**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

(СПбГУТ)

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ СЕТЕЙ И СИСТЕМ **(ИКСС)**

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ **(ПИ И ВТ)**

Дисциплина: «ООП»

Лабораторная работа №2-4.

# Тема: «ОТНОШЕНИЕ ВКЛЮЧЕНИЯ»

Вариант 16

Выполнил: Студент группы ИКПИ-95

Новиков С.С.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Приняла:

Петрова О.Б.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

2020г

*Постановка задачи. Лабораторная 2*

Разработать определения двух класса COne и CTwo, которые связаны отношением включения. Поля разрабатываемых классов считаются заданными и приводятся в таблицах 2.1 и 2.2. Для всех классов требуется написать три вида конструкторов (умолчания, с параметрами и конструктор копирования), деструктор, перегруженный оператор присваивания, методы доступа и метод print(), распечатывающий значения полей объекта. Написать тестовую программу для проверки работоспособности разработанных классов.

*Задача 1*

**Поля класса COne**

float f

char \*ps

**Поля класса CTwo**

Double d

COne \*p

*Постановка задачи. Лабораторная 3*

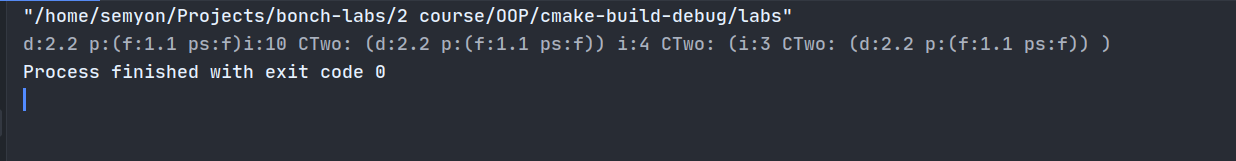
Дополнить систему, состоящую из двух классов COne и CTwo, которые были разработаны в лабораторной работе 2, новым классом CThree. Новый класс должен быть связан public наследованием с классом CTwo. Класс CThree должен имеет одно поля, которое выбирается студентом самостоятельно. Для разрабатываемого класса написать конструкторы умолчания, с параметрами и конструктор копирования, деструктор, методы доступа и метод print(). Написать тестовую программу для проверки работоспособности разработанных классов.

*Постановка задачи. Лабораторная 4*

Дополнить систему, состоящую из трех классов COne, CTwo и CThree, которые были разработаны в лабораторной работе 3, новым классом CFour. Новый класс должен быть связан **public** наследованием с классом CThree. Класс CFour должен иметь одно поля, которое выбирается студентом самостоятельно. Для разрабатываемого класса написать конструкторы умолчания, с параметрами и конструктор копирования, деструктор, методы доступа и метод print(). Метод print() в классах CTwo, CThree и CFour должен быть виртуальным. Написать тестовую программу для проверки работоспособности разработанных классов. Разработать глобальную функцию printAll(), имеющую два параметра: массив указателей типа CTwo\* и количество элементов в этом массиве **int** n.

В тестовой программе массив указателей должен быть инициализирован адресами объектов типа CTwo, CThree и CFour.

*Результаты работы программы*



*Программа на языке* ***c++***

Файл main.cpp

#include <iostream>

#include "CTwo.hpp"

#include "COne.hpp"

#include "CThree.hpp"

#include "CFour.hpp"

using std::cout;

using std::endl;

void printAll(CTwo \*\*ctwo, size\_t n) {

for (int i = 0; i < n; i++) {

ctwo[i]->print();

}

}

int main() {

COne cone(1.1, "f");

CTwo ctwo(2.2, &cone);

CThree cThree(3, ctwo);

CFour cFour(4, cThree);

cThree.setInt(10);

CTwo \*ctwos[3] = {&ctwo, &cThree, &cFour};

printAll(ctwos, 3);

return 0;

}

Файл COne.hpp

#pragma once

#include <string>

class COne {

public:

COne();

COne(float f, const char \*ps);

COne(const COne &cone);

virtual ~COne();

COne &operator=(const COne &r);

float getF() const;

char \*getChar() const;

void print() const;

private:

float f;

char \*ps;

};

Файл COne.cpp

#include <iostream>

using std::cout;

using std::endl;

#include <cstring>

#include "COne.hpp"

#include <memory>

COne::COne() :

f(0.0), ps(new char[1]) {

ps[0] = 0;

}

COne::COne(float f, const char \*str) :

f(f) {

const auto len = strlen(str) + 1;

ps = new char[len];

memcpy(ps, str, len);

}

COne::COne(const COne &cone) :

f(cone.f) {

const auto len = strlen(cone.ps) + 1;

ps = new char[len];

memcpy(ps, cone.ps, len);

}

COne::~COne() {

delete[] ps;

}

COne &COne::operator=(const COne &r) {

if (this == &r) {

return \*this;

}

delete[] ps;

const auto len = strlen(r.ps) + 1;

ps = new char[len];

memcpy(ps, r.ps, len);

f = r.f;

return \*this;

}

float COne::getF() const {

return f;

}

char \*COne::getChar() const {

return ps;

}

void COne::print() const {

cout << "f:" << f << " ps:" << ps;

}

Файл CTwo.hpp

#pragma once

#include <string>

#include "COne.hpp"

class CTwo {

public:

CTwo();

CTwo(const double &d, const COne \*p);

CTwo(const CTwo &ctwo);

virtual ~CTwo();

CTwo &operator=(const CTwo &ctwo);

const double &getDouble() const;

const COne \*getCOne() const;

virtual void print() const;

private:

COne \*p;

double d;

};

Файл CTwo.cpp

#include <iostream>

using std::cout;

using std::endl;

#include "CTwo.hpp"

#include "COne.hpp"

CTwo::CTwo() :

d(0.0), p(new COne) {

}

CTwo::CTwo(const double &d, const COne \*p) :

d(d), p(new COne(\*p)) {

}

CTwo::CTwo(const CTwo &ctwo) :

d(ctwo.d), p(new COne(\*(ctwo.p))) {

}

CTwo::~CTwo() {

delete p;

}

CTwo &CTwo::operator=(const CTwo &ctwo) {

if (this == &ctwo) {

return \*this;

}

d = ctwo.d;

\*p = \*(ctwo.p);

return \*this;

}

const double &CTwo::getDouble() const {

return d;

}

const COne \*CTwo::getCOne() const {

return p;

}

void CTwo::print() const {

cout << "d:" << d << " p:(";

p->print();

cout << ")";

}

Файл CThree.hpp

#pragma once

#include "CTwo.hpp"

class CThree : public CTwo {

public:

CThree();

CThree(const int i, const CTwo &p);

CThree(const CThree &cthree);

~CThree() override;

int getInt() const;

void setInt(int i);

void print() const override;

private:

int i;

};

Файл CThree.cpp

#include <iostream>

using std::cout;

using std::endl;

#include "CTwo.hpp"

#include "CThree.hpp"

CThree::CThree() :

i(0), CTwo() {

}

CThree::CThree(const int i, const CTwo &p) :

i(i), CTwo(p) {

}

CThree::CThree(const CThree &cThree) :

i(cThree.i), CTwo(cThree) {

}

CThree::~CThree() {

}

int CThree::getInt() const {

return i;

}

void CThree::setInt(int i) {

this->i = i;

}

void CThree::print() const {

cout << "i:" << i << " CTwo: (";

CTwo::print();

cout << ") ";

}

Файл CFour.hpp

#pragma once

#include "CThree.hpp"

class CFour : public CThree {

public:

CFour();

CFour(const int i, const CThree &p);

CFour(const CFour &cFour);

~CFour() override;

int getInt() const;

void setInt(int i);

void print() const override;

private:

int i;

};

Файл CFour.cpp

#include <iostream>

using std::cout;

using std::endl;

#include "CThree.hpp"

#include "CFour.hpp"

CFour::CFour() :

i(0), CThree() {

}

CFour::CFour(const int i, const CThree &p) :

i(i), CThree(p) {

}

CFour::CFour(const CFour &cFour) :

i(cFour.i), CThree(cFour) {

}

CFour::~CFour() {

}

int CFour::getInt() const {

return i;

}

void CFour::setInt(int i) {

this->i = i;

}

void CFour::print() const {

cout << "i:" << i << " CTwo: (";

CThree::print();

cout << ") ";

}

*Выводы*

В результате проведенной работы изучены классы в C++.