Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«Воронежский государственный архитектурно-строительный университет»

РАЗРАБОТКА ПРОГРАММНЫХ ПРИЛОЖЕНИЙ В СРЕДЕ MICROSOFT VISUAL C++ 2010

Методические указания
к выполнению лабораторных работ
для студентов бакалавриата направления09.03.02
«Информационные системы и технологии»

Воронеж 2015

УДК 004.424
ББК 32.973.26-018

Составитель О.В. Минакова, О.В. Курипта

Разработка программных приложений в среде Microsoft Visual C++ 2010: метод. указания к выполнения лабораторных работ для студ. бакалавриата направления 09.03.02 «Информационные системы и технологии»/ Воронежский ГАСУ ; сост.: О.В. Минакова, О.В. Курипта. – Воронеж, 2015. – 31 с.

Приводятся основные положения разработки консольных приложений и программ, использующие графический интерфейс WinAPI. Даются рекомендации и последовательность выполнения лабораторных работ по дисциплинам «Введение в программирование» и «Алгоритмы и структуры данных» для получения навыков использования среды Microsoft Visual C++ 2010 при написании и отладки программ на С.

Предназначены для студентов бакалавриата направления 09.03.02 «Информационные системы и технологии» всех форм обучения.

Ил. 19. Табл. 5. Библиогр. 5 назв.

УДК 004.424
ББК 32.973.26-018

Печатается по решению учебно-методического совета Воронежского ГАСУ

Рецензент – Е.А. Шипилова, доцент кафедры информатики и графики Воронежского государственного университета инженерных технологий

ВВЕДЕНИЕ

Получение навыков программирования является неотъемлемой частью инженерной подготовки и особенно важно для будущих ИТ-специалистов. Языки программирования C и C++ являются инструментами, сочетающими гибкость и мощь «низкоуровневого» программирования, с удобством и возможностями абстрагирования, предоставляемыми современными проблемно-ориентированными языками. Это позволяет с одинаковым успехом создавать мобильные приложения, сложные прикладные программы, распределенные системы, а также программные компоненты системного уровня и реального времени.

Методические рекомендации составлены для вводных курсов разработки программного обеспечения с использованием современных языков программирования C и C++, стандартных и прикладных библиотек и средств разработки и не ставят целью знакомство с синтаксисом и грамматикой языка, для чего имеются достаточно емкие и проверенные многолетней практикой учебники [1]. Целью данного издания является формирование базовых приемов программирования – построение структуры приложения, организации взаимодействия с пользователем и использования инструментальных средств для редактирования и отладки кода.

В данном пособии рассматриваются только вопросы использования инструментальной среды разработки Microsoft Visual C++ 2010 для написания нативных (соответствующих стандарту языка ANSIC) программных приложений на языке C. Исходные коды таких программ не зависят от среды разработки и могут быть просто перекомпилированы для различных платформ. Полученные в ходе предложенных работ навыки программирования консольных приложений и разработки программ, использующий графический интерфейс WinAPI, необходимы для большинства дисциплин направления 09.03.02 «Информационные системы и технологии».

ЛАБОРАТОРНАЯ РАБОТА №1

Тема работы: Создание консольного приложения

Цель работы: Получение практических навыков в создании консольной программы, использующей функции вывода библиотеки стандартных функций языка С.

Программные средства: MICROSOFT VISUAL C++ 2010

1.1. Указания по созданию программы в среде Microsoft Visual C++ 2010

В меню **Файл** необходимо выбрать **Создать → Проект** (или нажать комбинацию клавиш **Ctrl + Shift + N**). Результат выбора пунктов меню для создания нового проекта показан на рис. 1.

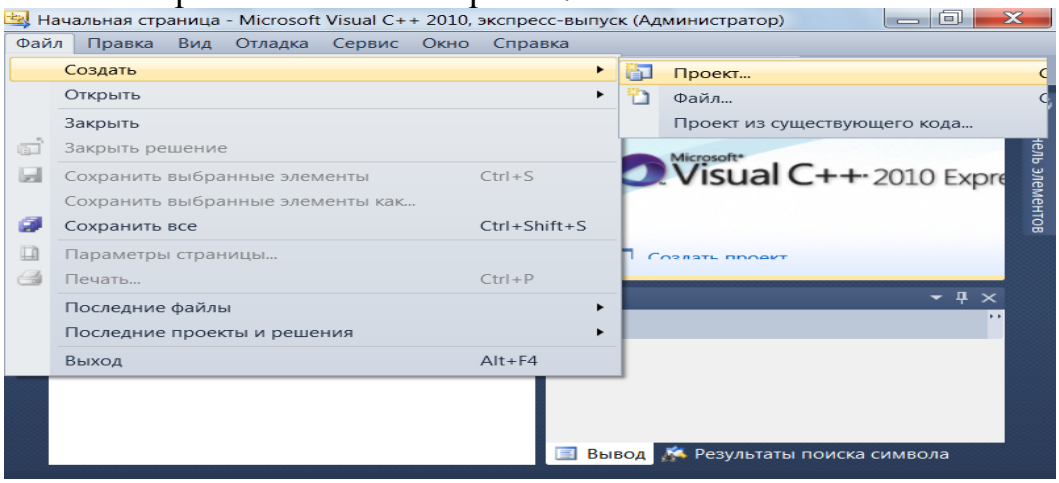


Рис.1. Окно с выбором нового проекта

Среда Microsoft Visual C++ 2010 отобразит окно **Создать проект**, в котором необходимо выбрать тип создаваемого проекта. *Проект* (project) используется в Microsoft Visual C++ 2010 для логической группировки нескольких файлов, содержащих исходный код, на одном из поддерживаемых языков программирования, а также любых вспомогательных файлов. Обычно после сборки проекта (которая включает компиляцию всех входящих в проект файлов исходного кода) создается один исполняемый модуль.

В окне **Создать проект** следует развернуть узел **Microsoft Visual C++**, затем обратиться к пункту **Win32** и на центральной панели выбрать **Консольное приложение** (рис. 2).

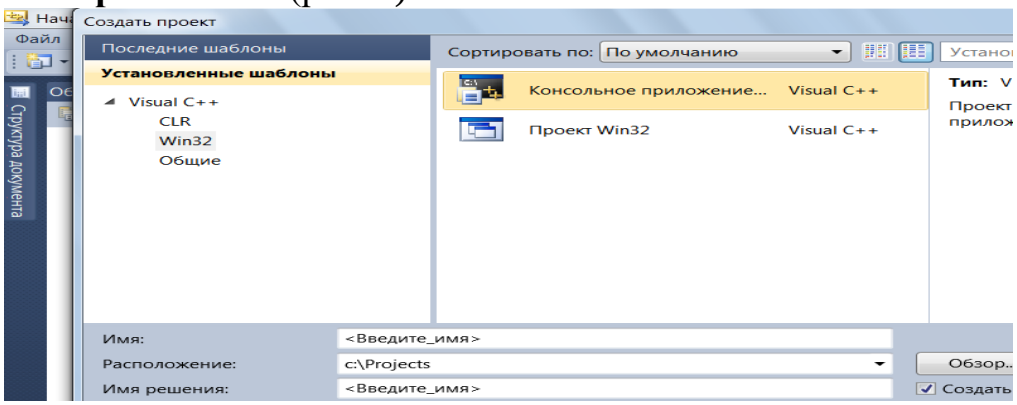


Рис. 2. Выбор типа проекта

После выбора типа проекта в поле редактора **Имя** (где по умолчанию имеется <Введите_имя>) необходимо ввести его имя, например **hello**. В поле **Расположение** можно указать путь размещения проекта или выбрать его (путь) с помощью клавиши (кнопки) **Обзор**. По умолчанию проект сохраняется в специальной папке **Projects**. Выбор имени проекта может быть достаточно произвольным: допустимо использовать числовое значение, допустимо имя задавать через буквы русского алфавита. Рекомендуется давать проекту имя, набранное с помощью букв латинского алфавита и с добавлением цифр (начинать с букв!). Пример выбора имени проекта показан на рис.3.

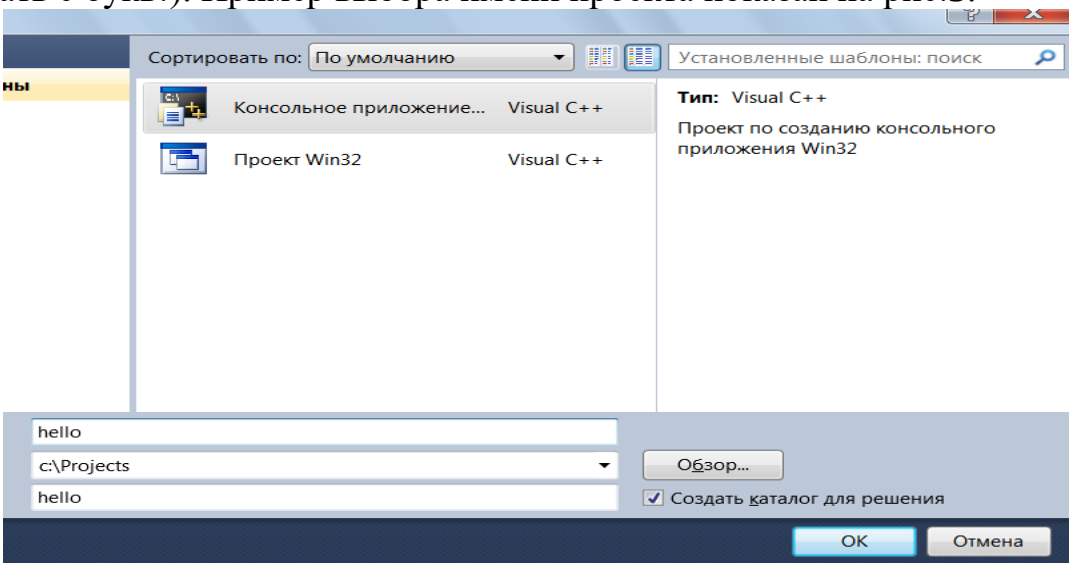


Рис.3. Пример задания имени проекта

Одновременно с проектом Microsoft Visual C++ 2010 создает решение. ***Решение*** (solution) – это способ объединения нескольких проектов для организации более удобной работы с ними.

После нажатия кнопки **OK** откроется окно **Мастер приложений Win32**. Справа расположено меню, состоящее из пунктов **Обзор** и **Параметры приложения**. На странице **Обзор** представлена информация о создаваемом проекте, на второй странице (**Параметры приложения)** можно сделать его первичные настройки **(**рис. 4**).**

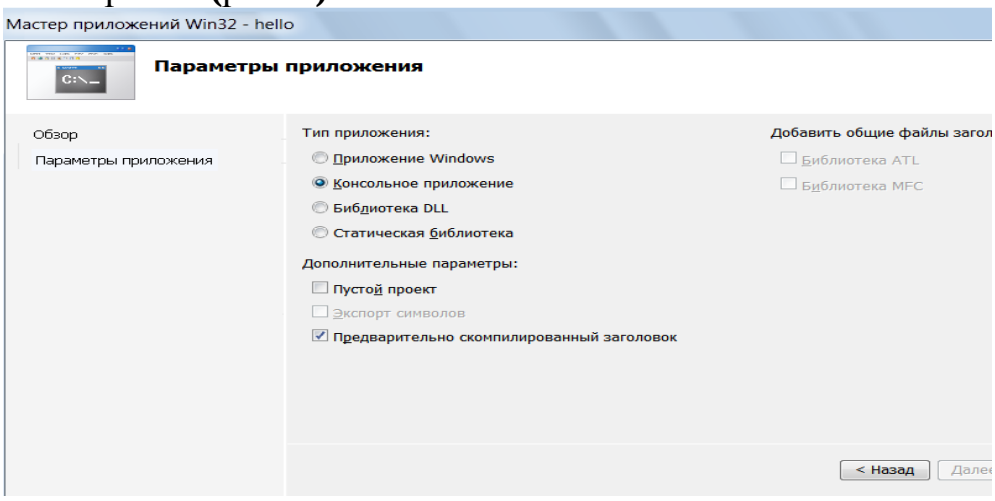


Рис. 4. Страница мастера настройки проекта по умолчанию

Для создания консольного приложения в **дополнительных параметрах** следует поставить галочку в поле **Пустой проект** и снять (убрать) ее в поле **Предварительно скомпилированный заголовок** (рис.5).

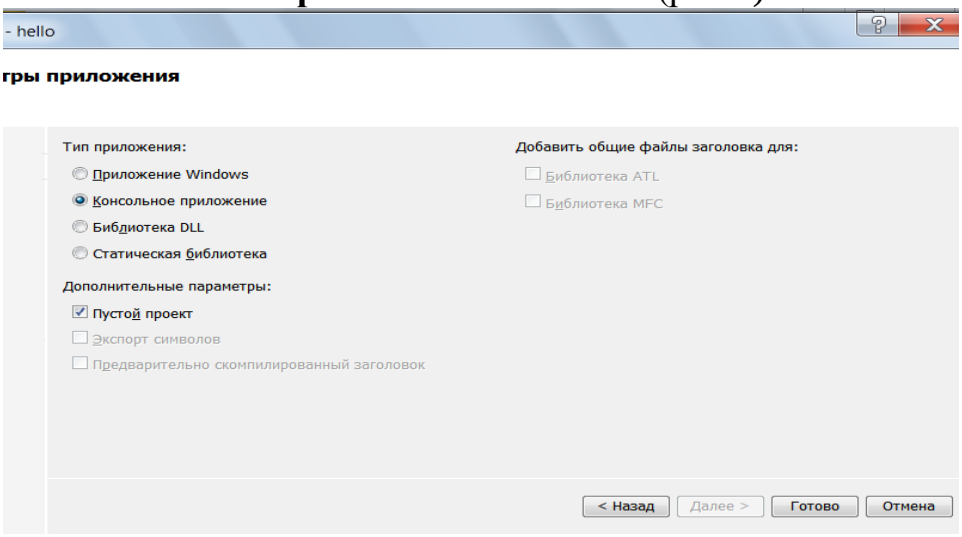


Рис.5. Страница мастера с заданными настройками

После нажатия кнопки **Готово** получим экранную форму, содержащую пустые папки для размещения файлов проекта. Для добавления исходного кода программы к проекту необходимо подвести курсор мыши к папке **Файлы исходного кода (Source Files)** из узла **hello** в левой части открытого проекта приложения, выбрать **Добавить**, затем **Создать элемент** (рис. 6**)**.

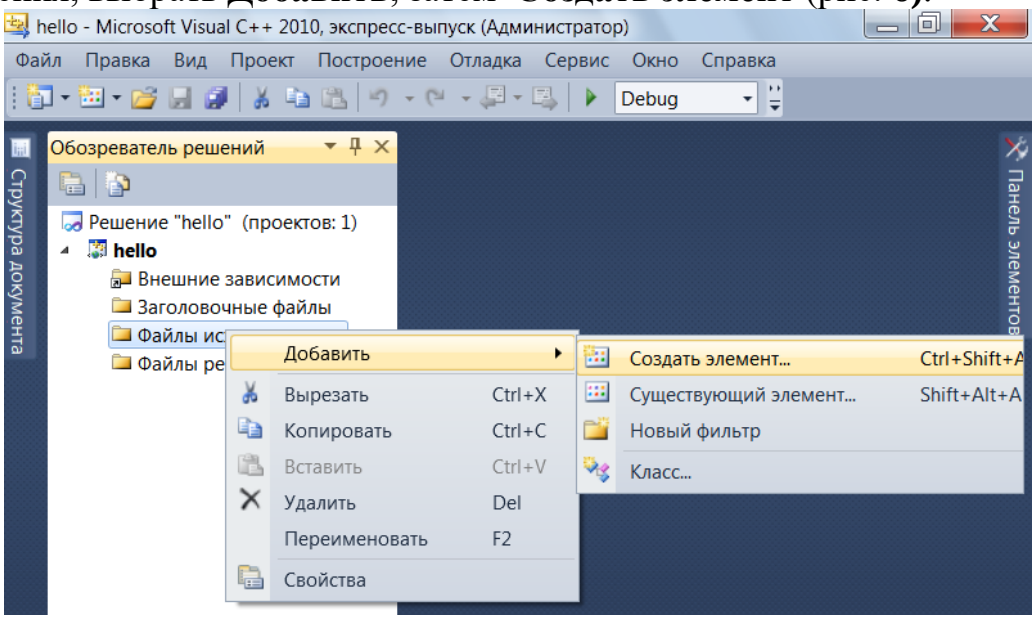


Рис.6.Меню добавления нового элемента к проекту

Выбрав (нажав) **Создать элемент**, получим окно (рис. 7), где через пункт меню **Код** узла **Microsoft Visual C++ 2010** выполнено обращение к центральной части панели, в которой осуществляется выбор типа файлов. В данном случае требуется обратиться к закладке **C++ File** (.cpp).

Теперь в поле **Имя** (в нижней части окна) следует задать имя нового файла и указать расширение **.c**. Например, **main.c**. Имя может быть достаточно произвольным, но имеется негласное соглашение, что оно должно отражать

назначение файла и логически описывать исходный код, который в нем содержится. В проекте, состоящем из нескольких файлов, есть смысл выделить файл, содержащий главную функцию программы, т.е. ту, с которой она начнет выполняться. В данном пособии такому файлу мы будем задавать имя **main.c**, где расширение **.c** указывает на то, что этот файл содержит исходный код на языке **C**, и он будет транслироваться соответствующим компилятором. Программам на языке **C** принято давать указанное расширение.

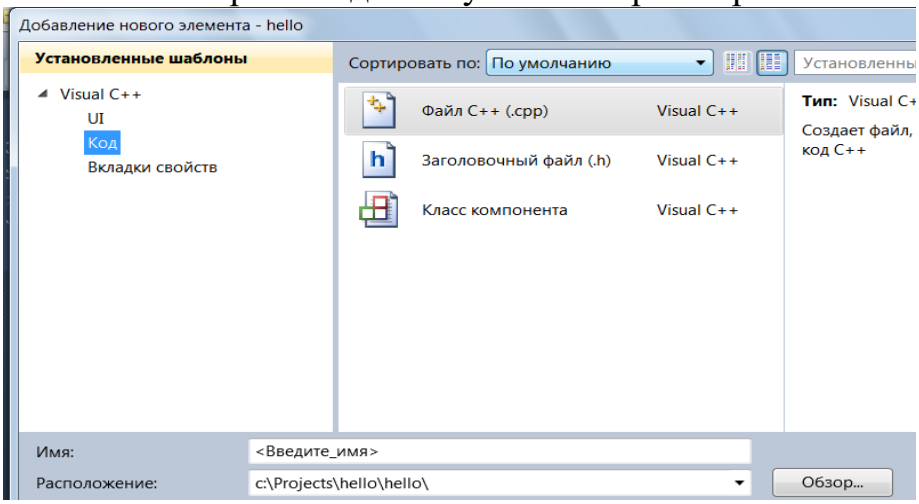


Рис. 7.Окно выбора типа файла для подключения к проекту

После задания имени файла в поле редактора **Имя** следует нажать кнопку **Добавить**. Вид среды Microsoft Visual C++ 2010 после добавления первого файла к проекту показан на рис.8. Добавленный файл отображается в дереве **Обозреватель решений** под узлом **Файлы с исходным кодом**, и для него автоматически откроется редактор в центре окна.

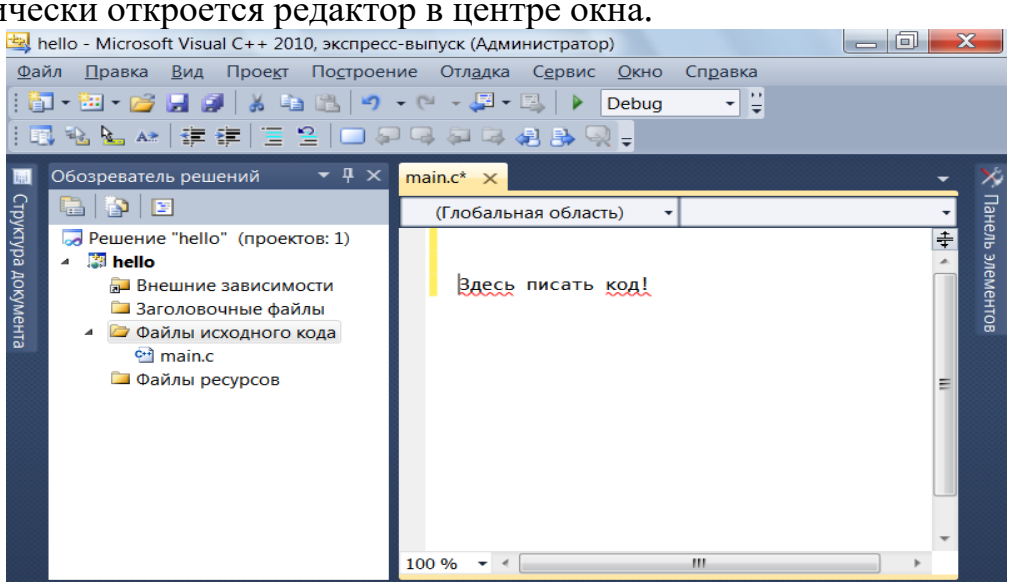


Рис. 8. Подключение файла проекта

На рис. 8 в левой панели в папке **Обозреватель решений** отображаются файлы, включенные в проект в папках. Приведем их описание.

Папка **Файлы исходного кода** (Source Files**)** предназначена для файлов с исходным кодом. В ней отображаются файлы с расширением **.c**.

Папка **Заголовочные файлы** (Header Files)содержит заголовочные

файлы с расширением **.h**.

В папке **Файлы ресурсов** (Resource Files) представлены файлы ресурсов, например изображения и т.д.

Папка **Внешние зависимости** (External Dependencies) отображает файлы, не добавленные явно в проект, но использующиеся в файлах исходного кода, например включенные при помощи директивы #include. Обычно в этой папке присутствуют заголовочные файлы стандартной библиотеки, применяющиеся в проекте.

Следующий шаг состоит в настройке проекта. Для этого в пункте **Проект** главного меню следует выбрать **hello Свойства** (или одновременно нажать клавиши Alt + F7).

После того как откроется окно свойств проекта, следует обратиться (с левой стороны) к **Свойства конфигурации**, появится ниспадающий список. Далее нужно обратиться к узлу **Общие** и через него в левой панели выбрать **Набор символов**, где установить свойство **Использовать многобайтовую кодировку**. Настройка **Набор символов** позволяет определиться, какая кодировка символов – ANSI или UNICODE – будет использована при компиляции программы. Для совместимости со стандартом языка C89 следует выбрать **Использовать многобайтовую кодировку**.

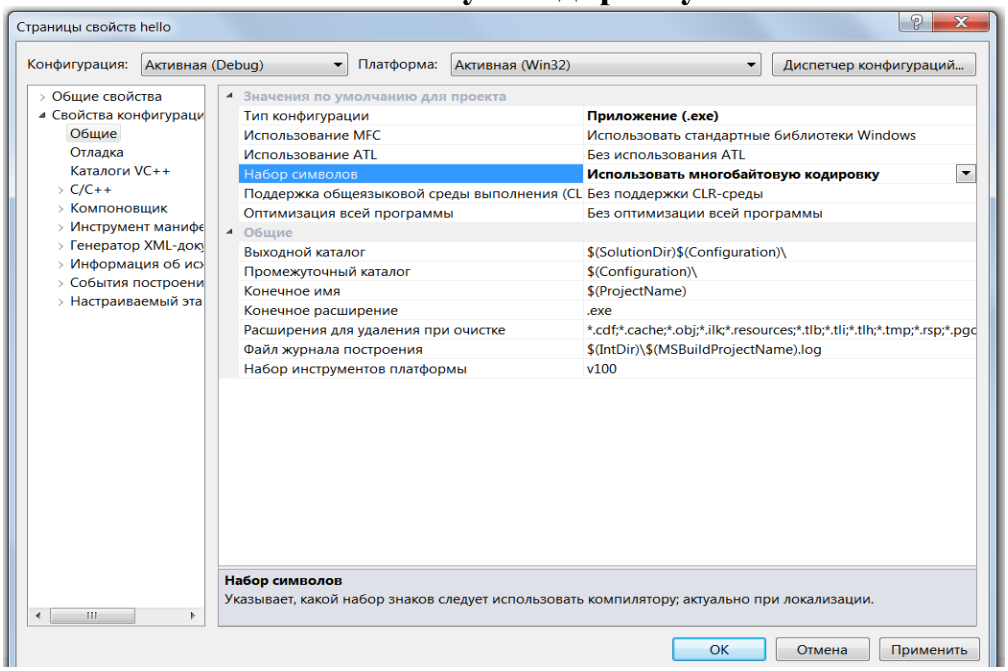


Рис. 9. Меню списка свойств проекта

После осуществления выбора следует нажать кнопку **Применить**. Затем необходимо выбрать узел **C/C++** и в ниспадающем меню выбрать пункт **Создание кода**, через который в правой части панели обратиться к закладке **Включить C++ Исключения**, установив для нее **Нет** (запрещение исключений C++). Затем нужно нажать кнопку **Применить**.

Далее в ниспадающем меню узла **C/C++** необходимо выбрать пункт **Язык** и через него обратиться в правую часть панели, где установить следующие значения свойств:

Отключить расширения языка (дополнительные языковые расширения фирмы Microsoft) – **Yes(/Да)**,

Считать wchar_t встроенным типом (базовый тип не поддерживаемый в языке С) – **No (/Z c:wchar_t–)**,

Обеспечить согласование видимости переменных (соответствие стандарту определения локальных переменных в операторе цикла for) –**Да (Z c:for Scope)**,

Включить информацию о типах во время выполнения – **No(/GR–)**,

Поддержка Open MP (используется при написании программ для многопроцессорных систем) – **No(/open mp–)**.

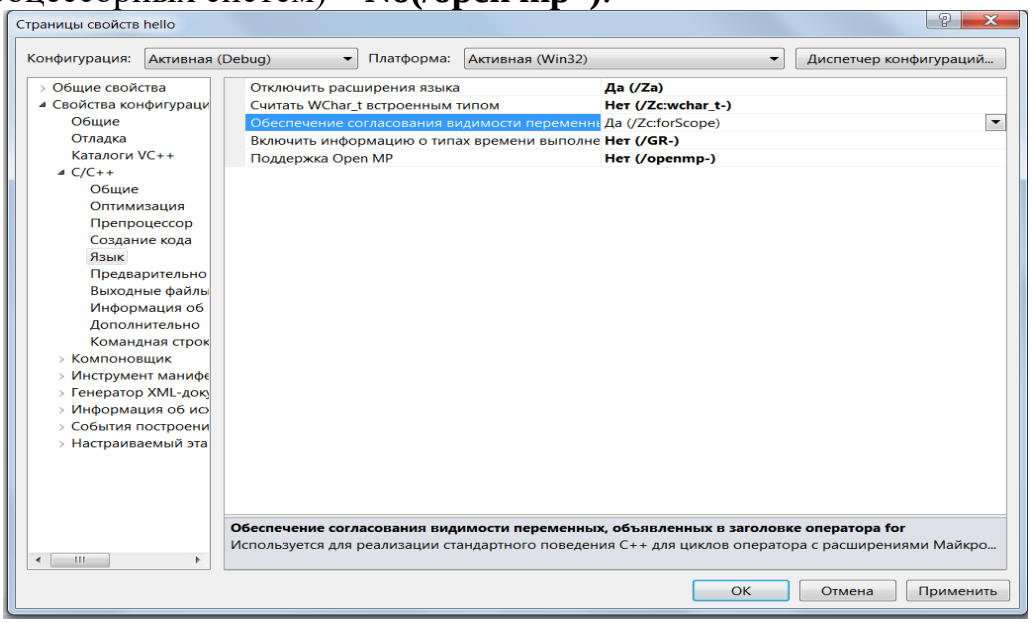


Рис. 10. Настройка языковой поддержки

После выполнения указанных действий следует нажать клавишу **Применить**.

Далее в ниспадающем списке узла **C/C++** следует выбрать пункт **Дополнительно** и в правой панели изменить свойство **Компилировать как** в свойство компиляции языка **C**, т.е. **Компилировать как код C (/TC)**.

После нажатия клавиш **Применить** и **ОК** откроется подготовленный проект с пустым полем редактора кода, в котором можно начать писать программы.

1.2. Разработка консольного приложения на языке С

Программа, написанная на языке C/C++, состоит из одной или нескольких функций, одна из которых обязательно имеет идентификатор (имя) ***main*** – основная, главная. Ее назначение – управление всей работой программы (проекта).

Наиболее ощутимые отличия консольного приложения – это организация ввода-вывода данных.

Для вывода информации в консольном приложении чаще всего используются следующие функции:

***puts* (*S*)** – вывод строки символов *S* с переходом на начало новой строки и вывод данных с форматированием;

printf (*управляющая строка*, *список объектов вывода*);

управляющая строка – заключенная в кавычки строка, содержащая спецификации преобразования объектов вывода, управляющие символы (признак «\») и любой набор символов, использующийся в качестве поясняющего текста – указывает компилятору вид выводимой информации;

список объектов вывода – переменные или выражения, вычисляемые перед выводом на консоль.

Данные, указанные в списке объектов вывода, печатаются в соответствии со спецификациями, указанными в управляющей строке. Также в управляющей строке может быть размещен текст и управляющие символы.

***Наиболее часто используемые управляющие символы*:**

n – переход на новую строку;

t – горизонтальная табуляция.

Спецификации преобразования выводимых данных имеют вид:

%<флаг><размер поля>.<точность>спецификатор.

Обязательным после знака **%** является лишь спецификатор формата, его значения указаны в табл. 1.

Таблица 1

Спецификаторы формата вывода данных

%c	символ
%d или %i	целое десятичное число со знаком
%e или %E	десятичное число в виде x.xxe+xx
%f или %F	десятичное число с плавающей запятой xx.xxxx
%g или %G	%f или %e, с выбором по принципу, что короче
%s	строка символов
%x или %X	шестнадцатеричное число

Для вывода длинных целых и вещественных чисел добавляется дополнительный спецификатор ***l*.**

Для форматирования вывода используется флаг – **-** (минус) выравнивание влево, так как по умолчанию выполняется выравнивание вправо.

Целое число, указанное после знака % задает ширину поля вывода (количество символов), при недостаточном значении выполняется автоматическое расширение. Для задания количества цифр, выводимых после запятой в вещественном числе, используется точность, которая задается числом после знака**.** (точка).

Пример вывода данных различного типа:

```
int a=5;//целое
printf ( "%d ", a); // будет напечатано 5
double b=3.4567; // вещественное
printf ( "%5.1f %s", b); // будет напечатано 3,4
char str1[]="Только";
printf ("% s%d%% предприятий не работало. \n",5)// будет
```

напечатано Только 5% предприятий не работало.

Для ввода информации с консоли используют следующие функции:

***gets* (S)** обеспечивает ввод строки символов *S* до нажатия клавиши *Enter*, т.е. позволяет ввести строку, содержащую пробелы.

scanf (*управляющая строка*, *список адресов объектов ввода*) предназначена для форматированного ввода данных. В *управляющей строке* указываются только спецификации преобразований, а в *списке объектов ввода* – ***адреса*** вводимых скалярных переменных, для чего перед *ID* переменной указывается операция &, обозначающая «взять адрес». Для ввода значений строковых (составных) переменных символ & не указывается. При использовании формата %s строка вводится до первого пробела.

Пример ввода данных различного типа:

```
int a;//целое
scanf ( "%d ", &a);
double b;//вещественное
scanf ( "%5.1f %s", &b);
char str[20]; // строка не более 19 символов
scanf (%s", str);
```

Вводить данные можно как в разных строках так и в одной строке через пробел.

1.3. Пример разработки программы, использующей ввод/вывод в консоль

В редакторе наберем программу, выводящую традиционное сообщение «Это моя первая программа».

Пример программы – вывода строки на консоль

```
#include <stdio.h>
#include <locale.h>
void main()
{
      setlocale(LC_ALL, "RUS"); // для переключения русской кодировки
      puts ("Это моя первая программа"); // вывод строки
      getchar(); // задержка экрана
}
```

Для компиляции созданной программы можно обратиться в меню **Build** или, например, нажать клавиши **Ctrl + F7**. В случае успешной компиляции получим экранную форму, показанную на рис. 11.

Для приведенного кода программы запуск на ее исполнение из окна редактора в Microsoft Visual C++ 2010 выполняется нажатием клавиши **F5**.

Пример программы, использующей ввод/вывод данных

```
double a, b, res; //объявление переменных
puts ("Программа выбора наибольшего");
printf ("Введите два значения через пробел");//приглашение к вводу данных
scanf ("%lf, %lf", &a, &b); // Ввод значений
```

```
if (a>b) res=a; // Проверка
else res=b;
printf ("Наибольшее %lf", res); //Вывод результата
```

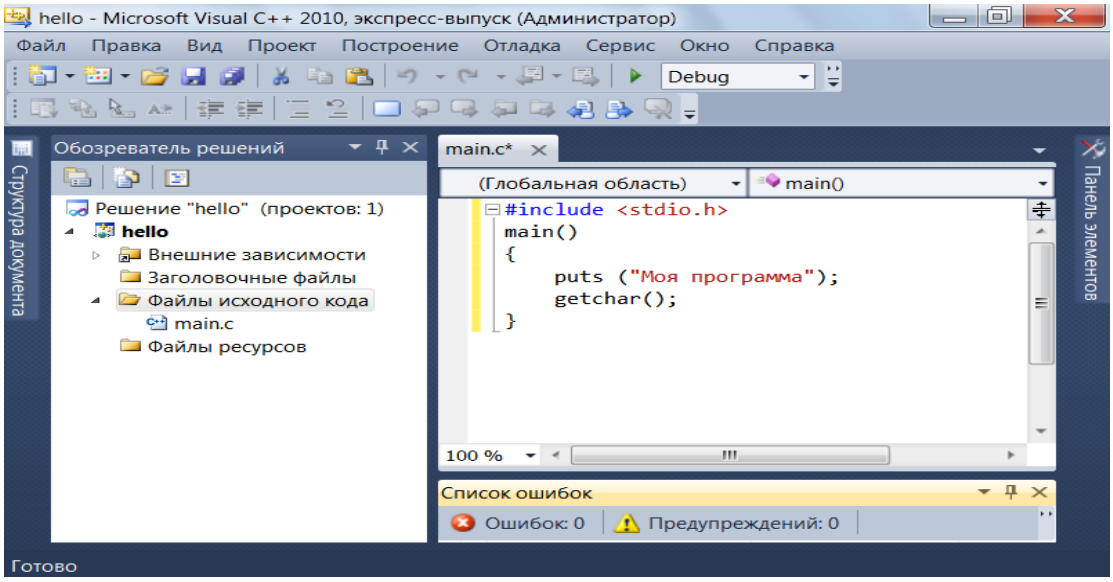


Рис.11. Успешно скомпилированная программа на языке **C**

1.4. Практические задания

Задача 1.

Напишите программу, которая выводит тему лабораторной работы, и информацию об ее исполнителе (группа, студент).

Реализуйте форматирование темы работы по центру консоли, а информацию об исполнителе – слева к краю.

Окружите текст рамкой из символов. Пример форматирования на рис. 12.

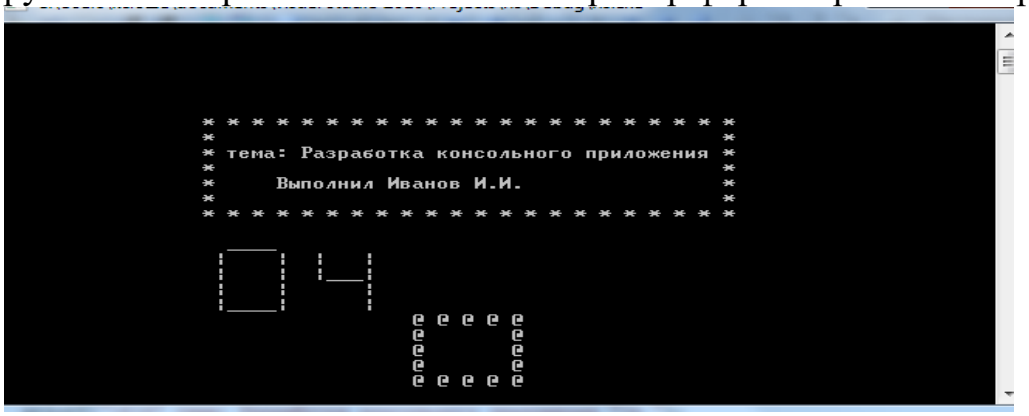


Рис. 12. Результаты работы программы консольного вывода

Задача 2.

Напишите программу для вычисления значения функции $F(x)=2*x^2-1$ для **n** вводимых с консоли вещественных чисел с выводом результатов в заданном формате – длиной ***k*** символов и ***m*** цифр в дробной части. (Вывод приглашения к вводу данных и указания на результат обязателен)

Вариант	n	k	m	Вариант	n	k	m
1	2	3	1	7	2	2	4

Вариант	n	k	m	Вариант	n	k	m
2	3	4	2	8	3	3	5
3	4	5	3	9	4	2	6
4	2	3	3	10	2	3	7
5	3	4	2	11	3	2	2
6	4	5	1	12	4	3	1

ЛАБОРАТОРНАЯ РАБОТА №2

Тема работы: Создание оконного приложения

Цель работы: Получение практических навыков написания программ с использованием ввода и вывода в окно

Программные средства: MICROSOFT VISUAL C++ 2010

2.1. Указания по созданию оконного приложения

При создании проекта выбираем **Проект Win32** и задаем имя проекта и нажимаем кнопку **OK (**рис. 13).

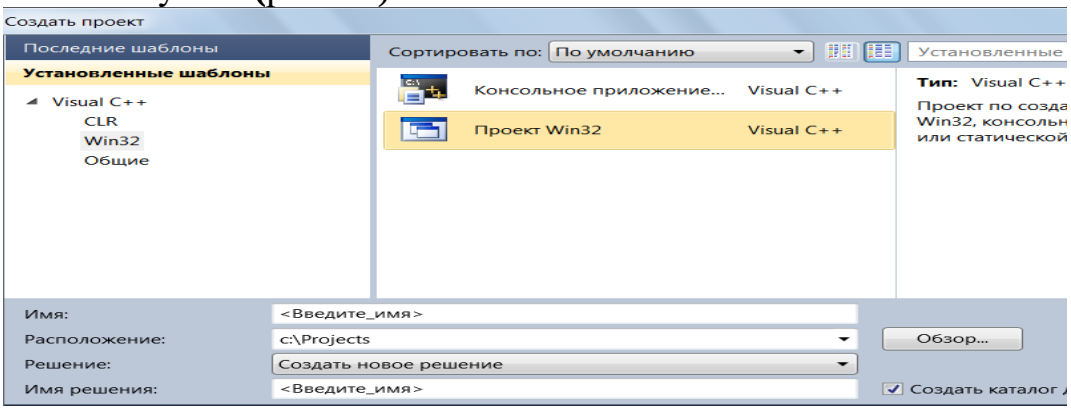


Рис. 13. Выбор типа проекта

В появившемся окне слева выбираем раздел **Параметры приложения** и отмечаем галочку **Дополнительные параметры: Пустой проект** и нажимаем кнопку **Далее**.

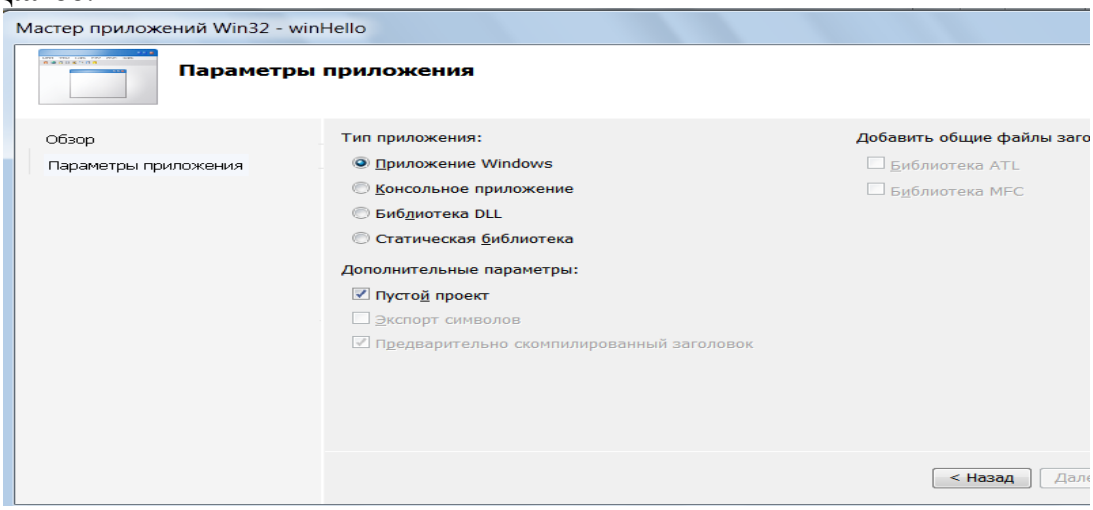


Рис.14. Окно настройки проекта

В левой части появившегося окна отображается **Обозреватель решений**. Для добавления нового файла программы в проект выбираем по правой кнопке мыши на папке **Файлы исходного кода** меню **Добавить->Создать элемент**.

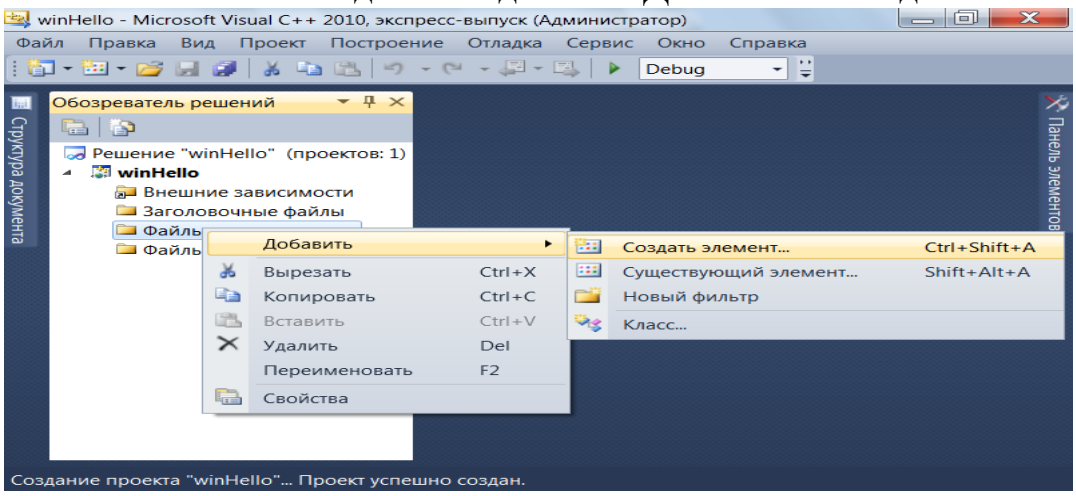


Рис. 15. Окно выбора элемента

В появившемся окне выбираем **Файл C++ (.cpp)**, задаем имя файла и нажимаем кнопку **Добавить**. В появившемся окне набираем текст программы.

2.2.Разработка оконного приложения

Оконные приложения строятся по принципам событийно-управляемого программирования, при котором поведение компонента системы, в данном случае – окна, определяется набором возможных внешних событий и ответных реакций компонента на них. В процессе работы Windows взаимодействует с каждым объектом посредством системных сообщений. При возникновении определенных событий Windows сообщает об этом окну приложения, посылая ему соответствующее сообщение. Окно, после получения сообщения, должно обработать его и возвратить результат обратно в систему. Поэтому на протяжении всей работы программы необходимо проверять, имеются ли в очереди сообщения. Для этого в программе создается цикл, в котором проверяется наличие сообщений в очереди, и организуется их поочередная обработка. Структура оконного приложения на языке **C** аналогична консольному, но состоит как минимум из двух функций – создания главного окна и цикла обработки сообщений.

Структура оконного приложения

```
#include <windows.h>
int   WINAPI   WinMain(HINSTANCE   hInst,  /* Точка старта
HINSTANCE, LPSTR cmdline, int ss) {        /* Блок инициализации:
                                           создание главного окна,
                                           загрузка ресурсов и
                                           т.п. */
MSG msg;                                   /*    Цикл    обработки
while   (GetMessage(&msg,(HWND)NULL,0,0))  событий: */
{
TranslateMessage(&msg);
```

```
DispatchMessage(&msg);
 }
 return msg.wParam;
}
LRESULT     CALLBACK     MainWinProc(HWND  /* Обработка сообщений
hw,UINT msg,WPARAM wp,LPARAM lp) {         главного окна */
switch (msg) {
case WM_CREATE:
   /* ... */
   return 0;
  case WM_COMMAND:
   /* ... */
   return 0;
  case WM_DESTROY:
   /* ... */
PostQuitMessage(0);
   return 0;
  /* ... */
 }
 return DefWindowProc(hw,msg,wp,lp);
}
```

В качестве начальной точки вместо функции **main** приложение на базе Win32 должно иметь функцию **WinMain**. Прототип функции **WinMain:**

```
int WINAPI WinMain
(
HINSTANCE hInstance,     // дескриптор текущего экземпляра
окна
HINSTANCE hPrevInstance, // дескриптор предыдущего экземпляра
окна
LPSTR lpCmdLine,         // указатель на командную строку
int nCmdShow             // показывает состояние окна
);
```

Обычно в функции WinMain выполняются следующие действия:

1. Создание и регистрация класса окна.
2. Создание и открытие окна.
3. Выделение ресурсов необходимых для работы приложения.
4. Запуск цикла обработки сообщений.
5. Освобождение выделенных ресурсов и завершение работы приложения.

В качестве параметров этой функции могут быть переданы следующие значения:

hInstance – идентифицирует текущую прикладную программу;

hPrevInstance – идентифицирует предыдущий экземпляр этой программы и может быть использован, если необходимо обнаружить уже запущенный в среде экземпляр этой программы, по умолчанию этот параметр всегда имеет значение ПУСТО (NULL).

lpCmdLine – указывает на строку с нулевым символом в конце, определяющую командную строку для прикладной программы.

nCmdShow – указывает режим отображения, значения представлены в

табл. 2.

Таблица 2

Параметры отображения оконного приложения

Значение	Режим
SW_HIDE	Скрывает окно и активизирует другое окно
SW_MINIMIZE	Минимизирует определенное окно и активизирует окно верхнего уровня в списке системы
SW_RESTORE	Активизирует и отображает окно. Если окно минимизировано или развернуто, Windows восстанавливает его в первоначальном размере и позиции (то же самое, что и SW_SHOWNORMAL)
SW_SHOW	Активизирует и отображает окно на экране в его текущем размере и позиции
SW_SHOWMAXIMIZED	Активизирует окно и отображает его как развернутое окно
SW_SHOWMINIMIZED	Активизирует окно и отображает его как пиктограмму
SW_SHOWNORMAL	Активизирует и отображает окно. Если окно минимизировано или развернуто, Windows восстанавливает его в первоначальный размер и позицию (то же самое, что и SW_RESTORE).

Если функция достигла цели, она завершается тогда, когда примет сообщение WM_QUIT, она должно возвратить значение выхода, содержащееся в параметре этого сообщения wParam. Если функция завершается перед вводом цикла сообщения, она должна возвратить 0.

Обработка сообщений, получаемых приложением из среды Win32, выполняется в специальной оконной функции, которая имеет следующий прототип:

```
LRESULT CALLBACKW in Func (
HWND hwnd, // значение дескриптора окна, для обработки сообщения которого эта функция была вызвана
UINT message, // тип сообщения (сообщение таймера, мыши, клавиатуры и т.д.)
WPARAM wParam, // структура и содержание которой определяется типом конкретного сообщения
LPARAM lParam // структура и содержание которой определяется типом конкретного сообщения
);
```

В простейшем случае эта функция должна обрабатывать сообщение WM_DESTROY, которое посылается, когда пользователь завершает работу приложения.

```
LRESULT CALLBACK WinFunc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{          switch (message) {
           case WM_DESTROY:
```

```
                    PostQuitMessage (0);
                    break;
            default:
                    return DefWindowProc (hwnd, message,
wParam, lParam);}
            return 0;}
```

Вывод информации на экран может быть осуществлен специальной функцией API – MessageBox, которая создает, отображает на экране окно сообщения, и возвращает в качестве результата значение типа int, отражающее действие пользователя с этим окном. Окно сообщений содержит определяемое программой сообщение и заголовок, плюс любую комбинацию предопределенных пиктограмм и командных кнопок.

Прототип функции MessageBox следующий:

```
int MessageBox
(
     HWND hWnd,           // дескриптор окна владельца
     LPCTSTR lpText,      // адрес текста в окне сообщений
     LPCTSTR lpCaption,   // адрес заголовка в окне сообщений
     UINT uType           // стиль окна сообщений);
```

Параметрами функции являются:

hWnd – идентифицирует окно, которым было создано текущее окно сообщения;

lpText – указывает на строку с символом нуля в конце, которая должна быть выведена в качестве сообщения;

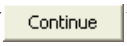
lpCaption – указывает на строку с символом нуля в конце, являющуюся заголовкомданногодиалогового окна (по умолчанию используется заголовок Ошибка (Error));

uType – управляет содержанием и поведением диалогового окна с использованием флажков, представленных в табл. 3.

Таблица 3

Параметры отображения окна сообщения

Значение	Отображаемые кнопки
MB_ABORTRETRYIGNORE	Прервать (Abort), Повторить (Retry) и Проигнорировать (Ignore).
MB_OK	Только OK (OK)
MB_OKCANCEL	OK и Отменить (Cancel)
MB_RETRYCANCEL	Повторить (Retry) и Отменить (Cancel)
MB_YESNO	Да (Yes) и Нет (No)
MB_YESNOCANCEL	Да (Yes), Нет (No) и Отменить (Cancel)
MB_HELP	Помощь (Help)

MB_CANCELTRYCONTINUE	Отменить(), повторить (), продолжить ()

Если функция завершается успешно, то возвращается значение, соответствующее нажатой кнопке (табл. 4).

Таблица 4

Возвращаемое значение

Выбранная кнопка	Значение
Прервать (Abort)	IDABORT
Отменить (Cancel)	IDCANCEL
Игнорировать (Ignore)	IDIGNORE
Нет (No)	IDNO
Кнопка OK	IDOK
Повторить (Retry)	IDRETRY
кнопка Да (Yes)	IDYES

Ввести информацию можно посредством диалогового окна с использованием функции DialogBox, создающей модальное диалоговое окно из шаблона в ресурсах (такое окно не вернет управление пока не будет вызвана функция EndDialog), ее прототип следующий:

```
INT_PTR DialogBox
{
      HINSTANCE hInstance,     // указатель на модуль
      LPCTSTR lpTemplate, // шаблон диалога
      HWND hWndParent,         // окно-владелец
      DLGPROC lpDialogFunc     // процедура диалогового окна);
```

В качестве второго параметра следует передавать структуру MAKEINTRESOURCE (идентификатор), где идентификатор указывает на имя диалога, предварительно созданного и хранящегося в файле ресурса.

Очень важно создать дополнительную функцию, обрабатывающую сообщения диалогового окна, которая передается последним параметром.

2.3. Примеры разработки диалога с пользователем

Пример 1. Вывод на экран сообщения "Привет всем!".

`#include <windows.h>`	Включение заголовочного файла функции Windows API
`#include <tchar.h>`	Включение заголовочного файла функции символьных преобразований
`int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int) {`	Главная функция
`MessageBox(`	Вызов функции для отображения окна сообщения
`NULL,`	Пустой первый передаваемый параметр, т.е. без идентификации родительского окна

`_T("Привет всем!"),`	Второй параметр – строка сообщения, формируется через вызов специальной функции
`_T("Программа"),`	В качестве третьего параметра используется вызов специальной функции для формирования строки заголовка
`MB_OK);`	Передается флаг для отображения в окне сообщения кнопки OK
`return 0;` `}`	Возврат значения 0, поскольку обработки событий в программе не предусмотрено

*Обратите внимание! В функциях WinAPI невозможно ввести строку явно, только через вызов функции _T(char *), которая возвращает указатель на строку, переданную ей в качестве аргумента.*

После написания кода, для компиляции программы выбираем меню **Отладка->Построить решение**. Для запуска приложения выбираем меню **Отладка->Начать отладку**.

Пример 2. Вывод вопроса и обработка полученного ответа.

Для реакции на действие пользователя в окне сообщения необходимо сравнивать возвращаемое функцией MessageBox значение с ожидаемым (табл. 4).

Так, если пользователь не выбрал кнопку OK, можно сформировать другое окно:

```
if(MessageBox(NULL,_T("Закрыть окно?"),_T("Оконная программа"),MB_YESNO|MB_ICONQUESTION)!=MB_OK)
    MessageBox(NULL,_T("все равно закрыть!"),_T("Оконная программа"),MB_ICONEXCLAMATION);
```

Флаг MB_ICONQUESTION использован для вывода знака вопроса в окне сообщения, а MB_YESNO обеспечивает формирование двух кнопок.

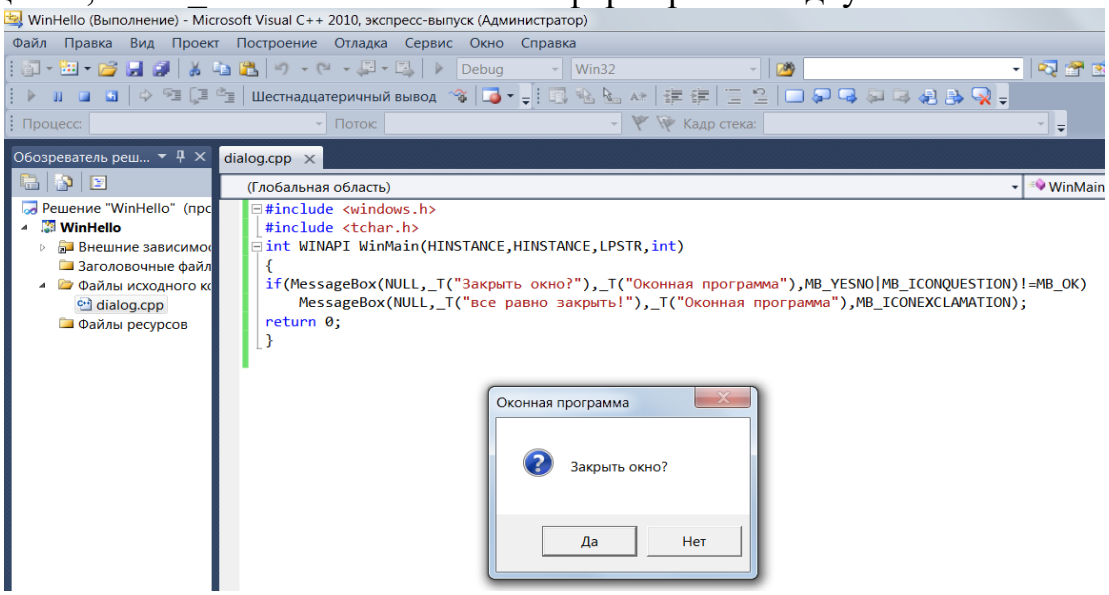


Рис.16. Пример запуска приложения с диалоговым окном

Пример 3. Ввод данных в программу.

Для организации ввода данных потребуется создание отдельного ресурса, содержащего описание содержимого окна диалога, написание пользовательской функции для обработки сообщений диалогового окна и реализация процедуры обработки полученных символов в соответствии с задачей. Последовательность шагов ввода данных с использованием DialogBox следующая:

1. Создать пустой проект, как описано в разделе 2.1

2. Создать диалоговое окно с помощью ресурса.

Для добавления ресурса в проект выбираем по правой кнопке мыши на папке **Файлы ресурсов** меню **Добавить->Создать элемент** и откроется редактор ресурсов.

Из предложенных управляющих элементов выберите **Dialog** и нажмите **New**. Щелчком правой кнопки во вновь созданном диалоговом окне выберите **Properties** и установите **ID** идентификатор **DD_DLGFIRST** и нажмитеTab. В **Caption** укажите имя окна, в **X pos box**– начальную позицию 260 и **Y pos box** – 200. Закройте и сохраните файл ресурса под любым именем.

Если редактор ресурсов не откроется, то зайдите в папку этого проекта с помощью Блокнота и создайте файл с расширением **.rc** со следующим кодом:

```
#include <windows.h>
#include "Resource.h"
IDD_DLGFIRST DIALOG 260, 200, 188, 95
STYLE WS_POPUP|WS_CAPTION|DS_MODALFRAME
CAPTION "Окно ввода"
FONT 10, "Arial"
BEGIN
LTEXT "Введите что-нибудь",-1, 10, 20, 120, 14
CONTROL "значение", IDD_EDITFIRST, "EDIT", ES_LEFT|WS_BORDER,
10, 50, 100, 14
DEFPUSHBUTTON   «OK», IDOK, 130, 10, 50, 14
END
```

Включите его в состав проекта путем перетаскивания с текущего места в папку **Файлы ресурсов** рабочего проекта.

Также в папке **Заголовочные файлы** рабочего проекта создайте файл Resource.h и запишите в него числовые значения идентификаторов:

```
#define IDD_EDITFIRST 102
#define IDD_DLGFIRST 101
```

3. Создать оконное приложение, записав его код:

```
#include <windows.h>
#include "Resource.h"
HWND hWnd;// дескриптор текущего приложения
TCHAR str[10];//буфер для введенных данных
/*-------объявление функции DlgProc
для обработки сообщений диалогового окна--------*/
LRESULT CALLBACK DlgProc (HWND hWnd, UINT Msg, WPARAM wParam,
LPARAM lParam);
//главная функция оконного приложения
INT   WINAPI   WinMain   (HINSTANCE   hInstance,   HINSTANCE
```

```
hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
     {
     DialogBox (hInstance, MAKEINTRESOURCE(IDD_DLGFIRST), hWnd,
(DLGPROC)(DlgProc));//вызов диалогового окна
     MessageBox (NULL, str, L"Моя программа", MB_ICONEXCLAMATION);
//вывод полученных данных
     return FALSE;
     }
     // функция обработки диалога
     LRESULT CALLBACK DlgProc (HWND hWndDlg, UINT Msg, WPARAM
wParam, LPARAM lParam)
     {
          switch(Msg)
          {
          case WM_INITDIALOG:
               return TRUE;
          case WM_COMMAND:
               switch(wParam)
               {
               case IDOK://обработка нажатия кнопки Ok
                    if(!GetDlgItemText(hWndDlg, IDD_EDITFIRST, str,
5)) *str=0;// чтение данных на которые указывает идентификатор
IDD_EDITFIRST в буфер str
               End Dialog (hWndDlg, 0);// закрытие диалогового окна
                    return TRUE;
               }
               break;
          }
          return FALSE;
     }
```

4. Запустить программу на выполнение.

2.4. Практические задания

1. Создайте окно сообщения «О программе» и разместите в нем информацию об авторе (на нескольких строках и с иконкой).

Параметры задания вида иконок представлены ниже.

Значение	Иконка	Возможное применение
MB_ICONWARNING		Предупреждение
MB_ICONINFORMATION		Информация о событии
MB_ICONQUESTION		Вопрос
MB_ICONERROR		Критическая ситуация или ошибка

2. Разработайте программу, предназначенную для табличного отображения 10 значений функции в заданном пользователем интервале. Шаг определяется автоматически, границы интервала задаются в диалоговом окне, результаты выводятся в окне сообщений.

Вариант	***функция***	***Вариант***	***функция***
1	синус	***7***	1-sin(x)
2	косинус	***8***	1-cos(x)
3	тангенс	***9***	1-tg(x)
4	котангенс	***10***	$\sin^2(x)$
5	секанс	***11***	$\cos^2(x)$
6	косеканс	***12***	$tg^2(x)$

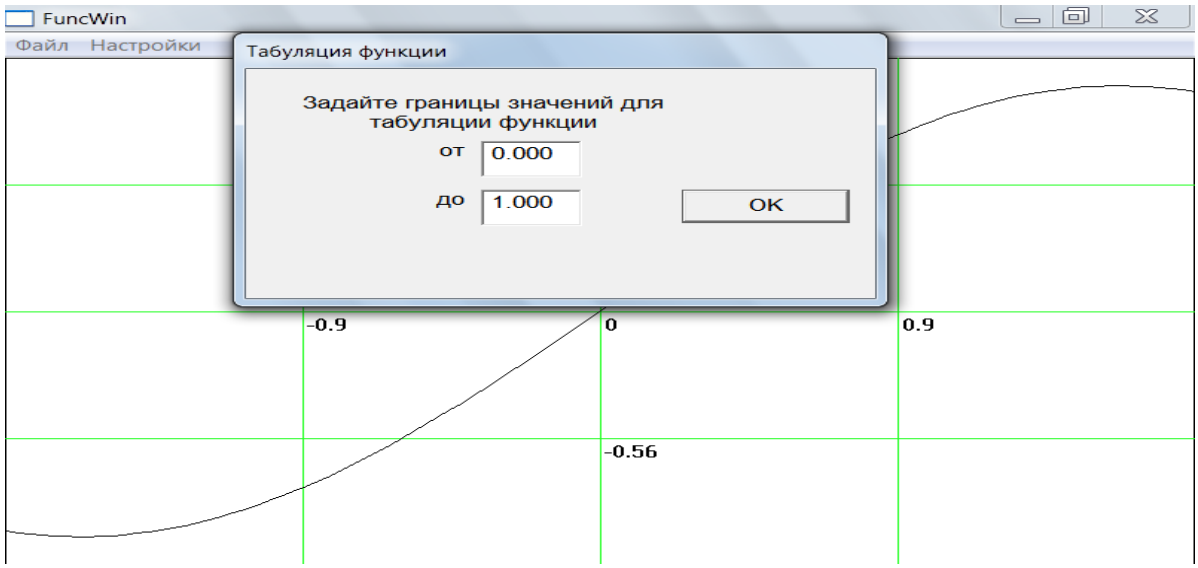


Рис.17. Пример окна задания интервала

ЛАБОРАТОРНАЯ РАБОТА №3

Тема работы: Рисование графических примитивов

Цель работы: Получение практических навыков использования графических библиотек.

Программные средства: MICROSOFT VISUAL C++ 2010

3.1. Указания по использованию графического режима в программах на C/C++

Для использования графического вывода необходимо создать **проект Win32** и проследить, чтобы в разделе **Параметры приложения** была снята галочка **Пустой проект**. В таком случае среда MICROSOFT VISUAL C++ 2010 сформирует заготовку проекта, содержащего основной файл оконного приложения с главной функцией WinMain и функцией WndProc, обрабатывающей сообщения этого окна, а также различные ресурсы (рис. 18). Файл readme.txt подробно описывает содержимое этого проекта.

Автоматически созданный проект представляет собой главное окно с возможностью использования графического режима и меню, позволяющее вызвать дополнительное окно. В исходном файле имеются метки «**TODO:**», указывающие на возможности расширения функционала этого приложения.

Построение проекта выполняется также через **Отладка -> Построение экземпляра проекта**.

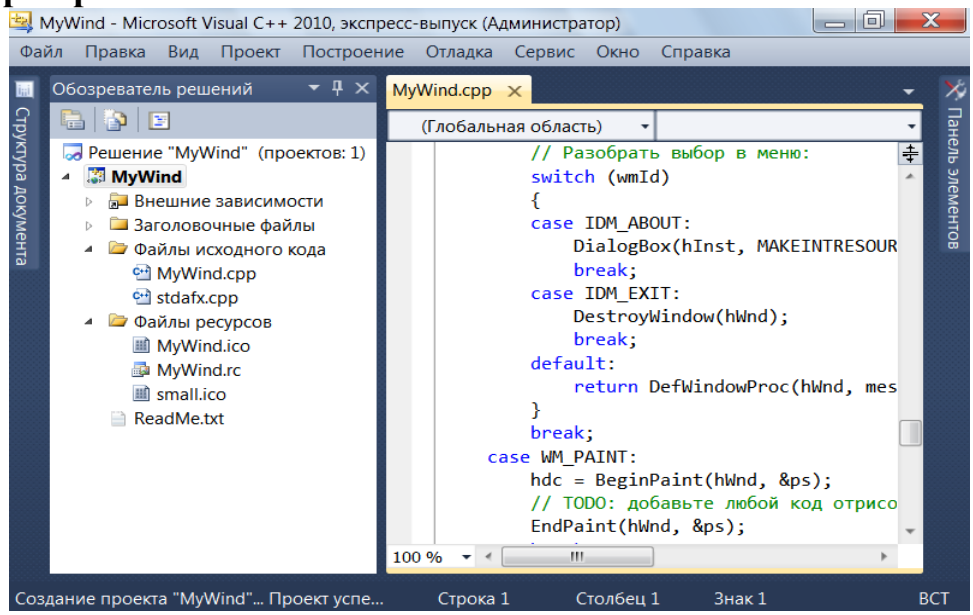


Рис.18. Заготовка стандартного проекта win32

Ввод и вывод в рабочую область окна следует осуществлять при обработке сообщения WM_PAINT:

```
case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        // TODO: добавьте любой код отрисовки...
        EndPaint(hWnd, &ps);
        break;
```

Для использования окна необходимо получить контекст устройства, который представляет собой структуру типа HDC, описывающую свойства текущего устройства вывода. Его значения будут обязательно указываться во всех используемых в программе функциях рисования в окне.

3.2.Разработка графического приложения

Единицей измерения координат в окне является логически условная единица – пиксел (pixel). Начало координат (0,0) находится в левом верхнем углу экрана, при этом ось 0x направлена традиционно (слева направо), а ось 0y – сверху вниз (рис.19).

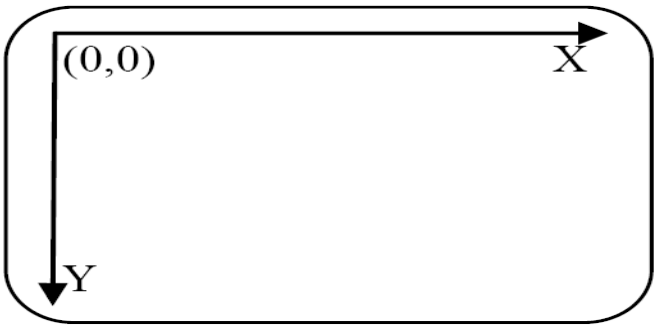


Рис. 19. Система графических (экранных) координат

Рисование в окне осуществляется с помощь линий, которые рисуются

текущим пером. С помощью функции ***MoveToEx (HDC hdc, int x, int y, LPPOINT oldPoint)*** устанавливается текущая позиция, координаты которой передаются в структуру POINT, состоящую из координат x и утипа long.

Для **рисования линии** от текущей позиции до заданной позиции с координатами (X, Y) используется функция

LineTo (HDC hdc, int x, int y) – черчение линии от текущего положения графического курсора до точки (X, Y).

Черчение **прямоугольника**, который задаётся координатами верхней левой вершины (xl, yl) и правой нижней вершины (xr, yr) осуществляется функцией

Rectangle (HDC hdc, int xl, int yl, int xr, int yr).

Отображение **эллипса**, ограниченного прямоугольником осуществляется с помощью функции

Ellipse (HDC hdc, int xl, int yl, int xr, int yr).

Если имеется массив точек и требуется соединить линией каждую следующую точку с предыдущей, можно использовать функцию PolyLine, в которой вторым параметром задается указатель на массив элементов типа POINT, а третьим – количество точек. При выполнении этой функции, в отличие от функции PolyLine то текущая позиция пера не изменяется.

Пересоздается при помощи функции HPENCreatePen (int style, int width, COLORREF color), где стиль определяется значениями, представленными в табл.5, толщина указывается в пикселях вторым параметром, а цвет указывается значениями структуры COLORREF или стандартными константами или функцией RGB (int RED, int GREEN, int BLUE), которая задает цвет в виде 8-битовых значений, определяющих интенсивность ее цветных компонентов (от черного (0, 0, 0) до белого (256, 256, 256)).

Для закрашивания аналогично создаются кисти различных стилей, однако чаще всего используется сплошная кисть. Сплошная кисть создается при помощи функции ***HBRUSH CreateSolidBrush(COLORREF color).***

После создания перо и кисть выбираются в контексте устройства при помощи функции ***SelectObject(HDC hdc, HBRUSH hbrush)***, которые должны удаляться перед завершением программы. Для этого используется функция ***DeleteObject (HGDIOBJ handle)***.

Таблица 5

Типы линий, создаваемых пером

№	Макрос	Тип линии пера
1	PS_DASH	Пунктирная линия (равномерные отрезки)
2	PS_DASHDOT	Штрих-пунктирная линия (тире-точка-тире)
3	PS_DASHDOTDOT	Штрих-пунктирная линия (тире-точка-точка-тире)
4	PS_DOT	Точечная линия
5	PS_INSIDEFRAME	Сплошная линия внутри ограничивающей области
6	PS_NULL	Прозрачное перо
7	PS_SOLID	Сплошная линия

Текст выводится в окно с помощью функции ***TextOut (HDC hdc, int x, int y, LPCWSTR str, int len),***которая выводит текст, на который указывает специальный текстовый указатель str в координаты (x, y) длиной len символов. Для вывода строки непосредственно в функции используйте ***_T().***

Для корректного отображения необходимо знать текущий размер окна, для этого используют функцию GetClientRect (HWNDh Window, LPRECTlp Rect), которая возвращает значения в структуре RECT, указатель на которую необходимо передать в функцию. Структура RECT определена как

```
typedef struct _RECT {
LONG left; //x координата левого верхнего угла
LONG top; //y координата левого верхнего угла
LONG right; //x координата правого нижнего угла
LONG bottom; //y координата правого нижнего угла
} RECT;
```

3.3. Примеры построения изображений

Пример 1.Вывод круга в середину окна

Рисование круга осуществляется функцией Ellipse(..), которой необходимо передавать координаты прямоугольной области, в которую будет вписан круг. Его размер зададим переменной radius. Середина окна (width /2, height/2) должна приходиться на центр квадрата, т.е. в точку (radius, radius) относительно квадрата с размером 2*radius в который вписан круг. Тогда координата левого верхнего угла будет вычисляться как x1=(width-radius)/2 и y1=(height -radius)/2.

Для определения размеров окна воспользуемся функцией GetClientRect(…) и структурой rect типа RECT. Ширину окна width вычислим как разницу между правой (rect.right) и левой (rect.left) координатой x, а высоту height как нижней и верхней координатой y (rect.bottom-rect.top).

```
void DrawRaund(HWND hwnd, HDC hdc, int radius){
RECT rect;//объявление структуры координат окна
int width, height; ;//объявление переменных для хранения ширины и высоты окна
GetClientRect(hwnd, &rect);//получение координат окна
width=rect.right-rect.left;//вычисление ширины окна
height=rect.bottom-rect.top;//вычисление высоты окна
Ellipse(hdc, //рисование от середины окна
rect.left+width/2-radius,
rect.top+height/2-radius, rect.left+width/2+radius, rect.top+ height /2+radius);}
```

Пример 2. Рисование закрашенного пятиугольника

Рисование любой фигуры сводится к следующим шагам:

1. Определение координат фигуры
2. Создание пера для контура
3. Создание кисти для закрашивания
4. Выбор пера и/или кисти в контексте отображения
5. Рисование фигуры

6. Удаление пера и/или кисти из контекста отображения

Для практически всех этих шагов предусмотрены функции в графической библиотеке. Но самостоятельно необходимо вычислить координаты фигуры и задать их в виде массива значений типа POINT. Координаты вершин правильного пятиугольника вычисляются по формулам:

$$x_i = x_0 + r \cdot \sin\frac{2\pi \cdot i}{5}$$

$$y_i = y_0 - r \cdot \cos\frac{2\pi \cdot i}{5},$$

где x_0, y_0 – координаты центра фигуры, r– радиус. За центр прорисовки возьмем середину окна с координатами (width/2, height/2), радиус можно задавать произвольный, но не более половины окна, т.е. от 5 до height/2.

Для рисования закрашенной фигуры используется функция Polygon, ее прототип аналогичен PolyLine:

```
Polygon(HDC hdc, // контекст отображения
const POINT * lppt,// указатель на массив структур POINT, содержащий отображаемые точки
int cPoints); // количество отображаемых точек
```

При создании кисти и пера необходимо определить цвет с помощью макрокоманды RGB, комбинирующей цвет из отдельных однобайтовых компонентов:

Биты

Red - Красный

7	6	5	4	3	2	1	0

Green - Зеленый

7	6	5	4	3	2	1	0

Blue - Синий

7	6	5	4	3	2	1	0

Что в десятичном представлении соответствует значениям от 0 до 255, например красный цвет RGB (255, 0, 0), коричневый – RGB (255, 128, 0), зеленый – RGB (0, 255, 0), желтый – RGB (255, 255, 0), синий – RGB (0, 0, 255).

```
//функция рисования пентагона
void DrawPentagon(HWND hwnd, HDC hdc,int radius){
RECT rect; // для границ текущего окна
POINT pntpt[6];//для вершин пентагона
//определение координат середины окна
GetClientRect(hwnd, &rect);
const int x0 = (rect.right-rect.left)/2;
const int y0 = (rect.bottom-rect.top)/2;
//задание координат пятиугольника
for(int i=1; i<7;i++){
     pntpt[i-1].x=x0+radius*sin(2.*M_PI*i/5.);
     pntpt[i-1].y=y0-radius*cos(2.*M_PI*i/5.);
 }
//создание пера и кисти
HPEN hPen0 = CreatePen(PS_SOLID, 5, RGB(0,160,0));
HBRUSH hBrush0=CreateSolidBrush(RGB(200,10,0));
```

```
//выбор пера и кисти в контексте отображения
SelectObject(hdc, hPen0);
SelectObject(hdc, hBrush0);
//рисование фигуры по заданным точкам
Polygon(hdc,pntpt,6);
}
```

Обратите внимание, что вместо 5 вершин в функции Polygon используется 6 координат, последняя из которых совпадает с первой. Это необходимо для того, чтобы была создана замкнутая фигура, в противном случае – закрашивания не будет.

Пример 3.Вывод текста в центр окна

```
/*  функция отображение заданный параметром str текст длиной len символов в середину окна, указанного hwnd через контекст отображения hdc */
void DrawTextCenter(HWND hwnd, HDC hdc, LPCWSTR str, int len){
RECT rect;
GetClientRect(hwnd, &rect); // получение размеров окна в структуру RECT rect
const int x = rect.right;
const int y = rect.bottom;
SIZE size;
//  определение размеров текущего шрифта в структуру SIZE
GetTextExtentPoint32(hdc,str,_tcsclen(str),&size);
//вывод текста
TextOut (hdc, (x-size.cx)/2, (y-size.cy)/2,  str,len);
}
```

3.3. Практические задания

1. Постройте следующие фигуры:

А) диагональные зеленые линии, одну – идущую от верхнего левого угла окна к нижнему правому и другую – от нижнего левого к верхнему правому;

Б) закрашенный красным прямоугольник шириной 50 и высотой 20, расположенный внизу посередине окна;

В) штрихпунктирную стрелку, направленную из точки (100, 100) в точку (150, 100) и состоящую из равностороннего треугольника-острия. Сторона треугольника, пересекающая отрезок, образует с ним прямой угол. Точка пересечения делит отрезок в отношении 1:5.

Обратите внимание! При толщине линий больше 1 вне зависимости от выбранного стиля отрисовка все равно будет – PS_SOLID.

Выведите в центр окна сообщение «Моя программа».

2. Доработайте программу предыдущей работы для построения графика заданной функции в интервале, определяемом пользователем. Пример разработки программы приведен в приложении 1.

ЗАКЛЮЧЕНИЕ

Проведение лабораторных работ с использованием предложенных методических указаний позволяет студентам познакомиться с двумя парадигмами программирования – императивной и событийно-ориентированной в рамках построения консольного и оконного приложений для операционной системы Windows, а также позволят получить навыки разработки пользовательских интерфейсов с организацией символьного и графического вывода информации.

В методических указаниях даны теоретические сведения и описаны практические действия необходимые для освоения одной из современных сред разработки программных приложений Microsoft Visual C++ 2010.

Все представленные работы имеют подробные пояснения и разбор практических примеров написания консольных и оконных программных приложений, а также задачи для самостоятельной работы.

Методические указания предназначены для дисциплин первого года обучения направления 09.03.02 «Информационные системы и технологии» – «Введение в программирование» и «Алгоритмы и структуры данных», и могут быть использованы во всех дисциплинах профессионального цикла, где написание программных приложений является инструментом решения научных и инженерных задач.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Керниган, Б. И., Ритчи Д. М. Язык программирования С / Б.И. Керниган, Д. М. Ритчи. – 2-е издание: Пер. с англ. – М.:Издат. дом «Вильямс» , 2007. – 304 с.

2. Павловская, Т.А. С/С++. Программирование на языке высокого уровня/ Т.А. Павловская. – СПб: Питер, 2013. – 461 с.

3. Пахомов, Б.И. С/С++ и MS Visual C++ 2010 для начинающих./Б.И. Пахомов. – СПб: БХВ-Петербург, 2011. –736 с.

4. Хортон, А. Visual C++ 2010. Полный курс /А. Хортон. – Диалектика, 2011 – 2116 с.

5. Документация MSDN по Windows API https://msdn.microsoft.com/ru-ru/library/windows/desktop/hh920508(v=vs.85).aspx

ПРИЛОЖЕНИЕ

Листинг программы
ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ

```
// FuncWin.cpp: определяет точку входа для приложения.
#include "stdafx.h"
#include "FuncWin.h"
#include "Math.h"
#include <stdlib.h>
#define MAX_LOADSTRING 100
                // Глобальные переменные:
HINSTANCE hInst;                                // текущий экземпляр
TCHAR szTitle[MAX_LOADSTRING];                  // Текст строки заголовка
TCHAR szWindowClass[MAX_LOADSTRING];            // имя класса главного окна
double Xmin, Xmax;          // задание границ интервала построения графика
// Отправить объявления функций, включенных в этот модуль кода:
ATOM                MyRegisterClass(HINSTANCE hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);
                // Задать функцию для вывода графика
double F1(double x){
return    sin(x);
}
//  ФУНКЦИЯ: drawGraph(HWND hWnd,HDC hdc)
//  НАЗНАЧЕНИЕ: рисует график заданной функции F1 в окне на которое указывает hWnd
//        с использованием контекста hdc  в интервале от Xmin до Xmax
void drawGraph (HWND hWnd, HDC hdc){
        const int n = 100; // количество точек графика
        double x[n];      // массив координат x
        double y[n];    // массив координат y
        double scale;
        //double xmax = 0; // максимальное значение функции
        double ymax  = 0, ymin=0; // максимальное значение функции
        // определение шага построения графика
        double cdel = (Xmax - Xmin)/(n-1);
        // вычисление значений функции
        for( int c= 0; c<n; c++)
        {
                x[c] = Xmin + cdel*c;
        y[c] = F1(x[c]);
                // определение размаха функции
                if(y[c]>ymax)ymax=y[c];
                if(y[c]<ymin)ymin=y[c];
        }
                        //определение размеров окна
        RECT r ;
        GetClientRect(hWnd,&r);
        int width = r.right-r.left;
        int height = r.bottom-r.top;
                        //определение соотношений для пересчета осей
```

```
        double convertX =  width/(Xmax-Xmin);
        double convertY  = height/(ymax-ymin);
        //опредление начала координат
        int winX0=convertX*(0-Xmin);
        int winY0=convertY*(0-ymin);
                        //прорисовка координатной сетки
        HBRUSH myBr = CreateSolidBrush(RGB(255,255,255));
        SelectObject(hdc, myBr);
        Rectangle(hdc, 0, 0, width, height);
        HBRUSH green = CreateSolidBrush(RGB(255,255,255));
        HPEN gr = CreatePen(PS_SOLID, 1,RGB(34,255,34) );
        SelectObject(hdc, gr);
        MoveToEx(hdc, 0,winY0,NULL);
        LineTo(hdc,width,winY0);
    MoveToEx(hdc, winX0,0,NULL);
        LineTo(hdc,winX0,height);
        TextOut(hdc, winX0+3 ,winY0+3, _T("0.0"), 3);
                                // отрисовка графика
        HPEN gr1 = CreatePen(PS_SOLID, 1,RGB(0,0,0) );
        SelectObject(hdc, gr1);
        MoveToEx(hdc,winX0+x[0]*convertX,winY0+y[0]*convertY,NULL);
        for(int i=1;i<n; i++) LineTo(hdc,winX0+x[i]*convertX,y[i]*convertY+winY0);
}
int APIENTRY _tWinMain(HINSTANCE hInstance,
                HINSTANCE hPrevInstance,
                LPTSTR    lpCmdLine,
                int       nCmdShow)
{
        UNREFERENCED_PARAMETER(hPrevInstance);
        UNREFERENCED_PARAMETER(lpCmdLine);
        // TODO: поместите код здесь.
        MSG msg;
        HACCEL hAccelTable;
        Xmin=-1.;
        Xmax=1.;
                                // Инициализация глобальных строк
        LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
        LoadString(hInstance, IDC_FUNCWIN, szWindowClass, MAX_LOADSTRING);
        MyRegisterClass(hInstance);
                                // Выполнить инициализацию приложения:
        if (!InitInstance (hInstance, nCmdShow))
        {
                return FALSE;
        }
        hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_FUNCWIN));
                                        // Цикл основного сообщения:
        while (GetMessage(&msg, NULL, 0, 0))
        {
                if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
                {
```

```
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }
        return (int) msg.wParam;
}
//  ФУНКЦИЯ: MyRegisterClass()
//  НАЗНАЧЕНИЕ: регистрирует класс окна.
//  КОММЕНТАРИИ:
//    Эта функция и ее использование необходимы только в случае, если нужно, чтобы данный
//  код был совместим с системами Win32, не имеющими функции RegisterClassEx'
//    которая была добавлена в Windows 95. Вызов этой функции важен для того,
//    чтобы приложение получило "качественные" мелкие значки и установило связь с ними.
ATOM MyRegisterClass (HINSTANCE hInstance)
{
        WNDCLASSEX wcex;
        wcex.cbSize = sizeof(WNDCLASSEX);
        wcex.style                  = CS_HREDRAW | CS_VREDRAW;
        wcex.lpfnWndProc    = WndProc;
        wcex.cbClsExtra             = 0;
        wcex.cbWndExtra             = 0;
        wcex.hInstance              = hInstance;
        wcex.hIcon          = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_FUNCWIN));
        wcex.hCursor        = LoadCursor(NULL, IDC_ARROW);
        wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
        wcex.lpszMenuName = MAKEINTRESOURCE(IDC_FUNCWIN);
        wcex.lpszClassName = szWindowClass;
        wcex.hIconSm        = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));
        return RegisterClassEx(&wcex);
}
//   ФУНКЦИЯ: InitInstance(HINSTANCE, int)
//   НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.
//   КОММЕНТАРИИ:
// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также
// создается и выводится на экран главное окно программы.
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
  HWND hWnd;
  hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной
hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
     CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
  if (!hWnd)
  {
    return FALSE;
  }
  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);
  return TRUE;
}
//  ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)
```

```
//  НАЗНАЧЕНИЕ:  обрабатывает сообщения в главном окне.
//  WM_COMMAND - обработка меню приложения
//  WM_PAINT         - отрисовка графика
//  WM_DESTROY     - ввести сообщение о выходе и вернуться.
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
        int wmId, wmEvent;
        PAINTSTRUCT ps;
        HDC hdc;
        switch (message)
        {
        case WM_COMMAND:
                wmId    = LOWORD(wParam);
                wmEvent = HIWORD(wParam);
                // Разобрать выбор в меню:
                switch (wmId)
                {
                case IDM_ABOUT:
                        DialogBox(hInst,      MAKEINTRESOURCE(IDD_ABOUTBOX),      hWnd,
About);
         InvalidateRect(hWnd,NULL,TRUE);
                        break;
                case IDM_EXIT:
                        DestroyWindow(hWnd);
                        break;
                default:
                        return DefWindowProc(hWnd, message, wParam, lParam);
                }
                break;
        case WM_PAINT:
                hdc = BeginPaint(hWnd, &ps);
                drawGraph(hWnd,hdc);
                EndPaint(hWnd, &ps);
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        default:
                return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}
// Обработчик сообщений для окна задания границ интервалов
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{
        TCHAR str1[10], str2[10];// тестовый буфер для границ
        UNREFERENCED_PARAMETER(lParam);
        switch (message)
```

```
	{
	case WM_INITDIALOG:
		return (INT_PTR)TRUE;
	case WM_COMMAND:
		switch(wParam)
			{
			case IDOK:
// считывает в текст буфера str из элемента управления, идентифицируется в ресурсе как
//IDD_EDITFIRST
				if(!GetDlgItemText(hDlg, IDD_EDITFIRST, str1, 5)) *str1=0;
				if(!GetDlgItemText(hDlg, IDD_EDITLAST, str2, 5)) *str2=0;
				Xmin=_wtof(str1);
				Xmax=_wtof(str2);
				  EndDialog(hDlg, 0);
				return TRUE;
			case IDCANCEL:
				EndDialog(hDlg, LOWORD(wParam));
				return (INT_PTR)TRUE;
		}
		break;
	}
	return (INT_PTR)FALSE;
}
```

ОГЛАВЛЕНИЕ

Разработка программных приложений в среде Microsoft Visual C++ 2010

Методические указания
к выполнению курсового проекта
для студентов бакалавриата
направления 09.03.02 «Информационные системы и технологии»
всех форм обучения

Составители: Минакова Ольга Владимировна
Курипта Оксана Валериевна

Подписано в печать 05.03. 2015.. Формат 60 х 84 1/16 Уч.-изд. л. 2,0.
Усл.-печ. л. 2,1. Бумага писчая. Тираж 60 экз. Заказ №_____

Отпечатано: отдел оперативной полиграфии
издательства учебной литературы и учебно-методических пособий
Воронежского ГАСУ
396006, Воронеж, ул. 20-летия Октября,84