Федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный аграрный университет имени П.А.Столыпина»

Университетский колледж агробизнеса

ООП по специальности 09.02.07 Информационные системы и программирование

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТОВ
по дисциплине
ОП.08 Объектно-ориентированное программирование

Специальность: **09.02.07 Информационные системы и программирование**

Ведущий преподаватель (руководитель) дисциплины	А.В.Кортусов

Омск 2023

Пояснительная записка

Методические рекомендации по учебной дисциплине **ОП.08 Объектно-ориентированное программирование** предназначены для выполнения самостоятельной работы обучающимися по специальности 09.02.07 Информационные системы и программирование.

Самостоятельная работа выполняется по заданию и при методическом руководстве преподавателя, но без его непосредственного участия.

Целью самостоятельной работы является овладение обучающимся умениями работать с источниками, обобщения и анализа юридической практики, аргументации собственной точки зрения.

Методические рекомендации по самостоятельной работе студентов содержат материалы для подготовки к лекционным, практическим занятиям, к формам текущего и промежуточного контроля. Наряду с методическими рекомендациями по подготовке и написанию курсовых работ, защите квалификационных работ составляют единый комплекс методического обеспечения студента по профессиональному модулю.

Предложенные в рекомендациях задания позволят успешно овладеть профессиональными знаниями, умениями и навыками, и направлены на формирование общих и профессиональных компетенций:

ОК. 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

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

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.

При выполнении самостоятельной работы обучающийся самостоятельно осуществляет сбор, изучение, систематизацию и анализ информации, а затем оформляет информацию и представляет на оценку преподавателя или группы.

Виды самостоятельной работы

№	Вид самостоятельной работы	Форма контроля	Максимал кол-во ов
1.	Работа с источниками	Устный ответ на занятии Составление аннотации	5
2.	Составление опорного конспекта	Опорный конспект	5
3.	Составление сравнительной ицы	Сравнительная таблица	5
4.	Решение ситуационных задач	Письменный ответ	5
5.	Анализ судебной практики	Письменный отчет	5
6.	Участие в научно-едовательской деятельности*	Выступление на конференции	5

Методические рекомендации по работе с источниками

Работа с источниками осуществляется с целью приобретения обучающимся навыков самостоятельного изучения учебного материала. Работа с источниками является важной составляющей при подготовке к занятиям.

Для подготовки к устному опросу необходимо прочитать текст источника, выделить главное, составить план ответа, повторить текст несколько раз. На учебном занятии полно, точно, доступно, правильно, взаимосвязано и логично изложить материал, иллюстрируя при необходимости примерами.

Работа с источником может быть предложена в форме аннотирования. Аннотация позволяет составить обобщенное представление об источнике. Для составления аннотации необходимо ответить на следующие вопросы:

1. Фамилия автора, полное наименование работы, место и год издания.
2. Вид издания (статья, учебник, и пр.).
3. Цели и задачи издания.
4. Структура издания и краткий обзор содержания работы.
5. Основные проблемы, затронутые автором.
6. Выводы и предложения автора по решению выделенных проблем.

Источник аннотирования определяет преподаватель, он же оценивает аннотацию, сданную в письменной форме.

Методические рекомендации по составлению опорного конспекта

Опорный конспект составляется с целью обобщения, систематизации и краткого изложения информации. Составление опорного конспекта способствует более быстрому запоминанию учебного материала.

Составление опорного конспекта включает следующие действия:

1. Изучение текста учебного материала.
2. Определение главного и второстепенного в анализируемом тексте.
3. Установление логической последовательности между элементами.
4. Составление характеристики элементов учебного материала в краткой форме.
5. Выбор опорных сигналов для расстановки акцентов.
6. Оформление опорного конспекта.

Опорный конспект может быть представлен в виде схемы с использованием

стрелок для определения связи между элементами; системы геометрических фигур; логической лестницы и т.д.

Оценкой опорного конспекта может служить качество ответа, как самого студента, так и других студентов его использовавших. Преподаватель также может проверить опорные конспекты, сданные в письменной форме. Допускается проведение конкурса на самый лучший конспект по следующим критериям: краткость формы; логичность изложения; наглядность выполнения; универсальность содержания.

Методические рекомендации по составлению сравнительной таблицы

Сравнительная таблица составляется с целью выявления сходств, отличий, преимуществ и недостатков анализируемых объектов.

Критерии для составления сравнительной таблицы предлагает преподаватель. Студент, самостоятельно сформулировавший критерии для сравнения, получает дополнительные баллы.

Проверка и оценка сравнительной таблицы осуществляется преподавателем в письменной форме.

Методические рекомендации по решению ситуационных задач

Ситуационные задачи решаются с целью приобретения обучающимся навыков самостоятельной работы с источниками, обобщения и анализа юридической практики, а также умений аргументировать собственную точку зрения и делать выводы.

При решении задач студентам можно рекомендовать такую основную схему:

1) проанализировать приведенную в задаче ситуацию и поставленный вопрос;

2) найти оптимальный способ решения задачи;

4) составить в письменной форме мотивированный вывод по задаче.

Практическая работа № 1

Среда разработки программ Microsoft Visual Studio.

Цель работы: Научиться использовать Microsoft Visual Studio для разработ- ки программ на языке C++. Получить практические навыкиработы со средой визу- альной разработки программ.

Задание 1. Создание приложения

Запускаем Microsoft Visual C++ .

После запуска системы мы увидим начальный пользовательскийинтерфейс, показанный на рис. 1.1.

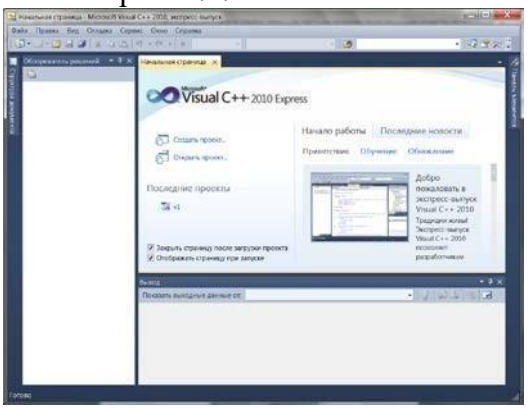

Рис. 1.1. Фрагмент стартовой страницы системы Visual Studio

Для создания приложения, необходимо в пункте меню Filевыполнить коман- ду New Project (Новый проект). В появившемся окне New Project в левой колонке находится список установленных шаблонов (Installed Templates). Среди них —шаблоны языков программирования, встроенных в Visual Studio, в том числе VisualBasic, Visual C#, Visual C++, Visual F# и др. Нам нужен язык Visual C++. В узле Visual C++ области типов проектов выберем средуCLR, а затем в области шабло- нов (в средней колонке) выберем шаблон (Templates) Windows Forms Application Visual C++.

Теперь введем имя проекта (Name) v4 и щелкнем на кнопке OK, в результате увидим окно, представленное на рис. 1.2.

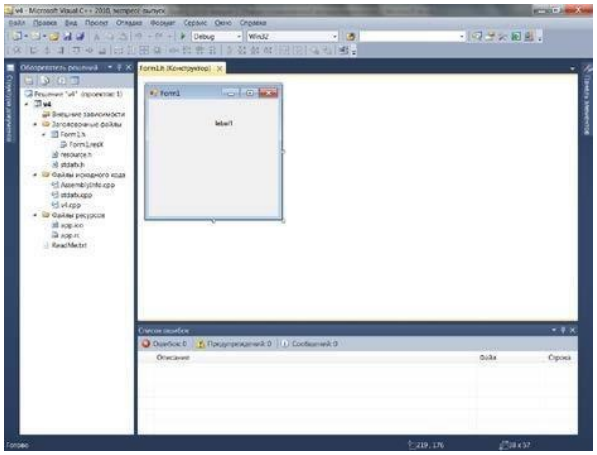

Рис. 1.2. Окно для проектирования пользовательского интерфейса

В этом окне изображена экранная форма — Form1. Первая программа будет отображать такую экранную форму, в которойбудет что-либо написано, например
«Microsoft Visual C++ 2010», также в форме будет расположена командная кнопка с надписью

«Нажми меня». При нажатии кнопки будет появляться диалоговое окно с со- общением «Всем привет!» В программе четыре объекта: форму Form, надпись на форме Label, кнопка Button и диалоговое окно MessageBox с текстом «Всем при- вет!» (окно с приветом).

Добавить в форму названные элементы управления. Для этого понадобится панель элементов управления Toolbox (Панель управления), ее можно добавить, например, с помощью комбинации клавиш Ctrl+Alt+x или ViewToolbox. Итак, до- бавьте метку Label и кнопку Button в форму, дважды щелкая на этих элементах на панели Toolbox. А затем следует рас положить их примерно так, как показано на рис. 1.3.

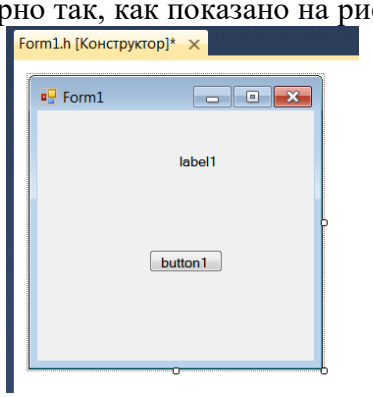

Рис. 1.3. Форма первого проекта

Каждый объект имеет свойства (properties- Свойства) . Свойств много, их можно увидеть, если щелкнуть правой кнопкой мыши в пределах формы и выбратьв контекстном меню команду Properties-Свойства, при этом появится панель свойств .

Для объекта label1 выбрать свойство Text и написать напротив этого поля «Microsoft Visual C++ 2010» (вместо текста label1). Для объекта button1 также в свойстве Text написать «Нажми меня».

Объекты не только имеют свойства, но и обрабатываются со- бытиями. В задаче событием, которым управляем, является щел- чок на командной кнопке. Для получения пустого обработчикаэтого события сле- дует в свойствах кнопки button1 щелкнуть назначке молнии Events (события) ив списке всех возможных событий кнопки button1 выбрать двойным щелчком событие Click. После этого попадаем на вклад- ку программного кода Form1.h (см.рис. 1.4).

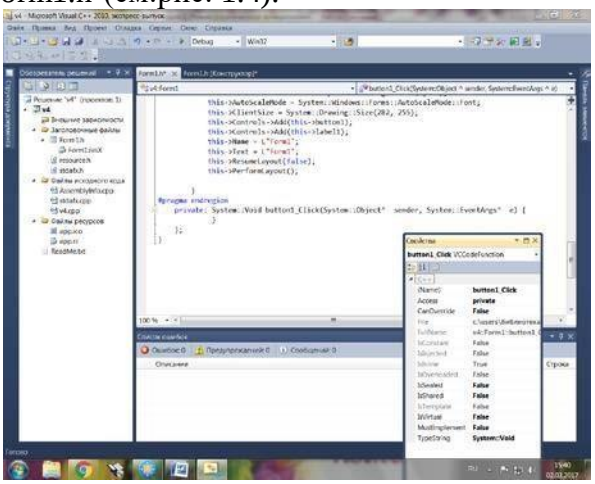

Рис 1.4 Вкладка программного кода

На вкладке Form1.h видно, что управляющая среда Visual C++ сгенерировала довольно таки много строк программного кода. В этом тексте уже можно найти те присваивания, которые сделали в панели свойств Properties. Например, для свойст- ва Text кнопки Button управляющая среда назначила строку ≪Нажми меня≫:

this->button1->Text = L"Нажми меня"**;**

Пустой обработчик события button1_Click:

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) { }

Здесь в фигурных скобках пишутся команды, подлежащие выполнению по-сле щелчка на кнопке. В фигурных скобках обработчика события напишите:

MessageBox::Show("Всем привет!");

Теперь нажмите клавишу F5 и проверьте работоспособность программы (рис. 1.5).

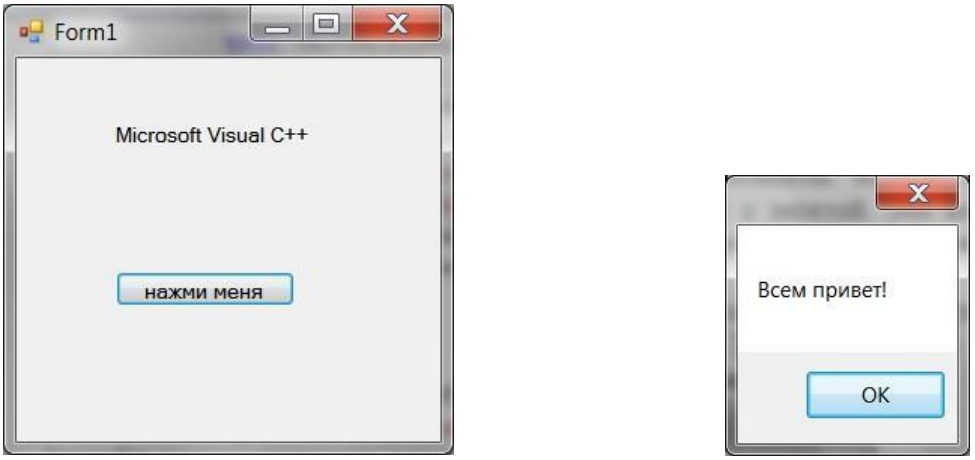

Рис. 1.5. Фрагмент работы программы

Задание 2. Обработка события MouseHover мыши

Событие MouseHover наступает тогда, когда пользователь указателем мыши «зависает» над каким-либо объектом, событие MouseHover происходит, когда ука- затель мыши наведен на элемент.

Таким образом, программа в данном примере должна содержать на экранной форме текстовую метку Label и кнопку Button. Метка должна отображать текст «Microsoft Visual C++ 2010»; при щелчке на командной кнопке, на которой по- прежнему будет написано «Нажми меня», появится диалоговое окно ссообщением «Всем привет!» Кроме того, когда указатель мышинаведен на текстовую метку (то самое событие MouseHover), должно появиться диалоговое окно с текстом «Собы- тие Hover».

Для решения этой задачи запустим Visual Studio 2010, щелкнем на пункте меню New Project. В появившемся окне New Project в левой колонке в узле Visual C++ выберем среду CLR, а затем в области шаблоны (в средней колонке) выберем шаблон (Templates) Windows Forms Application Visual C++. В качестве имени про- екта введем имя Hover и щелкнем на кнопке OK.

В дизайнере формы из панели Toolbox перетащим на форму метку Label и кнопку Button, а затем немного уменьшим размеры формы на свое усмотрение. Те- перь добавим три обработчика событий в программный код. Для этого в панели Properties следует щелкнуть на значке молнии (Events) и двойным щелчком после- довательно выбрать событие загрузки формы Form_Load, событие «щелчок на кнопке button1_Click» и событие label1_MouseHover.

При этом осуществится переход на вкладку программного кода Form1.h, и среда Visual Studio 2010 сгенерирует три пустых

обработчика события (рис.1.6). Например, обработчик последнегособытиябудет иметь вид:

private:System::Void lab el1_MouseHover(System::Object^ sender, System::EventArgs^ e){}

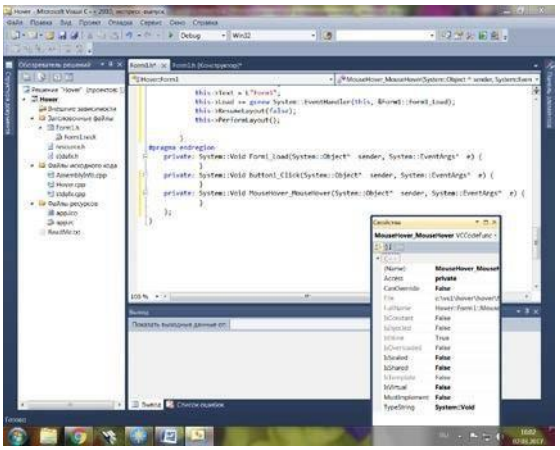

Рис. 1.6 Вкладка программного кода

Между фигурными скобками вставим вызов диалогового окна: MessageBox::Show("Событие Hover!");

Теперь проверим возможности программы: нажимаем клавишу F5, «зависаем» указателем мыши над label1, щелкаем на кнопке button1. Все работает! (рис.1.7)

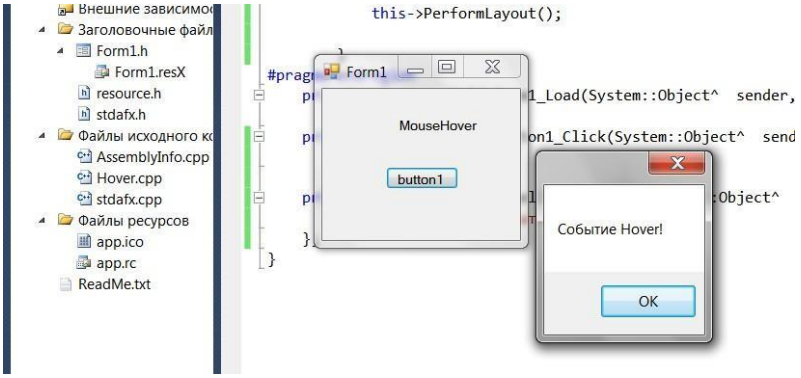

Рис.1.7 Работа приложения

Листинг. Фрагмент файла Form1.h, содержащего программный код с тремя обработчиками событий

```
// ..............
// Программный код, расположенный выше, создан средой Visual Studio
// автоматически, поэтому автором не приводитсяthis->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
```

```
     // Данная программа управляется тремя событиями. Событиеза-
грузки формы
     // Form1_Load инициализирует надписи заголовка формы, текстовой
метки
     // и кнопки. Событие щелчок на кнопке button1_Click вызывает
появление
     // диалогового окна с текстом "Всем привет!". Событие, когда
указатель
     // мыши наведен на метку, вызывает появление диалогового окна с
текстом
     // "Событие Hover".
     private: System::Void Form1_Load(System::Object^ sender,Sys-
tem::EventArgs^ e)
     { // Обработка события загрузки формы:
     this->Text = "Приветствие";
     // или Form1::Text = "Приветствие";
     label1->Text = "Microsoft Visual C++ 2010";button1->Text = "На-
жми меня";
     }
     private: System::Void button1_Click(System::Object^ sender,Sys-
tem::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++.
2. Что такое программа, основанная на диалоге.
3. Что такое Windows Forms Application.

Практическая работа № 2 Реализация простейшей программы на VC++, с использованием интерфейса и библиотеки CLR.

Цель работы: Научиться разрабатывать и реализовывать простейшие про- граммы на языке VC++. Получить практическиенавыки работы по использованию интерфейса CLR. Научиться связывать переменные и методы с элементами диало- гового окна.

Задание.Ввод данных через текстовое поле

TextBox с проверкой типа методом TryParse При работе с формой очень час- то ввод данных организуют через элемент управлениятекстовое поле TextBox . На- пишем типичную программу, которая вводит через текстовое поле число, при на- жатии командной кнопкиизвлекает из него квадратный корень и выводит результатна метку Label. В случае ввода не числа сообщает пользователю об этом.

Решая сформулированную задачу, запускаем Visual Studio, выбираем пункт меню File-New -Project. В окне New Project в узле Visual C++ выберем среду CLR, а затем в области шаблоны выберем шаблон (Templates) Windows Forms Application Visual C++. В качестве имени проекта введем имя Корень и щелкнем на кнопкеOK.

Далее из панели элементов управления Toolbox (если в данный момент вы не видите панель элементов управления, то ее можно добавить, например, с помо- щью комбинации клавиш Ctrl+Alt+x или меню View - oolbox) в форму с помощью указателя мыши перетаскиваем текстовое поле TextBox, метку Label и командную кнопку Button. Получить названные элементы на проектируемой экранной форме можно, также дважды щелкая указателем мыши на каждом элементе в панели Tools. Таким образом, в форме будут находиться три элемента управления. Распо- ложим их на экранной форме.

Теперь следует изменить некоторые свойства элементов управления. Чтобы получить пустой обработчик загрузки формы, дважды щелкнем по проектируемой экранной форме. Сразу после этого мы попадаем на вкладку программного кода Form1.h. Здесь задаем свойствам формы (к форме обращаемся посредством ссылки this), кнопкам button1 и текстового поля textBox1, метке label1 следующие значе- ния:

this->Text = "Извлечение квадратного корня"; button1->Text = "Извлечь ко- рень";

textBox1->Clear(); // Очистка текстового поля label1->Text = nullptr; // или = String::Empty;

Нажмем клавишу F5 для выявления возможных опечаток, то есть синтакси- ческих ошибок и предварительного просмотра дизайна будущей программы (рис.2.1).

Далее программируем событие button1_Click — «щелчок мышью на кнопке Извлечь корень». Создать пустой обработчикэтого события удобно, дважды щелк- нув мышью на этой кнопке. Между двумя появившимися строчками программиру- емдиагностику правильности вводимых данных, конвертирование строковой пере- менной в переменную типа Single и непосредственное извлечение корня (листинг). **Листинг.** Фрагмент программы извлечения корня с проверкой типаметодом
TryParse

```
     // ..............
     // Программный код, расположенный выше, создан средой Visual
Studio
     // автоматически, поэтому автором не приводитсяthis-
>ResumeLayout(false);
     this->PerformLayout();
     }
     #pragma endregion
     // Программа вводит через текстовое поле число, при щелчке на
командной
     // кнопке извлекает из него квадратный корень и выводитрезуль-
тат
     // на метку label1. В случае ввода не числа сообщаетпользова-
телю об
     // этом, выводя красным цветом предупреждение также на метку
label1.
     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 "Извлечько-
рень":
     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 ипро-
чее).
     // 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 Xif (Число_ли == false)
     { // Если пользователь ввел не число:
     label1->Text = "Следует вводить числа";
     label1->ForeColor = Color::Red; // - цвет текста на меткеreturn;
// - выход из процедуры
     }
     Single Y = (Single)Math::Sqrt(X); // - извлечение корня label1-
>ForeColor = Color::Black; // - черный цвет текстана метке
     label1->Text = String::Format("Корень из {0} равен {1:F5}", X,
Y);
     }
     };
     }
```

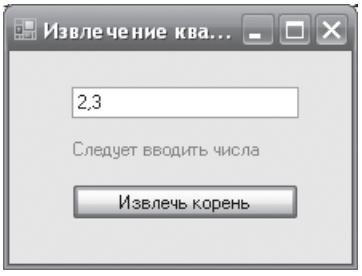

Рис.2.1 Фрагмент работы программы

Если пользователь ввел все-таки число, то будет выполняться следующий оператор извлечения квадратного корня Math::Sqrt(X). Математические функции Visual Studio являются методами класса Math. Их можно увидеть, набрав Math и введя так называемый *оператор разрешения области действия* (::). В раскрываю- щемся списке вы увидите множество математических функций: Abs, Sin, Cos, Min и т. д. и два свойства — две константы E = 2,718... (основание натуральных лога- рифмов) и PI = 3,14... (число диаметров, уложенных вдоль окружности). Функция Math::Sqrt(X)возвращает значение типа double (двойной точности с плавающей за- пятой), которое мы *приводим* с помощью неявного преобразования (Single) к пере- менной одинарной точности.

Последней строчкой обработки события button1_Click является формирова- ние строки label1->Text с использованием метода String::Format. Использованный формат ≪Корень из {0} равен

{1:F5}≫ означает: взять нулевой выводимый элемент, то есть переменную X, и записать эту переменную вместо фигурных скобок; после чего взять первый выводимый элемент, то есть Y, и

записать его вместо вторых фигурных скобок в формате с фиксированной точкой и пятью десятичными знаками после запятой.

Нажав клавишу F5, проверяем, как работает программа. Результат работаю- щей программы представлен на рис. 2.2.

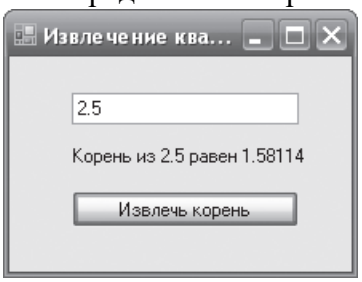

Рис.2.2. Извлечение квадратного корня

Контрольные вопросы

1. Опишите шаги, которые вы сделали, чтобы открыть диалоговуюпанель программы для ее визуальной настройки.
2. Что такое Class Wizard.
3. Как добавить элемент для ввода данных.

Практическая работа № 3

Создание графического интерфейса на базе диалогового окна VC++

Цель работы: Научиться разрабатывать и реализовывать программу, ис- пользующую вкладки и переключатели, изменять шрифт вкладок.

Разработать программу, позволяющую выбрать текст из двух вариантов, за- дать цвет и размер шрифта этого текста на трех вкладках TabControl с использова- нием переключателей RadioButton.

Программируя поставленную задачу, запустим Visual Studio и выберем при- ложение в среде CLR шаблона Windows Forms Application Visual C++. Назовем этот проект Вкладки. Используя панель элементов Toolbox, в форму перетащим с помощью мыши элемент управления TabControl. Как видно, по умолчанию имеем две вкладки, а по условию задачи, как показано на рисунке, три вкладки. Добавить третью вкладку можно в конструкторе формы, а можно программно (рис.3.1).

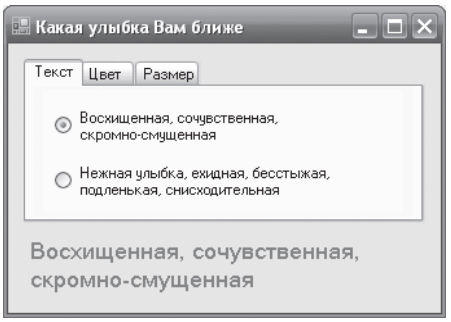

Рис.3.1 Программа с переключателями и вкладками

Чтобы добавить третью вкладку в конструкторе, необходимо всвойствах (ок- но Properties) элемента управления tabControl1 выбрать свойство TabPages, в ре- зультате попадаем в диалоговоеокно TabPage Collection Edit, где добавляем (кноп- ка Add) третью вкладку (первые две присутствуют по умолчанию). Эти вкладки нумеруются от нуля, то есть третья вкладка будет распознаваться как TabPages(2). Название каждой вкладки будем указывать в программном коде.

Рассмотрим, как добавить третью вкладку не в конструкторе, а в программ- ном коде при обработке события загрузки формы (листинг). Однако прежде чем перейти на вкладку программного кода, для каждой вкладки выбираем из панели Toolbox по два переключателя RadioButton, а в форму перетаскиваем метку Label. Теперь с помощью щелчка правой кнопкой мыши в пределах формы переключа- емся на редактирование программного кода.

Листинг Фрагмент программы, управляющей вкладками и переключателями

```
     // ..............
     // Программный код, расположенный выше, создан средой Visual
Studio
     // автоматически, поэтому автором не приводитсяthis-
>ResumeLayout(false);
     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 RadioButtonprivate: 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, снисхо-
дительная";
     // 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,Sys-
tem::EventArgs^ e)
     { label1->Text = radioButton1->Text; }
     private:                                                System::Void
     radioButton2_CheckedChanged(System::Object^ sender,Sys-
tem::EventArgs^ e)
     { label1->Text = radioButton2->Text; }
     private:                                                System::Void
     radioButton3_CheckedChanged(System::Object^ sender,Sys-
tem::EventArgs^ e)
     { label1->ForeColor = Color::Red; }
     private:                                                System::Void
     radioButton4_CheckedChanged(System::Object^ sender,Sys-
tem::EventArgs^ e)
     { label1->ForeColor = Color::Blue; }
     private:                                                System::Void
     radioButton5_CheckedChanged(System::Object^ sender,Sys-
tem::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); }
     };
     }
```

Как видно из текста программы, при обработке события

загрузки формы Form1_Load (этот участок программного кода можно было бы задать сразу после вызова процедуры InitializeComponent) создаем «программ- но» третью вкладку. Заметьте, что мы ее объявили как auto, то есть тип переменнойtabPage3 выводится из выражения инициализации в Visual C++. Далее добавляем новую вкладку tabPage3 в набор вкладок tabControl1, созданный в конструкторе. Затем «привязываем» пятый и шестой переключатели к третьей вкладке.

Каждая пара переключателей, расположенных на каком-либо элементе управления (в данном случае на различных вкладках),

«отрицают» друг друга, то есть если пользователь выбрал один, то другой переходит в противоположное состояние. Отслеживать изменения состояния пере- ключателей удобно с помощью обработки событий переключателей CheckChanged (см. листинг). Чтобы получить пустой обработчик этого события в конструкторе формы, следует дважды щелкнуть на соответствующем переключателе и таким об- разом запрограммировать изменения состояния переключателей.

Контрольные вопросы

1. Опишите, как добавить в программу вкладки.
2. По какому принципу работают переключатели RadioButton.
3. Опишите, как организовать работу группы переключателей.

Практическая работа № 4 Программирование консольныхприложений.

Цель работы: Научиться программировать консольное приложение, для вы- числения математических функций, вводимых значений и вывода результата на эк- ран. Получить практические навыки по использованию различных элементов гра- фического интерфейса и операторов языка VC++.

Иногда, например для научных расчетов, требуется организовать какой- нибудь самый простой ввод данных, выполнить весьма сложную математическую обработку введенных данных и оперативно вывести на экран результат вычисле- ний.

Можно по-разному организовать такую программу, в том числепрограмми- руя так называемое *консольное приложение* (от англ. *console* — пульт управления). Под консолью обычно подразумевают экран компьютера и клавиатуру.

Напишем консольное приложение, которое приглашает пользователя ввести два числа, складывает их и выводит результат вычислений на консоль. Для этого запускаем Visual C++ 2010, далее создаем новый проект (New Project), в узле Visual C++ в среде CLR выбираем шаблон Console Application CLR, задаем имя решения (Name) — Сумма. После щелчка на кнопке OK попадаем сразу на вкладку программного кода (рис. 4.1).

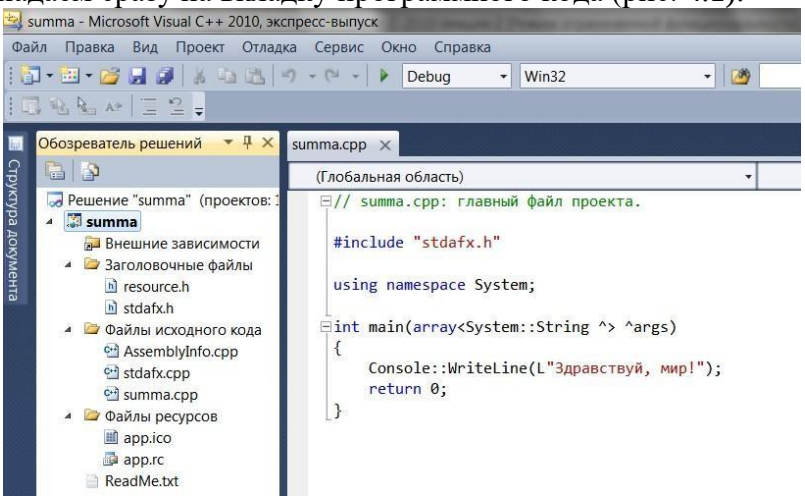

Рис. 4.1. Вкладка программного кода

Как видите, здесь управляющая среда Visual Studio приготовила несколько строк программного кода. Это вполне работоспособная программа. При запуске консольного или Windows-приложения C++ метод Main() является первым вызы- ваемым методом. В фигурные скобки после Main() мы вставим собственный про- граммный код (листинг). Фрагмент работы программы на рисунке 4.2.

Листинг Ввод и вывод данных в консольном приложении

```
// 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:"; Con-
sole::BackgroundColor = ConsoleColor::Cyan; // - цвет фона Con-
sole::ForegroundColor = ConsoleColor::Black; // - цвет текстаCon-
sole::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;
}
```

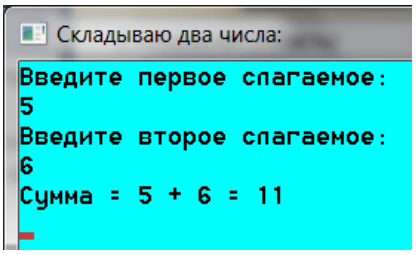

Рис. 4.2. Фрагмент работы консольного приложения

Итак, в данной программе Main() — это стартовая точка, с которой начинает- ся ее выполнение. Обычно консольноеприложение выполняется в окне на черном фоне. Чтобы как-то украсить традиционно черное окно консольного приложения, установим цвет фона окна BackgroundColor сине-зеленым (Cyan), а цвет символов, выводимых на консоль, черным (Black). Выводим строки в окно консоли методом WriteLine, а для считывания строки символов, вводимых пользователем, использу- ем метод ReadLine.Далее объявляем три переменных типа Single для соответствен- но первого числа, второго и значения суммы. Тип данных Single применяется тогда, когда число, записанное в переменную, может иметь це- лую и дробную части.

Переменная типа Single занимает 4 байта. Для преобразованиястроки симво- лов, введенных пользователем в числовое значение, используем метод Parse.

После вычисления суммы необходимо вывести результат вычислений из оперативной памяти на экран. Для этоговоспользуемся форматированным выводомв фигурных скобках метода WriteLine объекта Console:

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

Затем выдаем звуковой сигнал Beep, символизирующий об окончании про- цедуры и выводе на экран результатов вычислений. Последняя строка в программе Console::ReadKey(); предназначена для приостановки выполнения программы до нажатия какой- нибудь клавиши. Если не добавить эту строку, окно с командной строкой сразу исчезнет, и пользователь не сможет увидеть вывод результатов вы- полнения. Программа написана. Нажмите клавишу F5, чтобы увидеть результат.

Контрольные вопросы

1. Что такое форматированный ввод?
2. Какая команда помогает задерживать результат работы программы на экране?
3. Опишите вывод результата.

Практическая работа № 5 Инициирование и обработка событий Цель работы: Научиться создавать элементы управления в форме

«программным» способом, обрабатывать несколько событий одной процеду- рой. Получить практические навыки по использованию различных элементов гра- фического интерфейса и операторов языкаVC++.

1. Co

здать новый проект с формой. При этом, как обычно, запускаем Visual Studio 2010, в окне New Project выбираем в среде CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Чтобы к программному коду добавить пустой обработчик события загрузки формы, дважды щелкнем на проектируемой экранной форме. Далее вводим программный код, представленный в листинге :

```
// ..............
// Программный код, расположенный выше, создан средой Visual Studio
// автоматически, поэтому автором не приводится
this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
this->ResumeLayout(false);
}
#pragma endregion
// Программа создает командную кнопку в форме «программным» способом,
// т.е. с помощью написания непосредственно программного кода, не
// используя при этом панель элементов управления Toolbox. Программа
// задает свойства кнопки: ее видимость, размеры, положение, надпись
// на кнопке и подключает событие "щелчок на кнопке"
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
// Создание кнопки без панели элементов управления:Button^ button1 = gcnew Button();
// Задаем свойства кнопки:
button1->Visible = true;
// Ширина и высота кнопки:
button1->Size = Drawing::Size(100, 30);
// Расположение кнопки в системе координат формы:button1->Location = Drawing::Point(100, 80); button1->Text = "Новая кнопка";
// Добавление кнопки в коллекцию элементов управлениятhis->Controls->Add(button1);
// Подписку на событие Click для кнопки можно делать "вручную".
// Связываем событие Click с процедурой обработки этого события:
button1->Click += gcnew EventHandler(this, &Form1::ЩелчокНаКнопке);
}
private: System::
Void ЩелчокНаКнопке(System::Object^ sender, System::EventArgs^
```

```
e)
        {
        MessageBox::Show("Нажата новая кнопка");
        }
        };}
```

Мы видим, что при обработке события загрузки формы создаем новый объект button1 стандартного класса кнопок. Задаем свойства кнопки: ее видимость (Visible), размеры (Size), положение (Location) относительнолевого нижнего угла формы, надпись на кнопке — ≪Новая кнопка≫.

Далее необходимо организовать корректную работу с событием «щелчок на созданной нами командной кнопке». В предыдущих примерах мы для этой цели в конструкторе формы дважды щелкали на проектируемой кнопке, и исполняемая среда автоматически генерировала пустой обработчик этого события в программ- ном коде. Или опять же в конструкторе формы в панели свойств проектируемой кнопки щелкали мышью на значке молнии (Events) и в появившемся списке всех событий выбирали необходимое событие. Однако согласно условию задачи мы должны организовать обработку события «программным» способом без использования конструктора формы. Для этого в программном коде сразу после добавления командной кнопки вколлекцию элементов управления поставим оператор стрелки (->) после имени кнопки button1 и в раскрывающемся списке выберем необходимоесобытие Click. Затем, как приведено в листинге 3.2, осуществляем так называемую «подписку» на данное событие, то есть с помощью ключевого слова EventHandler связываем событие Click с процедурой обработки события. Мы его назвали ≪Щелчок- НаКнопке≫. Теперь создадим обработчик события «щелчок на кноп- ке», как показано в листинге. В этой процедуре предусматриваем вывод сообщения ≪Нажата новая кнопка≫. На рисунке 5.1 приведен фрагмент работы программы.

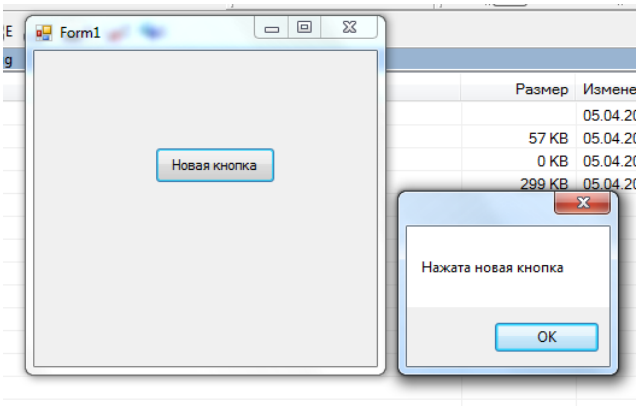

Рис.5.1 Фрагмент работы программы

В заключение отметим, что в случае создания пустого обработчика события в конструкторе формы строка подписки на событие формируется автоматически в методе InitializeComponent вфайле Form1.h проекта.

2. За

пустить Visual Studio 2010, в окне New Project выберем в среде

CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Затем из панели элементов перенесем вформу две командные кнопки и тек- стовую метку. Далее через двойной щелчок мышью в пределах проектируемой формы создать пустой обработчик загрузки формы и перейдем к вкладке про-граммного кода (листинг).

```
// .............
// Программный код, расположенный выше, создан средой Visual Studio
// автоматически, поэтому автором не приводитсяthis->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
// В форме имеем две командные кнопки, и при нажатии указателем мыши
// любой из них получаем номер нажатой кнопки. При этом в программе
// предусмотрена только одна процедура обработки событий
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
Form1::Text = "Щелкните на кнопке"; label1->Text = nullptr;
// Связываем события Click от обеих кнопок с одной процедурой КЛИК:
button1->Click += gcnew EventHandler(this, &Form1::КЛИК);button2->Click += gcnew EventHandler(this, &Form1::КЛИК);
// Подпиской на событие называют связывание названия события
// с названием процедуры обработки события посредством EventHandler
}
private: System::Void КЛИК(System::Object^ sender, System::EventArgs^ e)
{
// String S = Convert.ToString(sender);
// получить текст, отображаемый на кнопке, можно таким образом:
Button^ Кнопка = (Button^)sender;
// или String^ НадписьНаКнопке = ((Button^)sender)->Text;label1->Text = "Нажата кнопка " + Кнопка->Text; // или Кнопка->Name
}
};}
```

Как видно из текста программы, при обработке события загрузки формы мы осуществляем так называемую подписку на событие, то есть связываем название события с названием процедуры обработки события КЛИК посредством метода (делегата) EventHandler. Этот метод делегирует (передает полномочия) обработку события button1->Click процедуре КЛИК. Заметим, что события Click от обеих кнопок мы связали с одной и той же процедурой КЛИК. Далее создаем процедуру обработки события КЛИК, ее параметр sender содержит *ссылку на объект-источник события,* то есть кнопку,

нажатую пользователем.

С помощью неявного преобразования можно конвертировать параметр sender в экземпляр класса Button и, таким образом, выяснить все свойства кнопки, которая инициировала событие. На рисунке 5.2 приведен пример работы написанной про- граммы.

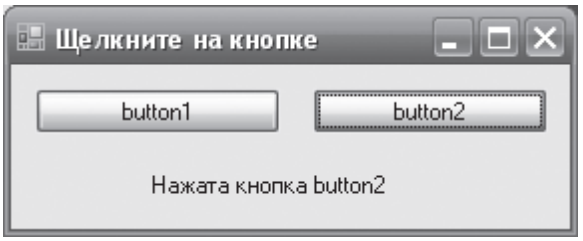

Рис.5.2 Фрагмент работы программы, определяющей нажатую кнопку

Контрольные вопросы

1. Опишите создание кнопок Button.
2. Опишите свойства Size, Point.
3. Что такое обработчик событий?
4. Опишите процедуру gcnew EventHandler.
5. Для чего нужен параметр sender?
6. Опишите связывание событий.

Практическая работа № 6.

Обработка событий клавиатуры.

Цель работы: Научиться создавать приложение для обработки событий кла- виатуры. Получить практические навыки по использованию различных элементов графического интерфейса иоператоров языка VC++.

Напишем программу, информирующую пользователя о тех клавишах и ком- бинациях клавиш, которые тот нажал. Запустим Visual Studio 2010, в окне New Project выберем в среде CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Затем из панели Toolbox перетащим в форму две текстовых метки Label.

Далее, поскольку нам потребуются клавишные события формы: KeyPress, KeyDown, KeyUp, уже привычным способом получим пустые обработчики этих событий.

То есть в панели Properties щелкнем на пиктограмме молнии (Events), а затем в списке всех возможных событий выберем каждоеиз названных событий клавиа- туры.

Программный код приведен **в листинге** :

```
// ..............
// Программный код, расположенный выше, создан средой Visual
Studio автоматически
this->ResumeLayout(false);this->PerformLayout();
}
#pragma endregion
// Программа, информирующая пользователя о тех клавишах
// и комбинациях клавиш, которые тот нажалprivate: System::
Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
// Устанавливаем шрифт с фиксированной шириной (моноширинный):
Form1::Font = gcnew Drawing:: Font(FontFamily::GenericMonospace,
14.0F);
// Поскольку мы задали этот шрифт увеличенным (от 8 по умолчанию
// до 14), форма окажется пропорционально увеличеннойForm1::Text
= "Какие клавиши нажаты сейчас:";
label1->Text = String::Empty; label2->Text = String::Empty;
}
private: System::Void Form1_KeyPress(System::Object^ sender,Sys-
tem::Windows::Forms::KeyPressEventArgs^ e)
```

```
    {
    // Здесь событие нажатия клавиши: при удержании
    // клавиши генерируется непрерывно
    label1->Text = "Нажатая клавиша: " + e->KeyChar;
    }
    private: System::Void Form1_KeyDown(System::Object^ sender,Sys-
tem::Windows::Forms::KeyEventArgs^ e)
    {
    // Здесь обрабатываем мгновенное событие первоначального
    // нажатия клавиши
    label2->Text = String::Empty;
    // Если нажата клавиша Alt
    if (e->Alt == true) label2->Text += "Alt: Yes\n";else label2-
>Text += "Alt: No\n";
    // Если нажата клавиша Shift
    if (e->Shift == true) label2->Text += "Shift: Yes\n";else la-
bel2->Text += "Shift: No\n";
    // Если нажата клавиша Ctrl
    if (e->Control == true) label2->Text += "Ctrl: Yes\n";
    else label2->Text += "Ctrl: No\n";label2->Text +=
String::Format(
    "Код клавиши: {0} \nKeyData: {1} \nKeyValue: {2}",e->KeyCode, e-
>KeyData, e->KeyValue);
    }
    private: System::Void Form1_KeyUp(System::Object^ sender,Sys-
tem::Windows::Forms::KeyEventArgs^ e)
    {
    // Очистка меток при освобождении клавиши
    label1->Text = String::Empty; label2->Text = String::Empty;
    }
    };
    }
```

В первую метку label1 записываем сведения о нажатой обычной (то есть не модифицирующей и не функциональной) клавише при обработке события KeyPress.

Во вторую метку из аргумента события e (e->Alt, e->Shift и e->Control) получаем сведения, была ли нажата какая-либо модифицирующая клавиша (либо их комбинация). Обработчик события KeyUp очищает обе метки при освобождении клавиш.

На рисунке 6.1 приведен фрагмент работы программы.

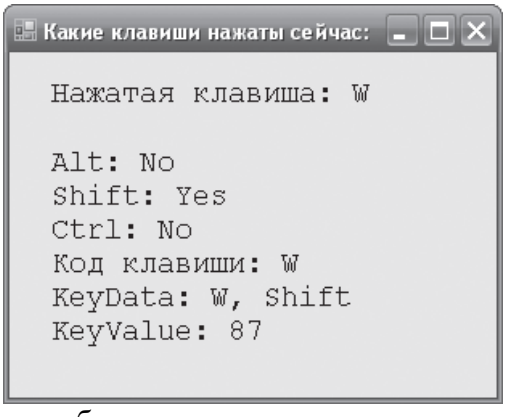

Рис. 6.1 Фрагмент работы программы, определяющей нажатую клавишу

Контрольные вопросы

1. Что означает модифицирующая клавиша?
2. Какое событие необходимо обработать, чтобы узнать нажатали модифицирующая клавиша?
3. В какой момент нажатия клавиши генерируется событие KeyDown, KeyUp?

Практическая работа № 7.

Работа с файлами в VC++. Использование файлов для хранения данных.

Цель работы: Изучить приемы работы с файлами и способы создания фай- лов. Получить практические навыки по использованию файлов для хранения ин- формации.

Напишем программу, содержащую на экранной форме текстовое поле и две командные кнопки. При щелчке мышью на первой кнопке происходит чтение тек- стового файла в текстовоеполе в кодировке Unicode. При щелчке на второй кнопке отредактированный пользователем текст в текстовом поле сохраняется в файл на диске.

Запустим систему Visual Studio 2010 и в окне New Project выберемв среде CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Далее в форму из панели Toolbox перенесемтекстовое поле и две командные копки. Для текстового поля в окнеProperties сразу укажем для свойства Multiline значение True, чтобытекстовое поле имело не одну строку, а столько, сколько поместится в растянутом указателем мыши поле. Одна кнопка предназначена для открытия фай- ла, а другая — для сохраненияфайла на машинном носителе. В листинге приведен текст данной программы «Чтение/запись текстового файла в кодировке Unicode».

Листинг:

```
     // ..............
     // Программный код, расположенный выше, создан средой Visual
Studio
     // автоматически, поэтому автором не приводитсяthis-
>ResumeLayout(false);
     this->PerformLayout();
     }
     #pragma endregion
     // Программа для чтения/записи текстового файла в кодировке
Unicode
     String ^ filename;
     // Объявляем filename здесь, чтобы эта переменная была "видна"
     // в процедурах обработки обоих событий.
     private: System::Void Form1_Load(System::Object^ sender,Sys-
tem::EventArgs^ e)
     {
     // Установка начальных значений:
     textBox1->Multiline = true; textBox1->Clear();textBox1->Size =
Drawing::Size(268, 112);
     button1->Text = "Открыть"; button1->TabIndex = 0;button2->Text =
"Сохранить";
```

```
     Form1::Text = "Здесь кодировка Unicode";filename =
"C:\\Text1.txt";
     }
     private: System::Void button1_Click(System::Object^ sender,Sys-
tem::EventArgs^ e)
     {
     // Щелчок на кнопке Открыть.
     // Русские буквы будут корректно читаться,
     // если открыть файл в кодировке UNICODE:
     try
     {
     // Создание объекта StreamReader для чтения из файла:auto Чита-
тель = gcnew IO::StreamReader(filename);
     // Непосредственное чтение всего файла в текстовое поле:
textBox1->Text = Читатель->ReadToEnd();
     Читатель->Close(); // закрытие файла
     // Читать текстовый файл в кодировке UNICODE в массив строк
     // можно также таким образом (без Open и Close):
     // array <String^>^ МассивСтрок =
     // IO::File::ReadAllLines("C:\\Text1.txt");
     }
     catch (IO::FileNotFoundException^ Ситуация)
     { // Обработка исключительной ситуации:
MessageBox::Show(Ситуация->Message + «\nНет такого файла»,"Ошибка",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
     }
     catch (Exception^ Ситуация)
     {
     // Отчет о других ошибках: MessageBox::Show(Ситуация->Message,
"Ошибка",
     MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
     }
     }
     private: System::Void button2_Click(System::Object^ sender,Sys-
tem::EventArgs^ e)
     {
     // Щелчок на кнопке Сохранить:
     try
     {
     // Создание объекта StreamWriter для записи в файл:auto Писатель
= gcnew
     IO::StreamWriter(filename, false);Писатель->Write(textBox1-
>Text); Писатель->Close();
     // Сохранить текстовый файл можно также таким образом
     // (без Close), причем, если файл уже существует,
     // то он будет заменен:
     // IO::File::WriteAllText("C:\\tmp.tmp", textBox1->Text);
     }
     catch (Exception^ Ситуация)
     {
     // Отчет обо всех возможных ошибках:
```

```
    MessageBox::Show(Ситуация->Message, "Ошибка",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
    }
    }
    };
    }
```

Блоки try, которые, как мы видим, используются в данном программном коде. Логика использования try следующая: *попытаться* (try) выполнить некоторую задачу, например прочитать файл. Если задача решена не- корректно (например, файл не найден), то *«перехватить»* (catch) управление и *об- работать* возникшую (*исключительную*, Exception) ситуацию.

При обработке события «щелчок на кнопке Открыть» организован ввод фай- ла C:\Text1.txt. Обычно в этой ситуации пользуются элементом управления OpenFileDialog для выборафайла. Мы не стали использовать этот элемент управле- ния для того, чтобы *не «заговорить»* проблему, а также свести к минимуму про- граммный код.

Далее создаем объект (поток) Читатель для чтения из файла. Для большей выразительности операций с данным объектом мы назвали его русскими буквами.

При обработке события «щелчок на кнопке Сохранить» организована запись файла на диск аналогично через объект Писатель. При создании объекта Писатель первым аргументом является filename, а второй аргумент false указывает, что дан- ные следует *не добавить* (append) к содержимому файла (если он уже существует), а *перезаписать* (overwrite). Запись на диск производится с помощью метода Write() из свойства Text элемента управления textBox1. На рисунке 7.1 приведен фрагмент работы программы.

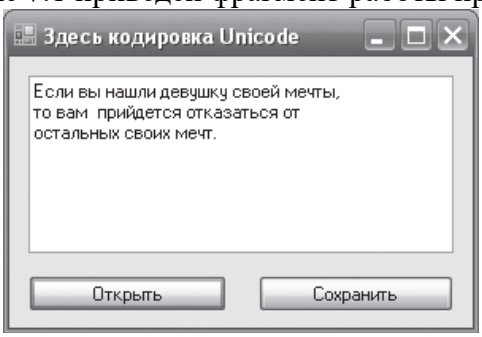

Рис.7.1 Чтение/запись текстового файла в кодировке Unicode

Запись текстового файла с помощью данной программы будет про- исходить *в формате (кодировке) Unicode*, как и чтение из файла. То есть вы сможете читать эти файлы Блокнотом, редактировать их, но каждый раз при сохранении файлов следить, чтобы кодировка была (оставалась)Unicode.

Контрольные вопросы

К чему свелась обработка исключительной ситуации ?Каким методом проис-ходит чтение файла filename?

Каким методом происходит закрытие файла?

Практическая работа № 8.

Создание меню в Windows Application.

Цель работы: Научиться создавать меню, передавать значения между диа- логовыми окнами и главным окном приложения. Получить практические навыки в разработке программ.

1. Создайте приложение с помощью Visual Studio 2010, в окне New Project выберте в среде CLR узла Visual C++ приложение шаблона Windows Forms Application

Для создания меню на панели инструментов выберите *MenuStrip* . Дважды кликните на появившемся в нижней области окна объекте, а затем перейдите на форму и в области *(Вводить здесь)* введите меню верхнего уровня с текстом *Цвет*.

Переместитесь на нижнюю область и введите текст *Черный.*

Заполните элемент MenuStrip следующим образом:

Цвет Черный КрасныйСиний Зеленый

Запустите программу и поэкспериментируйте: выбирайте разные элементы созданного объекта.

Запрограммируйте событие *Click* для каждого пункта; например, для элемен- та *Черный* необходимо написать следующий код (дважды щелкнув на пункте, что- бы открыть код):

```
private: System::Void черныйToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
this->txt->BackColor=System::Drawing::Color::Black;
}
```

Запрограммируйте аналогично остальные пункты *Цвет*. Запустите и отладь- те приложение. Сохраните проект.

Контрольные вопросы

1. Ка

ково основное назначение объекта MenuStrip?

2. Как запрограммировать необходимый пункт меню формы вVisual c++?
3. Ка

кое свойство служит для изменения фона объекта?

4. С

помощью какого свойства menu можно сделать недоступнымкакой-либо пункт?

Методические указания к лабораторным работамЛабораторная

работа № 1 Конфигурирование Microsoft Visual Studio.

Цель работы: Научиться использовать Microsoft Visual Studio для разработ- ки внешнего вида программы и написания исходного кода на языке C++. Получить практические навыки работы со средой визуальной разработки программ.

Порядок выполнения работы

Задание 1.Создание приложения.

Запустить оболочку Visual C++. В меню Tools / Options во вкладке Directories установить пути на заголовочные,библиотечные файлы, а также на файлы исход- ных текстов стандартной библиотеки и CRL, если они еще не установлены.

Создать новый проект приложения (меню File / New…, вкладка Projects, пункт Windows Forms Application).

Написать небольшую программу, которая будет отображать такую экранную форму, где будет что-либо написано, также в форме будет расположена командная кнопка с надписью,призывающей нажать на кнопку.

Написать обработчик события: при нажатии на кнопку выводится какое-либосообщение.

Запустить отладчик, убедиться, что программа работает. Добавить в приложение еще надпись, изменить название формы.

Запустить отладчик, убедиться, что программа работает.

Задание 2. Обработка события MouseHover мыши

Событие MouseHover наступает тогда, когда пользователь указателем мыши «зависает» над каким-либо объектом, событиеMouseHover происходит, когда ука-затель мыши наведен на элемент.

Запустить отладчик, убедиться, что программа работает.

Вопросы к защите

1. Основные компоненты Visual C++.
2. Создание приложения в Visual C++.
3. Использование отладчика.
4. Опишите свойства объекта Button:Font, ForeColor, Name.
5. Опишите методы объекта Button: MouseDown, MouseEnter,MouseLeave.
6. Как сгенерировать обработчик выше перечисленных событий?

Лабораторная работа № 2

Разработка приложений с использованием интерфейса ибиб-лиотеки CLR.

Цель работы: Изучить основные принципы разработки приложений с использованием интерфейса и библиотеки CLR. ***Порядоквыполнения работы:***

Задание 1.

Создать новый проект приложения Windows Forms ApplicationVisual C++.

Написать программу, которая вводит через текстовое поле число, (используя практическую работу №2), при нажатии командной кнопки возводит данное число в степень 3 и выводит результат на метку Label. В случае ввода не числа сообщает пользователю об этом.

Задание 2.

Ввести два значения через текстовые поля. При нажатии командной кнопки, произвести арифметические вычисления с введенными данными и вести результат на метку Label. В случае ввода не числа сообщает пользователю об этом.

Изучить структуру функции обработки сообщений главного окна приложе-ния .

Вопросы к защите

1. Математические функции Visual C++.
2. Соглашения об именах переменных и функций в программахдляWindows.
3. Сообщения Windows.
4. Функция обработки сообщений.

Лабораторная работа № 3

Создание графического интерфейса на базе диалогового окна VS

Цель работы: Научиться разрабатывать и реализовывать программу, ис-пользующую вкладки и переключатели, изменять шрифт вкладок.

Порядок выполнения работы

Создать новый проект приложения , с использованием мастераприложений.

Разработать программу, позволяющую использовать вкладки TabControl, пе-реключатели RadioButton. (тема выбирается произвольно).

Вопросы к защите

1. Назначение вкладок TabControl.
2. Обработка сообщений.
3. Организация работы RadioButton.
4. Обработка событий переключателей CheckChanged.

Лабораторная работа № 4 Создание консольного приложения

Цель работы: Изучить создание консольного приложения. Производить математические вычисления со значениями введенными с экрана, выводить результат на консоль.

Порядок выполнения работы

Создать новый проект приложения в узле Visual C++ в среде CLR выбрав шаблон Console Application CLR.

Согласно варианту задания (таблица4.1), написать программу для вычисле- ния значений выражений для вводимых с клавиатуры исходных данных. Вывести на экран значения исходных данных и результатов вычислений, сопровождая вы- вод именами переменных и комментариями.

Таблица 4.1 Варианты заданий

	$A = \frac{2\cos(X - \pi/6)}{Y}1/2 + \sin^2$		$\gamma = x^{\sqrt{y/x - 3}}\lvert y/x$
	$\varphi = (y - x)\frac{y - z/(y - x)}{1 + (y - x^2)}$		$\varphi = X(\sin X^3 + \cos^2 Y)$
	$s = b\sin(at^2\cos 2t) - 1$		$s = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}$
	$Q = \frac{bx^2 -}{ae^{ax} - 1}$		$y = e^{-bt}\sin(at + b)\sqrt{+}$
	$\sqrt{S} = \sqrt{xb/a +}$	0	$\sqrt{\omega =}$
1	$y = \sin^3(x^2 + a)^2 -$	2	$s = x^3 tg^2(x + b)^2 + a/$
3	$f = \sqrt[3]{mtgt + c}\sin t$	4	$y = \cos^2 x^3 - x/\sqrt{}$
5	$z = m\lvert\cos(b\lvert t\sin t) + c$	6	$R = x^2(x + 1)/b - \sin^2(x + a)$
7	$y = btg^2 x - \frac{}{\sin^2(x/a)}$	8	$\sqrt{S =}$
9	$d = ae^{-a}\cos(bx/a)$	0	$\sqrt{R =}$

Вопросы к защите

1. Создание консольного приложения.

2. Ввод значений с клавиатуры.
3. Форматированный вывод результатов вычислений.
4. Организация диалога в консольном приложении.

Лабораторная работа № 5 Инициирование и обработка событийЦель работы: Изучить методы инициирования и обработки событий.

Порядок выполнения работы

Создать новый проект с формой, в среде CLR на базе узла Visual C++ приложение шаблона Windows Forms Application Visual C++.

Создать командную кнопку в форме, используя Инструменты и Свойства. Установить свои размеры, место положения кнопки, используя свойства. Подклю- чить событие «зависание мышкой» используя панель Свойства. Добавить кнопку в коллекцию элементов управления. Вывести сообщение о действиях с кнопкой.

Вопросы к защите

1. Какие командные кнопки можно использовать в форме?
2. Какие значения в панели Свойства были изменены?
3. Какие методы при работе приложения были использованы?
4. Перечислите назначения командных кнопок и их установку спомощью Мастера.

Лабораторная работа № 6 Обработка событий клавиатуры.

Цель работы: Изучить методы обработки событий клавиатуры. Получить практические навыки по использованию различных элементов графического ин- терфейса и операторов языка VC++.

Порядок выполнения работы

События клавиатуры (клавишные события) создаются в момент нажатия или отпускания ее клавиш. Различают событие KeyPress, которое *генерируется в мо- мент нажатия клавиши*. При удержании клавиши в нажатом состоянии оно гене- рируется непрерывно с некоторой частотой. С помощью этого события можно рас- познать нажатую клавишу.

Создать приложение, написать обработчик события, распознающий нажата ли клавиша CapsLok, NumLok.

Вопросы к защите

1. Какое событие необходимо обработать, чтобы распо- знатьнажата ли клавиша Alt, Shift?
2. С помощью какой процедуры можно определить какуюклавишу нажал пользователь?

Лабораторная работа № 7Работа с файлами в VC++.Цель работы:

Изучить приемы работы с файлами и способы создания файлов. Полу- чить практические навыки по использованию файлов для хранения информа- ции.

Порядок выполнения работы

Запустите Visual Studio 2010, в окне New Project выберите в среде CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Расположите на экранной форме текстовое поле и две командные кнопки, используя приложение Практической работы № 7, создайте приложение

«Чтение/запись текстового файла в кодировке Windows 1251».

Вопросы к защите

1. Какой объект был введен в программу для работы приложения?
2. Какие аргументы используются для создания данного объекта?

Критерии оценки внеаудиторной (самостоятельной) работы

Процент результативности	Балл (оценка)	Критерии оценивания
90-100%	5	— глубокое изучение учебного материала, литературы и нормативных актов по вопросу; — правильность формулировок, точность определения понятий; — последовательность изложения материала; — обоснованность и аргументированность выводов; — правильность ответов на дополнительные вопросы; — своевременность выполнения задания.
70-89%	4	— полнота и правильность изложения материала; — незначительные нарушения последовательности изложения; — неточности в определении понятий; — обоснованность выводов приводимыми примерами; — правильность ответов на дополнительные вопросы; — своевременность выполнения задания.
50-69%	3	— знание и понимание основных положений учебного материала; — наличие ошибок при изложении материала; — непоследовательность изложения материала; — наличие ошибок в определении понятий, неискажающих их смысл; — несвоевременность выполнения задания.
0-49%	2	— незнание, невыполнение или неправильное выполнение большей части учебного материала; — ошибки в формулировке определений, искажающие их смысл; — беспорядочное и неуверенное изложение материала; — отсутствие ответов на дополнительные вопросы; — отсутствие выводов и неспособность их сформулировать; — невыполнение задания.