МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

филиал «РКТ» МАИ в г. Химки Московской области

**Специальность 09.02.03 «Программирование в компьютерных системах»**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»**

Дата проведения с 16.02.2023 по 01.03.2023

**Студент**

**группы МП31-20 / Анциферов Н.А.**

**Руководитель**

**практики от филиала / Жилина Т.А .**

**2023г.**

**АТТЕСТАЦИОННЫЙ ЛИСТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент Анциферов Никита Александрович , обучающийся на 3курсе по специальности СПО 09.02.03 Программирование в компьютерных системах успешно прошел учебную практику в объеме 72 часов с «16»февраля2023г. по «01»марта 2023г.

В филиале «РКТ» МАИ в г. Химки Московской области

*Наименование предприятия/организации, юридический адрес*

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| Виды работ, выполненных обучающимся во время практики | Объём работ  (час.) | Качество выполнения работ в соответствии с технологией и (или) требованиями организации, в которой походила практика |
| Разработка кода программного модуля. Проведение тестирования и осуществление оптимизации программного кода. | 56 | *Соответствует* |
| Составление отчётной документации по практическим заданиям. Подготовка материала, иллюстрирующего ход выполнения проекта в виде презентации | 8 | *Соответствует* |
| Защита выполненных проектов | 8 | *Соответствует* |

Руководитель практики от филиала «РКТ» МАИ в г. Химки Московской области

Преподаватель Жилина Т.А

(должность, фамилия, имя, отчество)

|  |  |  |
| --- | --- | --- |
| Дата 01.03.2023 | *(подпись)* | Жилина Т.А.  *Расшифровка подписи* |

**ХАРАКТЕРИСТИКА**

Обучающийся Анциферов Никита Александрович\_ группы МП31-20по специальности СПО 09.02.03Программирование в компьютерных системахуспешно прошелучебную практикув объеме72  часов с «16»  февраля 2023г. по «01» марта 2023г.

в организации филиал «РКТ» МАИ в г. Химки Московской области

*Наименование предприятия /организации, юридический адрес*

За время работы проявилсебя какответственный/безответственный, коммуникабельный/замкнутый, исполнительный/неисполнительный, дисциплинированный/имеющий дисциплинарные замечания, доброжелательный/наглыйсотрудник.

Обучающийсяобладает общими компетенциями, включающими в себя способность:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 6. Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

В ходе выполнения всех видов работ обучающийся(аяся) показал(а) сформированность следующих профессиональных компетенций:

ПК 1.1. Выполнять разработку спецификаций отдельных компонент.

ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять оптимизацию программного кода модуля.

ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

**К работе относился**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**Цели и задачи практики** достигнуты/ достигнуты не в полном объеме.

**Оценка за практику**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Руководитель практики от организации «РКТ» МАИ в г. Химки Московской области

Преподаватель Жилина Т.А.

(должность, фамилия, имя, отчество)

|  |  |  |
| --- | --- | --- |
| Дата 01.03.2023 | *(подпись)* | Жилина Т.А.  *Расшифровка подписи* |

**Программа учебной практики**

По модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

По специальности СПО 09.02.03 Программирование в компьютерных системах.

Дата проведения с 16.02.2023 по 01.03.2023

|  |  |  |
| --- | --- | --- |
| №п/п | Наименование вида работ | Количество дней практики |
|  | Разработка кода программного модуля (задания 1-8). Проведение тестирования и осуществление оптимизации программного кода. | 10 |
|  | Составление отчётной документации по практическим заданиям. Подготовка материала, иллюстрирующего ход выполнения проекта в виде презентации | 1 |
|  | Защита выполненных проектов | 1 |

Руководитель практики от филиала «РКТ» МАИ в г. Химки Московской области

Преподаватель Жилина Т.А

(должность, фамилия, имя, отчество)

|  |  |  |
| --- | --- | --- |
| Дата 01.03.2023г. | *(подпись)* | Жилина Т.А  *Расшифровка подписи* |

**ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ**

По специальности СПО 09.02.03Программирование в компьютерных системах.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование вида работ | Количество дней практики | Прилагаемый материал в отчете |
|  | Выполнение задания 1. | 1 | Отчёт. 1 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Выполнение задания 2. | 1 | Отчёт. 2 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Выполнение задания 3. | 1 | Отчёт. 3 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Выполнение задания 4. | 1 | Отчёт. 4 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Выполнение задания 5. | 1 | Отчёт. 5 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Выполнение задания 6. | 1 | Отчёт. 6 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Выполнение задания 7. | 1 | Отчёт. 7 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Выполнение задания 8. | 1 | Отчёт. 8 Задание. Внешний накопитель |
|  | Проведение тестирования и оптимизации программного модуля. |
|  | Составление отчётной документации по практическим заданиям. Подготовка материала, иллюстрирующего ход выполнения проекта в виде презентации | 2 | Отчёт. Внешний накопитель |

**Параметры учета выполнения работ по заданию**

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Подпись руководителя предприятия/ОУ |
| 16.02.2023 | Выполнение задания 1. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 17.02.2023 | Выполнение задания 2. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 18.02.2023 | Выполнение задания 3. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 20.02.2023 | Выполнение задания 4. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 21.02.2023 | Выполнение задания 5. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 22.02.2023 | Выполнение задания 6. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 23.02.2023 | Выполнение задания 7. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 24.02.2023 | Выполнение задания 8. |  |
| Проведение тестирования и оптимизации программного модуля. |  |
| 25.02.2023 | Составление отчётной документации по практическим заданиям. Подготовка материала, иллюстрирующего ход выполнения проекта в виде презентации |  |

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 8](#_Toc136634753)

[**1 Задание. Диалоговые окна, реализация команд для вызова MSPaint и Calc, свойства CheckBox** 10](#_Toc136634754)

[**2 Задание. Рисование в окне курсором мыши, реализация смены цвета и толщины линии** 15](#_Toc136634755)

[**3 Задание. Отображение нажатой клавиши на клавиатуре в дочернем окне** 18](#_Toc136634756)

[**4 Задание. Отображение информации текстового поля из главного окна в дочернем** 21](#_Toc136634757)

[**5 Задание. Приложение – ответы на вопросы** 23](#_Toc136634758)

[**6 Задание. Непрерывное рисование окружности с изменением цвета** 28](#_Toc136634759)

[**7 Задание. Открытие проводника через меню и выбор картинки для фона окна** 32](#_Toc136634760)

[**8 Задание. Разработка модуля запуска семи прошлых модулей через меню** 35](#_Toc136634761)

[**ЗАКЛЮЧЕНИЕ** 41](#_Toc136634762)

### **ВВЕДЕНИЕ**

С момента появления первых образцов персональных компьютеров прошло не так уж и много времени, но сейчас без них уже немыслимо огромное количество областей человеческой деятельности — экономика, управление, наука, инженерное дело, издательское дело, образование, культура и так далее.

Интерес к персональным компьютерам постоянно растет, а круг их пользователей непрерывно расширяется.

Одновременно развиваются языки программирования. С 1985 года язык С был дополнен возможностями объектно-ориентированного программирования (ООП). Новая версия языка была названа С++. Использования ООП позволило не только определить типы данных пользователя, но и задать операции для этих типов.

На основе языка С++ разработаны визуальные системы С++ BUILDER, VISIAL C++. Использование этих систем позволило значительно упростить создания интерфейса, работу с базами данных и так далее. Несмотря на эти нововведения, по-прежнему программируют на WinAPI, который используется для решения системных задач. Только про его использование позволяет создавать программы, использующие всю мощь Windows, которая является сейчас самой популярной операционной системой.

Компьютерная графика является одним из передовых направлений в области технологий программного обеспечения. Сегодня нет человека имеющего отношения к компьютеру, который бы не знал, что это такое. Интерфейс программирования (API) для создания графики предоставляется самой операционной системой, точнее ее компонентом называемым “framebuffer”. В операционной системе Windows framebuffer называется GDI, его главная библиотека, предоставляющая программисту функции для программирования, называется gdi32.dll и находится в системном каталоге ОС. GDI предоставляет полный контроль над прорисовкой экранного окна, что предоставляет поистине безграничные возможности программирования внешнего вида программ.

Данная работа посвящена исследованиям, лежащим в области программирования, и касается программирования в компьютерный средах модулей в Windows API.

Актуальность исследования заключается в том, что современное общество невозможно представить без программируемых компьютерных сред. Основной задачей программирования является создания более благоприятных сред для жизни и развития человечества в целом.

Производственная практика студента проводится с целью закрепления теоретических знаний, полученных в процессе обучения; приобретения практических навыков, компетенций и опыта деятельности по направлению подготовки; ознакомления на практике с вопросами профессиональной деятельности, направленными на формирование знаний, навыков и опыта профессиональной деятельности.

Цель курсовой работы заключается в глубоком изучении программируемых инструментов, для создания модулей охватывающие большую часть программируемых возможностей в Windows API.

Для осуществления обозначенной цели служат следующие задачи:

* Задание 1 - Диалоговые окна, реализация команд для вызова MSPaint и Calc, свойства CheckBox;
* Задание 2 - Рисование в окне курсором мыши, реализация смены цвета и толщины линии;
* Задание 3 - Создание диалогового окна, в котором при нажатии показывает нажатую клавишу;
* Задание 4 - Разработать модуль с меню и окном редактирования с вводом только чисел;
* Задание 5 - Разработать модуль ответов на вопросы;
* Задание 6 - Разработать модуль круг, который динамически изменяющего свой радиус;
* Задание 7 - Разработать модуль выбора изображений из файла и изменении их размера под размеры окна;
* Задание 8 - Разработать модуль запуска семи прошлых модулей через меню.

Объект исследования - программируемая среда Windows API.

Предмет исследования - программируемые инструменты, которые применяются для программного кода в Windows API.

### **1 Задание. Диалоговые окна, реализация команд для вызова MSPaint и Calc, свойства CheckBox**

Разработать код программного модуля создающего диалоговую панель в которой при вводе в строку редактирования Paint и Calculator запускаются MSPaint и WindowsCalculator. На данной панели должны присутствовать чек CheckBox: видимость и блокировка. В активном состоянии строка видна и доступна для редактирования, при снятии галочек строка исчезает или блокируется.

Создается окно, а также необходимые кнопки, textbox и checkbox. Переменной inputVal присваивается значение из textbox. При активации (по нажатию лкм) button(запустить) запускается алгоритм сравнения значения из textboxс нужными «calculator»и «paint». В случае, если textbox имеет значение «calc»- открывается калькулятор, если «paint» – открываетсяpaint. Если textBox содержит неизвестную команду – происходит исключение и открывается MessageBoxс информацией об ошибке.

CheckBox(Блокировка) – По умолчанию не активен(false). При нажатии запрещает изменять TextBox.

В приведенном ниже коде реализация button(открыть), checkbox(блокировка) и алгоритм сравнения значения из textbox.

#pragmaendregion

//реализациякнопкиbutton(открыть)

private: System::Void btn\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ inputVal = this->textBoxInput->Text;

if (inputVal == "paint" || inputVal == "Paint" || inputVal == "PAINT" || inputVal == "пеинт" || inputVal == "Пеинт") {

//открытие mspaint

WinExec("mspaint.exe", 1);

String^ inputVal = "";

}

else if (inputVal == "calc" || inputVal == "Calc" || inputVal == "calculator" || inputVal == "Calculator" || inputVal == "Калькулятор" || inputVal == "калькулятор") {

//открытие калькулятора

WinExec("calc.exe", 1);

String^ inputVal = "";

}

else {

String^ s = "Ошибка, попробуй 'calc' или 'paint'";

MessageBox::Show(s);

String^ inputVal = "";

this->textBoxInput->Text = "";

}

}

//метод передающий начальные значения при запуске окна

private: System::Void MainCPP\_Load(System::Object^ sender, System::EventArgs^ e) {

String^ inputVal = "";

}

//метод checkbox(Блокировка)

private: System::Void chBox\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (chBox->Checked) {

//блокировка редактирования textbox

this->textBoxInput->ReadOnly = true;

}

else {

//отключение блокировки

this->textBoxInput->ReadOnly = false;

}

}

На рисунке 1 изображен результат выполнения кода. На нем показан TextBox, кнопка открыть и checkbox блокировать.

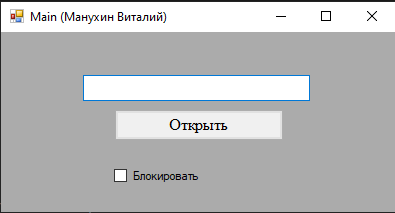
****

Рисунок 1 – Результат выполнения кода

Данный вариант диалогового окна имел ошибки. В нем неправильно реализованы checkbox. По заданию необходимо реализовать checkbox (Блокировать), а также checkbox (Видимость), с активным свойством checked(true).При нажатии на checkbox (Блокировать) – область textbox становится недоступной для редактирования. При нажатии на checkbox (Видимость) – область textbox скрывается. В представленном ниже участке кода присутствует метод реализации checkbox, в котором нужно изменить условия выполнения действий:

//реализация метода checkbox(Блокировать)

private: System::VoidchBox\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (chBox->Checked) {

this->textBoxInput->ReadOnly = true;

}

else {

this->textBoxInput->ReadOnly = false;

}

}

В представленном ниже участке кода реализованы недочеты предыдущего варианта кода:

//реализация метода checkbox(Блокировать)

private: System::Void chBox\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (chBox->Checked) {

//отключение блокировки

this->textBoxInput->ReadOnly = false;

}

else {

//блокировка редактирования textbox

this->textBoxInput->ReadOnly = true;

}

}

//реализация метода checkbox(Видимость)

private: System::Void checkBox1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (checkBox1->Checked) {

//показывает textbox

this->textBoxInput->Visible = true;

}

else {

//скрывает textbox

this->textBoxInput->Visible = false;

}

}

На рисунке 2 представлен результат выполнения исправленного кода.

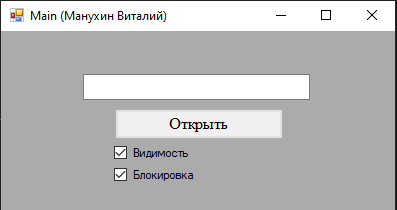


Рисунок 2 – Результат выполнения исправленного кода

На рисунке 3 представлен результат выполнения кода при отключенной checkbox(Видимость) – происходит скрытие textbox.

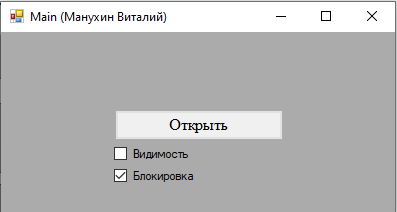


Рисунок 3 – Результат выполнения при отключенной видимости

На рисунке 4 представлен результат выполнения при выключенной checkbox (Блокировка) – происходит блокировка изменений textbox.

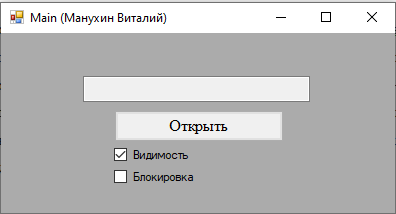


Рисунок 4 – Результат выполнения при выключенной блокировке

Данный вариант диалогового окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **2 Задание. Рисование в окне курсором мыши, реализация смены цвета и толщины линии**

Разработать программный модуль, в котором предусмотрена возможность для рисования с помощью линий, установленного цвета и размера в соответствии с движением мыши. Условия: наличие возможности выбора минимум 3-х цветов (из списка), наличие возможности выбора толщины линий, наличие кнопки «выход».

Создается окно, а также необходимые кнопки, в нижеприведенном варианте это кнопка exit и menu. По нажатию кнопки «Выйти» происходит выход из программы строкой Application::exit(). По нажатию кнопки menu – открывается панель элементов. Панель элементов – groupBox, на которой располагается название label(Параметры кисти), кнопка выбора цвета button(color), числовая панель выбора толщины линии numericupdows и кнопка принятия изменений размера, работая в связке с числовой панелью.

В приведенном ниже коде реализация метода смены толщины линии, цвета, также кнопка меню и выход из программы.

//реализация кнопки установки толщины линии (по умолчанию 1)

private: System::Void buttonApplySize\_Click(System::Object^ sender, System::EventArgs^ e)

{

//устанавливается значение, взятое из числовой панели выбора толщины

size = Convert::ToSingle(numericUpDownSize->Value);

pen = gcnew Pen(color, size);

returnSystem::Void();

}

//реализация кнопки смены цветов

private: System::Void buttonColor\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (color == Color::Black) {

//красный цвет

color = Color::Red;

}

else if(color == Color::Red) {

//черный цвет

color = Color::Blue;

}

else if (color == Color::Blue) {

//buttonColor->ForeColor = Color::Green;

color = Color::Green;

//зеленый цвет

}

else if (color == Color::Green) {

//buttonColor->ForeColor = Color::Green;

//черный цвет

color = Color::Black;

}

buttonColor->BackColor = color;

pen = gcnew Pen(color, size);

return System::Void();

}

//открытие меню

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

//условие при котором открывается меню

if (this->menuStrip1->Visible == false) {

//меню открывается

this->menuStrip1->Visible = true;

}

else {

//меню закрывается

this->menuStrip1->Visible = false;

}

}

//кнопка выхода

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

Application::Exit();

}

На рисунке 5 изображен результат выполнения кода. На нем показано окно для рисования, кнопка, отвечающая за выход из программы и кнопка открытия меню. В меню располагается вкладка Панель инструментов, в которой Панель элементов. На панели инструментов располагаются инструменты редактирования цвета и толщины линии.

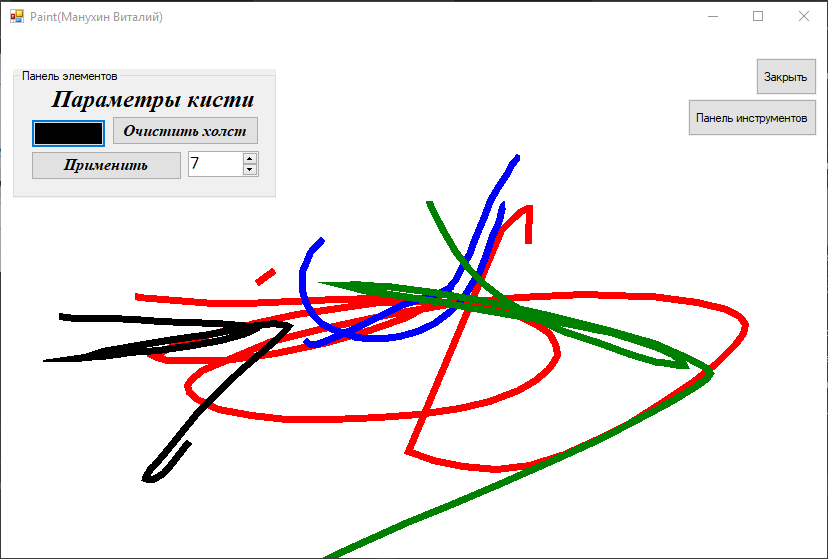
****

Рисунок 5 – Результат выполнения кода

Данный вариант диалогового окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **3 Задание. Отображение нажатой клавиши на клавиатуре в дочернем окне**

Разработка программного модуля, создающего диалоговую панель на которой должна быть кнопка запуска дочернего окна, в котором при нажатии на клавиатуру, показывается нажатый символ. В дочернем окне символ увеличиваетсяx4 от стандартного размера.

Создается главное окно, а также кнопка запуска дочернего окна. В дочернем окне по центру располагается пустой текстовый label, в параметры которого переданы размеры(x4 от базового). При нажатии клавиши на клавиатуре в дочернем окне – текстовому полю передается id нажатой клавиши, конвертированное в текстовой формат, то есть в название нажатой клавиши. При повторном нажатии – старый текст удаляется и заменяется на новый.

В приведенном ниже коде объявление дочернего окна, реализация метода обработки клавиш и изменение размера текстa label:

//объявление дочернего окна

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(462, 334);

this->Controls->Add(this->label1);

this->KeyPreview = true;

this->Name = L"windowKey";

this->Text = L"Диалоговое окно(Манухин)";

this->Load += gcnewSystem::EventHandler(this, &windowKey::windowKey\_Load);

this->KeyDown += gcnewSystem::Windows::Forms::KeyEventHandler(this, &windowKey::windowKey\_KeyDown);

this->ResumeLayout(false);

this->PerformLayout();

//Обработканажатыхклавиш

#pragma endregion

private:System::VoidwindowKey\_KeyDown(System::Object^ sender, System::Windows::Forms::KeyEventArgs^ e)

//изменение размера выводимого текста

{

this->label1->Font = (gcnewSystem::Drawing::Font(L"Times new roman", 32, System::Drawing::FontStyle::Regular,System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(0)));

this->label1->Text = e->KeyCode.ToString();

}

На рисунке 6 изображен результат выполнения кода. На нем показана кнопка, активация которого открывает дочернее окно.

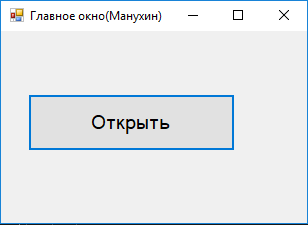
****

Рисунок 6 – Результат выполнения кода

На рисунке 7 изображен результат нажатия кнопки «Запуск» из главного окна. На нем показано текстовое поле, отображающее нажатую клавишу LWin.

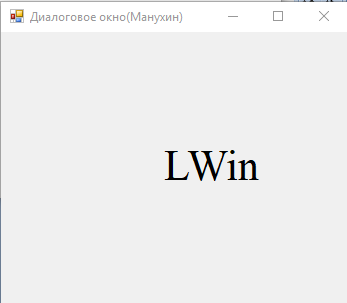


Рисунок 7 – Результат нажатия кнопки «Запуск» из главного окна

Данный вариант главного и дочернего окон являются верными, так как реализованы все необходимые методы и соблюдены условия задачи.

### **4 Задание. Отображение информации текстового поля из главного окна в дочернем**

Разработать модуль, содержащий меню из двух кнопок «файл» и «помощь», в рабочей области расположено окно редактирования (текстовое поле в середине главного окна), которое позволяет вводить только числа без знаков препинания, букв, спец клавиш, пункт меню «файл» содержит только одну команду «показать число», эта команда вызывает окно с сообщением, в которое передаётся информация из окна редактирования.

Создается главное окно, на нем будет располагаться меню с разделами «файл» и «помощь». В главном окне создаем textbox с параметрами, не позволяющими вводить ничего кроме цифр (от 0-9). В разделе «файл» содержится команда «показать число», активируя которую, открывается дочернее окно с информацией по центру о том, что вводил пользователь в главном окне в textbox.

В приведенном ниже коде метод обработки символов, вводимые пользователем и активация команды «показать число»:

//Объявление метода

richTextBox1->KeyPress+=gcnewSystem::KeyPressEventHandler(this, &windowMainTextbox::richTextBox1\_KeyPress);

//Обработчик символов, вводимых пользователем

public: System::Void richTextBox1\_KeyPress(System::Object^ sender, System::::KeyPressEventArgs^ e) {

if (!Char::IsDigit(e->KeyChar) && e->KeyChar != 0x08) {

e->Handled = true;

}}

//команда «показать число»

private: System::Void ToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

//объявление дочернего окна

windowEdit^ windowEditText = gcnewwindowEdit();

windowEditText->label1->Text = this->richTextBox1->Text;

windowEditText->Show();

}

На рисунке 8 изображен результат выполнения кода. На нем показан textbox с числовым значением и команда показать число из меню.

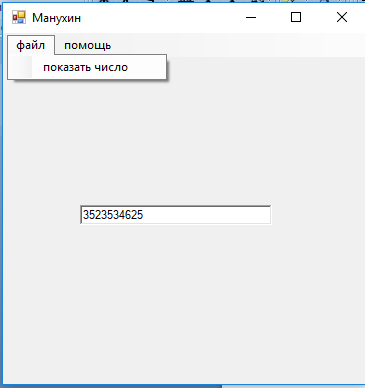
****

Рисунок 8 – Результат выполнения кода

На рисунке 9 изображен результат запуска команды «показать число» из меню главного окна. На нем показано текстовое поле, отображающее информацию из текстового поля главного окна.

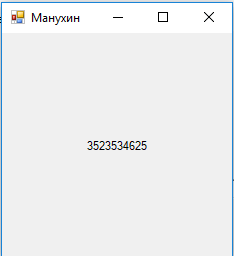


Рисунок 9 – Информация из текстового поля главного окна

Данный вариант главного и дочернего окон являются верными, так как реализованы все необходимые методы и соблюдены условия задачи.

### **5 Задание. Приложение – ответы на вопросы**

Разработать программный модуль, в главном окне которого расположено 3 кнопки "вопрос 1" "вопрос 2" "вопрос 3",каждая кнопка вызывает отдельное окно, в котором расположен некоторый вопрос и две кнопки ответа. При выборе кнопки с неправильным ответом, выдаётся сообщение о том, что ответ неверный и закрывается окно. При выборе кнопки с правильным ответом появляется сообщение о том, что ответ верный и вопрос "выбрать ли другой вопрос?", где будет 3 варианта - две кнопки с оставшимися вопросами и кнопка "нет", закрывающая окно. Для того, чтобы закрыть главное окно используется пункт меню файл->выход. При выборе пункта меню "помощь" должно открываться справочное окно, где написано, что надо делать в этой программе.

Создается главное окно, на котором располагаются 3 кнопки(Вопрос 1, Вопрос 2, Вопрос 3), в меню есть 2 вкладки – «Файл», в котором находится команда «выход», нажимая которую, программа закрывается, и вкладка «Помощь», нажимая на которую, открывается диалоговое окно с описание программы. Нажимая на одну из кнопок, пользователя перебрасывает в дочернее окно, на котором label с вопросом и 2 кнопки с разными вариантами ответа. Если нажать на правильный ответ – открывается диалоговое окно с информацией о том, что ответ правильный и программа предлагает продолжить отвечать на вопросы. Если нажать на неправильный ответ – открывается дочернее окно с информацией об этом, после чего программа завершается.

В приведенном ниже коде метод обработки ответов пользователя в дочернем окне:

else if ((menu)wParam == menu::button1)

{

auto id = MessageBox(hwndA, L"2+2=4?", L"Вопрос 1", MB\_YESNO);

// Если ответ правильный

if (id == IDYES)

{

//появляется окно с выбором следующего вопроса

ShowWindow(hwndB, SW\_SHOW);

but2 = CreateWindow(L"button", L"Вопрос 2", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 20, 200, 100, 22, hwndB, (HMENU)menu::but2, NULL, NULL);

but3 = CreateWindow(L"button", L"Вопрос 3", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 120, 200, 100, 22, hwndB, (HMENU)menu::but3, NULL, NULL);

cancel = CreateWindow(L"button", L"Отмена", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 220, 200, 100, 22, hwndB, (HMENU)menu::cancel, NULL, NULL);

}

// Если ответ не правильный

elseif (id == IDNO)

{

//появляется дочернее, сообщающий, что ответ неверный

MessageBox(hwndA, L"Ответневерный!", L"", MB\_OK);

}

}

else if ((menu)wParam == menu::button2)

{

autoid = MessageBox(hwndA, L"кореньиз 324 = 5?", L"Вопрос 2", MB\_YESNO);

if (id == IDYES)

{

MessageBox(hwndA, L"Ответневерный!", L"", MB\_OK);

}

else if (id == IDNO)

{

ShowWindow(hwndB, SW\_SHOW);

but1 = CreateWindow(L"button", L"Вопрос 1", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 20, 200, 100, 22, hwndB, (HMENU)menu::but1, NULL, NULL);

but3 = CreateWindow(L"button", L"Вопрос 3", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 120, 200, 100, 22, hwndB, (HMENU)menu::but3, NULL, NULL);

cancel = CreateWindow(L"button", L"Отмена", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 220, 200, 100, 22, hwndB, (HMENU)menu::cancel, NULL, NULL);

}

}

else if ((menu)wParam == menu::button3)

{

auto id = MessageBox(hwndA, L"Выпоставите 5?", L"Вопрос 3", MB\_YESNO);

if (id == IDYES)

{

ShowWindow(hwndB, SW\_SHOW);

but1 = CreateWindow(L"button", L"Вопрос 1", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 20, 200, 100, 22, hwndB, (HMENU)menu::but1, NULL, NULL);

but2 =CreateWindow(L"button", L"Вопрос 2", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 120, 200, 100, 22, hwndB, (HMENU)menu::but2, NULL, NULL);

cancel = CreateWindow(L"button", L"Отмена", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 220, 200, 100, 22, hwndB, (HMENU)menu::cancel, NULL, NULL);

}

else if (id == IDNO)

{

MessageBox(hwndA, L"Ответневерный!", L"", MB\_OK);

}

}

На рисунке 10 изображен результат выполнения кода. На нем показаны 3 кнопки (Вопрос 1, Вопрос 2, Вопрос 3) и меню с вкладками «Файл» с командой «Выход» и «Помощь» с информацией о программе.

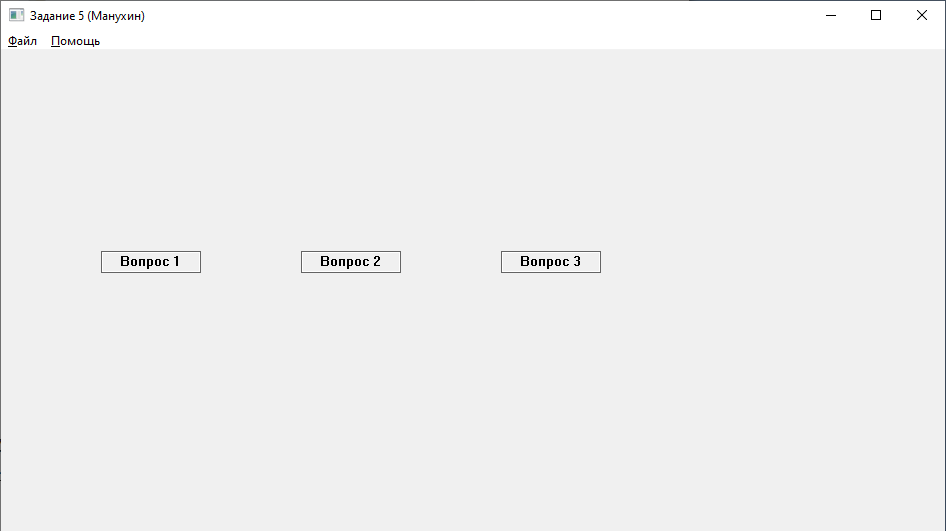
****

Рисунок 10 – Результат выполнения кода

На рисунке 11 изображен результат нажатия кнопки «Помощь» - открытие диалогового окна с инструкцией по использованию программы.

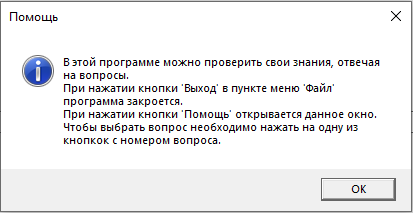


Рисунок 11 – Диалоговое окно «Помощь»

На рисунке 12 изображен результат нажатия на кнопку «Вопрос 1» - дочернее окно с вопросом и вариантами ответа.

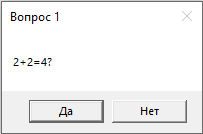


Рисунок 12 – Дочернее окно «Вопрос 1»

На рисунке 13 изображен результат нажатия на кнопку с правильным ответом.

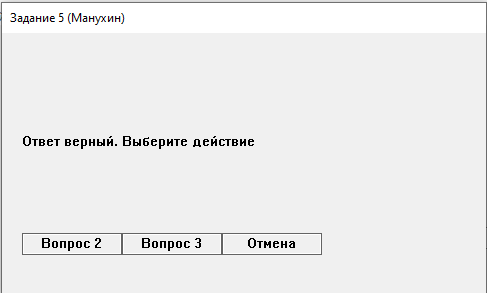


Рисунок 13 – Диалоговое окно при выборе правильного ответа

На рисунке 14 изображен результат нажатия на кнопку с неправильным ответом.

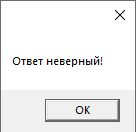


Рисунок 14 – Диалоговое окно «Ответ неверный»

Данный вариант главного, дочерних и диалоговых окон являются верными, так как реализованы все необходимые методы и соблюдены условия задачи.

### **6 Задание. Непрерывное рисование окружности с изменением цвета**

Разработать программный модуль создающий диалоговую панель, в которой в непрерывном режиме выполняется прорисовка окружности у которой радиус увеличивается, при достижении границ окна радиус начинает уменьшаться. У диалогового окна должна присутствовать кнопка выбора цвета, которая выводит диалоговое окно с выбором цвета круга с помощью радиокнопок. Для изменения цвета круга нужно подтвердить выбор кнопкой "ОК.

Создается главное окно, а также кнопка «Цвет», нажимая на которую открывается диалоговое окно с выбором цветов. На главном окне непрерывно рисуется окружность до границ окна и обратно до минимального значения радиуса по центру окна.

В приведенном ниже коде метод обработки главного окна:

LRESULT CALLBACK WindowProcessMessages(HWND hwnd, UINT msg, WPARAM param, LPARAM lparam) {

HDC hdc;

PAINTSTRUCT ps;

switch (msg) {

case WM\_COMMAND:

{

if ((buttons)param == buttons::color)

{

ShowWindow(hwndB, SW\_SHOW);

}

}

case WM\_CREATE:

{

SetTimer(hwnd, 1, 100, NULL);

break;

}

case WM\_SIZE:

{

width = LOWORD(lparam);

height = HIWORD(lparam);

break;

}

case WM\_TIMER:

{

InvalidateRect(hwndA, NULL, TRUE);

UpdateWindow(hwndA);

circle\_diameter += 10;

if (circle\_diameter>= width || circle\_diameter>= height)

circle\_diameter = -circle\_diameter;

hdc = GetDC(hwndA);

draw(hdc);

ReleaseDC(hwndA, hdc);

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

default:

returnDefWindowProc(hwnd, msg, param, lparam);

}

}

На рисунке 15 изображен результат выполнения кода. На нем показана окружность и кнопка «Цвет».

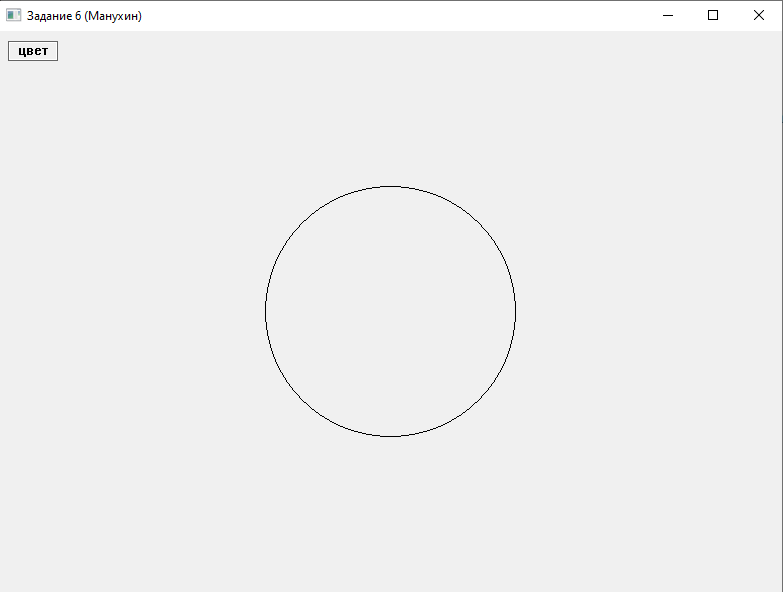
****

Рисунок 15 – Результат выполнения кода

На рисунке 16 изображен результат нажатия кнопки «Цвет» – открытие диалогового окна с выбором одного из трех цветов(красный, фуксия и синий).

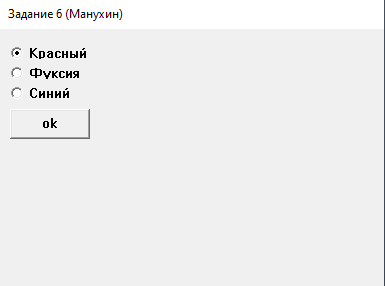


Рисунок 16 – Диалоговое окно с выбором цвета

На рисунке 17 изображен результат главного окна с выбранным цветом «фуксия».

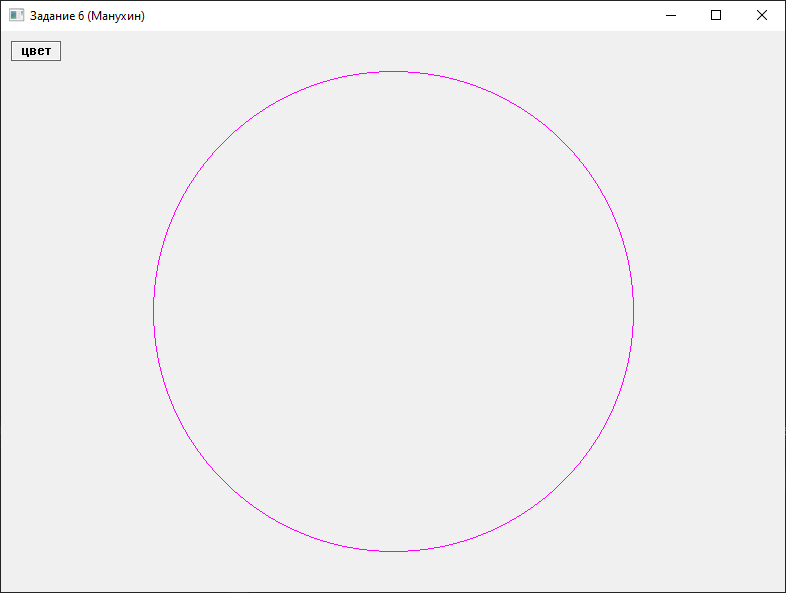


Рисунок 17 – Результат окна с выбранным цветом

Данный вариант главного окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **7 Задание. Открытие проводника через меню и выбор картинки для фона окна**

Создать модуль имеющий меню содержащий пункт меню«Файл», содержащий кнопки «Открыть» и «Выход». Кнопка «Открыть» позволяет открыть графический файл с расширением«.bmp». Кнопка «Выход» позволяет выйти из программы. Не должно быть изменений с файлом кроме изменения размера окна. Файл должен отрываться в главном окне этого модуля и подровняться под размеры этого окна.

Создается окно и меню с вкладкой «Файл», имеющий кнопки «Открыть» и «Выход». При нажатии кнопки «Выход» – программа прерывается и окно закрывается. При нажатии кнопки «Открыть» – открывается проводник, в котором можно выбрать картинки с расширением «.bmp». После выбора картинки она устанавливается на фон модуля и выравнивается по ширине окна, так же ее можно сменить, повторно нажав кнопку «Открыть».

В приведённом ниже коде представлена работа кнопок «Открыть» и «Выход»:

case WM\_COMMAND:

{

if ((menu)wParam == menu::exit)

{

PostQuitMessage(0);

}

if ((menu)wParam == menu::open)

{

OPENFILENAME open\_file\_name;

const LPCWSTR filters = L"BMP \0\*.bmp\0";

const auto flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;

ZeroMemory(&open\_file\_name, sizeof(OPENFILENAME));

std::memset(path, 0, sizeof(path));

open\_file\_name.lStructSize = sizeof(OPENFILENAME);

open\_file\_name.hInstance = (HINSTANCE)GetStockObject(NULL);

open\_file\_name.hwndOwner = NULL;

open\_file\_name.lpstrFile = path;

open\_file\_name.nMaxFile = MAX\_PATH;

open\_file\_name.lpstrFilter = filters;

open\_file\_name.nFilterIndex = 1;

open\_file\_name.lpstrFileTitle = NULL;

open\_file\_name.nMaxFileTitle = 0;

open\_file\_name.lpstrInitialDir = NULL;

open\_file\_name.Flags = flags;

if (GetOpenFileName(&open\_file\_name))

{

autohdc = GetDC(hwndA);

Gdiplus::Graphics gf(hdc);

Gdiplus::Bitmap bmp(path);

gf.DrawImage(&bmp, 0, 0, width, height);

return 1;

}

}

break;

}

На рисунке 18 изображен результат выполнения кода. На нем показано меню с командами «Открыть» и «Выход».

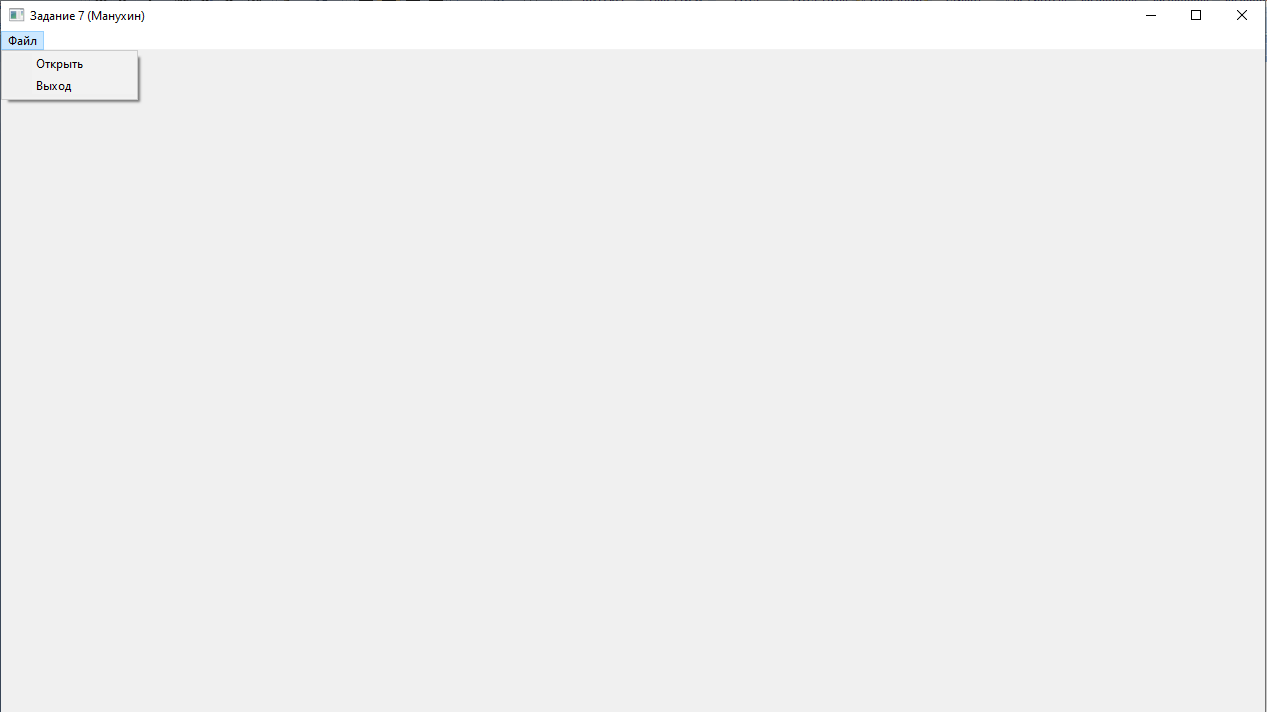
****

Рисунок 18 – Результат выполнения кода

На рисунке 19 изображен результат нажатия кнопки «Открыть».

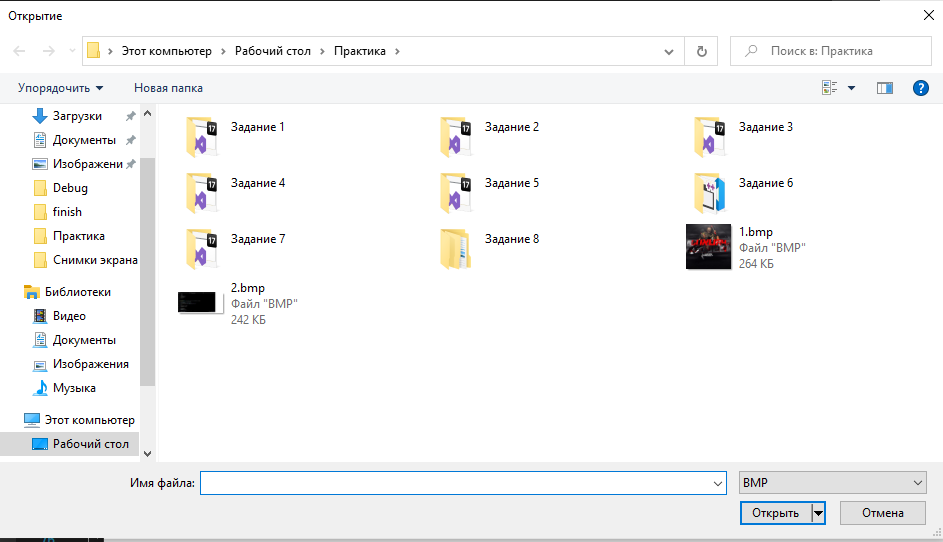


Рисунок 19 – Результат нажатия кнопки «Открыть»

На рисунке 20 изображен результат установки картинки на фон модуля.



Рисунок 20 – Результат установки картинки на фон модуля

Данный вариант главного окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **8 Задание. Разработка модуля запуска семи прошлых модулей через меню**

Необходимо создать программу, объединяющую в себе все 7 модулей следующим образом: главное окно программы имеет в качестве фона приветственную надпись, строку меню с двумя пунктами: «Программы» и «Информация». По нажатию кнопки «Информация» открывается окно с данными ФИО Группа Специальность и список программ с кратким описанием.Кнопка «Программы» открывает список меню, в котором расположены все модули и кнопка «Выход», отделённая линией.По нажатию на пункт меню открывается выбранная программа в новом окне. Закрытие этой программы в ходе работы не должно вызывать закрытие всей программы.

Создается окно и меню с вкладками «Программы» и «Информация». В подменю «Программы» содержатся все работы по порядку и кнопка «Выход», подчеркнутая линией. При нажатии кнопки «Информация» открывается диалоговое окно с информацией о студенте и краткое описание заданий. При нажатии на одну из кнопок подменю «Программы» открывается выбранное задание.

В приведённом ниже коде меню с кнопками «Программы» и «Информация»:

this->информацияToolStripMenuItem->Name = L"информацияToolStripMenuItem";

this->информацияToolStripMenuItem->Size = System::Drawing::Size(87, 20);

this->информацияToolStripMenuItem->Text = L"Программы";

//

// Task1ToolStripMenuItem

//

this->Task1ToolStripMenuItem->Name = L"Task1ToolStripMenuItem";

this->Task1ToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->Task1ToolStripMenuItem->Text = L"Задание 1";

this->Task1ToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::Task1ToolStripMenuItem\_Click);

//

// Task2ToolStripMenuItem

//

this->Task2ToolStripMenuItem->Name = L"Task2ToolStripMenuItem";

this->Task2ToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->Task2ToolStripMenuItem->Text = L"Задание 2";

this->Task2ToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::Task2ToolStripMenuItem\_Click);

//

// Task3ToolStripMenuItem

//

this->Task3ToolStripMenuItem->Name = L"Task3ToolStripMenuItem";

this->Task3ToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->Task3ToolStripMenuItem->Text = L"Задание 3";

this->Task3ToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::Task3ToolStripMenuItem\_Click);

//

// Task4ToolStripMenuItem

//

this->Task4ToolStripMenuItem->Name = L"Task4ToolStripMenuItem";

this->Task4ToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->Task4ToolStripMenuItem->Text = L"Задание 4";

this->Task4ToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::Task4ToolStripMenuItem\_Click);

//

// Task5ToolStripMenuItem

//

this->Task5ToolStripMenuItem->Name = L"Task5ToolStripMenuItem";

this->Task5ToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->Task5ToolStripMenuItem->Text = L"Задание 5";

this->Task5ToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::Task5ToolStripMenuItem\_Click);

//

// Task6ToolStripMenuItem

//

this->Task6ToolStripMenuItem->Name = L"Task6ToolStripMenuItem";

this->Task6ToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->Task6ToolStripMenuItem->Text = L"Задание 6";

this->Task6ToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::Task6ToolStripMenuItem\_Click);

//

// Task7ToolStripMenuItem

//

this->Task7ToolStripMenuItem->Name = L"Task7ToolStripMenuItem";

this->Task7ToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->Task7ToolStripMenuItem->Text = L"Задание 7";

this->Task7ToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::Task7ToolStripMenuItem\_Click);

//

// toolStripMenuItem1

//

this->toolStripMenuItem1->Name = L"toolStripMenuItem1";

this->toolStripMenuItem1->Size = System::Drawing::Size(177, 6);

//

// выходToolStripMenuItem

//

this->выходToolStripMenuItem->Name = L"выходToolStripMenuItem";

this->выходToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->выходToolStripMenuItem->Text = L"Выход";

this->выходToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::выходToolStripMenuItem\_Click\_1);

//

// программыToolStripMenuItem

//

this->программыToolStripMenuItem->Name = L"программыToolStripMenuItem";

this->программыToolStripMenuItem->Size = System::Drawing::Size(93, 20);

this->программыToolStripMenuItem->Text = L"Информация";

this->программыToolStripMenuItem->Click += gcnew System::EventHandler(this, &helloWindow::программыToolStripMenuItem\_Click);

На рисунке 21 изображен результат выполнения кода. На нем показано меню с кнопками «Программы» и «Информация». А также список в подменю «Программы» – 7 заданий и кнопка «Выход».

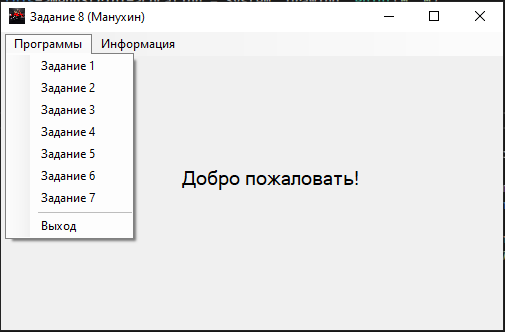
****

Рисунок 21 – Результат выполнения кода

На рисунке 22 изображен результат нажатия кнопки «Информация» – открытие диалогового окна с информацией о студенте и заданиях.

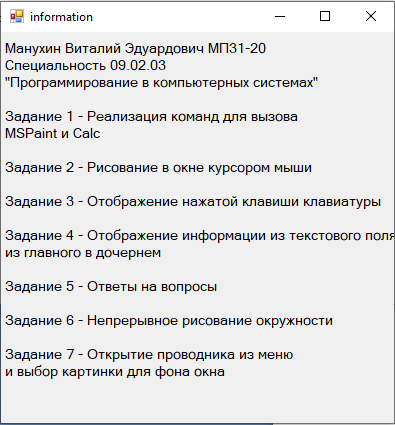


Рисунок 22 – Диалоговое окно «Информация»

На рисунке 23 изображен результат нажатия на кнопку «Задание 4» – запуск программы четвертого задания в отдельном окне.

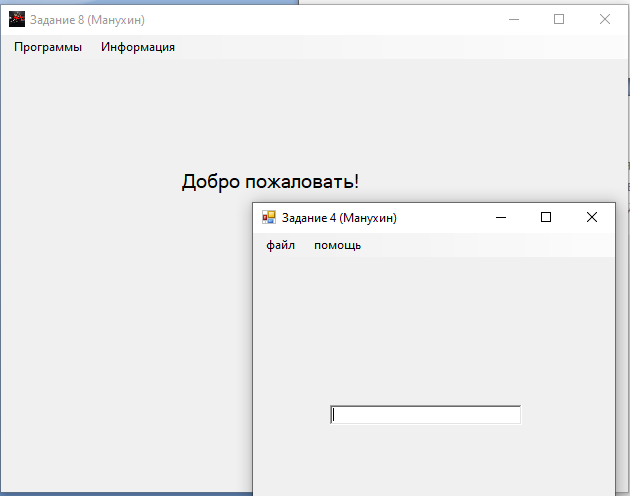


Рисунок 23 – Результат нажатия на кнопку «Задание 4»

Данный вариант окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **ЗАКЛЮЧЕНИЕ**

В процессе курсовой работы была изучена техника программирования графики в середе Visual C++. В качестве интерфейса программирования был использован Windows API, который доступен для использования в этой среде с помощью заголовочного файла windows.h. Разработаны программы:

* Задание 1 - Диалоговые окна, реализация команд для вызова MSPaint и Calc, свойства CheckBox;
* Задание 2 - Рисование в окне курсором мыши, реализация смены цвета и толщины линии;
* Задание 3 - Создание диалогового окна, в котором при нажатии показывает нажатую клавишу;
* Задание 4 - Разработать модуль с меню и окном редактирования с вводом только чисел;
* Задание 5 - Разработать модуль ответов на вопросы;
* Задание 6 - Разработать модуль круг, который динамически изменяющего свой радиус;
* Задание 7 - Разработать модуль выбора изображений из файла и изменении их размера под размеры окна;
* Задание 8 - Разработать модуль запуска семи прошлых модулей через меню.

Несмотря на то, что MFC не был задействован в этих программах, алгоритмы и API использовавшиеся могут быть применены и в программе на базе MFC, так как из классов MFC можно всегда получить идентификаторы окон и прочие необходимые для работы с API данные. Это подтверждает эффективность и универсальность интерфейса Windows API.