|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по домашнему заданию №** | 3 |

**Название:** Программирование на С++ с использованием классов

**Дисциплина:** Объектно-ориентированное программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22Б |  |  | Д.С.Твердюк |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.А. Веселовский |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Часть 1. Композиция**

Разработать и реализовать диаграмму классов для описанных объектов предметной области, используя механизм композиции. Протестировать все методы каждого класса. Все поля классов должны быть скрытыми (private) или защищенными (protected). Методы не должны содержать операций ввода/вывода, за исключением процедуры, единственной задачей которой является вывод информации об объекте на экран.

Объект – товарный вагон. Поля: грузоподъемность вагона, масса находящегося в нем груза и тип груза. Методы: процедура инициализации, процедура вывода параметров вагона на экран, функция расчета процента заполнения вагона и функции, возвращающие значение каждого поля по запросу.

Объект – товарный поезд. Содержит число вагонов в поезде (по умолчанию 0) и сами вагоны. Методы объекта должны позволять: прицепить к хвосту поезда вагон с заданными параметрами, отцепить последний вагон, вывести на экран информацию обо всех вагонах, рассчитать общую грузоподъемность поезда, получить количество вагонов, заполненных более чем наполовину.

Объектная декомпозиция:

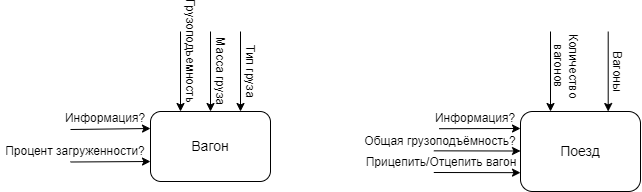


Диаграмма классов:

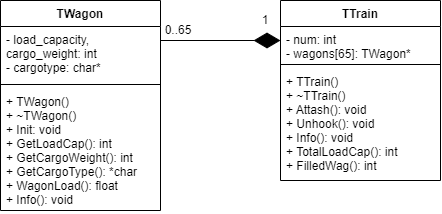
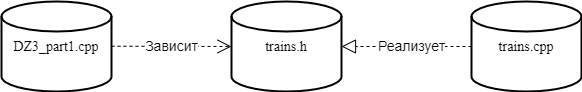


Диаграмма компоновки:



Файл trains.h:

#ifndef TRAINS\_h

#define TRAINS\_h

#include <string>

#include <iostream>

class TWagon {

private:

int load\_capacity, cargo\_weight;

char\* cargotype;

public:

TWagon() {

load\_capacity = 0;

cargo\_weight = 0;

cargotype = nullptr;

}

TWagon(int l, int c, const char\* t) {

Init(l, c, t);

}

~TWagon() { delete [] cargotype; }

void Init(int l, int c, const char\* t) {

load\_capacity = l;

cargo\_weight = c;

if (cargotype != nullptr)

delete[] cargotype;

cargotype = new char[sizeof(t) + 1];

strcpy\_s(cargotype, sizeof(t)+1, t);

}

int GetLoadCap() {

return load\_capacity;

}

int GetCargoWeight() {

return cargo\_weight;

}

char\* GetCargoType() {

return cargotype;

}

float WagonLoad() {

return (float)cargo\_weight / (float)load\_capacity \* 100;

}

void Info() {

std::cout << "Info about wagon:\n" << "Cargo type: " << cargotype << "\nLoad capacity: " << load\_capacity <<

"\nCargo weight: " << cargo\_weight << "\nWagon load(%):" << WagonLoad() << "\n-------\n";

}

};

class TTrain {

private:

int num;

TWagon\* wagons[65];

public:

TTrain();

~TTrain();

void Attash(int l, int c, const char\* t);

void Unhook();

void Info();

int TotalLoadCap();

int FilledWag();

};

#endif

Файл train.cpp:

#include "trains.h"

TTrain::TTrain() {

int i;

num = 0;

for (i = 0; i < 65; i++)

wagons[i] = nullptr;

}

TTrain::~TTrain() {

int i;

for (i = 0; (i < 65); i++) {

if (wagons[i] == nullptr)

break;

delete wagons[i];

}

}

void TTrain::Attash(int l, int c, const char\* t) {

if (num < 65) {

wagons[num] = new TWagon(l, c, t);

num++;

std::cout << "Wagon is successfully attached...\n";

}

else std::cout << "The maximum number of wagons exceeded!\n";

}

void TTrain::Unhook() {

if (num > 0) {

num--;

delete wagons[num];

wagons[num] = nullptr;

std::cout << "Wagon has been successfully unhooked...\n";

}

else std::cout << "There are no wagons on the train.\n";

}

void TTrain::Info() {

int i;

if (num > 0) {

std::cout << "Info about train:\n" << "Total number of wagons: " << num <<

"\nTotal load capacity: " << TotalLoadCap() << "\nNumber of filled wagons: " <<

FilledWag() << "\nWagons:\n";

for (i = 0; (i < 65); i++) {

if (wagons[i] == nullptr)

break;

std::cout << i+1 << ":\n";

wagons[i]->Info();

}

}

else std::cout << "There are no wagons on the train.";

}

int TTrain::TotalLoadCap() {

int i, lc = 0;

for (i = 0; i < 65; i++) {

if (wagons[i] == nullptr)

break;

lc += wagons[i]->GetLoadCap();

}

return lc;

}

int TTrain::FilledWag() {

int i, cnt=0;

for (i = 0; i < 65; i++) {

if (wagons[i] == nullptr)

break;

if (wagons[i]->WagonLoad() > 50)

cnt += 1;

}

return cnt;

}

Файл DZ3\_part1.cpp:

#include <iostream>

#include "trains.h"

int main()

{

using namespace std;

int lc, cw;

char ct[50];

//Работа с объектом класса TWagon

cout << "<< Working with an object of the TWagon class >>\n";

TWagon w;

cout << "Enter the wagon's load capacity, weight and type of cargo: ";

cin >> lc >> cw >> ct;

w.Init(lc, cw, ct);

w.Info();

//Работа с объектом класса TTrain

cout << "\n<< Working with an object of the TTrain class >>\n";

TTrain t;

int choice;

while (true) {

cout << "1. Attach wagon to the train\n" <<

"2. Unhook wagon from the train\n" <<

"3. Print information about the train\n" <<

"0. End the program\n" <<

"\nEnter a number -> ";

cin >> choice;

switch (choice) {

case 1: {

cout << "Enter the wagon's load capacity, weight and type of cargo: ";

cin >> lc >> cw >> ct;

t.Attash(lc, cw, ct);

break;

}

case 2: {

t.Unhook();

break;

}

case 3: {

t.Info();

break;

}

case 0: {

return 0;

}

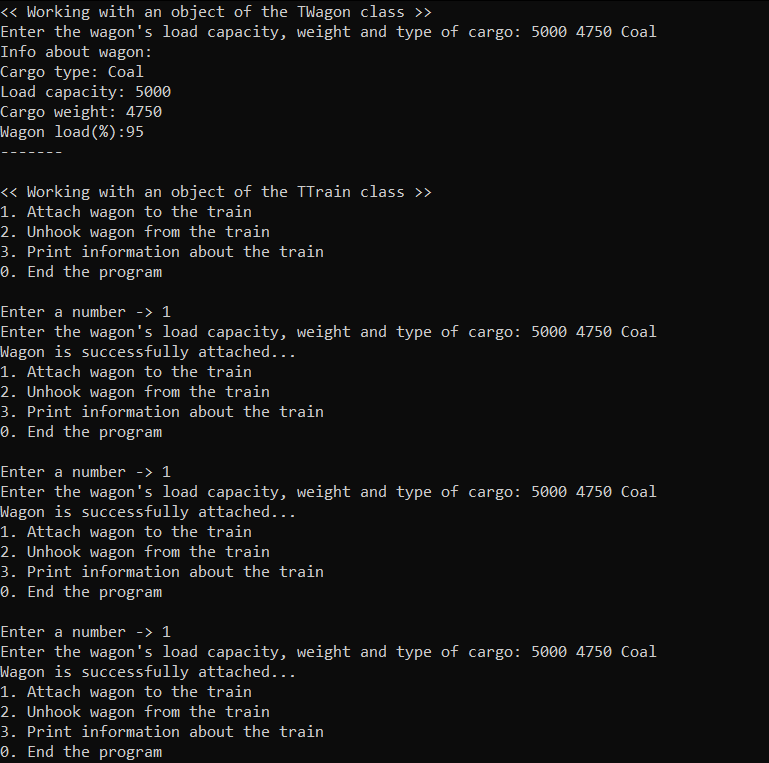
default: cout << "Wrong option!\n\n";

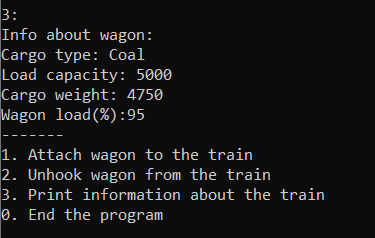
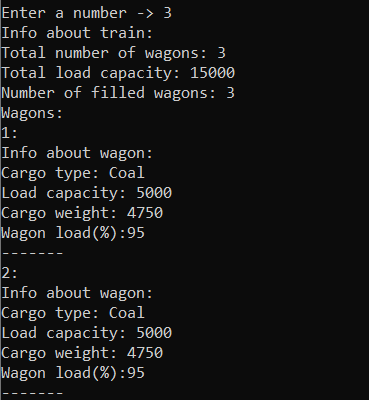
}

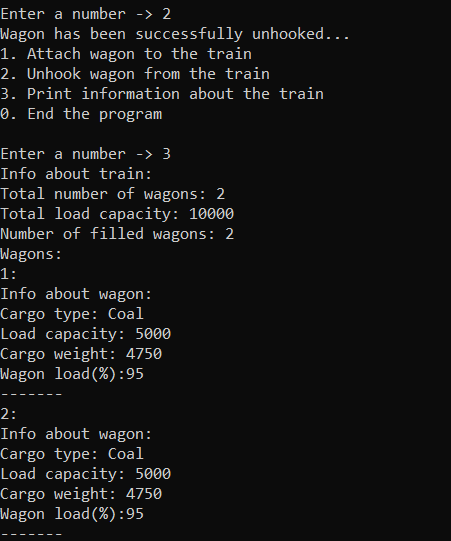
}

}

Результат работы написанной программы:







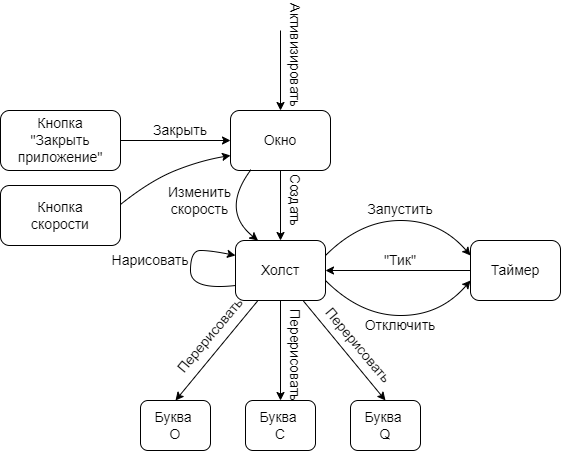
**Вывод:** написанная программа работает корректно на заданном отрезке исходных данных.

**Часть 2. Qt. Полиморфное наследование**

Разработать программу, содержащую описание трех графических объектов: изображения букв: С, О, Q.

Реализуя механизм полиморфизма, привести объекты в вертикальное движение по экрану с различными скоростями с отражением от границ экрана.

Объектная декомпозиция:



Граф состояний интерфейса:

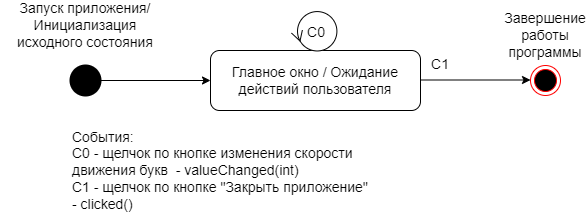
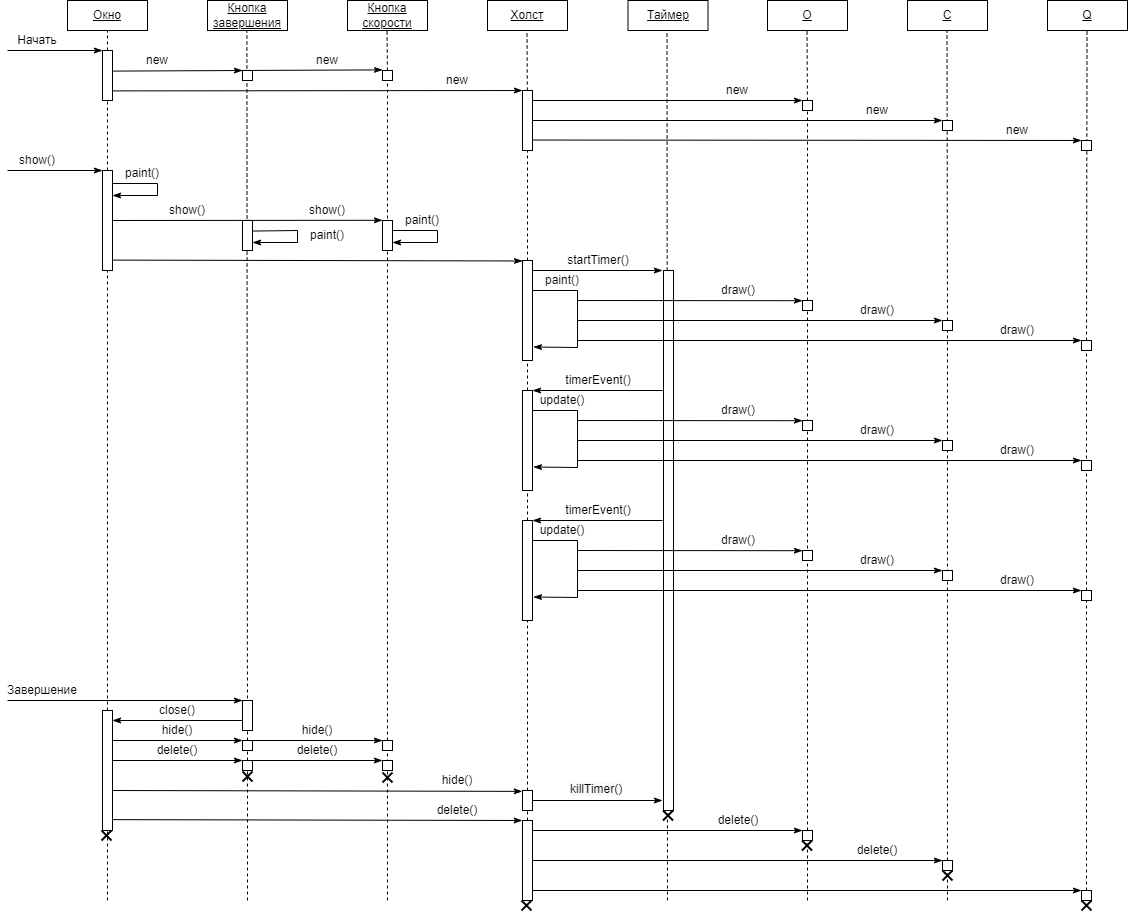


Диаграмма последовательностей:



Объектная декомпозиция:

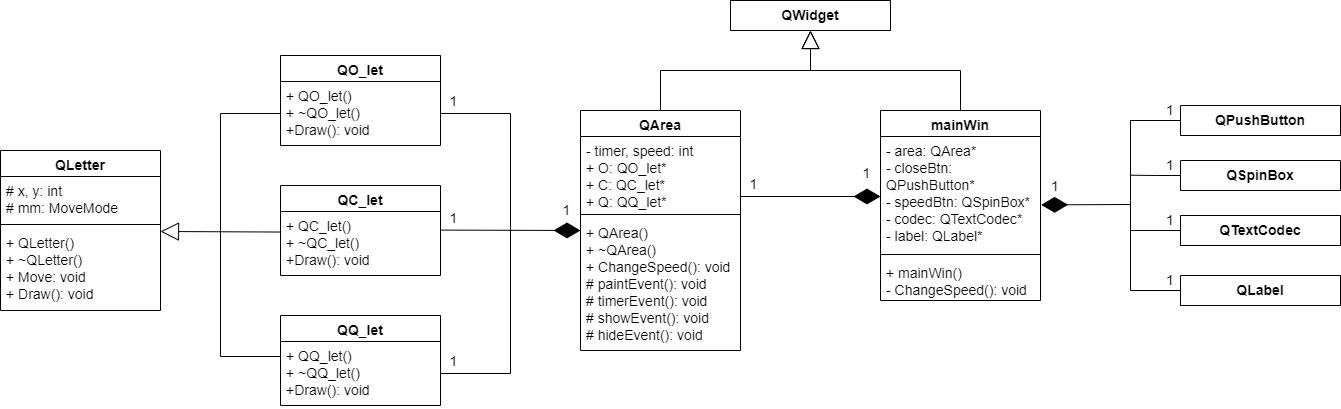
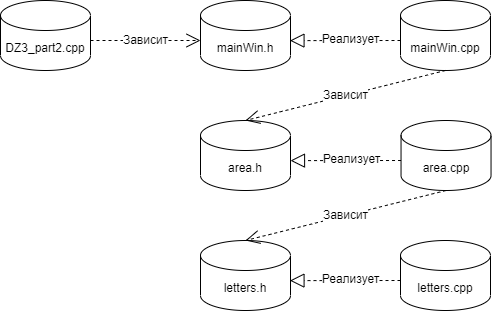


Диаграмма компоновки:



Файл letters.h:

#ifndef LETTERS\_H

#define LETTERS\_H

#include <QPainter>

enum MoveMode {Down, Up};

class QLetter {

protected:

int x, y;

MoveMode mm;

public:

QLetter() {

x = y = 0;

mm = Down;

}

QLetter(int ax, int ay, MoveMode amm) {

x = ax;

y = ay;

mm = amm;

}

virtual ~QLetter(){};

void Move(int u, QPainter \*painter);

virtual void Draw(QPainter \*painter) = 0;

};

class QO\_let: public QLetter {

public:

QO\_let(int ax, int ay, MoveMode amm) :

QLetter(ax, ay, amm) {};

~QO\_let(){};

void Draw(QPainter \*painter) override;

};

class QC\_let: public QLetter {

public:

QC\_let(int ax, int ay, MoveMode amm) :

QLetter(ax, ay, amm) {};

~QC\_let(){};

void Draw(QPainter \*painter) override;

};

class QQ\_let: public QLetter {

public:

QQ\_let(int ax, int ay, MoveMode amm) :

QLetter(ax, ay, amm) {};

~QQ\_let(){};

void Draw(QPainter \*painter) override;

};

#endif // LETTERS\_H

Файл letters.cpp:

#include "letters.h"

void QLetter::Move(int u, QPainter \*painter) {

*switch* (mm) {

*case* *Down*: {

*if* (y < 120)

y += u;

*else* mm = *Up*;

*break*;

}

*case* *Up*: {

*if* (y > 5)

y -= u;

*else* mm = *Down*;

*break*;

}

}

*Draw*(*painter*);

}

void QO\_let::*Draw*(QPainter \*painter) {

painter->drawEllipse(x, y, 40, 75);

}

void QC\_let::*Draw*(QPainter \*painter) {

painter->drawArc(x, y, 35, 75, 85\*16, 200\*16);

}

void QQ\_let::*Draw*(QPainter \*painter) {

painter->drawEllipse(x, y, 45, 75);

painter->drawLine(x+23, y+74, x+33, y+80);

painter->drawLine(x+33, y+80, x+23, y+77);

painter->drawLine(x+23, y+77, x+23, y+74);

}

Файл area.h:

#ifndef AREA\_H

#define AREA\_H

#include "letters.h"

#include <QWidget>

*class* QArea: *public* QWidget {

*private*:

int timer, speed;

*public*:

QO\_let \*O;

QC\_let \*C;

QQ\_let \*Q;

QArea(QWidget \*parent = 0);

~QArea();

void ChangeSpeed(int s);

*protected*:

void *paintEvent*(QPaintEvent \*event);

void *timerEvent*(QTimerEvent \*event);

void *showEvent*(QShowEvent \*event);

void *hideEvent*(QHideEvent \*event);

};

#endif *//* *AREA\_H*

Файл area.cpp:

#include "area.h"

#include <QTimerEvent>

QArea::QArea(QWidget \*parent): QWidget(*parent*) {

setFixedSize(QSize(300,200));

QPainter painter(*this*);

O = *new* QO\_let(10, 50, *Up*);

O->*Draw*(&*painter*);

C = *new* QC\_let(110, 50, *Down*);

C->*Draw*(&*painter*);

Q = *new* QQ\_let(210, 50, *Up*);

Q->*Draw*(&*painter*);

speed = 0;

}

QArea::~QArea() {

*delete* O;

*delete* C;

*delete* Q;

}

void QArea::*paintEvent*(QPaintEvent \*)

{

QPainter painter(*this*);

painter.setPen(QPen(Qt::*black*, 3));

O->Move(speed, &*painter*);

C->Move(speed, &*painter*);

Q->Move(speed, &*painter*);

}

void QArea::*timerEvent*(QTimerEvent \*event)

{

*if* (event->timerId() == timer)

{

update();

}

*else*

QWidget::*timerEvent*(*event*);

}

void QArea::*showEvent*(QShowEvent \*)

{

timer=startTimer(10);

}

void QArea::*hideEvent*(QHideEvent \*)

{

killTimer(timer);

}

void QArea::ChangeSpeed(int s) {

speed = s;

}

Файл mainWin.h:

#ifndef MAINWIN\_H

#define MAINWIN\_H

#include "area.h"

#include <QWidget>

#include <QPushButton>

#include <QSpinBox>

#include <QTextCodec>

#include <QLabel>

*class* mainWin: *public* QWidget {

Q\_OBJECT

*public*:

mainWin(QWidget \* parent = 0) ;

*private*:

QArea \*area;

QPushButton \*closeBtn;

QSpinBox \*speedBtn;

QTextCodec \*codec;

QLabel \*label;

*private* slots:

void ChangeSpeed(int);

};

#endif *//* *MAINWIN\_H*

Файл mainWin.cpp:

#include "mainWin.h"

#include <QVBoxLayout>

mainWin::mainWin(QWidget \* parent): QWidget(*parent*) {

codec = QTextCodec::codecForName("UTF-8");

*this*->setWindowTitle("Буквы-Побегуквы");

label = *new* QLabel(codec->toUnicode("Скорость"), *this*);

area = *new* QArea(*this*);

speedBtn = *new* QSpinBox;

speedBtn->setMaximum(3);

speedBtn->setMinimum(0);

connect(speedBtn, SIGNAL(valueChanged(int)), *this*, SLOT(ChangeSpeed(int)));

closeBtn = *new* QPushButton(codec->toUnicode("Закрыть приложение"), *this*);

connect(closeBtn, SIGNAL(clicked()), *this*, SLOT(close()));

QVBoxLayout \*layout = *new* QVBoxLayout(*this*);

layout->addWidget(*label*);

layout->addWidget(*speedBtn*);

layout->addWidget(*area*);

layout->addWidget(*closeBtn*);

}

void mainWin::ChangeSpeed(int i) {

area->ChangeSpeed(i);

}

Файл DZ3\_part2.cpp:

#include "mainWin.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication appl(*argc*, *argv*);

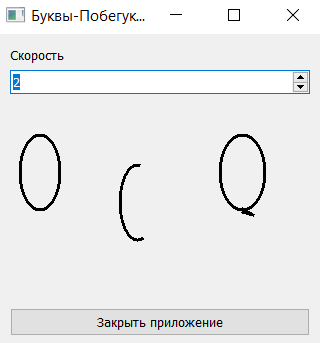
mainWin win;

win.show();

*return* appl.exec();

}

Результат работы написанной программы:



**Вывод**: написанная программа работает корректно на заданных исходных данных.