Министерство образования и науки Российской Федерации

Федеральное государственное образовательное автономное учреждение высшего образования

"Пермский национальный исследовательский политехнический университет"

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Дисциплина: Основы алгоритмизации и программирования

Тема: Классы и объекты. Использование конструкторов.

|  |  |
| --- | --- |
| Выполнил работу | |
| Студент группы РИС-22-1б | |
| Деревнин И.В. | |
|  | |
| Проверил работу | |
| Доцент кафедры ИТАС | |
| Полякова О.А. | |
|  | |

Пермь – 2023

**Анализ предметной области**

**Постановка задачи**

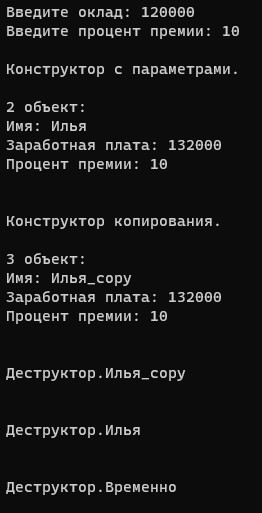
1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Написать демонстрационную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

Вариант 15: Пользовательский класс зарплата

Имя – string, оклад -double, премия (% от оклада) – int.

**Анализ задачи**

Класс Class реализовывает единицу начисления зарплаты какому-либо человеку. Для этого были выбраны поля string name, double salary, int percent для хранения данных о зарплате человека. Для того, чтобы было быстрее взаимодействовать с объектами класса были реализованы конструкторы – с параметрам, без параметров, конструктор копирования. Также для каждого поля были реализованы сеттеры и геттеры, в геттере GetSalary возвращается не стандартное значение заработной платы, а значение с учетом премии.

****Тестирование программы**

*Рис. 1 – Тестирование программы с классом Class.*

**Заключение**

Была разработана программа, которая рассчитывает зарплату для конкретного человека. Для реализации данной программы был описан класс, который хранит в себе 3 поля для подсчета данных о зарплате, а также методы, благодаря которым можно управлять объектом класса.

**Ответы на контрольные вопросы**

1. **Для чего нужен конструктор?** Конструктор в языке программирования С++ нужен для инициализации объектов класса.
2. **Сколько типов конструкторов существует в С++?** В языке С++ существует три типа конструкторов: конструктор без параметров (default constructor), конструктор с параметрами (parameterized constructor) и конструктор копирования (copy constructor).
3. **Для чего используется деструктор? В каких случаях деструктор описывается явно?** Деструктор в С++ используется для освобождения памяти, занятой объектом класса. Деструктор описывается явно в тех случаях, когда объект класса содержит динамические данные, которые должны быть освобождены при уничтожении объекта.
4. **Для чего используется конструктор без параметров? Конструктор с параметрами? Конструктор копирования?** Конструктор без параметров используется для создания объекта класса с значениями по умолчанию. Конструктор с параметрами используется для создания объекта класса с заданными значениями атрибутов. Конструктор копирования используется для создания копии объекта класса.
5. **В каких случаях вызывается конструктор копирования?** Конструктор копирования вызывается в тех случаях, когда объект класса передается в качестве аргумента в функцию по значению, возвращается из функции объект класса по значению или происходит инициализация одного объекта класса другим объектом класса.
6. **Перечислить свойства конструкторов.** Свойства конструкторов:

* Имя конструктора совпадает с именем класса
* Не имеют возвращаемого значения
* Могут быть перегружены

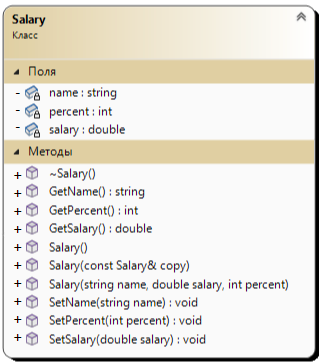
1. **Перечислить свойства деструкторов.** Свойства деструкторов:

* Имя деструктора совпадает с именем класса с добавлением символа ~ в начале
* Не имеют параметров и возвращаемого значения
* Могут быть перегружены
* Вызываются автоматически при уничтожении объекта

1. **К каким атрибутам имеют доступ методы класса?** Методы класса имеют доступ к атрибутам объекта класса.
2. **Что представляет собой указатель this?** Указатель this представляет адрес объекта, для которого был вызван метод.
3. **Какая разница между методами определенными внутри класса и вне класса?** Методы, определенные внутри класса, имеют доступ к приватным и защищенным атрибутам класса, а методы, определенные вне класса, могут иметь доступ только к публичным атрибутам класса.
4. **Какое значение возвращает конструктор?** Конструктор не возвращает никакого значения.
5. **Какие методы создаются по умолчанию?** Методы, создаваемые по умолчанию: конструктор без параметров, конструктор копирования, деструктор, оператор присваивания и оператор сравнения.
6. **Какое значение возвращает деструктор?** Деструктор не возвращает никакого значения.
7. **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 определение метода конструктора по умолчанию.
8. **Какой метод будет вызван при выполнении следующих операторов: student\*s; s=new student;** Будет вызван конструктор без параметров.
9. **Какой метод будет вызван при выполнении следующих операторов: student s(“Ivanov”,20);** Будет вызван конструктор с параметрами, переданными в качестве аргументов.
10. **Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20); student s2=s1;** Будет вызван конструктор копирования.
11. **Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20); student s2; s2=s1;** Будет вызван конструктор с параметрами затем будет вызван конструктор копирования для создания объекта s2 на основе объекта s1, а затем будет вызван оператор присваивания для копирования значений полей объекта s1 в объект s2.
12. **Какой конструктор будет использоваться при передаче параметра в функцию print(): void print(student a) {a.show();}** Будет вызван конструктор копирования, так как объект a передается в функцию print() по значению, а не по ссылке.
13. **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?** Для того чтобы присвоить новое значение атрибуту name объекта р, можно использовать метод set\_name(), например: p.set\_name("New name"), а также метод set\_age.

**Приложения**

Приложение UML-диаграмма



Приложение Б – код программы

Main.cpp:

#include "Class.h"

int main()

{

system("chcp 1251 >> null");

string name;

double salary;

int percent;

Salary sal;

cout << "1 объект: " << "\nИмя: " << sal.GetName() << "\nЗаработная плата: " << sal.GetSalary() << "\nПроцент премии: " << sal.GetPercent() << endl << endl; //вывод объекта

cout << "Введите имя: "; cin >> name; //установка объекту новых полей

cout << "Введите оклад: "; cin >> salary;

cout << "Введите процент премии: "; cin >> percent;

Salary sal2(name, salary, percent); //конструктор с параметрами

cout << "2 объект: " << "\nИмя: " << sal2.GetName() << "\nЗаработная плата: " << sal2.GetSalary() << "\nПроцент премии: " << sal2.GetPercent() << endl << endl; //вывод объекта

Salary sal3(sal2); //отработка конструктора копирования

cout << "3 объект: " << "\nИмя: " << sal3.GetName() << "\nЗаработная плата: " << sal3.GetSalary() << "\nПроцент премии: " << sal3.GetPercent() << endl << endl; //вывод объекта

return 0;

}

#pragma once

#include <iostream>

#include <cmath>

using namespace std;

class.h:

#pragma once

#include <iostream>

using namespace std;

class Salary

{

private:

string name; //имя

double salary; //зарплата

int percent; //процент премии относительно зарплаты

public:

Salary(); //конструктор

Salary(const Salary& copy); //конструктор копирования

Salary(string name, double salary, int percent); //конструктор с параметрами

~Salary(); //деструктор

void SetName(string name) { this->name = name; } //сеттер имени

void SetSalary(double salary) { this->salary = salary; } //сеттер зарплаты

void SetPercent(int percent) { this->percent = percent; } //сеттер премии

string GetName() { return this->name; } //геттер имени

double GetSalary() { return this->salary + this->salary \* (float(this->percent) / 100); } //геттер зарплаты

int GetPercent() { return this->percent; } //геттер премии

};

class.cpp:

#include "Class.h"

Salary::Salary()

{

cout << endl << "Конструктор без параметров." << endl << endl;

name = "Временно";

salary = 0;

percent = 0;

}

Salary::Salary(const Salary& copy)

{

cout << endl << "Конструктор копирования." << endl << endl;

this->name = copy.name + "\_copy";

this->salary = copy.salary;

this->percent = copy.percent;

}

Salary::Salary(string name, double salary, int percent)

{

cout << endl << "Конструктор с параметрами." << endl << endl;

this->name = name;

this->salary = salary;

this->percent = percent;

}

Salary::~Salary()

{

cout << endl << "Деструктор." << name << endl << endl;

name = "";

salary = 0;

percent = 0;

}