**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»**

**ХАБАРОВСКИЙ ИНСТИТУТ ИНФОКОММУНИКАЦИЙ (ФИЛИАЛ)**

**СРЕДНЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ**

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №15

по МДК 01.01 Системное программирование

на тему

Разработка приложений Windows Forms.

Выполнил: А.В. Медведев

Проверил: Н.А. Насонова

Группа: ПКС-320

2023

*Практическая работа № 1*

*Среда разработки программ Microsoft Visual Studio.*

#pragma once

#include <windows.h>

namespace MyForms {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

// Программный код, расположенный выше, создан средой Visual Studio

// автоматически, поэтому автором не приводится

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

public ref class MyForm : public System::Windows::Forms::Form

{

private: System::Void Form1\_Load(System::Object^ sender, System::EventArgs^ e)

{ // Обработка события загрузки формы:

this->Text = "Приветствие";

// или Form1::Text = "Приветствие";

label1->Text = "Microsoft Visual C++ 2010";

button1->Text = "Нажми меня";

}

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

{ // Обработка события щелчок на кнопке:

MessageBox::Show("Всем привет!");

}

private: System::Void label1\_MouseHover(System::Object^ sender, System::EventArgs^ e)

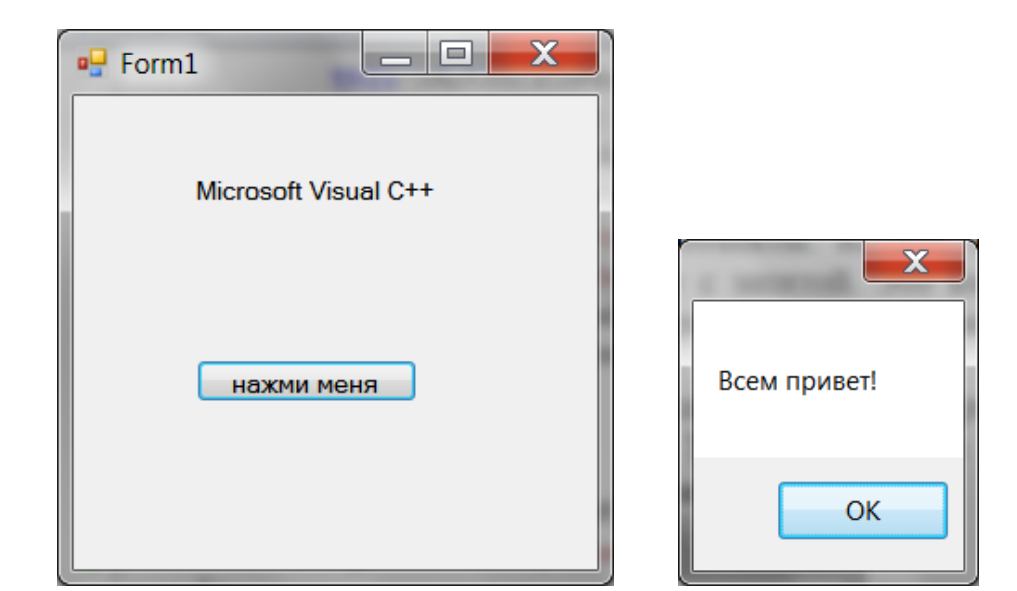
{ // Обработка события, когда указатель мыши наведен на метку:

MessageBox::Show("Событие Hover!");

}

};

}



**Контрольные вопросы.**

1. Из каких двух этапов состоит процесс проектирования программы Visual C++?

Два шага, связанные с проектированием программы Visual C++:

Дизайн пользовательского интерфейса: Этот шаг включает в себя проектирование интерфейса, с которым будет взаимодействовать пользователь. Это включает в себя проектирование расположения окон, меню, кнопки, и другие графические элементы. Цель состоит в том, чтобы создать интуитивно понятный и удобный интерфейс, в котором легко ориентироваться и понимать.

Дизайн логики приложения: Этот шаг включает в себя разработку базовой логики программы. Сюда входит разработка алгоритмов и структур данных, которые программа будет использовать для выполнения своих задач. Цель состоит в том, чтобы создать надежную и эффективную программу, которая может обрабатывать широкий спектр входных данных и выдавать точные выходные данные.

Эти два шага тесно связаны, поскольку дизайн пользовательского интерфейса часто влияет на логику приложения. Например, дизайн интерфейса может диктовать типы ввода, которые может принимать программа, что, в свою очередь, повлияет на разработку алгоритмов и структур данных. Следуя структурированному процессу проектирования, разработчики могут создавать высококачественные программы Visual C++, отвечающие потребностям их пользователей.

1. Что такое программа, основанная на диалоге?

Диалоговая программа в C++ Windows Form — это тип приложения, использующего диалоговые окна в качестве основного интерфейса для взаимодействия с пользователем.

В программах этого типа пользовательский интерфейс создается с помощью диалоговых окон, которые представляют собой окна, отображающие сообщение или подсказку и позволяющие пользователю выбирать параметры или вводить информацию. Диалоговые окна можно настроить с помощью различных элементов управления, таких как кнопки, текстовые поля и раскрывающиеся списки, чтобы обеспечить более интерактивное взаимодействие с пользователем.

Диалоговые программы часто используются для простых приложений, не требующих сложного пользовательского интерфейса. Они также полезны для создания всплывающих окон или окон сообщений, предоставляющих пользователям важную информацию или предупреждения.

В C++ Windows Form разработчики могут создавать диалоговые программы с помощью библиотеки MFC (Microsoft Foundation Classes) или с помощью встроенного редактора диалоговых окон в Visual Studio. Библиотека MFC предоставляет набор классов и функций, упрощающих создание диалоговых приложений, а редактор диалоговых окон в Visual Studio позволяет разработчикам создавать и настраивать диалоговые окна с помощью визуального интерфейса.

1. Что такое Windows Forms Application?

Приложение Windows Forms — это тип приложения с графическим пользовательским интерфейсом (GUI), разработанного для операционной системы Microsoft Windows с использованием платформы .NET Framework или .NET Core. Это тип приложения, который позволяет разработчикам создавать многофункциональные интерактивные настольные приложения с пользовательскими интерфейсами, включающими окна, меню, кнопки и другие элементы управления.

Приложения Windows Forms обычно разрабатываются с использованием таких языков программирования, как C# или Visual Basic .NET, и предоставляют разработчикам широкий спектр функций и инструментов для создания мощных настольных приложений. Эти приложения можно развернуть на локальных компьютерах или в сети, и они могут взаимодействовать с другими приложениями и службами в системе.

Приложения Windows Forms обеспечивают высокую степень настройки и гибкости, позволяя разработчикам создавать пользовательские интерфейсы, отвечающие потребностям их пользователей. Они также предлагают такие функции, как привязка данных, поддержка операций перетаскивания и возможность создавать собственные элементы управления и компоненты. Приложения Windows Forms обычно используются в различных отраслях, включая финансы, здравоохранение, образование и многие другие.

*Практическая работа № 2*

*Реализация простейшей программы на VC++, с использованием интерфейса и библиотеки CLR.*

#pragma once

#include <windows.h>

namespace MyForms {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

// Программный код, расположенный выше, создан средой Visual Studio

// автоматически, поэтому автором не приводится

this->PerformLayout();

}

#pragma endregion

private: System::

Void Form1\_Load(System::Object^ sender, System::EventArgs^ e)

{

button1->Text = "Извлечь корень"; label1->Text = String::Empty;

// или label1->Text = nullptr;

this->Text = "Извлечение квадратного корня";

textBox1->Clear(); // Очистка текстового поля

textBox1->TabIndex = 0; // Установка фокуса в текстовом поле

}

private: System:: // Обработка щелчка на кнопке "Извлечь корень":

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

{

Single X; // - из этого числа будем извлекать корень

// Преобразование из строковой переменной в Single:

bool Число\_ли = Single::TryParse(textBox1->Text,

System::Globalization::NumberStyles::Number,

System::Globalization::NumberFormatInfo::CurrentInfo, X);

// Второй параметр - это разрешенный стиль числа (Integer,

// шестнадцатеричное число, экспоненциальный вид числа и прочее).

// Третий параметр форматирует значения на основе текущего языка

// и региональных параметров из Панели управления - Язык и

// региональные стандарты - число допустимого формата; метод

// возвращает значение в переменную X

if (Число\_ли == false)

label1->Text = "Следует вводить числа";

label1->ForeColor = Color::Red; // - цвет текста на метке return; // - выход из процедуры

}

Single Y = (Single)Math::Sqrt(X); // - извлечение корня

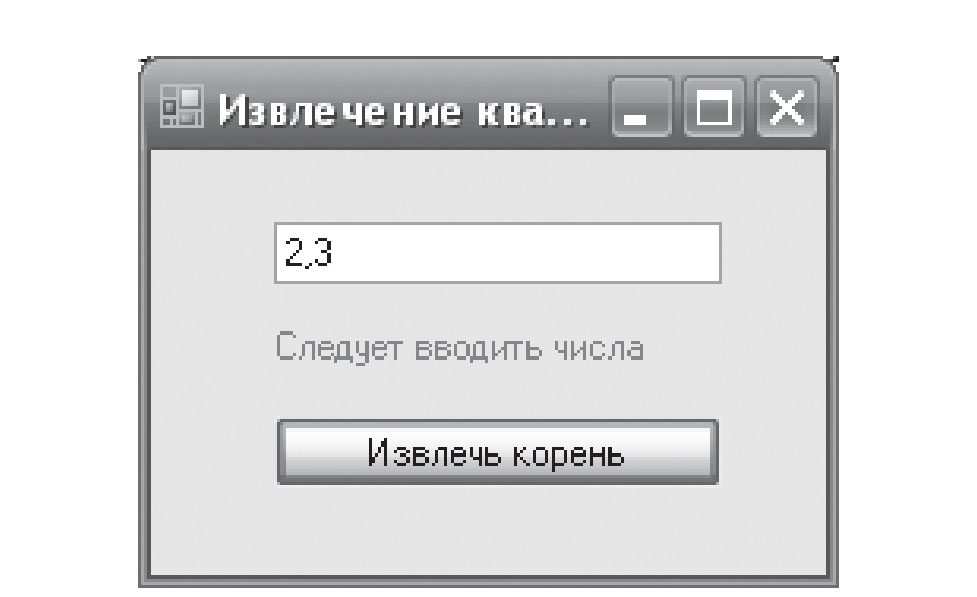
label1->ForeColor = Color::Black; // - черный цвет текста на метке

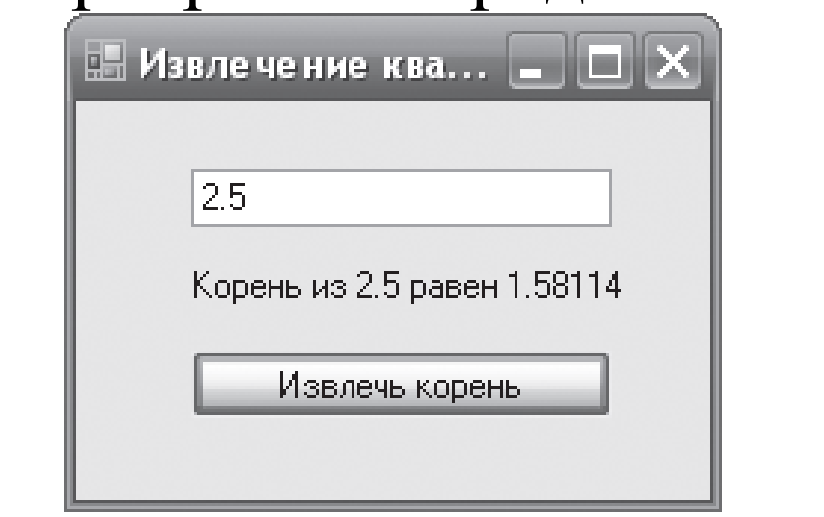
label1->Text = String::Format("Корень из {0} равен {1:F5}", X, Y);

}

};

}





**Контрольные вопросы.**

1. Опишите шаги, которые вы сделали, чтобы открыть диалоговую панель программы для ее визуальной настройки

Чтобы открыть диалоговое окно программы для визуальной настройки в C++ Windows Forms, выполните следующие действия:

1. Откройте проект C++ Windows Forms в Microsoft Visual Studio.
2. В окне обозревателя решений найдите объект формы, который вы хотите настроить, и дважды щелкните его, чтобы открыть конструктор.
3. В дизайнере щелкните правой кнопкой мыши объект формы и выберите «Свойства» в контекстном меню.
4. Появится окно «Свойства», показывающее список свойств для выбранного объекта «Форма». Найдите раздел «Внешний вид» в окне «Свойства» и разверните его, щелкнув значок маленького треугольника рядом с заголовком раздела.
5. В разделе «Внешний вид» вы найдете несколько свойств, позволяющих настроить внешний вид объекта формы. Например, вы можете изменить цвет фона формы, изменив свойство «BackColor», или изменить шрифт, используемый для текста формы, изменив свойство «Шрифт».
6. Внесите необходимые изменения в свойства внешнего вида формы. Вы можете предварительно просмотреть изменения в режиме реального времени, вернувшись в режим конструктора.
7. Когда вы будете удовлетворены своими изменениями, сохраните проект, выбрав «Файл» > «Сохранить» в строке меню или нажав сочетание клавиш «Ctrl» + «S».

Выполнив следующие действия, вы сможете открыть диалоговое окно программы для визуальной настройки в C++ Windows Forms и внести изменения во внешний вид объекта формы. Обратите внимание, что эти шаги охватывают только основные свойства внешнего вида объекта формы — для более сложной настройки может потребоваться дополнительное кодирование или использование сторонних инструментов.

1. Что такое класс Wizard?

Класс Wizard в C++ Windows Form Application — это тип диалогового окна, которое направляет пользователя через серию шагов для выполнения определенной задачи или набора задач. Он предоставляет простой в использовании интерфейс, который может помочь пользователям выполнять сложные задачи, разбивая их на более мелкие и более управляемые шаги.

Класс Wizard — это предварительно созданный класс в .NET Framework, включенный в пространство имен System::Windows::Forms. Он является производным от класса Form и предоставляет набор свойств и методов для создания диалоговых окон мастера и управления ими.

Чтобы использовать класс Wizard в приложении Windows Form C++, вы можете создать новый экземпляр класса Wizard, задать его свойства для определения внешнего вида и поведения диалогового окна, а затем добавить в диалоговое окно элементы управления и обработчики событий для обработки. пользовательский ввод и поток управления.

Некоторые из ключевых функций класса Wizard включают в себя возможность:

* Отображения ряда страниц, которые помогут пользователю выполнить набор шагов.
* Включения кнопки для перехода между страницами, такие как кнопки «Далее» и «Назад».
* Проверки введенных пользователем данных, не позволяйте пользователю переходить на следующую страницу до тех пор, пока не будет предоставлена ​​вся необходимая информация.
* Настройки внешнего вида диалогового окна мастера, включая его заголовок, значок и цвет фона.
* Реагирования на действия и события пользователя, например, когда пользователь нажимает кнопку или выбирает параметр.

В целом, класс Wizard предоставляет удобный и удобный способ создания диалоговых окон в стиле мастера в приложении C++ Windows Form.

1. Как добавить элемент для ввода данных.

Чтобы добавить элемент для ввода данных в приложении C++ Windows Form, выполните следующие действия:

* Откройте свой проект в Microsoft Visual Studio.
* В окне обозревателя решений откройте объект формы, к которому вы хотите добавить элемент ввода данных.
* В режиме конструктора найдите окно Toolbox. Это окно содержит список элементов управления, которые вы можете добавить к объекту формы.
* В окне Toolbox найдите элемент управления, соответствующий типу элемента ввода данных, который вы хотите добавить. Например, если вы хотите добавить текстовое поле для ввода текстовых данных, вы можете выбрать элемент управления "TextBox".
* Перетащите выбранный элемент управления на объект формы. Элемент управления появится в объекте формы, и вы можете изменить его размер и положение по мере необходимости.
* В окне «Свойства» вы можете изменить различные свойства элемента управления, чтобы настроить его внешний вид и поведение. Например, вы можете изменить размер элемента управления, шрифт, цвет и значение по умолчанию.
* Если вы хотите связать данные, введенные в элемент управления, с переменной в вашем коде C++, вы можете установить для свойства «Имя» элемента управления уникальный идентификатор. Затем в коде C++ вы можете ссылаться на этот идентификатор, чтобы читать данные, введенные в элемент управления, или управлять ими.
* Сохраните изменения и запустите приложение для проверки элемента ввода данных. Когда приложение запускается, пользователь сможет взаимодействовать с элементом управления, чтобы вводить данные и отправлять их по мере необходимости.

Следуя этим шагам, вы можете легко добавить элемент ввода данных в приложение C++ Windows Form и настроить его внешний вид и поведение в соответствии с вашими потребностями.

*Практическая работа № 3*

*Создание графического интерфейса на базе диалогового окна VC++*

#pragma once

#include <windows.h>

namespace MyForms {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

// Программный код, расположенный выше, создан средой Visual Studio

// автоматически, поэтому автором не приводится

this->PerformLayout();

}

#pragma endregion

// Программа, позволяющая выбрать текст из двух вариантов, задать цвет

// и размер шрифта для этого текста на трех вкладках TabControl

// с использованием переключателей RadioButton private: System::

Void Form1\_Load(System::Object^ sender, System::EventArgs^ e)

{

// Создание третьей вкладки "программно":

auto tabPage3 = gcnew System::Windows::Forms::TabPage(); tabPage3->UseVisualStyleBackColor = true;

// Добавление третьей вкладки в существующий набор

// вкладок tabControl1:

this->tabControl1->Controls->Add(tabPage3);

// Добавление переключателей 5 и 6 на третью вкладку: tabPage3->Controls->Add(this->radioButton5); tabPage3->Controls->Add(this->radioButton6);

// Расположение переключателей 5 и 6:

this->radioButton5->Location = System::Drawing::Point(20, 15); this->radioButton6->Location = System::Drawing::Point(20, 58); this->Text = "Какая улыбка вам ближе";

// Задаем названия вкладок:

tabControl1->TabPages[0]->Text = "Текст"; tabControl1->TabPages[1]->Text = "Цвет"; tabControl1->TabPages[2]->Text = "Размер";

// Эта пара переключателей изменяет текст: radioButton1->Text =

"Восхищенная, сочувственная,\nскромно-смущенная"; radioButton2->Text = "Нежная улыбка, ехидная, бес" + "стыжая,\nподленькая, снисходительная";

// или

// radioButton2->Text = "Нежная улыбка, бесстыжая," +

// Environment::NewLine + "подленькая, снисходительная";

// Эта пара переключателей изменяет цвет текста: radioButton3->Text = "Красный";

radioButton4->Text = "Синий";

// Эта пара переключателей изменяет размет шрифта: radioButton5->Text = "11 пунктов";

radioButton6->Text = "13 пунктов"; label1->Text = radioButton1->Text;

}

private: System::Void

radioButton1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

label1->Text = radioButton1->Text;

}

private: System::Void

radioButton2\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

label1->Text = radioButton2->Text;

}

private: System::Void

radioButton3\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

label1->ForeColor = Color::Red;

}

private: System::Void

radioButton4\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

label1->ForeColor = Color::Blue;

}

private: System::Void

radioButton5\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

label1->Font = gcnew System::Drawing::

Font(label1->Font->Name, 11);

}

private: System::Void

radioButton6\_CheckedChanged(System::Object^ sender,

System::EventArgs^ e)

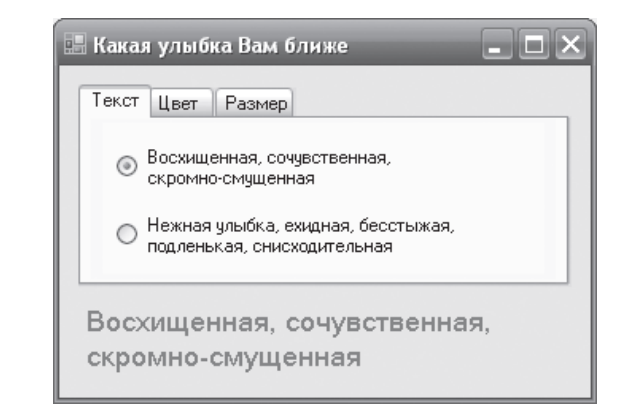
{

label1->Font = gcnew System::Drawing::Font(label1->Font->Name, 13);

}

};

}

****

**Контрольные вопросы.**

1. Опишите, как добавить в программу вкладки.

Чтобы добавить вкладки в приложение C++ Windows Forms, выполните следующие действия.

1. Откройте свой проект в Microsoft Visual Studio.
2. В окне обозревателя решений найдите объект формы, к которому вы хотите добавить вкладки.
3. В режиме конструктора найдите окно Toolbox. Это окно содержит список элементов управления, которые вы можете добавить к объекту формы.
4. В окне панели инструментов найдите элемент управления «TabControl» и перетащите его на объект формы.
5. В окне «Свойства» вы можете изменить различные свойства TabControl, чтобы настроить его внешний вид и поведение. Например, вы можете изменить размер, шрифт, цвет TabControl и добавить новые вкладки.
6. Чтобы добавить новые вкладки, щелкните свойство TabPages элемента TabControl в окне свойств, а затем нажмите кнопку с многоточием. Откроется редактор коллекции вкладок, который позволяет добавлять, удалять и изменять отдельные вкладки.
7. В редакторе коллекции вкладок нажмите кнопку «Добавить», чтобы создать новую вкладку. Затем вы можете установить свойства новой вкладки, такие как ее заголовок, значок и содержимое.
8. Повторите предыдущий шаг, чтобы добавить столько вкладок, сколько вам нужно.
9. Сохраните изменения и запустите приложение, чтобы протестировать вкладки. Когда приложение запускается, пользователь сможет переключаться между вкладками для доступа к различному контенту и функциям.

Следуя этим шагам, вы можете добавить вкладки в приложение C++ Windows Forms и настроить их в соответствии со своими потребностями. Вкладки — это полезный способ организации сложных пользовательских интерфейсов и упрощения доступа пользователей к различным частям вашего приложения и взаимодействия с ними.

1. По какому принципу работают переключатели RadioButton.

В приложении C++ Windows Forms переключатели RadioButton работают по принципу взаимоисключающего выбора. Это означает, что когда пользователь выбирает один элемент управления RadioButton, все остальные элементы управления RadioButton в той же группе автоматически отменяются.

Чтобы использовать элементы управления RadioButton в приложении, необходимо создать группу элементов управления, связанных друг с другом. Для этого вы можете использовать элемент управления GroupBox или элемент управления Panel, чтобы сгруппировать элементы управления RadioButton вместе. Затем вы можете установить для свойства «GroupName» каждого элемента управления RadioButton одно и то же значение, чтобы гарантировать, что все они являются частью одной и той же группы.

Когда пользователь взаимодействует с элементами управления RadioButton, одновременно может быть выбран только один элемент управления в группе. Это означает, что если пользователь выбирает другой элемент управления в той же группе, выбор ранее выбранного элемента управления автоматически отменяется.

Вы можете использовать событие "CheckedChanged" элемента управления RadioButton, чтобы определить, когда пользователь выбирает или отменяет выбор элемента управления. Это событие возникает всякий раз, когда пользователь изменяет состояние выделения элемента управления, щелкая его или нажимая клавишу пробела, когда элемент управления находится в фокусе.

В целом, переключатели RadioButton в приложении C++ Windows Forms предоставляют пользователям простой и интуитивно понятный способ выбора из множества взаимоисключающих параметров. Используя соответствующую группировку и обработку событий, вы можете создавать мощные и гибкие пользовательские интерфейсы, которые осмысленно реагируют на вводимые пользователем данные.

1. Опишите, как организовать работу группы переключателей.

Чтобы организовать группу переключателей в приложении C++ Windows Forms, выполните следующие действия.

1. Откройте свой проект в Microsoft Visual Studio.
2. В режиме конструктора найдите окно Toolbox. Это окно содержит список элементов управления, которые вы можете добавить к объекту формы.
3. Перетащите элемент управления GroupBox на объект формы. Этот элемент управления будет служить контейнером для переключателей.
4. В окне «Свойства» задайте для свойства «Текст» элемента управления GroupBox описательную метку для группы переключателей.
5. Перетащите элементы управления RadioButton на элемент управления GroupBox. Эти элементы управления будут представлять отдельные переключатели в группе.
6. В окне «Свойства» задайте для свойства «Текст» каждого элемента управления RadioButton метку, описывающую значение этого параметра.
7. Для каждого элемента управления RadioButton задайте для свойства «GroupName» одно и то же значение. Это гарантирует, что радиокнопки будут сгруппированы вместе и что одновременно можно будет выбрать только одну из них.
8. При желании можно установить для свойства Checked одного из элементов управления RadioButton значение True, чтобы указать выбор по умолчанию.
9. Сохраните изменения и запустите приложение, чтобы проверить группу переключателей. Когда приложение запустится, пользователь сможет выбрать один из переключателей в группе, а другие параметры будут автоматически отменены.

Следуя этим шагам, вы можете легко организовать группу переключателей в приложении C++ Windows Forms. Это полезный способ представить пользователю несколько взаимоисключающих опций, и его можно настроить разными способами в соответствии с вашими конкретными потребностями.

*Практическая работа № 4*

*Программирование консольных приложений.*

// Сумма.cpp: главный файл проекта.

// Программа организует ввод двух чисел, их сложение и вывод суммы на консоль

#include "stdafx.h" using namespace System;

int main(array<System::String^>^ args)

{

// Задаем строку заголовка консоли:

Console::Title = "Складываю два числа:"; Console::BackgroundColor = ConsoleColor::Cyan; // - цвет фона Console::ForegroundColor = ConsoleColor::Black; // - цвет текста Console::Clear();

// Ввод первого слагаемого: Console::WriteLine("Введите первое слагаемое:"); String^ Строка = Console::ReadLine();

Single X, Y, Z;

// Преобразование строковой переменной в число: X = Single::Parse(Строка);

// Ввод второго слагаемого: Console::WriteLine("Введите второе слагаемое:"); Строка = Console::ReadLine();

Y = Single::Parse(Строка); Z = X + Y;

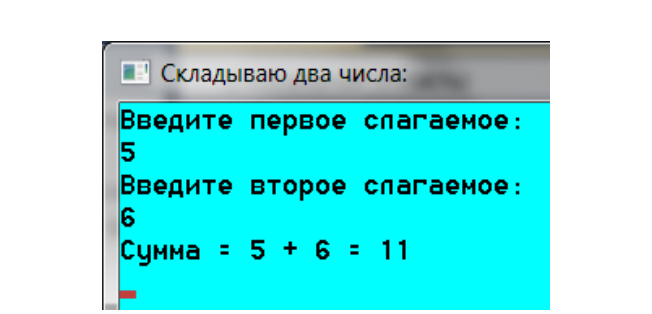
Console::WriteLine("Сумма = {0} + {1} = {2}", X, Y, Z);

// Звуковой сигнал частотой 1000 Гц и длительностью 0.5 секунды: Console::Beep(1000, 500);

// Приостановить выполнение программы до нажатия какой-нибудь клавиши:

Console::ReadKey(); return 0;

}



**Контрольные вопросы.**

1. Что такое форматированный ввод?

Форматированный ввод в C++ Windows Forms относится к способу чтения входных данных от пользователя или из файла структурированным и организованным образом. Форматированный ввод позволяет указать точный формат входных данных, включая тип данных, количество элементов для чтения и ожидаемые разделители или разделители между элементами.

В C++ Windows Forms вы можете использовать форматированные функции ввода, такие как «cin», для чтения входных данных из консоли или из файла. Синтаксис форматированного ввода подобен синтаксису форматированного вывода, который использует такие функции, как «cout», для вывода выходных данных на консоль или в файл.

Функции форматированного ввода в C++ Windows Forms принимают в качестве параметра строку формата, которая указывает ожидаемый формат входных данных. Строка формата может включать спецификаторы преобразования, такие как "%d" для целых чисел, "%f" для значений с плавающей запятой и "%s" для строк. Вы также можете включить другие символы в строку формата, чтобы указать ожидаемые разделители или разделители между элементами.

1. Какая команда помогает задерживать результат работы программы на экране?

В C++ Windows Forms вы можете использовать функцию «Сон», чтобы задержать результат работы программы на экране. Функция «Сон» — это функция Windows API, которая заставляет программу приостанавливаться на определенное время в миллисекундах.

Синтаксис использования функции «Сон» в C++ Windows Forms следующий:

#include <windows.h>

// ...

// Delay for 1000 milliseconds (1 second)

Sleep(1000);

В этом примере функция «Сон» используется для приостановки программы на 1000 миллисекунд или 1 секунду. В течение этого времени программа не будет реагировать на ввод данных пользователем или обновлять экран.

Важно отметить, что использование функции «Сон» для задержки программы, как правило, не является хорошей практикой программирования, потому что это может привести к тому, что программа перестанет отвечать на запросы и приведет к плохому взаимодействию с пользователем. Вместо этого часто лучше использовать методы программирования, управляемые событиями, такие как таймеры или фоновые потоки, для выполнения действий через определенные промежутки времени или в ответ на вводимые пользователем данные.

В целом, хотя функция «Сон» может быть полезна в определенных ситуациях, ее следует использовать с осторожностью и только тогда, когда нет другого решения. Для большинства задач программирования в C++ Windows Forms существуют лучшие и более эффективные способы обработки задержек и проблем со временем.

1. Опишите вывод результата.

Сумма = 5 + 6 = 11;

Воспользовались форматированным выводом {0}, {1} = {2} , X, Y, Z.