МИНОБРНАУКИ РОССИИ

ФГБОУ ВО Череповецкий государственный университет

Институт информационных технологий

Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: Объектно-ориентированное программирование

ЛАБОРАТОРНАЯ РАБОТА № 5

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ОТНОШЕНИЯ

ПРОСТОГО НАСЛЕДОВАНИЯ

Выполнил:

студент гр.

1ПИб-01-21оп

МартынюкМ.В.

Проверил :

ЕршовЕ.В.

Череповец, 2020 г.

**Цель работы**:

Изучить способы создания производного класса и особенности работы с ним, правила инициализации и доступа к элементам производного класса; приобрести практические навыки наследования.

**Задание:**

Создайте производный класс для АТД, реализованного по заданию лабораторной работы 4, используя одиночное наследование. Проверьте работоспособность АТД на тестовом наборе данных.

**Код программы:**

**Исправленный код.**

#include"iostream"

#include<string>

#include<string.h>

#include<cstring>

using namespace std;

template<class T>

class printer {

public:

printer(); //по умолчанию

printer(int a, int n, T b, T c, T d, T e, char\* f, char\* g); //конструктор с параметрами

printer(const printer& m); //копирующий конструктор

printer& operator=(const printer& m);

~printer(); //деструктор

void print();

printer& operator+=(const printer& m2);

printer& operator>(const printer& m);

printer& operator+(const printer& m);

protected:

int font\_size;//размершрифта

int number\_of\_pages;//количествостраниц

T indent\_on\_the\_left;//отступслева

T indent\_on\_the\_right;//отступсправа

T indent\_on\_the\_from\_above;//отступсверху

T indent\_on\_the\_from\_below;//отступснизу

char\* color;//цветшрифта

char\* text;// текст

private:

};

template<class T>printer<T>::printer() {

font\_size = 1;

number\_of\_pages = 2;

indent\_on\_the\_left = 3;

indent\_on\_the\_right = 4;

indent\_on\_the\_from\_above = 5;

indent\_on\_the\_from\_below = 6;

char f[100] = "not text";

//delete[]color;

color = new char[strlen(f) + 1];

strcpy(color, f);

char g[100] = "not text";

//delete[]text;

text = new char[strlen(g) + 1];

strcpy(text, g);

}

template<class T>printer<T>::printer(int a, int n, T b, T c, T d, T e, char\* f, char\* g) {

font\_size = a;

number\_of\_pages = n;

indent\_on\_the\_left = b;

indent\_on\_the\_right = c;

indent\_on\_the\_from\_above = d;

indent\_on\_the\_from\_below = e;

//delete[]color;

color = new char[strlen(f) + 1];

strcpy(color, f);

//delete[]text;

text = new char[strlen(g) + 1];

strcpy(text, g);

}

template<class T>printer<T>::printer(const printer<T>& m) {

if (&m != this) {

font\_size = m.font\_size;

number\_of\_pages = m.number\_of\_pages;

indent\_on\_the\_left = m.indent\_on\_the\_left;

indent\_on\_the\_right = m.indent\_on\_the\_right;

indent\_on\_the\_from\_above = m.indent\_on\_the\_from\_above;

indent\_on\_the\_from\_below = m.indent\_on\_the\_from\_below;

color = new char[strlen(m.color) + 1];

strcpy(color, m.color);

text = new char[strlen(m.text) + 1];

strcpy(text, m.text);

}

}

template <class T> printer<T>::~printer() {delete[]color;delete[]text;}

template<class T>printer<T>& printer<T>::operator=(const printer<T>& m) {

if (&m != this) { //если адрес не равен содержимому адресу this А если равен ?

font\_size = m.font\_size;

number\_of\_pages = m.number\_of\_pages;

indent\_on\_the\_left = m.indent\_on\_the\_left;

indent\_on\_the\_right = m.indent\_on\_the\_right;

indent\_on\_the\_from\_above = m.indent\_on\_the\_from\_above;

indent\_on\_the\_from\_below = m.indent\_on\_the\_from\_below;

color = new char[strlen(m.color) + 1];

strcpy(color, m.color);

text = new char[strlen(m.text) + 1];

strcpy(text, m.text);

}

return \*this;

}

template<class T>printer<T>& printer<T>::operator>(const printer<T>& m)

{

if (font\_size < m.font\_size) cout << "Размершрифта = small" << endl;

if (font\_size > m.font\_size) cout << "Размершрифта = big" << endl;

else cout << "Размершрифта = ravni" << endl;

if (number\_of\_pages < m.number\_of\_pages) cout << "Количествостраниц = small" << endl;

if (number\_of\_pages > m.number\_of\_pages) cout << "Количествостраниц = big" << endl;

else cout << "Количество страниц = ravni" << endl;

if (indent\_on\_the\_left < m.indent\_on\_the\_left)cout << "Отступслева = small" << endl;

if (indent\_on\_the\_left > m.indent\_on\_the\_left)cout << "Отступслева = big" << endl;

else cout << "Отступслева = ravni" << endl;

if (indent\_on\_the\_right < m.indent\_on\_the\_right)cout << "Отступсправа = small" << endl;

if (indent\_on\_the\_right > m.indent\_on\_the\_right)cout << "Отступсправа = big" << endl;

else cout << "Отступсправа = ravni" << endl;

if (indent\_on\_the\_from\_above < m.indent\_on\_the\_from\_above)cout << "Отступсверху = small" << endl;

if (indent\_on\_the\_from\_above > m.indent\_on\_the\_from\_above)cout << "Отступсверху = big" << endl;

else cout << "Отступсверху = ravni" << endl;

if (indent\_on\_the\_from\_below < m.indent\_on\_the\_from\_below)cout << "Отступснизу = small" << endl;

if (indent\_on\_the\_from\_below > m.indent\_on\_the\_from\_below)cout << "Отступснизу = big" << endl;

else cout << "Отступснизу = ravni" << endl;

if (strlen(text) < strlen(m.text))cout << "Длинатекста = small" << endl;

if (strlen(text) > strlen(m.text))cout << "Длинатекста = big" << endl;

else cout << "Длина текста = ravni" << endl;

return (\*this);

}

template<class T>printer<T>& printer<T>::operator+=(const printer<T>& m)

{

int dlin;

font\_size = font\_size + m.font\_size;

number\_of\_pages = number\_of\_pages + m.number\_of\_pages;

indent\_on\_the\_left = indent\_on\_the\_left + m.indent\_on\_the\_left;

indent\_on\_the\_right = indent\_on\_the\_right + m.indent\_on\_the\_right;

indent\_on\_the\_from\_above = indent\_on\_the\_from\_above + m.indent\_on\_the\_from\_above;

indent\_on\_the\_from\_below = indent\_on\_the\_from\_below + m.indent\_on\_the\_from\_below;

dlin = strlen(color) + strlen(m.color);

if (strlen(color) < strlen(m.color)) color = new char[strlen(m.color) + 1];

else color = new char[strlen(color) + 1];

sprintf(color, "%d", dlin);

dlin = strlen(color) + strlen(m.color);

if (strlen(text) < strlen(m.text)) text = new char[strlen(m.text) + 1];

else text = new char[strlen(text) + 1];

sprintf(text, "%d", dlin);

return (\*this);

}

template<class T>printer<T>& printer<T>::operator+(const printer<T>& m)

{

printer n;

int dlin;

n.font\_size = font\_size + m.font\_size;

n.number\_of\_pages = number\_of\_pages + m.number\_of\_pages;

n.indent\_on\_the\_left = indent\_on\_the\_left + m.indent\_on\_the\_left;

n.indent\_on\_the\_right = indent\_on\_the\_right + m.indent\_on\_the\_right;

n.indent\_on\_the\_from\_above = indent\_on\_the\_from\_above + m.indent\_on\_the\_from\_above;

n.indent\_on\_the\_from\_below = indent\_on\_the\_from\_below + m.indent\_on\_the\_from\_below;

dlin = strlen(color) + strlen(m.color);

if (strlen(color) < strlen(m.color)) n.color = new char[strlen(m.color) + 1];

else color = new char[strlen(color) + 1];

sprintf(n.color, "%d", dlin);

dlin = strlen(color) + strlen(m.color);

if (strlen(text) < strlen(m.text)) n.text = new char[strlen(m.text) + 1];

else n.text = new char[strlen(text) + 1];

sprintf(n.text, "%d", dlin);

return n;

}

template<class T>void printer<T>::print() { //определениефункции

cout << "Размершрифта " << font\_size << endl;//размершрифта

cout << "Количествостраниц " << number\_of\_pages << endl;

cout << "Отступслева " << indent\_on\_the\_left << endl;//отступслева

cout << "Отступсправа " << indent\_on\_the\_right << endl;//отступсправа

cout << "Отступсверху " << indent\_on\_the\_from\_above << endl;//отступсверху

cout << "Отступснизу " << indent\_on\_the\_from\_below << endl;//отступснизу

cout << "Цветшрифта " << color << endl;//цветшрифта

cout << "Текст " << text << endl;// текст

}

//-------------------------------class lazerni\_printer----------------------------------

class lazerni\_printer : public printer<int>{

public://копирующий конструктор и деструктор, вызывающий деструктор базового класса

lazerni\_printer();

lazerni\_printer(int a, int n, int b, int c, int d, int e, char\* f, char\* g, int k, int s); //k -kol\_ener, s - skorost

lazerni\_printer(const lazerni\_printer& m); //копирующий конструктор

void print();

~lazerni\_printer();

lazerni\_printer& operator=(const lazerni\_printer& m);

lazerni\_printer& operator+(const lazerni\_printer& m);

private:

int kol\_ener;

int skorost;

};

lazerni\_printer::lazerni\_printer() :printer<int>::printer() {

kol\_ener = 360;

skorost = 100;

}

lazerni\_printer::lazerni\_printer(int a, int n, int b, int c, int d, int e, char\* f, char\* g, int k, int s):printer<int>::printer(a, n, b, c, d, e, f, g) {

kol\_ener = k;

skorost = s;

}

lazerni\_printer::~lazerni\_printer() { printer<int>::~printer(); }

lazerni\_printer::lazerni\_printer(const lazerni\_printer& m){

if (&m != this) {

printer<int>::printer(m);

kol\_ener = m.kol\_ener;

skorost = m.skorost;

}

}

void lazerni\_printer::print() {

printer<int>::print();

cout << "kol\_ener " << kol\_ener << endl;

cout << "skorost " << skorost << endl;

}

lazerni\_printer& lazerni\_printer::operator=(const lazerni\_printer& m) {

if (&m != this) {

printer<int>::operator=(m);

kol\_ener = m.kol\_ener;

skorost = m.skorost;

}

return \*this;

}

lazerni\_printer& lazerni\_printer::operator+(const lazerni\_printer& m)

{

lazerni\_printer n;

n.kol\_ener = kol\_ener + m.kol\_ener;

n.skorost = skorost + m.skorost;

return n;

}

int main() {

setlocale(LC\_ALL, "Russian");

char col[10] = "red";

char tex[100] = "redtextred";

char tex1[100] = "redstextred";

cout << "Вводим информацию о первом объекте" << endl << endl;

lazerni\_printer print1(12, 25, 2.5, 3.2, 1.2, 2.5, col, tex, 360, 100);

cout << "Выводим информацию о первом объекте:" << endl << endl;

print1.print();

cout << endl;

cout << "Копируем информацию из первого объекта во второй" << endl << endl;

lazerni\_printer print2(print1);

cout << "Выводим информацию о втором объекте:" << endl << endl;

print2.print();

cout << "Вводим информацию о третьем объекте" << endl << endl;

lazerni\_printer print3;

cout << "Копируем информацию из третьего объекта в первый" << endl << endl;

lazerni\_printer print1=print3;

cout << "Выводим информацию о первом объекте:" << endl << endl;

print1.printer::print();

system("pause");

}

**Изначальный код.**

//#include "stdafx.h"

#include"iostream"

#include<string>

#include<string.h>

#include<cstring>

usingnamespace std;

template<classT>

class printer {

public:где копирующий конструктор?

printer();

printer(inta, intn, Tb, Tc, Td, Te, char\* f, char\* g);

printer&operator=(constprinter&m);

//~printer(); деструктор должен быть

voidprint();

printer&operator+=(constprinter&m2);

printer&operator>(constprinter&m);

printer&operator+(constprinter&m);

protected:

intfont\_size;//размершрифта

intnumber\_of\_pages;//количествостраниц

Tindent\_on\_the\_left;//отступслева

Tindent\_on\_the\_right;//отступсправа

Tindent\_on\_the\_from\_above;//отступсверху

Tindent\_on\_the\_from\_below;//отступснизу

char\* color;//цветшрифта

char\* text;// текст

private:

};

template<classT>printer<T>::printer() {

font\_size = 1;

number\_of\_pages = 2;

indent\_on\_the\_left = 3;

indent\_on\_the\_right = 4;

indent\_on\_the\_from\_above = 5;

indent\_on\_the\_from\_below = 6;

charf[100] = "not text";

//delete[]color;

color = newchar[strlen(f) + 1];

strcpy(color, f);

charg[100] = "not text";

//delete[]text;

text = newchar[strlen(g) + 1];

strcpy(text, g);

}

template<classT>printer<T>::printer(inta, intn, Tb, Tc, Td, Te, char\* f, char\* g) {

font\_size = a;

number\_of\_pages = n;

indent\_on\_the\_left = b;

indent\_on\_the\_right = c;

indent\_on\_the\_from\_above = d;

indent\_on\_the\_from\_below = e;

//delete[]color;

color = newchar[strlen(f) + 1];

strcpy(color, f);

//delete[]text;

text = newchar[strlen(g) + 1];

strcpy(text, g);

}

/\*template <class T> printer<T>::~printer() {delete[]color;delete[]text;}\*/

template<classT>printer<T>&printer<T>::operator=(constprinter<T>&m) { //(полноеимя) ("::" операторрасширенияобластивидимости)

//если адрес не равен содержимому адресу this

А если равен ?

font\_size = m.font\_size;

number\_of\_pages = m.number\_of\_pages;

indent\_on\_the\_left = m.indent\_on\_the\_left;

indent\_on\_the\_right = m.indent\_on\_the\_right;

indent\_on\_the\_from\_above = m.indent\_on\_the\_from\_above;

indent\_on\_the\_from\_below = m.indent\_on\_the\_from\_below;

color = newchar[strlen(m.color) + 1];

strcpy(color, m.color);

text = newchar[strlen(m.text) + 1];

strcpy(text, m.text);

return \*this;

}

template<classT>printer<T>&printer<T>::operator>(constprinter<T>&m)

{

if (font\_size<m.font\_size) cout<<"Размершрифта = small"<<endl;

if (font\_size>m.font\_size) cout<<"Размершрифта = big"<<endl;

elsecout<<"Размершрифта = ravni"<<endl;

if (number\_of\_pages<m.number\_of\_pages) cout<<"Количествостраниц = small"<<endl;

if (number\_of\_pages>m.number\_of\_pages) cout<<"Количествостраниц = big"<<endl;

elsecout<<"Количество страниц = ravni"<<endl;

if (indent\_on\_the\_left<m.indent\_on\_the\_left)cout<<"Отступслева = small"<<endl;

if (indent\_on\_the\_left>m.indent\_on\_the\_left)cout<<"Отступслева = big"<<endl;

elsecout<<"Отступслева = ravni"<<endl;

if (indent\_on\_the\_right<m.indent\_on\_the\_right)cout<<"Отступсправа = small"<<endl;

if (indent\_on\_the\_right>m.indent\_on\_the\_right)cout<<"Отступсправа = big"<<endl;

elsecout<<"Отступсправа = ravni"<<endl;

if (indent\_on\_the\_from\_above<m.indent\_on\_the\_from\_above)cout<<"Отступсверху = small"<<endl;

if (indent\_on\_the\_from\_above>m.indent\_on\_the\_from\_above)cout<<"Отступсверху = big"<<endl;

elsecout<<"Отступсверху = ravni"<<endl;

if (indent\_on\_the\_from\_below<m.indent\_on\_the\_from\_below)cout<<"Отступснизу = small"<<endl;

if (indent\_on\_the\_from\_below>m.indent\_on\_the\_from\_below)cout<<"Отступснизу = big"<<endl;

elsecout<<"Отступснизу = ravni"<<endl;

if (strlen(text) <strlen(m.text))cout<<"Длинатекста = small"<<endl;

if (strlen(text) >strlen(m.text))cout<<"Длинатекста = big"<<endl;

elsecout<<"Длина текста = ravni"<<endl;

return (\*this);

}

template<classT>printer<T>&printer<T>::operator+=(constprinter<T>&m)

{

intdlin;

font\_size = font\_size + m.font\_size;

number\_of\_pages = number\_of\_pages + m.number\_of\_pages;

indent\_on\_the\_left = indent\_on\_the\_left + m.indent\_on\_the\_left;

indent\_on\_the\_right = indent\_on\_the\_right + m.indent\_on\_the\_right;

indent\_on\_the\_from\_above = indent\_on\_the\_from\_above + m.indent\_on\_the\_from\_above;

indent\_on\_the\_from\_below = indent\_on\_the\_from\_below + m.indent\_on\_the\_from\_below;

dlin = strlen(color) + strlen(m.color);

if (strlen(color) <strlen(m.color)) color = newchar[strlen(m.color) + 1];

else color = newchar[strlen(color) + 1];

sprintf(color, "%d", dlin);

dlin = strlen(color) + strlen(m.color);

if (strlen(text) <strlen(m.text)) text = newchar[strlen(m.text) + 1];

else text = newchar[strlen(text) + 1];

sprintf(text, "%d", dlin);

return (\*this);

}

template<classT>printer<T>&printer<T>::operator+(constprinter<T>&m)

{

printer n;

intdlin;

n.font\_size = font\_size + m.font\_size;

n.number\_of\_pages = number\_of\_pages + m.number\_of\_pages;

n.indent\_on\_the\_left = indent\_on\_the\_left + m.indent\_on\_the\_left;

n.indent\_on\_the\_right = indent\_on\_the\_right + m.indent\_on\_the\_right;

n.indent\_on\_the\_from\_above = indent\_on\_the\_from\_above + m.indent\_on\_the\_from\_above;

n.indent\_on\_the\_from\_below = indent\_on\_the\_from\_below + m.indent\_on\_the\_from\_below;

dlin = strlen(color) + strlen(m.color);

if (strlen(color) <strlen(m.color)) n.color = newchar[strlen(m.color) + 1];

else color = newchar[strlen(color) + 1];

sprintf(n.color, "%d", dlin);

dlin = strlen(color) + strlen(m.color);

if (strlen(text) <strlen(m.text)) n.text = newchar[strlen(m.text) + 1];

elsen.text = newchar[strlen(text) + 1];

sprintf(n.text, "%d", dlin);

return n;

}

template<classT>voidprinter<T>::print() { //определениефункции

cout<<"Размершрифта "<<font\_size<<endl;//размершрифта

cout<<"Количествостраниц "<<number\_of\_pages<<endl;

cout<<"Отступслева "<<indent\_on\_the\_left<<endl;//отступслева

cout<<"Отступсправа "<<indent\_on\_the\_right<<endl;//отступсправа

cout<<"Отступсверху "<<indent\_on\_the\_from\_above<<endl;//отступсверху

cout<<"Отступснизу "<<indent\_on\_the\_from\_below<<endl;//отступснизу

cout<<"Цветшрифта "<< color <<endl;//цветшрифта

cout<<"Текст "<< text <<endl;// текст

}

Class lazerni\_printer:public printer<int> {

public:копирующий конструктор и деструктор, вызывающий деструктор базового класса

lazerni\_printer();

lazerni\_printer(inta, intn, intb, intc, intd, inte, char\* f, char\* g, intk, ints);

voidprint();

lazerni\_printer&operator=(constlazerni\_printer&m);

lazerni\_printer&operator+(constlazerni\_printer&m);

private:

intkol\_ener;

intskorost;

};

lazerni\_printer::lazerni\_printer():printer<int>::printer() {

kol\_ener = 360;

skorost = 100;

}

lazerni\_printer::lazerni\_printer(inta, intn, intb, intc, intd, inte, char\* f, char\* g, intk, ints):printer<int>::printer(a, n, b, c, d, e, f, g) {

kol\_ener = k;

skorost = s;

}

voidlazerni\_printer::print(){

printer<int>::print();

cout<<"kol\_ener "<<kol\_ener<<endl;

cout<<"skorost "<<skorost<<endl;

}

lazerni\_printer&lazerni\_printer::operator=(constlazerni\_printer&m) {

printer<int>::operator=(m);

kol\_ener = m.kol\_ener;

skorost = m.skorost;

return \*this;

}

lazerni\_printer&lazerni\_printer::operator+(constlazerni\_printer&m)

{

lazerni\_printer n;

n.kol\_ener = kol\_ener + m.kol\_ener;

n.skorost = skorost + m.skorost;

return n;

}

intmain() {

setlocale(LC\_ALL, "Russian");

charcol[10] = "red";

chartex[100] = "redtextred"; //"Красный цвет это цвет текста";

char tex1[100] = "redstextred";//"Другой текст для третьего объекта";

cout<<"Вводим информацию о первом объекте"<<endl<<endl;

lazerni\_printer print1(12, 25, 2.5, 3.2, 1.2, 2.5, col, tex, 360, 100);

cout<<"Выводим информацию о первом объекте:"<<endl<<endl;

print1.print();

cout<<endl;

cout<<"Копируем информацию из первого объекта во второй"<<endl<<endl;

lazerni\_printer print2 = print1;

cout<<"Выводим информацию о втором объекте:"<<endl<<endl;

print2.print();

cout<<"Вводим информацию о третьем объекте"<<endl<<endl;

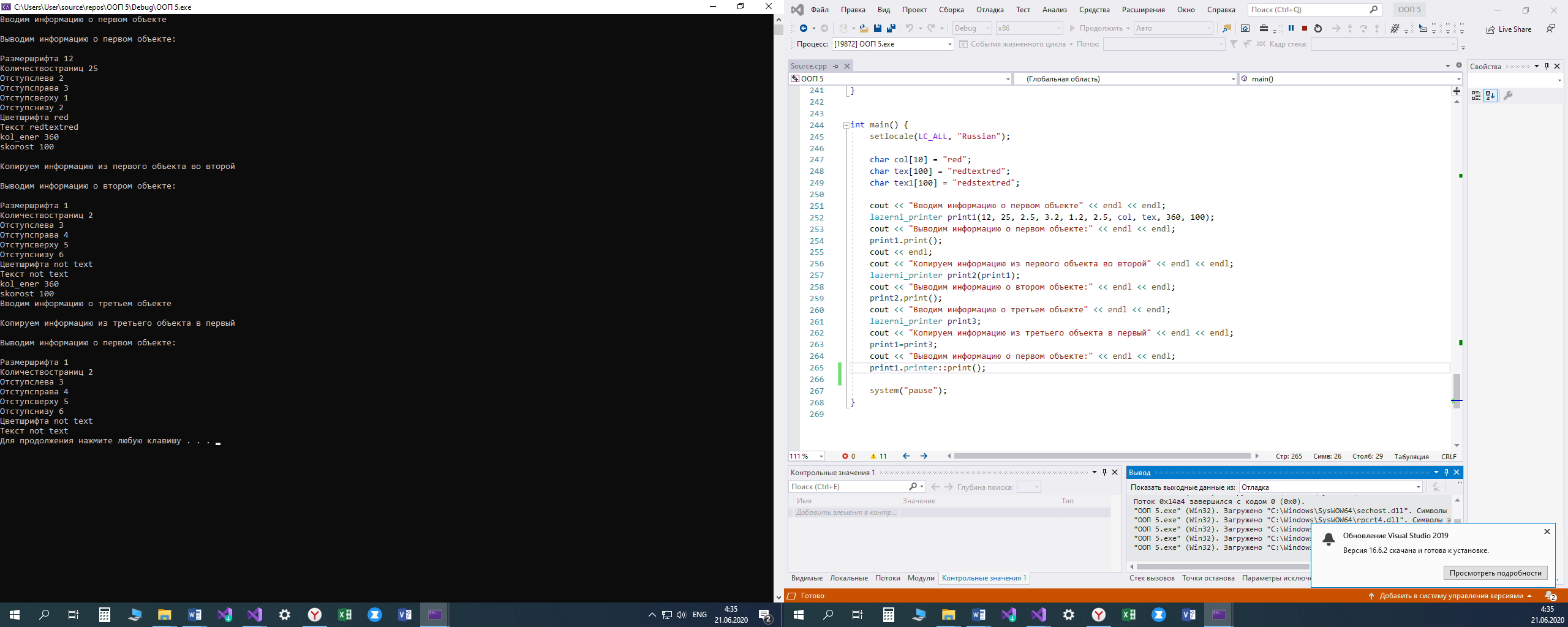
lazerni\_printer print3;

system("pause");

}

необходимо проверять функции базового и производного классов, вызывая их через объекты производного класса

**Тестирование программы:**



**Контрольные вопросы:**

1. Дайте определение наследования.

Наследование (inheritance) – это механизм получения нового класса на основе существующего класса. Существующий класс может быть дополнен или изменен для создания производного класса.Наследование осуществляется с помощью конструкции:

classимя\_класса:

public|protected|private)optимя\_базового\_класса

{

объявления членов

};

1. Какие модификаторы прав доступа к производному классу вы знаете и в чем их особенности?

Модификаторы прав доступа к членам класса (public, pro-tected и private) могут применяться в объявлении класса в любом порядке и сколько угодно раз. Открытый член доступен во всей области видимости, где виден класс. Закрытый член доступен другим функциям-членам своего класса. Защищенный член доступен не только другим функциям-членам своего класса, но и функциям-членам класса, унаследованного непосредственно от данного класса. Если все члены открыты, то ключевое слово class можно заменить ключевым словом struct.

Если в базовом классе отсутствует конструктор по умолчанию, то производный класс обязательно должен иметь конструктор. Если же в базовом классе имеются конструкторы, требующие аргументы, то производный класс явно обращается к конструктору базового класса в своем инициализирующем списке:

*имя\_класса (список\_арг) :*

*имя\_базового\_классаopt (список\_арг\_базового\_класса)*

Список\_арг\_базового\_класса используется при вызове соответствующего конструктора базового класса и реализуется перед тем как будет выполняться тело конструктора производного класса.

Открыто наследуемый класс является подтипом своего базового класса. Во многих случаях переменная производного класса может рассматриваться как переменная типа базового класса. Указатель, тип которого «указатель на базовый класс», может указывать на объекты открыто наследуемого класса. Ссылка на производный класс неявно может быть преобразована в ссылку на открытый базовый класс. Можно объявить ссылку на базовый класс и инициализировать ее ссылкой на объект открыто наследуемого класса.

1. Как выполняется конструктор при наследовании?

При наследовании и инициализации членов класса конструкторы выполняются в следующем порядке:

1. Базовые классы инициализируются в порядке объявления.

2. Члены инициализируются в порядке объявления.

Виртуальные базовые классы создаются до того, как создан любой из производных классов, и до того, как созданы не виртуальные базовые классы. Порядок их создания – «из глубины, слева направо». Деструкторы вызываются в обратном выполнению конструкторов порядке.

1. Дайте определение чисто виртуальной функции.

Ключевое слово virtual служит спецификатором функции, который обеспечивает механизм для динамического выбора на этапе выполнения подходящей функции-члена среди функций базового и производного классов. Оно может применяться для изменения объявлений только функций-членов. Виртуальная функция должна иметь исполняемое тело и имеет семантику вызова. Виртуальная функция может замещаться в производном классе. Выбор того, какое определение функции вызвать для виртуальной функции, происходит динамически на этапе выполнения. Самый распространенный случай – это когда базовый класс содержит виртуальную функцию, а производные классы имеют свои версии этой функции. Указатель базового класса может указывать либо на объект базового класса, либо на объект производного класса. Выбор вызываемой функции-члена будет произведен на этапе выполнения и будет зависеть от типа объекта, а не от типа указателя. При отсутствии члена производного типа по умолчанию используется виртуальная функция базового класса.

1. Какие операторы используются для разыменования указателя на член класса?

Указатель на член класса имеет тип T::\*, где T – имя класса. Для разыменования указателя на член класса используются два оператора: .\* и →\*.

Рассмотрим, например, выражение z.\*fet. Здесь сначала разыменовывается указатель для получения переменной-члена, а затем происходит доступ к члену объекта z.