МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский Государственный Технологический Университет

**РЕФЕРАТ**

по дисциплине: «Основы Алгоритмизации и Программирования»

на тему: “Отличие перегрузки функции от переопределения”

Выполнил студент группы

Оглавление

[Перегрузка функций 2](#_Toc158244373)

[Переопределение функций 3](#_Toc158244374)

[Сравнение 4](#_Toc158244375)

[Список литературы: 4](#_Toc158244376)

# Перегрузка функций

**перегрузка функций (function overloading)** – это возможность определять функции с одним и тем же именем, но разным набором параметров.

!Но для того, чтобы определить несколько различных версий функции с одним и тем же именем, версии должны иметь разное кол-во параметров или параметры должны иметь разный тип.

Пример:

#include <iostream>

#include <string>

int sum(int, int);

double sum(double, double);

std::string sum(const std::string&, const std::string&);

int main()

{

int result1{ sum(5, 8) };

std::cout << result1 << std::endl;

double result2{ sum(4.2, 7.3) };

std::cout << result2 << std::endl;

std::string result3{ sum("Good ", "morning!") };

std::cout << result3 << std::endl;

}

int sum(int a, int b)

{

return a + b;

}

double sum(double a, double b)

{

return a + b;

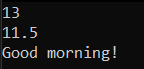
}

std::string sum(const std::string& a, const std::string& b)

{

return a + b;

}



При вызове функции sum в main(), **компилятор выбирает подходящую версию функции в зависимости от типов переданных аргументов**. Код включает использование трех версий функции sum:

* int result1 {sum(5, 8)}; - вызывается версия с двумя целочисленными аргументами.
* double result2 {sum(4.2, 7.3)}; - вызывается версия с двумя значениями с плавающей запятой.
* std::string result3 {sum("Good ", "morning!")}; - вызывается версия с двумя строковыми аргументами.

# Переопределение функций

**Переопределение функций в C++ позволяет в производных классах предоставлять собственные реализации для методов, которые уже имеются в базовых классах.** Суть в том, что можно использовать тот же интерфейс (название и параметры функций) для взаимодействия с объектами как базового, так и производного классов, однако каждый класс может определить свою собственную логику выполнения этой функции.

Ключевое слово virtual в базовом классе указывает, что данная функция может быть переопределена в производных классах, а ключевое слово override в производном классе явно указывает на то, что функция переопределена и предоставляет новую реализацию**.**

#include <iostream>

// Базовый класс

class Animal {

public:

// Виртуальная функция

virtual void makeSound() {

std::cout << "Generic animal sound\n";

}

};

// Производный класс

class Dog : public Animal {

public:

// Переопределение виртуальной функции базового класса

void makeSound() override {

std::cout << "Woof! Woof!\n";

}

};

int main() {

Animal genericAnimal;

genericAnimal.makeSound(); // Вывод: Generic animal sound

Dog myDog;

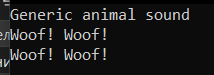
myDog.makeSound(); // Вывод: Woof! Woof!

Animal\* animalPtr = &myDog; // Указатель базового класса на объект производного класса

animalPtr->makeSound(); // Вывод: Woof! Woof!

return 0;

}



# Сравнение

Таким образом, переопределение и перегрузка функций в C++ предоставляют различные механизмы для обработки функций с одинаковыми именами в разных контекстах.

**Переопределение относится к изменению реализации функции в производном классе для сохранения интерфейса базового класса. Перегрузка относится к созданию нескольких функций с одним и тем же именем, но с разными параметрами.**

# Список литературы:

1. <https://metanit.com/cpp/tutorial/5.11.php>
2. <https://metanit.com/cpp/tutorial/4.14.php>
3. <https://cppforeach.wordpress.com/2014/06/18/overloading_vs_overriding/>