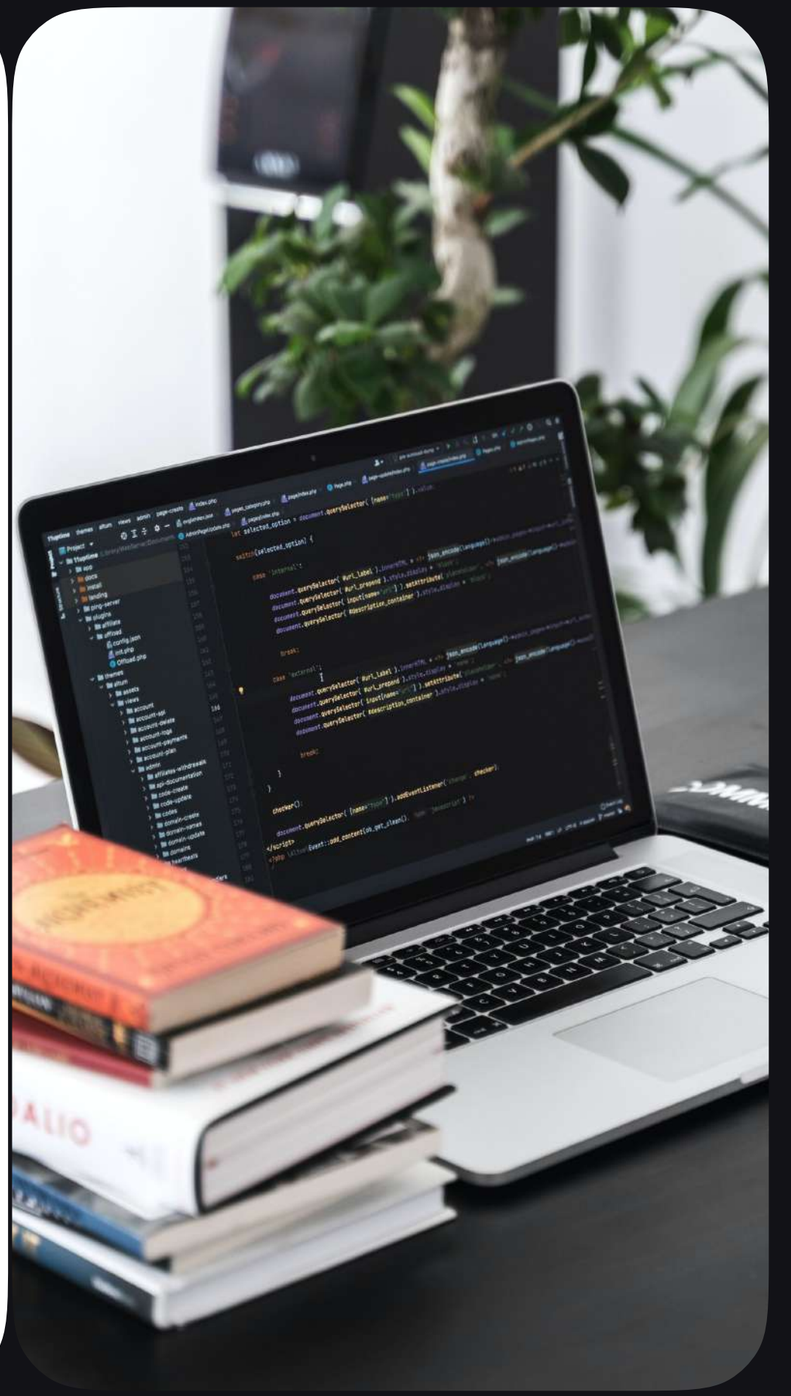
UCHi DOMA

20.35 УНИВЕРСИТЕТ

Урок 9 Модуль 3

Перегрузка

Полезные материалы



Цели урока

- изучить перегрузку функций
- отработать на практике написание перегруженных функций на C++

Перегрузка функций

```
int add(int x, int y)
{
    return x + y;
}
```

Функция складывает два числа **int** и возвращает результат **int**.

функция **add()** не подходит, если нам также нужна функция, которая может складывать два числа с плавающей запятой, поскольку любые параметры с плавающей запятой будут преобразованы в целые числа, что приведет к потере дробных частей аргументов с плавающей запятой.

Перегрузка функций

Один из способов обойти эту проблему — определить несколько функций с немного разными именами:

```
int addInteger(int x, int y)
{
    return x + y;
}
double addDouble(double x, double y)
{
    return x + y;
}
```

Перегрузка функций

Недостатки такого решения:

- необходимо определить единый стандарт именования для аналогичных функций, которые имеют параметры разных типов
- необходимо запомнить названия всех различных вариантов функций и вызывать правильную из них
- необходимо создавать новую функцию, если потребуется складывать три числа int вместо двух
- необходимо организовать управление уникальными именами для каждой функции

Перегрузка функций

Перегрузка функций позволяет нам создавать несколько функций с одним и тем же именем, при условии, что все функции с одинаковыми именами имеют разные параметры (или функции могут различаться иным образом).

Все функции, имеющие одинаковые имена (в одной и той же области видимости), называются **перегруженными функциями** (иногда для краткости называемыми **перегрузками**).

Перегрузка функций в C++

Чтобы перегрузить функцию **add()**, необходимо определить другую функцию **add()**, которая принимает параметры **double**:

```
double add(double x, double y)
{
    return x + y;
}
```

Перегрузка функций в C++

Теперь есть две версии **add**() в одной области видимости:

```
1   int add(int x, int y) // целочисленная версия
2   {
3       return x + y;
4   }
5
6   double add(double x, double y) // версия с плавающей запятой
7   {
8       return x + y;
9   }
10
11  int main()
12  {
13      return 0;
14  }
```

Перегрузка функций в C++

Созданные функции не приведут к конфликту имен, поскольку типы параметров этих функций различаются, компилятор может различать эти функции и рассматривать их как отдельные функции, которые просто имеют общее имя.

Разрешение перегрузки

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

int add(int x, int y)
{
    return x + y;
}

double add(double x, double y)
{
    return x + y;
}

int main()
{
    cout << add(1, 2);     // вызывает add(int, int)
    cout << '\n';
    cout << add(1.2, 3.4); // вызывает add(double, double)

    return 0;
}
```

Когда выполняется вызов для функции, которая была перегружена, компилятор на основе аргументов, используемых в вызове функции, попытается сопоставить этот вызов функции с соответствующей перегрузкой. Это называется **разрешением перегрузки**.

Разрешение перегрузки

Когда функция вызывается с целочисленными аргументами в вызове **add(1, 2)**, компилятор определит, что мы пытаемся вызвать **add(int, int)**. А когда аргументы с плавающей запятой в вызове **add(1.2, 3.4)**, компилятор определит, что мы пытаемся вызвать **add(double, double)**.

Как различаются перегруженные функции

Свойство функции	Используется для перегрузки
Количество параметров	Да
Тип параметров	Да
Тип возвращаемого значения	Нет

Тип возвращаемого значения функции не используется для различения перегруженных функций.

Пример перегрузки по количеству параметров

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

int add(int x, int y)
{
    return x + y;
}


int add(int x, int y, int z)
{
    return x + y + z;
}
int main()
{
    cout << add(1, 2);      // вызывает add(int, int)
    cout << '\n';
    cout << add(1, 2, 3); // вызывает add(int, int, int)

    return 0;
}
```