МИНОБРНАУКИ РОССИИ

ФГБОУ ВО Череповецкий государственный университет

Институт информационных технологий

Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: Объектно-ориентированное программирование

ЛАБОРАТОРНАЯ РАБОТА № 6

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ОТНОШЕНИЯ

МНОЖЕСТВЕННОГО НАСЛЕДОВАНИЯ

Выполнил:

студент гр.

1ПИб-01-21оп

МартынюкМ.В.

Проверил:

ЕршовЕ.В.

Череповец, 2020 г.

**Цель работы:**

Изучить принципы и механизмы множественного наследования, правила доступа к базовым классам; приобрести практические навыки работы с базовыми и производными классами при множественном наследовании.

**Задание:**

Создайте базовый класс для реализации множественного наследования по заданию лабораторной работы 5. Проверьте работоспособность АТД на тестовом наборе данных.

**Код программы:**

#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;

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

strcpy(color, f);

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

}

//-------------------------------сканер------------------------------------------

class scaner{

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

scaner();

scaner(int c\_c, char\* text);

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

void print();

~scaner();

private:

int s\_s; //скорость сканирования

int g\_c; //глубина цвета

protected:

int c\_c; //количество копий

char\* text; //просканированный текст

};

scaner::scaner(){

s\_s = 1.5;

g\_c = 24;

c\_c = 0;

char f[100] = "not text";

text=new char[strlen(f) + 1];; //просканированный текст

strcpy(text, f);

}

scaner::scaner(int a, char\* b) {

c\_c = a;

text = new char[strlen(b) + 1];; //просканированный текст

strcpy(text, b);

}

scaner::~scaner() { delete[]text; } //!!!!

scaner::scaner(const scaner& m) {

if (&m != this) {

s\_s = m.s\_s;

g\_c = m.g\_c;

c\_c = m.c\_c;

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

strcpy(text, m.text);

}

}

void scaner::print() {

cout << "Cкорость сканирования" << s\_s << endl;

cout << "Глубина цвета " << g\_c << endl;

cout << "Количество копий " << c\_c << endl;

cout << "Просканированный текст " << text << endl;

}

//-------------------------------объединение----------------------------------------

class printer\_and\_scaner : public printer<int>, public scaner{

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

printer\_and\_scaner();

printer\_and\_scaner(int a, int n, int b, int c, int d, int e, char\* f, char\* g, int c\_c,char\* text ,int k); //k -kol\_ener, s - skorost

printer\_and\_scaner(const printer\_and\_scaner& m); //копирующий конструктор

void print();

~printer\_and\_scaner();

private:

int kol\_ener;

};

printer\_and\_scaner::printer\_and\_scaner() :printer(), scaner(){

kol\_ener = 360;

}

printer\_and\_scaner::printer\_and\_scaner(int a, int n, int b, int c, int d, int e, char\* f, char\* g, int c\_c, char\* text, int k) : printer<int>(a, n, b, c, d, e, f, g), scaner(c\_c, text) {

kol\_ener = k;

}

printer\_and\_scaner::~printer\_and\_scaner() {

printer::~printer();

scaner::~scaner();

}

printer\_and\_scaner::printer\_and\_scaner(const printer\_and\_scaner& m) {

if (&m != this) {

printer::printer();

scaner::scaner();

kol\_ener = m.kol\_ener;

}

}

void printer\_and\_scaner::print() {

printer::print();

scaner::print();

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

}

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;

printer\_and\_scaner print1(12, 25, 2.5, 3.2, 1.2, 2.5, col, tex, 20, tex1 ,360);

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

print1.print();

cout << endl;

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

printer\_and\_scaner print2(print1);

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

print2.print();

system("pause");

}

**Тестирование программы:**

