Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Отличие перегрузки от переопределения»

Выполнила:

Студент(ка) 1 курса 7 группы

Подшиваленко Диана Игоревна

Проверил:

Белодед Николай Иванович

2023, Минск

**Содержание**

1. Предисловие … 3
2. Отличие перегрузки от переопределения … 4
3. Перегрузка методов… 4
4. Переопределение методов … 5
5. Связь полиморфизма с перегрузкой и переопределением … 6
6. Примеры … 7

**Предисловие**

В данном реферате будут рассмотрены отличия перегрузки от переопределения. А также приведены примеры.

**1.Отличие перегрузки от переопределения**

Перегрузка методов и переопределение методов очень важны в объектно-ориентированном программировании. Они позволяют разработчикам гибко использовать и изменять функциональность своих классов. Несмотря на то, что эти два понятия имеют общую задачу — изменение поведения метода, они имеют существенные отличия.

**2.Перегрузка методов**

**Перегрузка методов** (method overloading) — это возможность определить в классе несколько методов с одинаковым именем, но с разными параметрами. В зависимости от переданных аргументов, будет вызываться соответствующий метод.

Особенностью перегрузки методов является использование разных сигнатур — наборов параметров, которые позволяют различать методы. Сигнатура метода — это сочетание количества, типов и порядка параметров.

Зачем нужна перегрузка методов? Она позволяет создавать более удобные и гибкие интерфейсы для работы с классами. Например, если у нас есть класс, выполняющий какие-то математические действия, то перегрузка методов позволяет вызвать одну и ту же функцию с разными параметрами. Например, метод можно вызвать как с двумя числами, так и с тремя, как с целыми числами, так и с вещественными.

Для определения перегруженных методов необходимо использовать такие же имена, но разные параметры. Компилятор самостоятельно будет выбирать нужный метод при вызове, основываясь на переданных аргументах и их типах.

**Пример:**

1) public void printworld() - метод без аргументов, выводящий слово в консоль.

2) public void printworld(string w) - метод c 1 аргументом типа string, выводящий переданное слово в консоль.

3) public void printworld(string w, int kol) - метод c 2 аргументами – string и int, выводящий переданное слово в консоль указанное число раз.

В языках программирования, поддерживающих перегрузку методов, различные параметры, которые могут быть перегружены, включают в себя:

1) **Число параметров**: методы могут быть перегружены по количеству параметров. Класс может иметь несколько методов с одним и тем же именем, но с разным числом параметров. Таким образом, можно создать методы для разного числа параметров.

2) **Типы параметров**: методы могут быть перегружены по типу параметров. Класс может иметь несколько методов с одним и тем же именем, но с разными типами параметров. Таким образом, можно создать методы, работающие с разными типами данных.

3) **Порядок параметров**: методы могут быть перегружены по порядку параметров. Например, класс может иметь несколько методов с одним и тем же именем, но с разным порядком параметров. Таким образом, можно создать методы, для которых порядок передачи аргументов имеет значение.

**Пример:**

1) void printsum(int a, int b) – метод, который печатает сумму 2 целых чисел.

2) void printsum(int a, int b, int c) – метод, который печатает сумму 3 целых чисел.

3) void printsum(double a, double c) – метод, который печатает сумму 2 вещественных чисел.

В этом примере показана перегрузка по числу и по типу параметров.

**3.Переопределение методов**

При разработке программного кода на ЯП часто возникает необходимость изменять поведение уже существующих методов в классе. Для этого используется механизм переопределения методов.

**Переопределение метода** (method overriding) — это процесс замены реализации метода, уже определенного в родительском классе, в классе-наследнике. Когда методы с одинаковым именем и сигнатурой определены как в родительском, так и в дочернем классе, то при вызове метода на объекте дочернего класса будет использоваться реализация из дочернего класса. Для того чтобы переопределить метод, нужно использовать ключевое слово override.

Условия для переопределения метода:

1) Метод в дочернем классе должен иметь такое же имя, как и метод в родительском классе.

2) Методы в родительском и дочернем классах должны иметь одинаковую сигнатуру (необязательно одинаковый тип возвращаемого значения).

3) Метод в родительском классе должен быть помечен модификатором доступа virtual или abstract.

4) Метод в дочернем классе должен быть помечен модификатором доступа override.

Если данные условия не соблюдаются, то метод считается перегруженным, а не переопределенным.

**4.Связь полиморфизма с перегрузкой и переопределением**

**Перегрузка методов** — это механизм полиморфизма, который позволяет задавать несколько методов с одинаковым именем, но с разными параметрами в одном классе. Компилятор выбирает наиболее подходящий метод для вызова на основе типов аргументов, передаваемых при вызове метода. Таким образом, разные версии метода могут быть вызваны в зависимости от типов параметров, что позволяет сделать код более гибким и удобочитаемым.

**Переопределение методов** — это другой механизм полиморфизма, который позволяет классам-наследникам изменять реализацию унаследованного метода от родительского класса. Для переопределения метода в дочернем классе, необходимо использовать тот же самый сигнатуру (имя метода и типы его аргументов) и указать новую реализацию. При вызове метода на объекте дочернего класса, будет вызвана его переопределенная реализация, а не реализация из родительского класса. Это позволяет изменять поведение методов в разных классах-наследниках, сохраняя при этом одинаковый интерфейс во всех классах.

Таким образом, перегрузка методов и переопределение методов являются механизмами полиморфизма, который позволяет создавать универсальные и гибкие программы. Перегрузка методов позволяет использовать методы с одним именем, но с разными параметрами, в зависимости от контекста, а переопределение методов позволяет изменять поведение унаследованных методов в классах-наследниках.

**4.Примеры**

1) Перегрузка

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

int sum(int a, int b) {

return a + b;

}

int sum(int a, int b, int c) {

return a + b + c;

}

double sum(double a, double b) {

return a + b;

}

string sum(string a, string b) {

return a + b;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << sum(1, 2) << '\n';

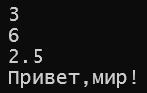
cout << sum(1, 2, 3) << '\n';

cout << sum(1.3, 1.2) << '\n';

cout << sum("Привет", ",мир!") << '\n';

}

Результат выполнения:



2) Переопределение

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

class Person

{

public: virtual void sayworld() {

cout << "I am person\n";

}

};

class Child : Person

{

public: void sayworld() override {

cout << "I am child";

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Person peeople1;

Child people2;

peeople1.sayworld();

people2.sayworld();

}

Результат выполнения:

