Министерство науки и высшего образования РФ

ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)

Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий

Кафедра прикладной математики и программирования

«Графическая оконная библиотека»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ   
по дисциплине «Объектно-ориентированное программирование»

ЮУрГУ–01.03.02.2021.101.ПЗ КР

|  |  |  |
| --- | --- | --- |
|  |  | *Руководитель,*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Демидов А.К.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.* |
|  |  | *Автор работы:*  *Студент группы: ЕТ – 212*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Никитин М.С.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.* |
|  |  | *Работа защищена с оценкой*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.* |

Челябинск 2023

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет   
(национальный исследовательский университет)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

Направление Прикладная математика и информатика

УТВЕРЖДАЮ

Заведующий кафедрой ПМиП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А.Замышляева

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

**на курсовую работу студента**

\_\_\_\_\_\_\_\_*Никитин М.С.*\_\_\_\_\_\_\_\_\_

Группа \_\_*ЕТ-212*\_\_\_\_

1. Дисциплина  *Объектно-ориентированное программирование*

2. Тема работы \_\_\_\_*Программа для обучения основам программирования* \_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Срок сдачи студентом законченной работы *28 января 2023 г.*

4. Перечень вопросов, подлежащих разработке

1. разработка иерархии и интерфейса классов;
2. реализация программы (библиотеки классов) на языке С++
3. оформление программной документации (описание программы (библиотеки классов), руководство пользователя, листинг кода) и отчета по курсовой работе
4. презентация проектных решений для защиты КР (иерархия и интерфейсы классов, особенности реализации)

5. Календарный план

|  |  |  |
| --- | --- | --- |
| **Наименование разделов**  **(этапов) курсовой работы** | **Срок выполнения**  **разделов (этапов) работы** | **Отметка**  **о выполнении**  **руководителя** |
| анализ предметной области | 01.09.2022-10.10.2022 |  |
| разработка иерархии и интерфейса классов | 20.09.2022-07.11.2022 |  |
| реализация основных классов, функций | 01.10.2022-20.11.2022 |  |
| тестирование программы и/или классов, улучшение и исправление ошибок | 20.10.2022-10.12.2022 |  |
| оформление программной документации и отчета по курсовой работе | 30.10.2022-20.12.2022 |  |
| защита курсовой работы | 20.12.2022-28.12.22 |  |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка)

АННОТАЦИЯ

Никитин М.С. Программа для обучения основам программирования. – Челябинск: ЮУрГУ, ЕТ-212, 2023. – 31с., 3 ил., библиографический список – 3 наим., 1 прил.

В курсовой работе описывается разработка программы для обучения основам программирования с помощью объектно-ориентированного подхода. Работа содержит результаты объектно-ориентированного анализа и проектирования, инструкции по установке и использованию библиотеки.

В результате работы была разработана графическая оконная библиотека, код которой приводится в приложении.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc104386000)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc104386001)

[2 ОПИСАНИЕ ПРОГРАММЫ 6](#_Toc104386002)

[3 ИНСТРУКЦИЯ ПО УСТАНОВКЕ И ТРЕБОВАНИЯ К СИСТЕМЕ 9](#_Toc104386003)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 10](#_Toc104386004)

[ЗАКЛЮЧЕНИЕ 13](#_Toc104386005)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 14](#_Toc104386006)

[ПРИЛОЖЕНИЕ А 15](#_Toc104386007)

# ВВЕДЕНИЕ

**Актуальность темы**. Объектно-ориентированный подход является наиболее прогрессивной технологией разработки программных систем, позволяет разрабатывать более сложные системы.

**Цель работы** – разработать программу для обучения основам программирования.

**Задачи работы**:

– изучить приемы объектно-ориентированного анализа;

– научиться разрабатывать программы в объектно-ориентированном стиле;

– овладеть технологиями объектно-ориентированного анализа и проектирования;

– изучить особенности объектной модели языка программирования C++;

– научиться самостоятельно и творчески использовать знания и полученные практические навыки;

– овладеть навыками самостоятельного получения новых знаний по теории и практике объектного подхода в программировании.

**Объект работы** – программа для обучения основам программирования.

**Предмет работы** – применение объектно-ориентированного подхода для разработки библиотеки.

**Результаты работы** можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика».

# 1 ПОСТАНОВКА ЗАДАЧИ

Необходимо написать библиотеку для оконного интерфейса в графическом режиме. Библиотека должна работать для произвольного графического режима, обеспечиваемого выбранной графической библиотекой. Анализ предметной области помогает выявить следующие необходимые объекты:

* кнопки управления, обладающие следующими характеристиками: координаты, размеры, видимость, текст, цвет самой кнопки и цвет текста кнопки, процедуру, вызываемую при нажатии;
* поле ввода, обладающее следующими характеристиками: координаты, ширина, видимость, текст, цвет поля и цвет вводимого текста;
* метка, которая имеет собственные координаты, видимость, цвет и текст;
* окно, которое обладает собственными координатами, размером, видимостью, цветом фона, текста и заголовка и процедурой, вызываемой для нераспознанных клавиш окна, а также окно может находиться поверх других окон и позволяет добавить другой элемент.

Таким образом, над объектами можно совершать следующие действия:

* создание и удаление объекта;
* изменение цвета и текста объекта;
* изменение размеров и координат объекта.

**2 ОПИСАНИЕ ПРОГРАММЫ**

2.1 Для разработки программы были использованы:

– компилятор MinGW C/C++ 11.3;

– библиотека winBGIm.

2.2 Программа состоит из 2 модулей:

1 Модуль **main** (файл main.cpp) содержит основную функцию:

int main();

2 Модуль **Lib** (интерфейсная часть в файле Lib.h, реализация в файле Lib.cpp) содержит следующие классы:

#ifndef LIB\_HPP

#define LIB\_HPP

#include <functional>

#include <vector>

#include "graphics.h"

using namespace std;

/// Абстрактный класс от которого наследуются все классы

class Object

{

protected:

int x, y, color, backgroundColor, width, height; // координаты, цвет, цвет фона, ширина и высота

string text; // текст

bool activity = false; // активность элемента (есть ли на нем фокус)

bool visiblity = true; // видимость элемента

virtual void draw() = 0; // отрисовка объекта

public:

Object(int x, int y, int c, int bc, int w, int h, string t); // конструктор объекта

void setVisibility(bool); // установка видимости объекта

bool isVisible() const; // получение видимости объекта

void setActivity(bool state); // установить активность объекта

bool isActive() const;

void setText(string); // установление текста объекта

string getText() const; // получение текста объекта

void setColor(int); // установление цвета объекта

int getColor() const; // получение цвета объекта

void setBackgroundColor(int); // установление цвета фона объекта

int getBackgroundColor() const; // получение цвета фона объекта

void setWidth(int); // установление ширины объекта

int getWidth() const; // получение ширины объекта

void setHeight(int); // установление высоты объекта

int getHeight() const; // получить высоту объекта

void setX(int); // установление координаты x объекта

int getX() const; // получить координату x объекта

void setY(int); // установление координаты y объекта

int getY() const; // получить координату y объекта

virtual bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры

virtual void handleMouse(int x, int y); // определение нажатия мыши по объекту

void show(); // отобразить объект

};

/// Класс окна

class Window: public Object

{

private:

std::function<void(int)> func; // функция которая вызывается при ошибке

vector <Object \*> controls; // массив объектов окна

int activeElementIndex = 0; // номер активного элемента в окне

protected:

void draw(); // отрисовка окна

public:

Window(int x, int y, int c, int bc, int w, int h, string t); // конструктор окна

void addControl(Object \*control); // добавить элемент к окну (кнопку, лейбл или текстовое поле)

void removeControl();// удалить элемент

void setAction(function<void(int)> f); // установка функции на кнопку

void handleMouse(int x, int y); // обработка нажатия мыши

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры (tab - перемещение между объeктами окна)

};

/// Класс рабочего стола

class Desktop: public Object

{

private:

vector<Window \*> windows; // массив окон

int activeElementIndex = 0; // номер активного окна на рабочем столе

Desktop(int width, int height); // Конструктор рабочего стола

protected:

void draw(); // отрисовка рабочего стола

public:

// Так как класс одиночка, его запрещено копировать

Desktop(const Desktop& rhs) = delete;

Desktop& operator=(const Desktop& rhs) = delete;

void addWindow(Window \*window); // добавление окна к рабочему столу

void removeWindow();// удаление окна у рабочего стола

void handleEvents(); // цикл обработки событий

void handleMouse(int x, int y); // определение нажатий мышью

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры (shift+tab - переключение окон, esc - закрытие программы)

static Desktop\* instance();

};

/// Класс кнопки

class Button: public Object

{

private:

function<void()> func; // функция, задаваемая пользователем, которая вызывается при нажатии на кнопку

protected:

void draw(); // отрисовка кнопки

public:

Button(int x, int y, int c, int bc, int w, int h, string t); // конструктор кнопки

void setAction(function<void()> f); // установка функции на кнопку

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры

void handleMouse(int x, int y); // обработка нажатия клавиши мыши по кнопке

};

/// Класс для текстового поля

class Text: public Object

{

protected:

void draw(); // отрисовка текстового поля

public:

Text(int x, int y, int c, int bc, int w, int h, string t); // конструктор текстового поля

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры (backspace - стереть символ)

};

/// Класс для надписи

class Label: public Object

{

protected:

void draw(); // отрисовка лейбла

public:

Label(int x, int y, int c, string t); // конструктор лейбла

};

#endif

2.3 Иерархия классов показана рисунке 1.

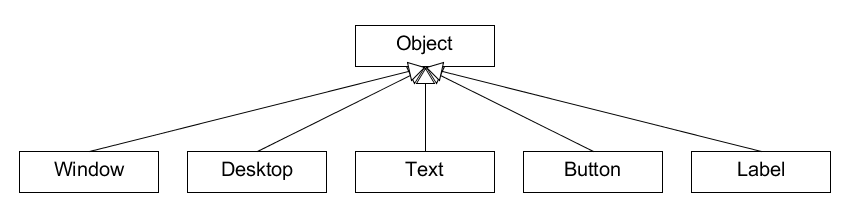


Рисунок 1 – Иерархия классов

**3** **ИНСТРУКЦИЯ ПО УСТАНОВКЕ И ТРЕБОВАНИЯ К СИСТЕМЕ**

3.1 Требования к компьютеру:

Процессор: Intel Pentium 4 1ГГц или выше

Память: 512+ MБ

Видеокарта: разрешение 1024x768 или выше, 24-битный цвет

Свободное место на диске: 10 Мб

Операционная система: Windows XP или выше

Дополнительное устройство: клавиатура, мышь

3.2 Для сборки библиотеки необходим компилятор, поддерживающий  
С++14. Для работы программы нужно добавить файлы «Lib.h» и «Lib.cpp» в компилятор и в файлах, в которых используется данная библиотека, подключить #include “Lib.h”. Для работы кнопок потребуется также подключение заголовочного файла #include <functional> в “Lib.h”.

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Для работы с библиотекой необходимо создать рабочий стол:

Desktop \*D = new Desktop(0, 0, 0, 0, 1400, 800, " ");

После чего создаем окна, используя конструктора, которому передаются его координаты, цвет, ширину и высоту окна, название окна.

Пример:

Window \*window1 = new Window(300, 300, 0, COLOR(255, 218, 185), 300, 300, "Confirm operation");

Для создания кнопок используем конструктор, которому передаются координаты, цвет текста, цвет кнопки, ширину и высоту, а также текст.

Пример:

Button \*button1 = new Button(150, 100, WHITE, COLOR(255, 160, 122), 200, 75, "ОK");

Поле ввода создается с помощью конструктора, куда передаются координаты, цвет строки и фона, ширина, высота и начальное текстовое значение.

Пример:

Text \*text1 = new Text(100, 200, MAGENTA, WHITE, 150, 30, "");

Метка создаётся с помощью конструктора, куда передаются её координаты, цвет и надпись.

Пример:

Label \*label1 = new Label(100, 150, LIGHTRED, "Enter new title");

С помощью метода addControl происходит добавление объектов на окна.

Пример:

window1->addControl(button1);

Метод removeControl удаляет последний добавленный объект окна.

Пример:

window2-> removeControl (button2);

Аналогично происходит добавление окон на рабочий стол.

Пример:

desktop->addWindow(window1);

И также удаление окна с рабочего стола.

Пример:

desktop-> removeWindow(window1);

Для обработки действий используется метод handleEvents ().

Пример:

desktop->handleEvents();

Для обработки нажатий клавиш клавиатуры используется метод handleKeyboard().

Пример:

windows[activeElementIndex]->handleKeyboard(keyCode); Для обработки нажатий мыши используется метод handleMouse(int x, int y);

Пример:

control->handleMouse(x, y);

Для того, чтобы установить видимость объекта необходимо вызвать метод setVisibility ()

Пример:

l1-> setVisibility (1) ;

Чтобы получить видимость объекта необходимо использовать метод isVisible().

Пример:

bool vis\_w = Window1-> isVisible ();

Для того, чтобы установить текст объекта (название окна или кнопки, например) используется метод setText().

Пример:

window2->setText(text);

Чтобы получить текст объекта необходимо использовать метод getText ().

Пример:

string text = text1->getText();

Для того, чтобы установить цвет объекта используется метод setColor ().

Пример:

button1-> setColor(COLOR(255, 168, 18));

Чтобы получить цвет объекта необходимо использовать метод getColor ().

Пример:

int color\_b = button1-> getColor();

Для того, чтобы установить цвет фона объекта используется метод setBackgroundColor ().

Пример:

window1->setBackgroundColor(COLOR(0, 255, 0));

Чтобы получить цвет фона объекта необходимо использовать метод getBackgroundColor ().

Пример:

int color\_w window1 -> getBackgroundColor();

Для того, чтобы установить высоту и ширину объекта используются методы setHeight () и setWidth () соответственно.

Пример:

window3->setWidth(w\*(i/100.0));

window3->setHeight(h\*(i/100.0));

Чтобы получить высоту и ширину объекта необходимо использовать методы getHeight () и getWidth () соответственно.

Пример:

int w = window3->getWidth();

int h = window3->getHeight();

Для того, чтобы установить координаты объекта x и y используются методы setX () и setY () соответственно.

Пример:

button1-> SetX(50);

button1-> SetY(50);

Чтобы получить координаты объекта x и y необходимо использовать методы getX () и getY () соответственно.

Пример:

int windowX = window->getX();

int windowY = window->getY();

Для определения функции, которую будет выполнять кнопка, необходимо прописать написать лямбда-выражение в методе setAction ().

Пример:

button1->setAction([&]()

{

window1->setBackgroundColor(COLOR(0, 255, 0));

});

При нажатии на кнопки произойдет следующие изменения. Кнопки button1 button2 меняют цвет фона окна window1 на зеленый и красный. Кнопка button3 изменяет название window2. А кнопка button4 изменяет размеры окна Window3. (см. рис. 4)

Управление с клавиатуры осуществляется следующим образом. При нажатии клавиши Esc происходит закрытие программы. При нажатии клавиши Tab происходит переход между элементами активного окна. Если активным элементом окна является кнопка, то при нажатии клавиши Enter выполняется функция, заданная в этой кнопке. При нажатии сочетания клавиш Shift и Tab происходит переход между окнами.

Управление с помощью мыши. При нажатии курсором на кнопку выполняется функция, заданная в этой кнопке. Текстовое поле становится активным при нажатии на него. Если нажать курсором на окно, то оно станет активным.

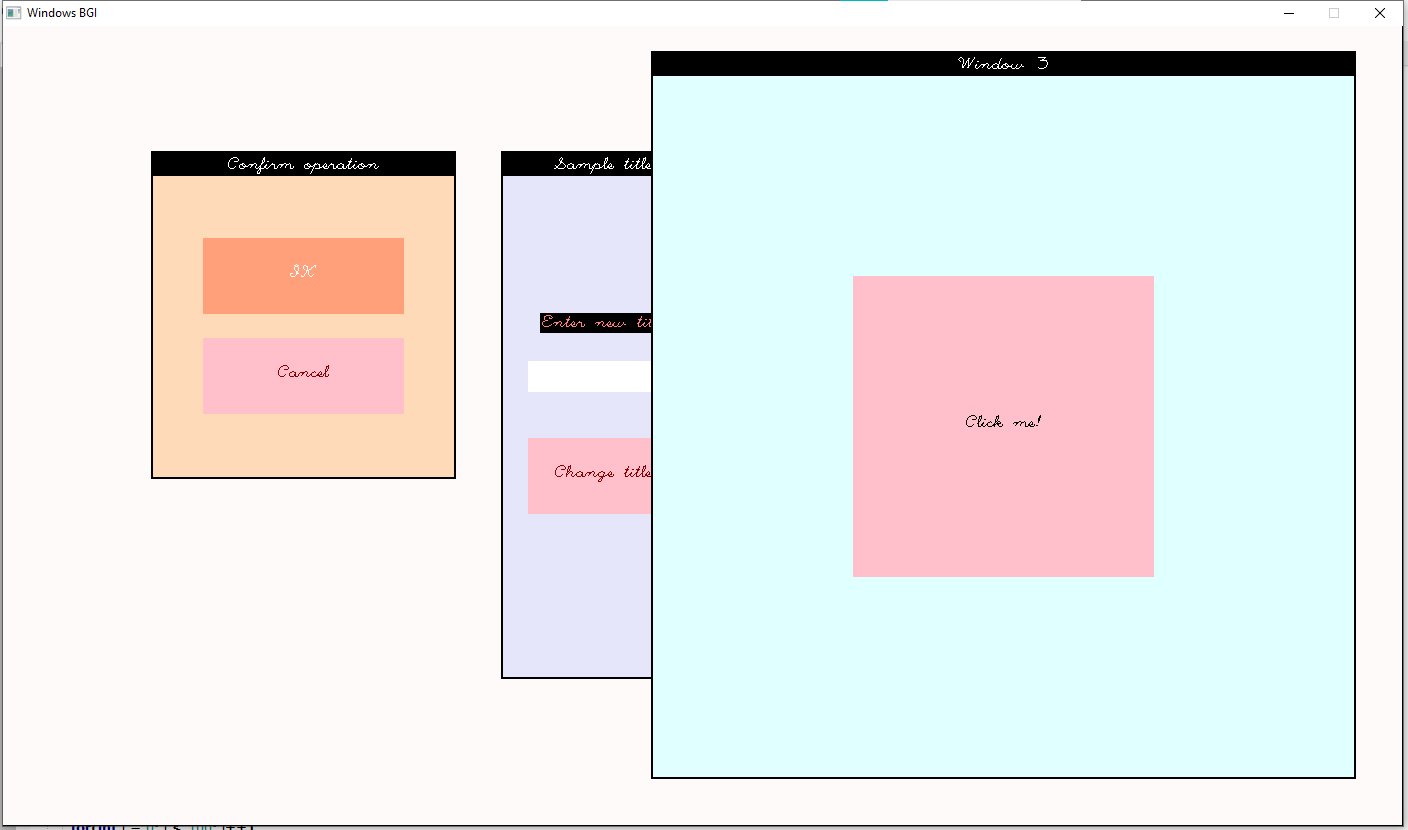


Рисунок 2 – Создание рабочего стола и его объектов

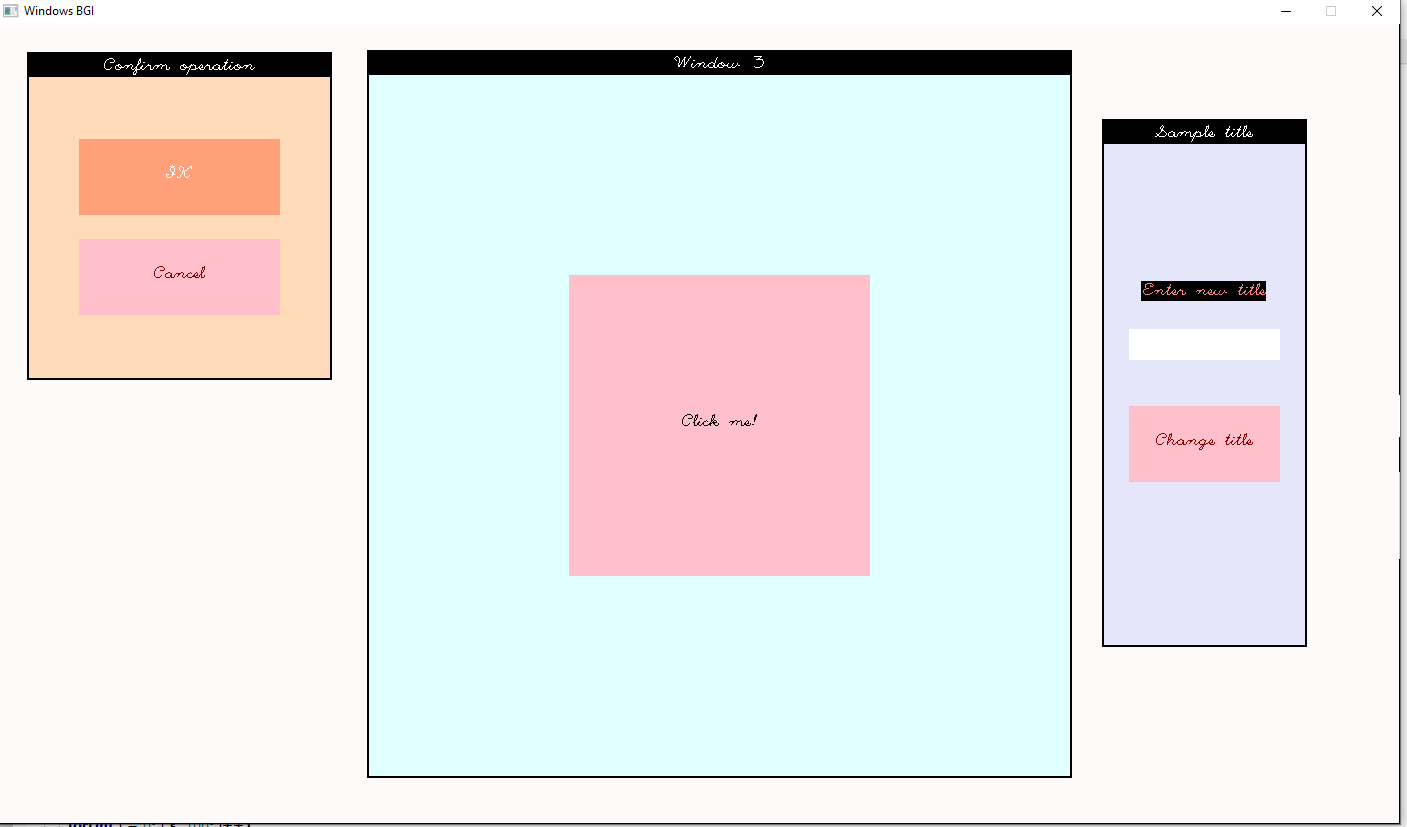


Рисунок 3 – Передвижение окон с помощью мыши

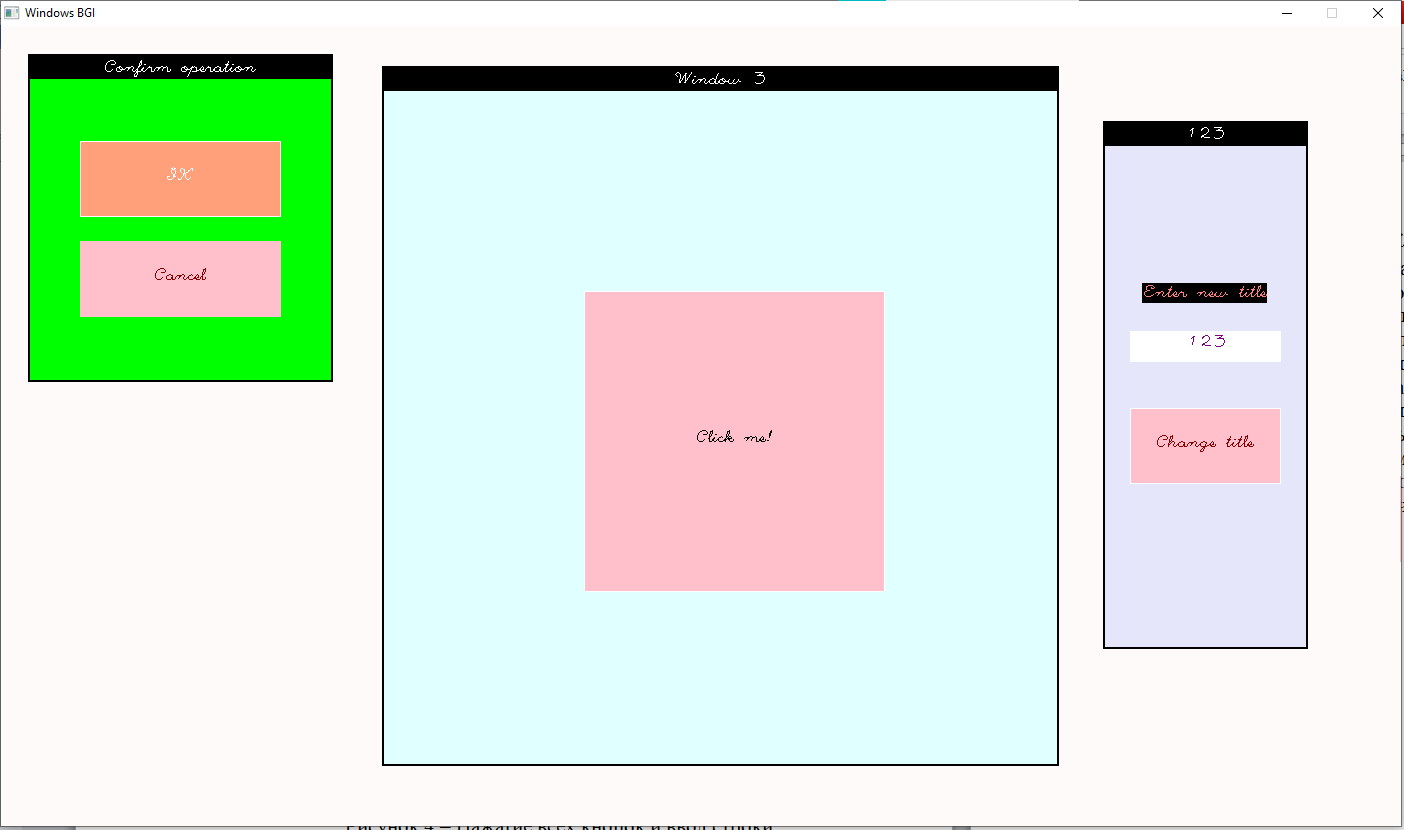


Рисунок 4 – Нажатие всех кнопок и ввод строки

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были выявлены объекты предметной области и определена система классов для них, разработан интерфейс программы. После объектно-ориентированного проектирования классы были реализованы на языке С++. Разработанный код был проверен на контрольных тестах и в код были внесены необходимые исправления. Для программы была разработана документация, описывающая её установку и использование. Таким образом, цель работы была достигнута, задачи – решены.

Результаты работы можно использовать в процессе последующего обучения в форме навыков практического применения объектно-ориентированного подхода для разработки сложных программных систем, понимания порядка этапов разработки программного обеспечения и достигаемых на каждом этапе результатов.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Графическая библиотека WinBGIm. – URL: https://ipc.susu.ru/20786.html (дата обращения: 20.05.2022).

2 Демидов, А. К. Объектно-ориентированное программирование на C++.[Текст]:учеб. пособие. – URL: [ftd (susu.ru)](https://lib.susu.ru/ftd?base=SUSU_METHOD&key=000557669&dtype=F&etype=.pdf) (дата обращения: 20.05.2022).

3 Липман, С. Язык программирования С++. Полное руководство. [Электронный ресурс] / С. Липман, Ж. Лажойе. — Электрон. дан. — М. : ДМК Пресс, 2006. — 1105 с. – URL: <http://e.lanbook.com/book/1216> (дата обращения: 20.05.2022).

# ПРИЛОЖЕНИЕ А

А.1 Файл main.cpp

#include <iostream>

#include "Lib.h"

using namespace std;

int main()

{

Desktop \*desktop = Desktop::instance();

Window \*window1 = new Window(300, 300, 0, COLOR(255, 218, 185), 300, 300, "Confirm operation");

Button \*button1 = new Button(150, 100, WHITE, COLOR(255, 160, 122), 200, 75, "ОK");

Button \*button2 = new Button(150, 200, COLOR(139, 0, 0), COLOR(255, 192, 203), 200, 75, "Cancel");

button1->setAction([&]()

{

window1->setBackgroundColor(COLOR(0, 255, 0));

});

button2->setAction([&]()

{

window1->setBackgroundColor(COLOR(255, 0, 0));

});

window1->addControl(button1);

window1->addControl(button2);

desktop->addWindow(window1);

Window \*window2 = new Window(600, 400, 0, COLOR(230, 230, 250), 200, 500, "Sample title");

Label \*label1 = new Label(100, 150, LIGHTRED, "Enter new title");

Text \*text1 = new Text(100, 200, MAGENTA, WHITE, 150, 30, "");

Button \*button3 = new Button(100, 300, COLOR(139, 0, 0), COLOR(255, 192, 203), 150, 75, "Change title");

button3->setAction([&]()

{

string text = text1->getText();

window2->setText(text);

});

window2->addControl(label1);

window2->addControl(text1);

window2->addControl(button3);

desktop->addWindow(window2);

Window \*window3 = new Window(1000, 400, 0, COLOR(224, 255, 255), 700, 700, "Window 3");

Button \*button4 = new Button(350, 350, COLOR(0,0,0), COLOR(255, 192, 203), 300, 300, "Click me!");

button4->setAction([&]()

{

int w = window3->getWidth();

int h = window3->getHeight();

for(int i = 0; i < 100; i++)

{

window3->setWidth(w\*(i/100.0));

window3->setHeight(h\*(i/100.0));

}

});

window3->addControl(button4);

desktop->addWindow(window3);

desktop->handleEvents();

return 0;

}

А.2 Файл Lib.h

#ifndef LIB\_HPP

#define LIB\_HPP

#include <functional>

#include <vector>

#include "graphics.h"

using namespace std;

/// Абстрактный класс от которого наследуются все классы

class Object

{

protected:

int x, y, color, backgroundColor, width, height; // координаты, цвет, цвет фона, ширина и высота

string text; // текст

bool activity = false; // активность элемента (есть ли на нем фокус)

bool visiblity = true; // видимость элемента

virtual void draw() = 0; // отрисовка объекта

public:

Object(int x, int y, int c, int bc, int w, int h, string t); // конструктор объекта

void setVisibility(bool); // установка видимости объекта

bool isVisible() const; // получение видимости объекта

void setActivity(bool state); // установить активность объекта

bool isActive() const;

void setText(string); // установление текста объекта

string getText() const; // получение текста объекта

void setColor(int); // установление цвета объекта

int getColor() const; // получение цвета объекта

void setBackgroundColor(int); // установление цвета фона объекта

int getBackgroundColor() const; // получение цвета фона объекта

void setWidth(int); // установление ширины объекта

int getWidth() const; // получение ширины объекта

void setHeight(int); // установление высоты объекта

int getHeight() const; // получить высоту объекта

void setX(int); // установление координаты x объекта

int getX() const; // получить координату x объекта

void setY(int); // установление координаты y объекта

int getY() const; // получить координату y объекта

virtual bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры

virtual void handleMouse(int x, int y); // определение нажатия мыши по объекту

void show(); // отобразить объект

};

/// Класс окна

class Window: public Object

{

private:

std::function<void(int)> func; // функция которая вызывается при ошибке

vector <Object \*> controls; // массив объектов окна

int activeElementIndex = 0; // номер активного элемента в окне

protected:

void draw(); // отрисовка окна

public:

Window(int x, int y, int c, int bc, int w, int h, string t); // конструктор окна

void addControl(Object \*control); // добавить элемент к окну (кнопку, лейбл или текстовое поле)

void removeControl();// удалить элемент

void setAction(function<void(int)> f); // установка функции на кнопку

void handleMouse(int x, int y); // обработка нажатия мыши

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры (tab - перемещение между объeктами окна)

};

/// Класс рабочего стола

class Desktop: public Object

{

private:

vector<Window \*> windows; // массив окон

int activeElementIndex = 0; // номер активного окна на рабочем столе

Desktop(int width, int height); // Конструктор рабочего стола

protected:

void draw(); // отрисовка рабочего стола

public:

// Так как класс одиночка, его запрещено копировать

Desktop(const Desktop& rhs) = delete;

Desktop& operator=(const Desktop& rhs) = delete;

void addWindow(Window \*window); // добавление окна к рабочему столу

void removeWindow();// удаление окна у рабочего стола

void handleEvents(); // цикл обработки событий

void handleMouse(int x, int y); // определение нажатий мышью

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры (shift+tab - переключение окон, esc - закрытие программы)

static Desktop\* instance();

};

/// Класс кнопки

class Button: public Object

{

private:

function<void()> func; // функция, задаваемая пользователем, которая вызывается при нажатии на кнопку

protected:

void draw(); // отрисовка кнопки

public:

Button(int x, int y, int c, int bc, int w, int h, string t); // конструктор кнопки

void setAction(function<void()> f); // установка функции на кнопку

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры

void handleMouse(int x, int y); // обработка нажатия клавиши мыши по кнопке

};

/// Класс для текстового поля

class Text: public Object

{

protected:

void draw(); // отрисовка текстового поля

public:

Text(int x, int y, int c, int bc, int w, int h, string t); // конструктор текстового поля

bool handleKeyboard(int); // обработка нажатия клавиши клавиатуры (backspace - стереть символ)

};

/// Класс для надписи

class Label: public Object

{

protected:

void draw(); // отрисовка лейбла

public:

Label(int x, int y, int c, string t); // конструктор лейбла

};

#endif

А.3 Файл Lib.cpp

#include <iostream>

#include "Lib.h"

#define NO\_MOUSE\_BUTTONS 0 // код если мышь не нажата

#define LEFT\_MOUSE\_BUTTON 1 // код нажатия левой кнопки мыши

#define RIGHT\_MOUSE\_BUTTON 2 // код, что нажата правая кнопка мыши

#define WIN\_TITLE\_BACKGROUND\_COLOR BLACK // Фон заголовка у окна

#define WIN\_TITLE\_Y\_OFFSET 8 // смещение текста по вертикали в заголовке окна

#define WIN\_BORDER\_Y\_THICKNESS 25 // высота границы у окна (высота заголовка)

#define WIN\_BORDER\_X\_THICKNESS 2 // ширина границы окна (полосок вокруг)

#define SHIFT\_KEY\_CODE 16 // Код шифта

#define DESKTOP\_WIDTH 1400 // ширина рабочего стола

#define DESKTOP\_HEIGHT 800 // высота рабочего стола

using namespace std;

/////////////////////////////////////////////////////////////

// DESKTOP /////////////////////////////////////////////////////////////

// Получить объект рабочего стола (единственный в программе)

Desktop \*Desktop::instance()

{

static Desktop desktop(DESKTOP\_WIDTH, DESKTOP\_HEIGHT);

return &desktop;

}

// конструктор

Desktop::Desktop(int width, int height) : Object(0, 0, 0, 0, width, height, " ")

{

initwindow(width, height);

settextstyle(SCRIPT\_FONT, HORIZ\_DIR, 3);

}

// Обработка действий

void Desktop::handleEvents()

{

Desktop::instance()->draw();

while (true)

{

int keyCode;

if (kbhit())

{

keyCode = getch();

int result = handleKeyboard(keyCode);

if (!result) break;

}

else if (mousebuttons() == LEFT\_MOUSE\_BUTTON)

{

handleMouse(mousex(), mousey());

while (mousebuttons() != 0);

}

}

}

// Отрисовка рабочего стола

void Desktop::draw()

{

setbkcolor(COLOR(255, 250, 250));

clearviewport();

for (auto window : windows)

window->show();

swapbuffers();

}

//добавление окна

void Desktop::addWindow(Window \*window)

{

windows.push\_back(window);

activeElementIndex = windows.size() - 1;

Desktop::instance()->show();

}

//удаление окна

void Desktop::removeWindow()

{

windows.pop\_back();

activeElementIndex = windows.size() - 1;

Desktop::instance()->show();

}

//обработка нажатий мыши, если курсор попадает на окно, то это окно становится активным

void Desktop::handleMouse(int x, int y)

{

for (int i = windows.size() - 1; i >= 0; i--)

{

Window \*window = this->windows[i];

bool isCursorIn = (window->getX() - window->getWidth()/2.0) < x && x < (window->getX() + window->getWidth()/2.0) &&(window->getY() - window->getHeight()/2.0 - WIN\_BORDER\_Y\_THICKNESS) < y && y < (window->getY() + window->getHeight()/2.0);

if (window->isVisible() && isCursorIn)

{

bool isCursorInTitle = (window->getX() - window->getWidth()/2.0) < x && x < (window->getX() + window->getWidth()/2.0) && (window->getY() - window->getHeight()/2.0 - WIN\_BORDER\_Y\_THICKNESS) < y && y < (window->getY() - window->getHeight()/2.0);

if (isCursorInTitle)

{

int windowX = window->getX();

int windowY = window->getY();

int oldX = x;

int oldY = y;

while (mousebuttons() != NO\_MOUSE\_BUTTONS)

{

int newX = mousex();

int newY = mousey();

window->setX(windowX + newX-oldX);

window->setY(windowY + newY-oldY);

}

}

else

{

windows.erase(windows.begin() + i);

windows.push\_back(window);

Desktop::instance()->show();

window->handleMouse(x - window->getX() + window->getWidth()/2.0, y - window->getY() + window->getHeight()/2.0);

break;

}

}

}

}

// Обработка нажатий клавиши клавиатуры

bool Desktop::handleKeyboard(int keyCode)

{

if (keyCode == KEY\_TAB && controlkeystate() == SHIFT\_KEY\_CODE)//если нажаты клавиши shift и tab, то происходит смена окон

{

windows[activeElementIndex]->setActivity(false);

Window \*window = windows[0];

windows.erase(windows.begin());

windows.push\_back(window);

activeElementIndex = windows.size() - 1;

Desktop::instance()->show();

}

if (keyCode == KEY\_ESC) //если нажата esc, то закрытие программы

{

closegraph();

return false;

}

else windows[activeElementIndex]->handleKeyboard(keyCode);

return true;

}

////////////////////////////////////////////////////////////////////

// OBJECT

////////////////////////////////////////////////////////////////////

//конструктор

Object::Object(int x, int y, int c, int bgc, int w, int h, string t):

x(x),

y(y),

color(c),

backgroundColor(bgc),

width(w),

height(h),

text(t)

{

}

void Object::handleMouse(int, int) { }; // была ли активирована клавиша мыши

bool Object::handleKeyboard(int) { return true; } // была ли активирована клавиша клавиатуры

int Object::getY() const { return y; } // получение координаты y объекта

int Object::getX() const { return x; } // получение координаты x объекта

int Object::getHeight() const { return height; } // получить высоту

int Object::getWidth() const { return width; } // получение ширины объекта

int Object::getBackgroundColor() const { return backgroundColor; }// получение цвета фона объекта

int Object::getColor() const { return color; } // получение цвета

string Object::getText() const { return text; } // получение текста

bool Object::isVisible() const { return visiblity; } // виден ли объект?

void Object::setActivity(bool state) { activity = state; } // установить активность объекта

bool Object::isActive() const { return activity; } // активен ли объект

void Object::show() { if (isVisible()) draw(); }; // отобразить объект

void Object::setText(string text)// установление текста

{

this->text = text;

Desktop::instance()->show();

}

void Object::setColor(int color) //установление цвета

{

this->color = color;

Desktop::instance()->show();

}

void Object::setBackgroundColor(int backgroundColor)//установление фона

{

this->backgroundColor = backgroundColor;

Desktop::instance()->show();

}

void Object::setWidth(int width)//установление ширины

{

this->width = width;

Desktop::instance()->show();

}

void Object::setHeight(int height) //установление высоты

{

this->height = height;

Desktop::instance()->show();

}

void Object::setX(int x)//установление координаты x

{

this->x = x;

Desktop::instance()->show();

}

void Object::setY(int y)// установление координаты y

{

this->y = y;

Desktop::instance()->show();

}

void Object::setVisibility(bool state)// установление видимости

{

this->visiblity = state;

Desktop::instance()->show();

}

////////////////////////////////////////////////////////////////////

// TEXT

////////////////////////////////////////////////////////////////////

// конструктор

Text::Text(int x, int y, int color, int backgroundColor, int width, int height, string text) : Object(x, y, color, backgroundColor, width, height, text){};

// Отрисовка текста

void Text::draw()

{

settextjustify(CENTER\_TEXT, CENTER\_TEXT);

setfillstyle(SOLID\_FILL, backgroundColor);

bar(x - width/2.0, y - height/2.0, x + width/2.0, y + height/2.0);

setcolor(this->color);

setbkcolor(backgroundColor);

outtextxy(x + 1, y, text.c\_str());

int textWidth = textwidth(text.c\_str());

while (textWidth > this->width - 2) //функция, которая проверяет и уменьшает

{

text.erase(text.end() - 1);

textWidth = textwidth(text.c\_str());

}

setcolor(WHITE);

if (isActive())

rectangle(x - width/2.0, y - height/2.0, x + width/2.0, y + height/2.0);

}

//если нажата клавиша Backspace, стираем

bool Text::handleKeyboard(int keyCode)

{

if (keyCode == KEY\_BACKSPACE && text.size() > 0)

text.pop\_back();

else if (keyCode >= ' ' && keyCode < 256)

{

int textWidth = textwidth(text.c\_str());

if (textwidth((text + (char)keyCode).c\_str()) < width - 2) text += (char)keyCode;

}

Desktop::instance()->show();

return true;

}

////////////////////////////////////////////////////////////////////

// WINDOW

////////////////////////////////////////////////////////////////////

// конструктор

Window::Window(int x, int y, int c, int bc, int w, int h, string t) : Object(x, y, c, bc, w, h, t)

{

}

// Установить функцию на окно, при нажатии некорректной клавиши

void Window::setAction(function<void(int)> f)

{

func = f;

}

// отрисовка окна

void Window::draw()

{

settextjustify(CENTER\_TEXT, CENTER\_TEXT);

int x1,y1,x2,y2;

setfillstyle(SOLID\_FILL, WIN\_TITLE\_BACKGROUND\_COLOR);

x1 = x - width/2.0 - WIN\_BORDER\_X\_THICKNESS;

y1 = y - height/2.0 - WIN\_BORDER\_Y\_THICKNESS;

x2 = x + width/2.0 + WIN\_BORDER\_X\_THICKNESS;

y2 = y + height/2.0 + WIN\_BORDER\_X\_THICKNESS;

bar(x1, y1, x2, y2);

setbkcolor(WIN\_TITLE\_BACKGROUND\_COLOR);

setcolor(WHITE);

settextstyle(SCRIPT\_FONT, HORIZ\_DIR, 1);

outtextxy(x, y-height/2.0 - WIN\_TITLE\_Y\_OFFSET, text.c\_str());

setfillstyle(SOLID\_FILL, this->backgroundColor);

x1 = x - width/2.0;

y1 = y - height/2.0;

x2 = x + width/2.0;

y2 = y + height/2.0;

bar(x1, y1, x2, y2);

//отрисовка объектов окна относительно этого окна

x1 = x - width/2.0;

y1 = y - height/2.0;

x2 = x + width/2.0 - WIN\_BORDER\_X\_THICKNESS;

y2 = y + height/2.0 + WIN\_BORDER\_Y\_THICKNESS;

setviewport(x1, y1, x2, y2, 1);

for (auto control : controls)

control->show();

setviewport(0, 0, getmaxx(), getmaxy(), 1);

}

// обработка клавиши клавиатуры у окна

bool Window::handleKeyboard(int keyCode)//если нажат tab, то переход к следующему элементу окна

{

if (keyCode == KEY\_TAB && controlkeystate() == 0)

{

controls[activeElementIndex]->setActivity(false);

activeElementIndex = (activeElementIndex + 1) % controls.size();

while (!controls[activeElementIndex]->isVisible() || dynamic\_cast<Label \*>(controls[activeElementIndex]) != nullptr)

{

activeElementIndex = (activeElementIndex + 1) % controls.size();

}

controls[activeElementIndex]->setActivity(true);

Desktop::instance()->show();

}

// если нажата неверная клавиша

else if (!controls[activeElementIndex]->handleKeyboard(keyCode))

{

if (func != nullptr)

func(keyCode);

Desktop::instance()->show();

}

return true;

}

//обработка нажатий мыши, если курсор попадает на элемент окна, то этот элемент становится активным

void Window::handleMouse(int x, int y)

{

for (int i = controls.size()-1; i >= 0; i--)

{

Object \*control = controls[i];

bool isControlVisible = control->isVisible();

bool isNotLabel = dynamic\_cast<Label \*>(control) == nullptr;

bool isCursorOnControl = (control->getX() - control->getWidth()/2.0) < x && x < (control->getX() + control->getWidth()/2.0) && (control->getY() - control->getHeight()/2.0) < y && y < (control->getY() + control->getHeight()/2.0);

if (isControlVisible && isNotLabel && isCursorOnControl)

{

controls[activeElementIndex]->setActivity(false);

activeElementIndex = i;

control->setActivity(true);

Desktop::instance()->show();

control->handleMouse(x, y);

break;

}

}

}

//добавление элемента в окно

void Window::addControl(Object \*control) { controls.push\_back(control); }

//удаление элемента с окна

void Window::removeControl() { controls.pop\_back(); }

////////////////////////////////////////////////////////////////////

// LABEL

////////////////////////////////////////////////////////////////////

// конструктор

Label::Label(int x, int y, int c, string t) : Object(x, y, c, 0, 0, 0, t)

{

};

// отрисовка метки

void Label::draw()

{

setcolor(color);

setbkcolor(backgroundColor);

outtextxy(x, y, text.c\_str());

}

////////////////////////////////////////////////////////////////////

// BUTTON

////////////////////////////////////////////////////////////////////

// конструктор

Button::Button(int x, int y, int c, int bc, int w, int h, string t) : Object(x, y, c, bc, w, h, t)

{

}; // конструктор

// установка функции кнопки

void Button::setAction(function<void()> f)

{

func = f;

}

//если курсор попадает на кнопку, то выполнить функцию

void Button::handleMouse(int x, int y)

{

if (func != nullptr) func();

}

// обработка нажатия клавиши клавиатуры у кнопки

bool Button::handleKeyboard(int keyCode)

{

if (keyCode == KEY\_ENTER)

{

if (func != nullptr)

{

func();

return true;

}

}

return true;

}

//отрисовка кнопки

void Button::draw()

{

settextjustify(CENTER\_TEXT, CENTER\_TEXT);

setfillstyle(SOLID\_FILL, backgroundColor);

bar(x - width/2.0, y - height/2.0, x + width/2.0, y + height/2.0);

setcolor(this->color);

setbkcolor(backgroundColor);

settextstyle(SCRIPT\_FONT, HORIZ\_DIR, 1);

int text\_width = textwidth(text.c\_str());

while (text\_width > width - 2) //функция, которая проверяет и уменьшает

{

text.erase(text.end() - 1);

text\_width = textwidth(text.c\_str());

}

outtextxy(x, y, text.c\_str());

setcolor(WHITE);

if (isActive())

rectangle(x - width/2.0, y - height/2.0, x + width/2.0, y + height/2.0);

}