ПНИПУ

Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №2.

Классы и объекты. Использование конструкторов.

Выполнил студент группы РИС-23-3Б

Буковский Денис Владимирович

Проверила доцент кафедры ИТАС О.А. Полякова

2024

1. 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.

Пользовательский класс ЗАРПЛАТА

ФИО — string

Оклад — double

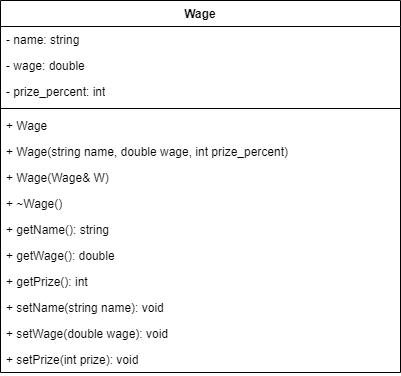
Премия (% от оклада) — int

1. Анализ задачи

Определим класс Wage со всеми необходимыми методами и атрибутами. Определим конструкторы и деструкторы в соответствии с заданием.

Протестируем в функции main().

1. Блок-схема



1. Код

Wage.h

#pragma once

#include <iostream>

class Wage {

private:

std::string name;

double wage;

int prize\_percent;

public:

Wage();

Wage(std::string, double, int);

Wage(Wage&);

~Wage();

std::string getName();

double getWage();

int getPrize();

void setName(std::string name);

void setWage(double wage);

void setPrize(int prize);

};

Wage.cpp

#include "Wage.h"

Wage::Wage() {

this->name = "NOT DEFINED";

this->wage = 0.0;

this->prize\_percent = 0;

}

Wage::Wage(std::string name, double wage, int prize\_percent) {

this->name = name;

this->wage = wage;

this->prize\_percent = prize\_percent;

}

Wage::Wage(Wage& T) {

this->name = T.name;

this->wage = T.wage;

this->prize\_percent = T.prize\_percent;

}

Wage::~Wage() {

std::cout << std::endl << this->name << " was fired... :(" << std::endl;

}

std::string Wage::getName() {

return this->name;

}

double Wage::getWage() {

return this->wage;

}

int Wage::getPrize() {

return this->prize\_percent;

}

void Wage::setName(std::string name) {

this->name = name;

}

void Wage::setWage(double wage) {

this->wage = wage;

}

void Wage::setPrize(int prize) {

this->prize\_percent = prize;

}

Main.cpp

#include <iostream>

#include "Wage.h"

using namespace std;

int main() {

Wage pay1;

Wage pay2("Vladimir", 54234.15, 30);

Wage pay3 = pay1;

pay1.setName("Gennadiy");

pay1.setWage(25496.50);

pay1.setPrize(50);

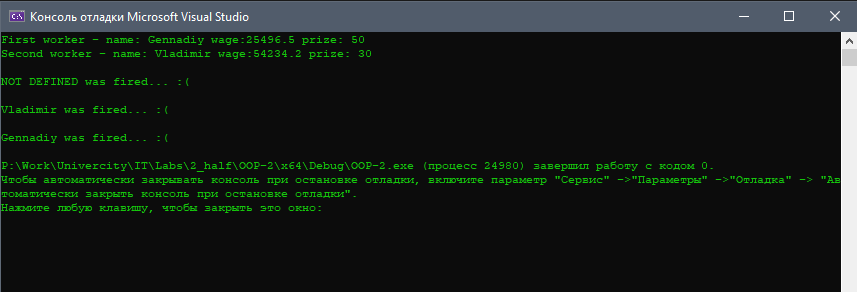
cout << "First worker - name: " << pay1.getName() << " wage:" << pay1.getWage() << " prize: " << pay1.getPrize() << endl;

cout << "Second worker - name: " << pay2.getName() << " wage:" << pay2.getWage() << " prize: " << pay2.getPrize() << endl;

return 0;

}

1. Вывод программы



1. Ответы на контрольные вопросы
2. Конструктор – это метод который автоматически вызывается при создании объекта класса. Он создаёт в объекте атрибуты и методы класса, а также может носить дополнительные функции в зависимости от своей роли.
3. В C++ существует 3 типа конструкторов.
4. Деструктор вызывается при удалении объекта класса. Его задача – очистить ненужную занятую память. Деструктор описывается явно в случаях, когда атрибуты класса выходят за пределы своей области памяти (например массивы, списки, очереди, деревья и т.д.) и деструктор не может их удалить самостоятельно.
5. Конструктор без параметров используется по умолчанию и создаёт объект со стандартными значениями либо без значений вовсе.

Конструктор с параметрами вызывается при передаче параметров при создании объекта и эти параметры могут влиять на поведение будущего объекта.

Конструктор копирования создаёт объект с такими же значениями атрибутов, что и у переданного объекта.

1. Конструктор копирования вызывается при передаче объекта того же типа как параметра конструктора.
2. Конструкторы нужны для: а) создания объектов, б) установления значений атрибутов и в) выполняют необходимые для работы объекта действия перед его созданием.
3. Деструкторы нужны для: а) очистки памяти, б) уничтожения побочных структур данных и в) выполнение необходимых действий после удаления объекта.
4. Методы класса имеют доступ ко всем атрибутам класса без исключений.
5. Указатель this ссылается на себя – объект класса.
6. Методы, определённые вне класса, не имеют отличий от методов, определённых внутри класса.
7. Конструктор не возвращает значений.
8. Конструктор и деструктор.
9. Деструктор не возвращает значений.
10. Конструктор без параметров.
11. Конструктор student().
12. Конструктор student(string, int).
13. Конструктор student(string, int) и конструктор копирования student(const student&).
14. Конструктор student(string, int) и конструктор student().
15. Конструктор student().
16. Set\_name(name).