Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**Пермский национальный исследовательский политехнический университет**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Использование конструкторов»

Семестр 2

*Работу выполнил: Учащийся группы ИВТ - 22-2б:*

Голубцов Никита Валерьевич

*Работу проверил: доцент кафедры ИТАС:*

*Полякова Ольга Андреевна*

Г. Пермь – 2023

**Постановка задачи**

1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров.

**Описание класса**

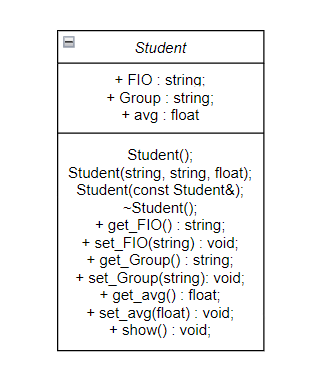
Пользовательский класс СТУДЕНТ

ФИО – string

Группа – string

Средний балл – float

**UML - Диаграмма**



**Рис. 1 – UML-диаграмма класса Student**

**Контрольные вопросы**

1. Для чего нужен конструктор?

Конструктор - специальный метод, который используется для инициализации объекта и вызывается автоматически при его создании.

1. Сколько типов конструкторов существует в С++?

В С++ существует три типа конструкторов:

- Конструктор без параметров

- Конструктор с параметрами

- Конструктор копирования

1. Для чего используется деструктор? В каких случаях он описывается явно?

Деструктор - специальный метод, используемый для освобождения памяти, выделенной объекту:

class grade

{

public:

int value;

grade(int Value);

~grade();

};

grade::grade(int Value)

{

value = Value;

}

grade::~grade() // Деструктор

{

cout << " Вызван деструктор";

}

Деструктор вызывается автоматически при выходе из области видимости объекта. Описывать деструктор явным образом необходимо, когда объект содержит указатели на динамическую память иначе при уничтожении объекта память, на которую ссылались его поля-указатели, не будет помечена как свободная. Указатель на деструктор определить нельзя.

1. Для чего используется конструктор без параметров? Конструктор с параметрами? Конструктор копирования?

- Конструктор без параметров используется для создания пустого объекта, он вызывается автоматически:

class pressure

{

public:

int value;

pressure();

};

pressure::pressure()

{

value = 0;

}

void main()

{

pressure first; // Вызов конструктора без параметров

}

- Конструктор с параметрами используется для инициализации объекта с требуемыми значениями полей:

class pressure

{

public:

int value;

pressure(int Value);

};

pressure::pressure(int Value)

{

value = Value;

}

void main()

{

pressure first(120); // Вызов конструктора с параметрами

}

- Конструктор копирования - это специальный вид конструктора, получающий в качестве единственного параметра указатель на объект этого же класса:

public:

int value;

pressure(int Value);

pressure(const pressure&);

};

pressure::pressure(int Value)

{

value = Value;

}

pressure::pressure(const pressure& first)

{

value = first.value;

}

void main()

{

pressure first(120); // Вызов конструктора с параметрами

pressure second = first; // Вызов конструктора копирования

}

1. В каких случаях вызывается конструктор копирования?

Этот конструктор вызывается в тех случаях, когда новый объект создаётся путём копирования существующего:

-при описании нового объекта с инициализацией другим объектом:

void main()

{

pressure first(120); // Вызов конструктора с параметрами

pressure second = first; // Вызов конструктора копирования

}

-при передаче объекта в функцию (метод) в качестве параметра по значению:

void print(pressure a)

{

a.PrintPressure();

}

void main()

{

pressure b(120);

print(b);

}

-при возврате объекта из функции:

Student Created()

{

Student first; // Создаётся временный объект класса

first.name = "Polina";

first.name = 18;

return first; // возвращает значения как значения функции

}

void main()

{

Student second = Created(); // Объект second - копия объекта first

}

1. Перечислить свойства конструкторов.

- Конструктор не возвращает значение. Нельзя получить указатель на конструктор.

- Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).

- Конструктор, вызываемый без параметров, называется конструктором по умолчанию.

- Параметры конструктора могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию, но их может содержать только один из конструкторов.

- Если программист не указал ни одного конструктора, компилятор создаёт его автоматически. Такой конструктор вызывает конструкторы по умолчанию для полей класса. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.

- Конструкторы не наследуются.

- Конструкторы нельзя описывать с модификаторами const, virtual и static.

- Конструкторы глобальных объектов вызываются до вызова функции main. Локальные объекты создаются, как только становится активной область их действия. Конструктор запускается и при создании временного объекта (на пример, при передаче объекта из функции).

- Конструктор вызывается, если в программе встретилась какая-либо из синтаксических конструкций:

pressure first(120); // Вызов конструктора с параметрами

pressure second = first; // Вызов конструктора копирования

pressure third; // Конструктор без параметров

1. Перечислить свойства деструкторов.

- Вызов деструктора осуществляется автоматически, когда объект удаляется из памяти;

- Деструктор не имеет аргументов и возвращаемого значения;

- Не наследуется;

- Не может быть объявлен как const или static (далее);

- Может быть виртуальным (далее).

- Если деструктор явным образом не определен, компилятор автоматически создает пустой деструктор.

1. К каким атрибутам имеют доступ методы класса?

Методы класса имеют неограниченный доступ ко всем элементам класса, независимо от спецификаторов доступа и порядка объявления методов в классе.

1. Что представляет собой указатель this?

Указатель \*this - это скрытый константный для адреса указатель, содержащий адрес объекта, который вызывает метод класса.

Может использоваться для обращения к полям класса.

Например, если мы будем присваивать значению элемента класса его название

this ->m = n;

1. Какая разница между методами, определёнными внутри и вне класса?

Определение метода внутри класса ничем не отличается от определения обычной функции. По умолчанию такой метод считается встроенной функцией (inline). Если метод определяется вне функции, то принадлежность метода классу указывается с помощью имени класса: Имя\_класса::Имя\_метода. В классе присутствует только прототип.

class music

{

string name = "Music";

int length = 300;

public:

string ShowName() // Метод, определённый внутри класса

{

return name;

}

int SetLength(int l); // Прототип метода

};

int music::SetLength(int length) // Метод, опеределённый вне класса

{

this->length = length;

}

void main()

{

int n = 200;

music first;

first.ShowName();

first.SetLength(n);

}

1. Какое значение возвращает конструктор?

Конструктор не возвращает значения.

1. Какие методы создаются по умолчанию?

По умолчанию создаётся конструктор с пустыми полями и деструктор.

1. Какое значение возвращает деструктор?

Деструктор не возвращает значений, так как он лишь освобождает память, выделенную какому-либо объекту.

1. Дано описание класса class Student

{

string name;

int group;

public:

student(string, int);

student(const student&)

~student();

};

Какой метод отсутствует в описании класса?

В описании класса отсутствует конструктор без параметров (он вызывается автоматически).

1. Какой метод будет вызван при выполнении следующих операторов: student\*s;

s=new student;

Здесь будет вызван пустой конструктор.

1. Какой метод будет вызван при выполнении следующих операторов: student s(“Ivanov”,20);

Здесь будет вызван конструктор с параметрами.

1. Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);

student s2=s1;

Сначала будет вызван конструктор с параметрами, а затем конструктор копирования, заданный явно.

1. Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);

student s2;

s2=s1;

Сначала будет вызван конструктор с параметрами, затем конструктор без параметров, а после конструктор копирования, заданный явно.

1. Какой конструктор будет использоваться при передаче параметра в функцию print():

void print(student a)

{

1. show();

}

Здесь будет использован конструктор копирования, поскольку параметр лишь один и его просто нужно вывести на экран.

1. Класс описан следующим образом:

class Student

{

string name;

int age;

public:

void set\_name(string);

void set\_age(int );

…..

};

Student p;

Каким образом можно присвоить новое значение атрибуту name объекта р?