Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича»

(СПбГУТ)

Кафедра Программной инженерии и вычислительной техники

**Лабораторные работы №2-4 по Объектно-ориентированному программированию**

**«Отношение включения. Отношения наследования. Виртуальные функции.»**

**Работу выполнил:**

Студент 2-го курса

Группа ИКПИ-73

Сударев Андрей Владимирович

**Приняла:**

Петрова Ольга Борисовна

Санкт-Петербург

2018 уч. г.

**Лабораторная работа №2. Отношение Включения.**

**Цель работы**

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.

**Лабораторная работа №3. Отношение Наследование.**

**Цель работы**

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.

**Лабораторная работа №4. Виртуальные функции**

**Цель работы**

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.

**Вариант 20 (Поля класса COne: long l; string s; Поля класса CTwo: 5 – COne\* p; double d )**

**Таблица идентификаторов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Идентификатор | Метод класса | Описание метода | Назначение |
| 1 | COne | void inpL(long) | Изменение параметра l | Классы |
| void inpS(string) | Изменение параметра s |
| operator=(COne) | Перегрузка оператора = |
| void display() | Вывод полей класса COne |
| 2 | CTwo | COne getP(); | геттер объекта класса COne |
| inpD(double) | Изменение параметра d |
| void inpS(long, string) | Изменение параметров объекта класса COne |
| operator=(CTwo) | Перегрузка оператора = |
| display() | вывод полей класса CTwo |
| 3 | CThree | double getS() | получить параметра s |
| operator=(CThree) | Перегрузка оператора = |
| display() | вывод полей класса CThree |
| 4 | CFour | void inpCent(int | Изменение параметра cent |
| operator=(CFour) | Перегрузка оператора = |
| display() | вывод полей класса CFour |

**Программа на С++**

**Main.cpp**

#include "stdafx.h"

void displayAll(CTwo\*\* const mass, int n)

{

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

{

cout << "mass[" << i << "]: \n";

mass[i]->display();

cout << endl;

}

}

int main()

{

CTwo a, b, c;

COne abc(23, "str");

a.display(); //1

cout << endl;

b.display(); //2

cout << endl;

c.display(); //3

cout << endl;

c.inpS(abc);

c.display(); //4

cout << endl;

c.inpD(99.999);

abc.inpS("proth");

c.display(); //5

cout << endl;

abc.display(); //6

cout << endl;

CTwo par(22, abc);

abc.inpS("expected");

abc.inpL(5);

par.display(); //7

cout << endl;

CTwo cpy = par;

par.inpS(abc);

cpy.display(); //8

cout << endl;

par.display();

cout << endl; //9

CThree Testing;

Testing.display();

cout << endl; //10

CThree Testing\_par(par);

Testing\_par.display();

cout << endl; //11

CThree Testing\_cpy(Testing\_par);

Testing\_par.inpD(100);

Testing\_par.display();

cout << endl; //12

CFour fo4(Testing\_par, 55);

fo4.display();

cout << endl; //13

cout << "Test displayAll\n";

CTwo\* mass[5];

mass[0] = &a;

mass[1] = &par;

mass[2] = &fo4;

mass[3] = &Testing;

mass[4] = &cpy;

displayAll(mass, 5);

a = a;

a.display();

return 0;

}

**COne.h**

#pragma once

class COne

{

private:

long l;

string s;

public:

COne();

COne(long, string);

COne(const COne &);

int getL();

string getS();

void inpL(long);

void inpS(string);

void display();

COne operator=(COne);

~COne();

};

**COne.cpp**

#include "stdafx.h"

#include "COne.h"

COne::COne():l(0), s("?")

{

}

COne::COne(long l, std::string s) : l(l), s(s)

{

}

COne::COne(const COne & cpy) : l(cpy.l), s(cpy.s)

{

}

void COne::display()

{

cout << l << endl << s << endl;

}

COne COne::operator=(COne cpy)

{

COne Temp;

Temp.l = cpy.l;

Temp.s = cpy.s;

return Temp;

}

int COne::getL()

{

return l;

}

string COne::getS()

{

return s;

}

void COne::inpL(long l)

{

this->l = l;

}

void COne::inpS(string s)

{

this->s = s;

}

COne::~COne()

{

}

**CTwo.h**

#pragma once

#include "COne.h"

class CTwo

{

private:

COne \* p;

double d;

public:

CTwo();

CTwo(double, COne &);

CTwo(const CTwo &);

double getD();

COne getP();

void inpD(double);

void inpS(COne &);

void inpS(long, string);

virtual void display();

CTwo& operator=(const CTwo&);

~CTwo();

};

**CTwo.cpp**

#include "stdafx.h"

#include "CTwo.h"

CTwo::CTwo()

{

p = new COne;

d = 0;

}

CTwo::CTwo(double dpy, COne &cpy) : d(dpy)

{

p = new COne;

\*p = cpy;

}

CTwo::CTwo(const CTwo & cpy) : d(cpy.d)

{

p = new COne;

\*p = \*cpy.p;

}

void CTwo::display()

{

cout << d << endl;

cout << "COne: \n";

p->display();

}

double CTwo::getD()

{

return d;

}

COne CTwo::getP()

{

return \*p;

}

void CTwo::inpD(double d)

{

this->d = d;

}

void CTwo::inpS(COne &cpy)

{

\*p = cpy;

}

void CTwo::inpS(long l, string s)

{

p->inpL(l);

p->inpS(s);

}

CTwo& CTwo::operator=(const CTwo& ch)

{

if (this == &ch)

{

cout << "cpy is nelzya!\n";

return \*this;

}

d = ch.d;

\*p = \*ch.p;

return \*this;

}

CTwo::~CTwo()

{

delete p;

}

**CThree.h**

#pragma once

#include "CTwo.h"

class CThree :

public CTwo

{

private:

double s;

public:

CThree();

CThree(const CTwo &);

CThree(const CThree &);

double getS();

virtual void display();

~CThree();

};

**CThree.cpp**

#include "stdafx.h"

#include "CThree.h"

CThree::CThree() : s(0), CTwo()

{

}

CThree::CThree(const CTwo &cpy) : CTwo(cpy)

{

s = getD() + getP().getL();

}

CThree::CThree(const CThree &cpy) : CTwo(cpy), s(cpy.s)

{

}

void CThree::display()

{

s = getD() + getP().getL();

cout << "sum = " << s << endl;

cout << "CTwo: \n";

CTwo::display();

}

CThree::~CThree()

{

}

**CFour.h**

#pragma once

class CFour :

public CThree

{

private:

int cent;

public:

int getCent(void);

void inpCent(int);

virtual void display(void);

CFour();

CFour(const CThree&, int);

CFour(const CFour&);

~CFour();

};

**CFour.cpp**

#include "stdafx.h"

#include "CFour.h"

CFour::CFour() : cent(0), CThree()

{

}

CFour::CFour(const CThree& cpy, int s) : CThree(cpy), cent(s)

{

}

CFour::CFour(const CFour& cpy) : CThree(cpy), cent(cpy.cent)

{

}

void CFour::inpCent(int s)

{

cent = s;

}

int CFour::getCent()

{

return cent;

}

void CFour::display(void)

{

cout << "cent = " << cent << endl;

cout << "CThree: \n";

CThree::display();

}

CFour::~CFour()

{

}

**stdafx.h**

#pragma once

#include "targetver.h"

#include <stdio.h>

#include <tchar.h>

#include <iostream>

#include <string>

using namespace std;

#include "COne.h"

#include "CTwo.h"

#include "CThree.h"

#include "CFour.h"