

**Министерство науки и высшего образования**  
**ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**  
**Физико-технический институт**

Индивидуальный отчет по курсовому проекту  
по дисциплине «Технология программирования»

Выполнили:  
студент гр. 21312  
Кадыко В.Л.  
Преподаватель:  
канд. физ.-мат. наук  
Бульба А.В.

**Петрозаводск 2020**

**Цель работы:** разработать игру на языке C++ с использованием библиотеки SFML.

**Программная реализация:**

Среда разработки: Visual Studio Express 2013;

Язык: C++;

Заголовочные файлы:

Entity.h – Содержит класс сущность. Данный класс является предком класса Игрок и Враг.

Объявляет координаты и размеры сущности, а также логические переменные для управления стартом игры

Player.h - Содержит класс Игрока (PacMan). Данный класс содержит поле очков, и выбранное направление. А также объявляет методы управления игроком, проверки на препятствия и обновления состояния.

Enemy.h - Содержит класс Врага (Призрака). Объявляет буферные поля для памяти предыдущих координат, поле направления, методы проверки на препятствие и обновления состояния.

Interface.h - Содержит класс самого игрового поля. Объявляет поля для текстур, спрайтов, шрифтов, таймеров, карты и список врагов, а также метод взаимодействия с игрой.

Map.h - Содержит класс карты. Объявляет поле карты.

Файлы реализации (.cpp):

Entity.cpp – Содержит конструктор, инициализирующий поля данного класса.

Player.cpp – Содержит реализацию методов получения полей, управления игроком, проверкой на препятствия и обновлением, так же содержит конструктор и деструктор

Enemy.cpp - Содержит реализацию конструктора, методов проверки на препятствия и обновления.

Interface.cpp – Содержит конструктор и деструктор, а так же реализацию метода взаимодействия с полем

Map.cpp – содержит в себе реализацию карты.

## Процесс разработки:

### Краткое словесное описание сюжета

К нам обратился владелец клуба со старыми игровыми аппаратами. Заказчик просит создать игру, похожую на классическую PacMan. В игре должно быть реализовано: игровое поле в виде лабиринта, в которых стенки являются препятствиями; точки, которые поедает главный персонаж и тем самым зарабатывает очки; 3 противника в виде приведений, при прикосновении с которыми заканчивается игра. Цель игры собрать все точки и не попасться призракам. Игра должна отображать текущий счет очков и возможность на подготовку к запуску игры и выход из нее по завершению»

### Список классов

- Экран – Данный класс будет содержать реализацию самой игры. Он будет отвечать за содержание игрового поля.
- Сущность – Данный класс будет являться предком для классов Игрок и Призрак. Он будет содержать их общие параметры. Например, координаты и размер.

## Код заголовочных файлов:

### Entity.h

```
#ifndef __ENTITY_H__
#define __ENTITY_H__
#include "stdafx.h"
#include "map.h"
class Entity { ///Класс сущности//
protected:
    float dx, dy, x, y, speed; //dx,dy изменение направления, x,y
    координаты, speed скорость изменения
    int w, h; //w-ширина, h-длина нашей сущности
    Texture texture;///текстура сущности
    Map mp; //объект класса карта
    float CurrentFrame;///текущий кадр
    bool life; //переменная жизни
    bool game;
public:
    Sprite sprite;///спрайт сущности
    Entity(Image &image, float X, float Y, int W, int H);///конструктор
    FloatRect getRect() //получение прямоугольника (координаты и размеры)
    {
        FloatRect FR(x, y, w, h);
        return FR;
    }
    bool getLife();
    void setLife(bool l);
    bool getGame();
    void setGame(bool l);
    virtual void update(float time) = 0; //метод обновления
};
```

```
#endif
```

### **Interface.h**

```
#ifndef INTERFACE_H
#define INTERFACE_H
#include "Enemy.h"
#include "Player.h"
#include "Entity.h"
#include "Player.h"
#include "stdafx.h"

using namespace sf;
class Interface
{
private:
    Player* p;
    Font font; //Шрифт для строк
    Texture map; //текстура карты
    Sprite s_map; //спрайт карты
    Clock clock, gameTimeClock; //таймеры
    int gameTime; //игровое время
    Map inter;
    Event event;
    Image map_image, heroImage, enemy1, enemy2, enemy3;
    std::list<Entity*> enemy; //список врагов
    std::list<Entity*>::iterator it; //итератор для списка врагов
public:
    Interface();
    ~Interface();
    void interact(); //интерфейс программы
};

#endif
```

### **Код исходных файлов:**

#### **Entity.cpp**

```
#include "stdafx.h"
#include "Entity.h"
using namespace sf;

Entity::Entity(Image &image, float X, float Y, int W, int H)
{
    x = X; y = Y;
```

```

        w = W; h = H;
        dx = 0.1; dy = 0.1; //изменение координат
        speed = 0.1; //скорость
        CurrentFrame = 0;
        life = false; //герой по умолчанию жив
        game = true;
        texture.loadFromImage(image); //изображение сущности
        sprite.setTexture(texture); //заливка спрайта для сущности
    }

    bool Entity::getLife(){
        return life;
    }

    void Entity::setLife(bool l){
        life = l;
    }

    bool Entity::getGame(){
        return game;
    }

    void Entity::setGame(bool l){
        game = l;
    }

```

### **Interface.cpp**

```

#include "stdafx.h"
#include "Interface.h"
#include "Player.h"
#include "Entity.h"
#include "Enemy.h"
using namespace sf;

Interface::Interface(){
    font.loadFromFile("fonts/Strenuous.ttf"); //установка шрифта
    map_image.loadFromFile("images/map.png"); //загружаем файл с текстурой
    карты
    map.loadFromImage(map_image); //заряжаем текстуру карты из картинки
    s_map.setTexture(map); //устанавливаем текстуру карты
    gameTime = 0; //начало игрового времени
    heroImage.loadFromFile("images/hero.png"); // загружаем изображение
    ПакМэна
    enemy1.loadFromFile("images/enemy1.png"); // загружаем изображение
    призрака
    enemy2.loadFromFile("images/enemy2.png");
    enemy3.loadFromFile("images/enemy3.png");
    p = new Player(heroImage, 288, 512, 30, 30);
}

```

```

Interface::~~Interface()
{
    delete p;
    while (!enemy.empty())
    {
        it = enemy.begin();
        delete *it;
        enemy.erase(it);
    }
};

void Interface::interact(){

    sf::VideoMode desktop = sf::VideoMode::getDesktopMode();
    sf::RenderWindow window(sf::VideoMode(608, 704, desktop.bitsPerPixel),
"PacMan");

    Text text("", font, 20), menu("", font, 30); //создаем объект текст
    text.setColor(Color::Yellow); //покрасили текст в красный
    text.setStyle(Text::Bold); //жирный текст.
    menu.setColor(Color::Yellow); //покрасили текст в красный
    menu.setStyle(Text::Bold); //жирный текст.
    //Player p(heroImage, 288, 512, 30, 30); //объект класса игрока
    srand(time(0));
    enemy.push_back(new Enemy(enemy1, 288, 288, 32, 32)); //создаем врагов
и помещаем в список
    enemy.push_back(new Enemy(enemy2, 256, 320, 32, 32));
    enemy.push_back(new Enemy(enemy3, 288, 320, 32, 32));

    while (window.isOpen()) { //пока открыто
        if (p->getGame() == true)
            if (Keyboard::isKeyPressed(Keyboard::Enter)){ p->setLife(true);
p->setGame(false); }
        float time = clock.getElapsedTime().asMicroseconds(); //таймер
логики
        if (p->getLife()) gameTime =
gameTimeClock.getElapsedTime().asSeconds(); //игровое время
        clock.restart(); //перезапуск таймера
        time = time / 800;

        while (window.pollEvent(event)) //обработчик событий на закрытие
        {
            if (event.type == sf::Event::Closed)
            {
                window.close();
            }
            if (Keyboard::isKeyPressed(Keyboard::Q)) {
                window.close();
            }
        }

        p->update(time); //обновление игрока
    }
}

```

```

        if (p->getLife())
        for (it = enemy.begin(); it != enemy.end(); it++)
        {
            (*it)->update(time); //запускаем метод update()
        }

        if (p->getLife() == true)
        { //если игрок жив
            for (it = enemy.begin(); it != enemy.end(); it++)
            { //бежим по списку врагов
                if ((p->getRect().intersects((*it)->getRect()))
                {
                    p->setLife(false);
                    std::cout << "Game over";
                }
            }
        }
    }

    window.clear();

    for (int i = 0; i < 22; i++) //отрисовка карты по шаблону из
map.cpp
    for (int j = 0; j < 19; j++)
    {
        if (inter.TileMap[i][j] == ' ')
s_map.setTextureRect(IntRect(0, 0, 32, 32)); //если пусто, то рисовать блок
земли
        if (inter.TileMap[i][j] == 's')
s_map.setTextureRect(IntRect(32, 0, 32, 32)); //если точка, то рисовать
кружок
        if ((inter.TileMap[i][j] == '0'))
s_map.setTextureRect(IntRect(64, 0, 32, 32)); //если препятствие, то рисовать
блок стены

        s_map.setPosition(j * 32, i * 32);
        window.draw(s_map); //рисовать
    }

    std::ostringstream playerScoreString, gameTimeString;
    playerScoreString << p->getScore(); gameTimeString <<
gameTime; //Получаем счёт и время в игре
    text.setString("Score: " + playerScoreString.str() + "
Time: " + gameTimeString.str()); //задаем строку тексту
    text.setPosition(5, 2); //задаем позицию текста
    window.draw(text); //рисуем этот текст

    if (p->getScore() == 176) p->setLife(false);
    if (p->getGame()){
        menu.setString("Press ENTER for start");
        menu.setPosition(100, 315); //задаем позицию текста
        window.draw(menu); //рисуем этот текст
    }
    if ((!p->getLife()) && (!p->getGame())){
        menu.setString("Press Q for exit");
        menu.setPosition(150, 315); //задаем позицию текста
    }

```

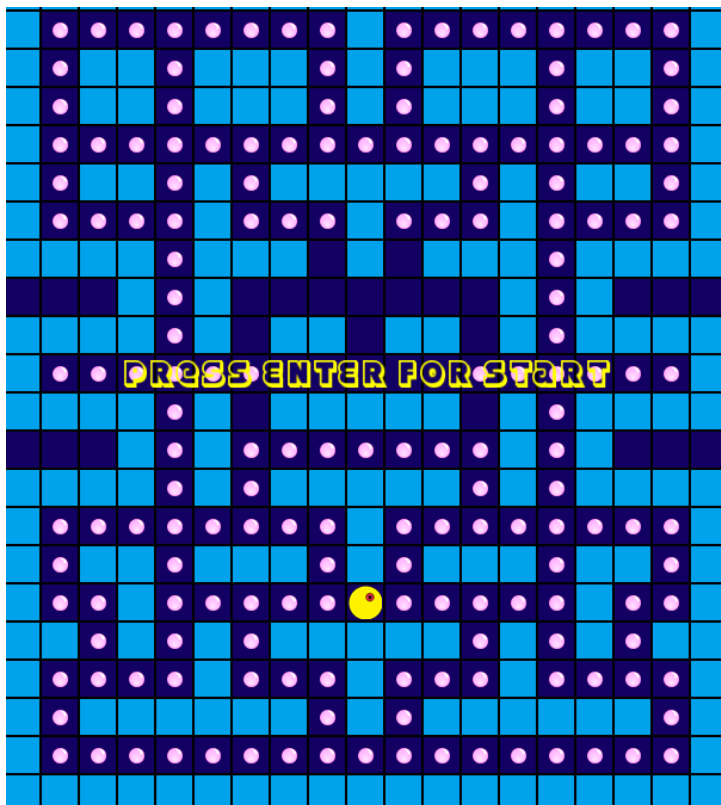
```

        window.draw(menu);//рисуем этот текст
    }
    window.draw(p->sprite);//рисуем спрайт ПакМэна
    for (it = enemy.begin(); it != enemy.end(); it++)
    {
        window.draw((*it)->sprite); //рисуем призраков
    }
    window.display();
}
}

```

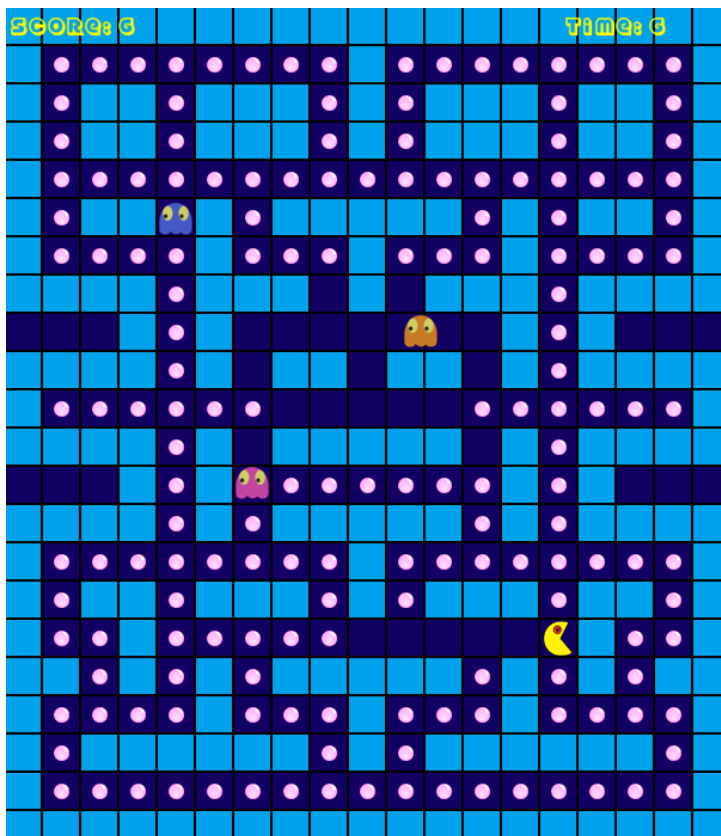
### Руководство пользователя

При запуске программы, первое что видит игрок это сообщение об запуске программы с помощью нажатия клавиши Enter:

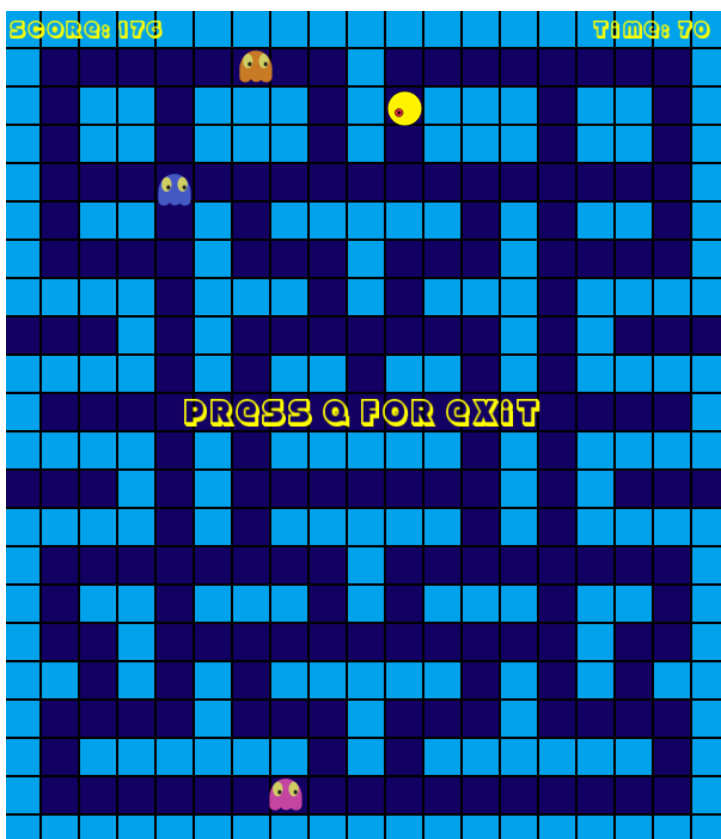


После нажатия клавиши запускается игра и на экране обновляются действия героя и призраков:





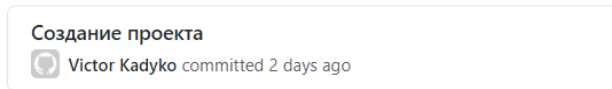
Собрав все точки либо погибнув от призрака, игра выдаст сообщения об выходе программы через кнопку Q:



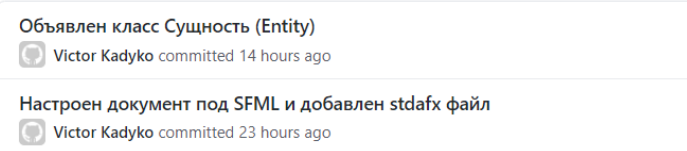
## История проекта на GitHub.

Адрес репозитория: <https://github.com/VitKad/Pac-man.git>

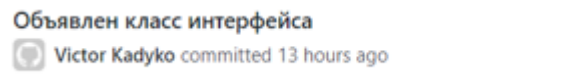
Victor Kadyko – Виктор Кадыко;



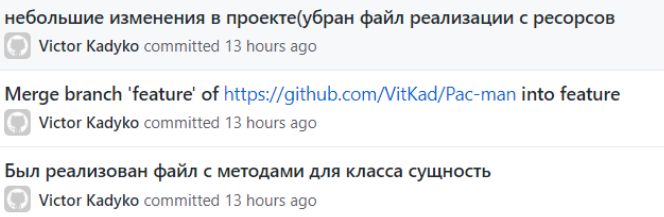
В ветке master был создан проект VS 2013.



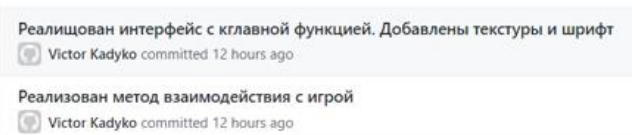
На этом этапе были произведены настройки документа для видимости графической sfml библиотеки. Так же был создан первый заголовочный файл сущности.



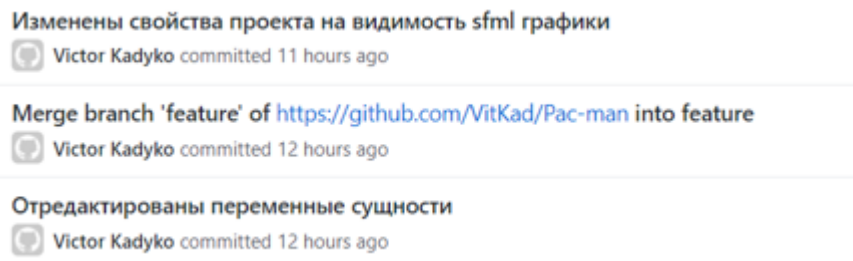
Здесь создан заголовочный файл класса экрана(интерфейса).



В данном коммите была добавлена реализация методов класса сущность, а так же создана функция main.



Здесь реализован метод взаимодействия (interact) для класса интерфейса. Так же в репозиторий были загружены текстуры и шрифт.



Были удалены неиспользованные переменные, а также было произведено небольшое редактирование проекта.

Исправлено отображение времени

 Victor Kadyko committed 11 hours ago

Далее мной было отредактировано отображение времени на игровом поле (не выходит за рамки).

Добавлены геттеры и сеттеры для класса сущности

 Victor Kadyko committed 3 hours ago

Добавлен старт и выход из игры


 Victor Kadyko committed 4 hours ago

На этом этапе появилась возможность старта и выхода из игры по нажатию кнопки, а так же добавлены геттеры и сеттеры для переменных логического типа.

Добавлены файлы SFML в Debug

 Victor Kadyko committed 1 hour ago

Избавился от спецификатора доступа public в классе врага

 Victor Kadyko committed 1 hour ago

Освобождение памяти для класса врагов

 Victor Kadyko committed 1 hour ago

Добавлен указатель на класс игрок, с освобождением памяти

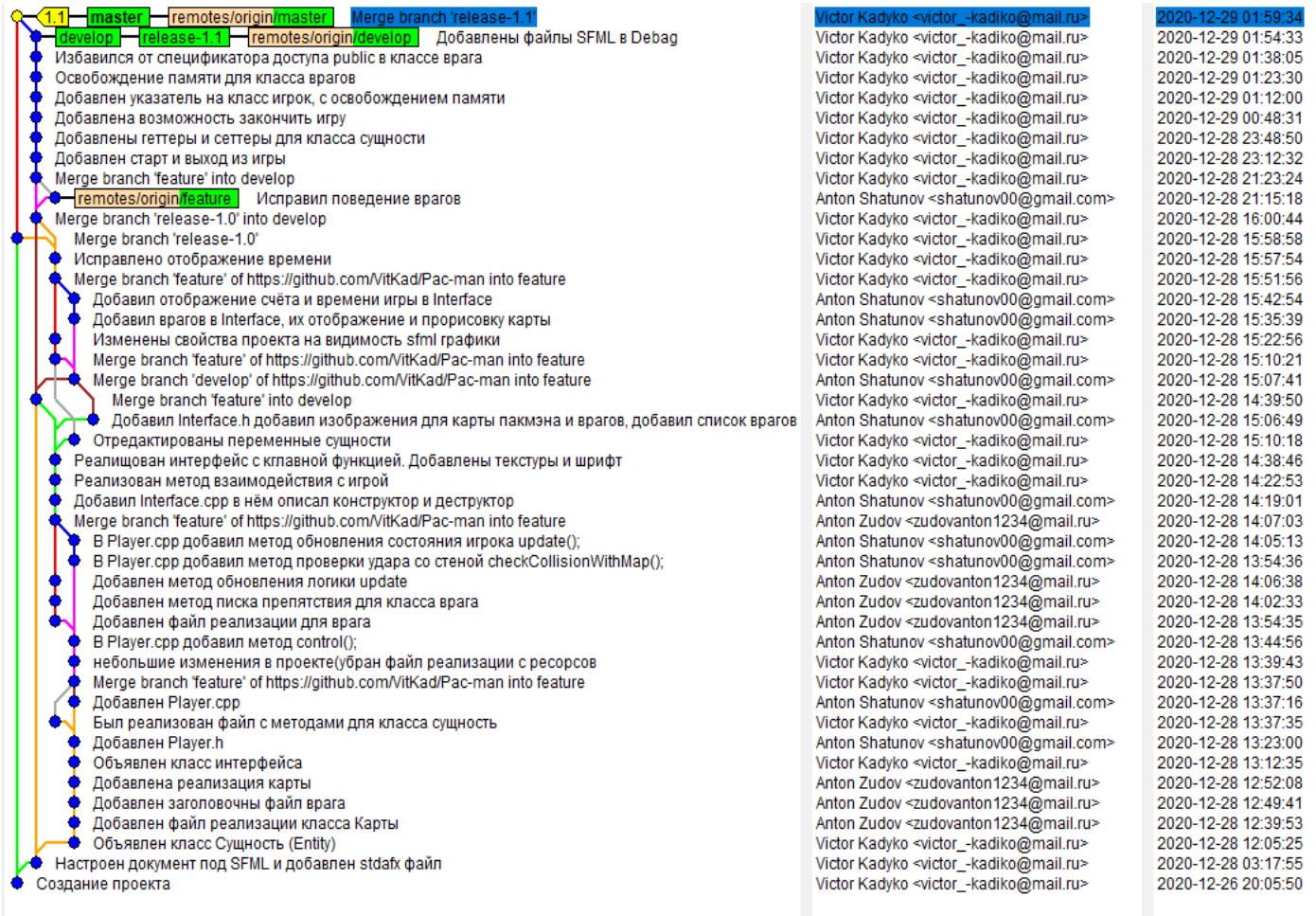
 Victor Kadyko committed 2 hours ago

Добавлена возможность закончить игру

 Victor Kadyko committed 2 hours ago

На последней стадии были внесены корректирующие исправления для программы. Например, теперь есть возможность завершить игру победой. Также проведена работа над выделением и освобождением памяти под классы игрока и врага. Еще в файле gitignore было убрано исключение папки Debug из списка. Данная версия послужила релизом игры с версией 1.1.

## Графическое отображение:



## Заключение.

Нашей командой была разработана игра, которая соответствует требованиям заказчика. Выбор среды разработки и языка программирования остался прежним. Была использована система контроля версий Git. Она применялась для совместной разработки программы. Сбоев и зависаний не наблюдается. Был использован принцип отдельной компиляции. Все классы разделены на отдельные заголовочные файлы, имеющие свою реализацию в соответствующих .cpp файлах. В программе реализована очистка динамической памяти. Неиспользованных переменных и избыточных алгоритмов не наблюдается. В отчете приведены диаграмма вариантов использования и диаграмма классов. Цель, поставленная заказчиком, выполнена.