Лабораторная работа №1
ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММЫ НА ЯЗЫКЕ С++

ЛИТЕРАТУРА:

1. Страуструп, Бьярне. Программирование: принципы и практика использования C++. : Пер. с англ. — М. : ООО "И.Д. Вильяме", 2011. — 1248 с.
2. Шилдт, Герберт. C++: руководство для начинающих, 2-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2005. — 672 с.
3. Программирование на С++: учеб. пособие. / В. П. Аверкин [и др.]; под ред. проф. А. Д. Хомоненко. – 2-е изд., испр. и доп. – СПб.: КОРОНА принт; М.: Альтекс-А, 2003
4. Архангельский А. Я., Тагин М. А. Программирование в C++ Builder 6 и 2006. – М.: БИНОМ-Пресс, 2007 г.
5. Архангельский А. Я. Программирование в C++ Builder. – 7-е изд. − М.: БИНОМ-Пресс, 2010.
6. Основы программирования в среде C++ Builder: лаб.практикум по курсу «Основы алгоритмизации и программирования» для студ. 1 – 2-го курсов БГУИР. В 2 ч. Ч. 1 / Бусько В. Л. [и др.] . – Минск: БГУИР, 2007. – 70 с.

1. СРЕДА ПРОГРАММИРОВАНИЯ

Разработка программ на языке С++ ведется с помощью специальных комплексов программ, которые называются системами программирования и позволяют создавать программы на определенной реализации языка. Системы программирования даже одного производителя имеют различные версии, которые отражают развитие технологии программирования и эволюцию среды выполнения программ. Это стимулирует стремление максимально использовать стандартные средства языка для того, чтобы снизить затраты на модификацию программ при изменении среды выполнения или при переходе на другую версию языка.

Вместе с тем многие важные аспекты языка определяются в *реализации* и не описываются стандартом. К их числу относится машинное кодирование символов, числовых и логических значений. Стандарт не определяет порядок создания программы для определенной среды выполнения. Детали процесса построения программ описаны в документации системы программирования. Если отвлечься от синтаксических, семантических и иных особенностей, присущих каждой конкретной системе программирования, процесс создания программ включает четыре этапа:

1. Написание и редактирование исходного текста программы с сохранением ее в виде *исходного файла* или *модуля.*

2. Компиляция программы и получение ее на определенном промежуточном языке с сохранением виде *объектного файла* или *модуля.*

3. Построение *исполнимого файла* или *модуля* путем объединения (компоновки) полученного объектного модуля программы с другими объектными модулями стандартных и специальных библиотек.

4. Отладка программы, которую можно проводить с помощью специального средства *(отладчика),* облегчающего обнаружение ошибок.

Схема получения исполнимого модуля программы в интегрированной среде показана на рис. 1.

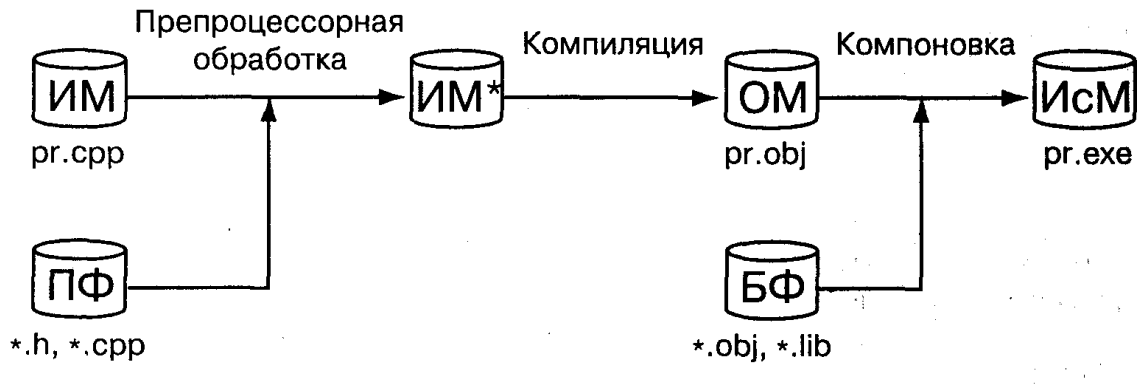


Рис.1. Схема получения исполнимого модуля

Исходный модуль (ИМ) программы подготавливается с помощью встроенного или внешнего текстового редактора и размещается в файле с расширением *cpp.* После этого ИМ обрабатывается препроцессором и, в случае необходимости, к исходному тексту программы присоединяются ***подключаемые файлы*** (ПФ). В дальнейшем ***модернизированный исходный модуль*** (ИМ*) обрабатывается компилятором. Выявленные синтаксические ошибки устраняются, и безошибочно откомпилированный ***объектный модуль*** (ОМ) помещается в файл с расширением *obj*. Затем ОМ обрабатывается компоновщиком, который дополняет программу нужными библиотечными функциями из ***библиотечных файлов*** (БФ). Полученный модуль называется ***исполнимым модулем*** (ИсМ) и помещается в файл с расширением *exe,* который в дальнейшем исполняется.

Программы-утилиты — это вспомогательные программы, которые могут потребоваться при создании программ. В качестве примера программы-утилиты следует назвать автономный отладчик. С помощью ***отладчика*** можно, в частности, выполнять программу в пошаговом режиме и контролировать содержимое всех переменных программы.

Интегрированная Среда Разработки является составной частью системы программирования. Под ***системой программирования*** понимают совокупность *языка программирования* и *программных средств*, обеспечивающих подготовку исходных текстов программ, их перевод на машинный язык, и последующую их отладку. Иными словами системы программирования создаются для удобства работы пользователя с выбранным языком программирования.

Как правило, системы программирования включают в свой состав:

- интегрированную среду разработки или программирования (Integrated Development Environment - IDE);
- компилятор;
- редактор связей или компоновщик;
- библиотеки заголовочных файлов;
- библиотеки классов и функций;
- программы-утилиты.

Интегрированная Среда Разработки (ИСР) − это среда, в которой есть все необходимое для проектирования, запуска и тестирования приложений и где все нацелено на облегчение процесса создания программ. ИСР интегрирует в себе редактор кодов, отладчик, инструментальные панели, редактор изображений, инструментарий баз данных − т.е. все, с чем приходится работать. Результатом является быстрая разработка сложных прикладных программ. Таким образом, IDE дает возможность получить EXE файл, не используя другие программы.

C++ BUILDER

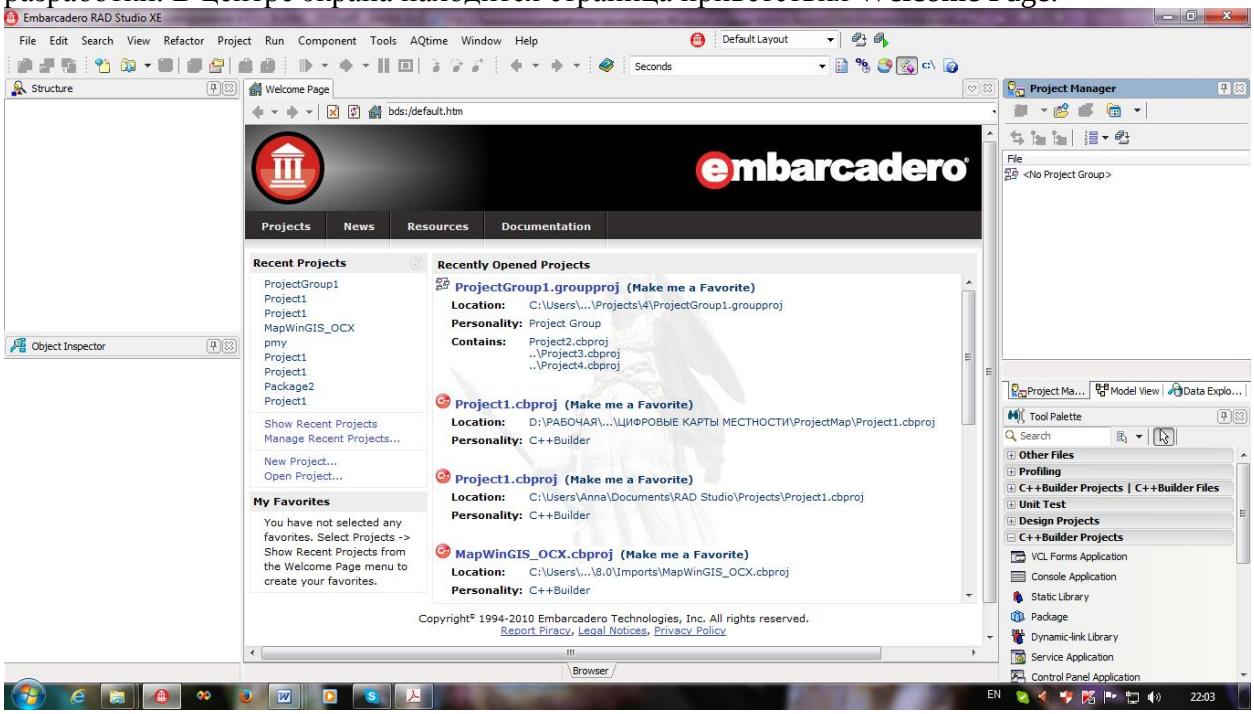
Для разработки программ на языке C++ будет использована среда программирования C++ Builder XE (ее английская аббревиатура − IDE: Integrated Development Environment), которая содержит в себе средства создания программы, ее компиляции, отладки и запуска на выполнение. В

этой связи рассмотрим кратко структуру этой среды, а точнее ее интерфейс. Интерфейс − это аппарат, который позволяет удобно взаимодействовать пользователю со средой.

После установки на своем компьютере среды C++ Builder XE вы можете ее загрузить, воспользовавшись командой главного меню **Пуск/Все программы/ Embarcadero RAD Studio XE/ C++ Builder XE** (после установки продукта его имя автоматически появится в списке команды **Программы)** или воспользовавшись ярлыком в меню **Пуск.**

Для удобства дальнейшей работы с установленным программным продуктом следует мышью перетянуть его значок на линейку быстрого запуска программ, которая находится на **Панели задач**. Находящийся на этой линейке любой программный продукт запускается одинарным щелчком мыши на значке соответствующего продукта

После загрузки C++ Builder на экране отображается основное окно Интегрированной среды разработки. В центре экрана находится страница приветствия Welcome Page.



Используя разделы меню данной страницы можно выполнить следующие действия:

− Recent Projects и Show Recent Projects − открыть один из последних проектов, с которыми вы работали.

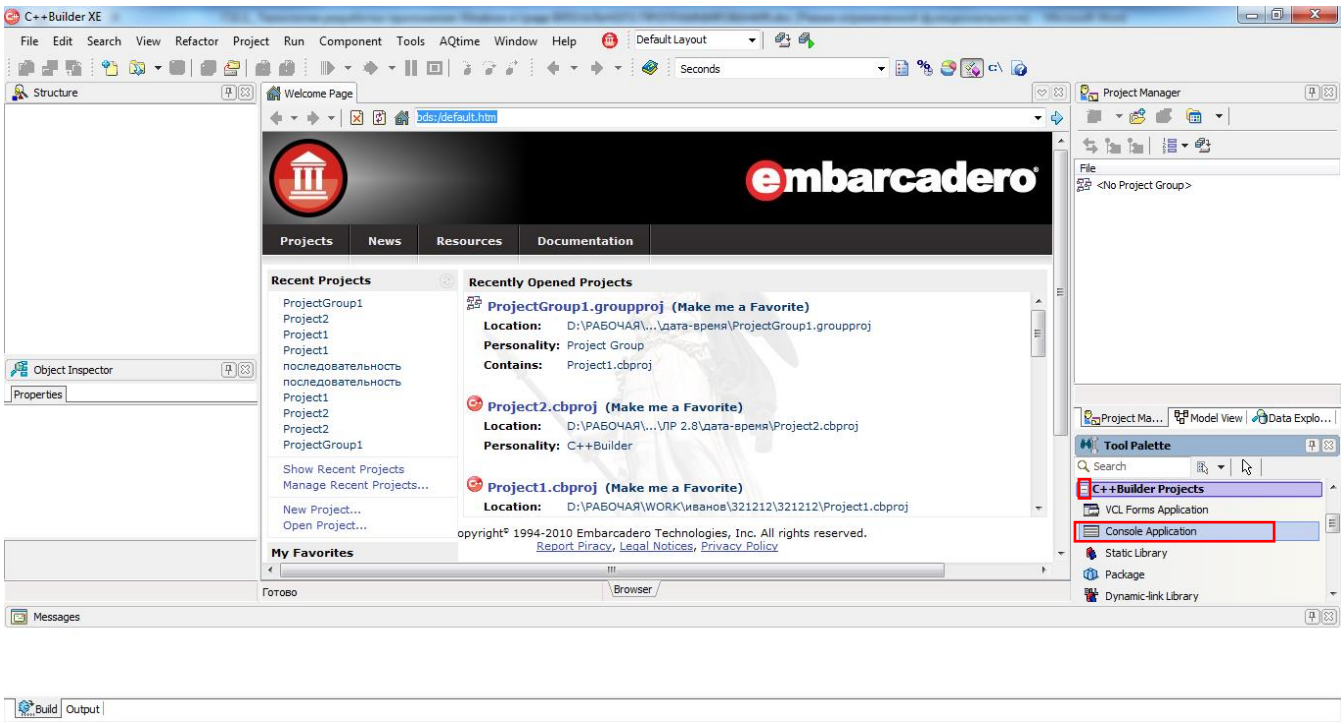
− New Projects − начать новый проект.

− Open Projects − открыть существующий проект.

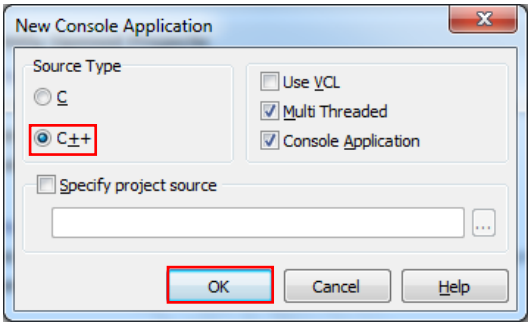
Также по умолчанию отображаются:

Tool Palette(палитра инструментов), содержимое которой изменяется в зависимости от выполняемой операции. При запуске C++ Builder список проектов, которые можно создать, выбрав их из соответствующих разделов.

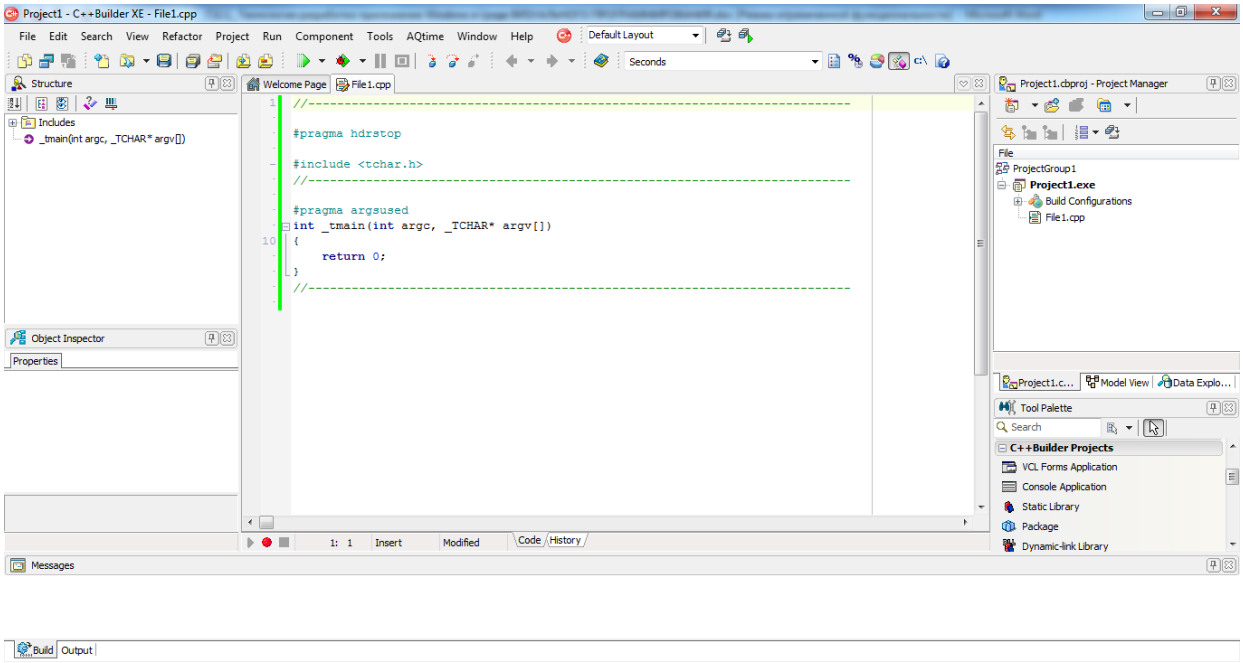
Создать проект можно несколькими способами. Например, путем выбора из **Tool Palette** раздела **C++ Builder Projects** и пункта **Console Application** (двойной клик).



В появившемся диалоговом окне должен быть выбран язык C++. Нажать кнопку OK.



окно интегрированной среды разработки (ИСР) приобретет следующий вид:



В верхней части окна располагается **Главное меню**, которое состоит из разделов:

- **File** (файл) – работа с файлами (создание, открытие и сохранение проектов, форм, фай-

лов);

- **Edit** (правка, редактирование) – работа с буфером обмена, размещение, упорядочение, выравнивание компонентов на форме;
- **Search** (поиск) – поиск, замена заданного символа (строки) в тексте;
- **View** (просмотр) – отображение различной информации, вызов **Менеджера проекта**, **Инспектора объектов** и других информационных окон;
- **Refactor**(рефакторинг) – различные структурные преобразования кода: переименование идентификаторов, преобразование фрагмента кода в метод, объявления переменных и элементов класса.
- **Project** – управление текущим проектом: добавление и удаление файлов, компиляция и сборка проекта, получение информации о текущем проекте; настройка параметров проекта, визуальной среды и хранилища объектов;
- **Run** (выполнение) – выполнение проекта, задание параметров командной строки, управление отладкой;
- **Component** (компонент) – разработка нового компонента и его инсталляция, управление библиотекой компонентов, конфигурирование **Палитры компонентов**;
- **Tools** (инструментарий) – запуск утилит, которые могут пригодиться в процессе разработки программ и настройка опций среды;
- **AQtime** – доступ к возможностям системы контроля версий, облегчающей разработку крупных проектов.
- **Windows** (окна) – переключение между окнами модулей, форм, **Инспектора объектов** и т.д;
- **Help** (справка) – получение справочной информации.

Вызвать многие команды главного меню можно также с помощью быстрых кнопок, которые располагаются на панели инструментов, или комбинаций клавиш, указываемых справа от названия соответствующей команды. Например, команду **Run/Run** можно выполнить с помощью клавиши <**F9**>.

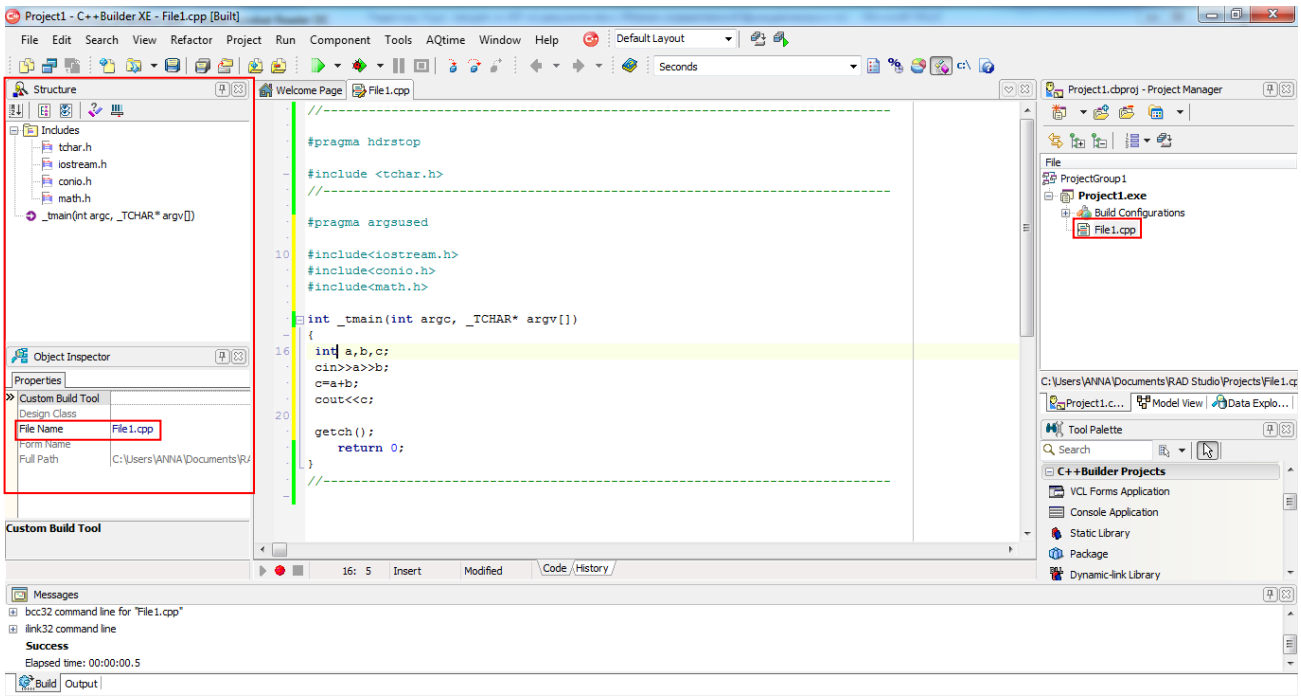
Во второй строке главного окна, расположенной под строкой главного меню, находятся кнопки быстрого вызова некоторых команд на исполнение. Все эти кнопки имеют всплывающие подсказки (надо навести курсор мыши на кнопку, немного подождать, после чего появится подсказка о том, для чего предназначена данная кнопка).

Рядом с такими кнопками могут быть дополнительные кнопки для раскрытия списка значений основной кнопки. Так как все кнопки не помещаются в отведенное им место на рабочем столе, то они свернуты в небольшие полосы с кнопками их развертывания точно так же, как это выполнено в Word.

Вид главного окна, в свою очередь, изменяется при задании типа создаваемого приложения.

Слева вверху располагается окно **Structure**(структура), в котором отображается иерархическая связь объектов приложения, подключенные заголовочные файлы и т.д.

Слева внизу − окно **Object Inspector** (инспектор объектов), в котором в консольном режиме отображаются свойства (имя и расположение) выделенного в окне Project Manager файла или проекта.



В центре экрана располагается область, где отображается код программы.

В верхней ее области находятся закладки, соответствующие открытым файлам проекта. На рисунке отображен вариант, когда доступны файл File1.cpp и страница приветствия.

Внизу расположены две закладки: Code – код программы, History − переключение в режим управления версиями проекта.

Справа от области проектирования сверху расположена панель с вкладками: Project Manager (менеджер проектов), который позволяет управлять проектами; Model View (Просмотр модели) − для представления в UML; Data Explorer − доступные приложению базы данных.

Справа внизу в окне **Tool Palette** (палитры инструментов) в консольном режиме представлены различные варианты файлов и приложений, которые можно создать или добавить в имеющийся проект.

В режиме разработки приложения с графическим интерфейсом там отображается палитра библиотеки визуальных компонентов (Visual Component Library − VCL), представленная в виде отдельных категорий, которые можно разворачивать или сворачивать, используя значки «+» и «−».

ВОПРОС 2. СОЗДАНИЕ ПРОЕКТА

Программы в C++ Builder называются приложениями (очевидно, приложениями к среде IDE). Мы так и дальше станем их называть. Приложения выглядят для пользователя как совокупность нескольких файлов.

Консольные (т. е. опорные, базовые) приложения − это приложения без графического интерфейса, которые взаимодействуют с пользователем через специальную командную строку или (если они работают в рамках IDE) запускаются специальной командой из главного меню среды. Такие приложения создаются с помощью специального шаблона, доступного из диалогового окна, открывающегося после выбора пункта **New Project…** и двойного клика по значку **Console Application**, а также выбрав в окне **Tool Palette(Палитра компонентов)** пункт **C++ Builder Projects/Console Application** после выполнения команд **File/New/Other…/ C++ Builder Projects/Console Application**. Новый проект можно создать и с помощью кнопки быстрого вызова **Создать проект** (первая слева в строке, расположенной ниже строки с опциями главного меню).

Шаблон консольного приложения добавляет в создаваемое приложение необходимые элементы (создается заготовка будущего приложения), после чего разработчик вставляет в этот шаблон свои операторы на языке C/C++. Затем приложение компилируется в автономный исполняемый файл и может быть запущено на выполнение. Общение с пользователем происходит через

специальное так называемое консольное окно, открывающееся средой после запуска приложения (в это окно выводятся сообщения программы, через него вводятся данные для расчета и в него же выводятся результаты расчетов).

Запуск на выполнение с одновременной компиляцией осуществляется с помощью команды **Run/Run** главного меню среды или нажатием на зеленую треугольную кнопку быстрого вызова команды.

Задание 1. Создание консольного приложения.

Создание и сохранение нового проекта:

1. Запустите ВИСР C++ Builder. Если C++ Builder уже работает и вы уже выполняли какие-либо эксперименты с файлом, то необходимо создать новое консольное приложение любым описанным способом**.**

2. Сохраните новый проект. Для этого воспользуйтесь командой: **File/Save All.** Можно также использовать соответствующую быструю кнопку. Используя команду контекстного меню, создайте папку с номером группы, а в ней папку LR1. Откройте созданную папку. При первом сохранении C++ Builder предложит по умолчанию имя файла *File1*, а затем – имя файла проекта *Project1*.

Обратите внимание:

- C++ Builder не допускает одинаковых имён файла модуля и файла проекта;
- для сохранения каждого нового проекта лучше всего использовать отдельную папку.

Среда оформляет создаваемое приложение в виде двух контейнеров, вложенных один в другой. Один (главный контейнер) называется **ProjectGroup(Группа проектов)**, а другой − **Проект**. **Проект** определен как конфигурация (каркас, контейнер), объединяющий группу файлов. Можно добавлять несколько проектов с помощью быстрой кнопки в окне **Project Manager** или контекстного меню к **ProjectGroup**. В данном случае можно запустить на выполнение нужный проект выбрав его в выпадающем списке кнопки запуска. Также в окне **Project Manager** можно удалить любой файл или проект, выбрав команду контекстного меню **Remove…**.

Такой подход к оформлению приложения позволяет работать с группой проектов как с одним целым, что ускоряет процесс разработки приложений.

Задание 2. Разработка программы вычисления произведения целых чисел.

1. Сохранить созданное ранее приложение **1.**
2. Закрыть приложение.
3. Открыть **1,** используя команды главного меню: **File/Reopen**.
4. В файл **1.cpp** к имеющемуся шаблону добавить текст программы.

Начинается консольное приложение директивой **#pragma hdrstop**, которая запрещает выполнение предварительной компиляции подключаемых файлов. Расположенные после директивы заголовочные файлы будут компилироваться при каждой компиляции данного модуля, а те что расположены выше - не будут. Перед этой директивой рекомендуется располагать те заголовочные файлы, которые являются общими для двух и более модулей, чтобы избежать их повторной компиляции. Включение данной директивы в коды больших проектов способствует уменьшению времени компиляции.

Директива **#pragma argsused** отключает предупреждение компилятора о том, что аргументы, указанные в заголовке функции, не используются. После этой директивы надо вставить директивы **#inciude**, обеспечивающие подключение необходимых библиотек (например, **#include <conio.h>**).

```
//программа вычисления произведения чисел
//---------------------------------------------------------------------
-------
```

```
#pragma hdrstop
//---------------------------------------------------------------------------
#pragma argsused
#include <tchar.h>

#include <iostream>   //подключается заголовочный файл для
                      //поддержки системы ввода/вывода C++
#include <conio.h>    //подключается заголовочный файл библиотеки для работы
                      //с консольным вводом и выводом – функция  getch();

#include<math.h>      //подключается заголовочный файл библиотеки для работы
                      //с математическими функциями

int _tmain(int argc, _TCHAR* argv[])   //заголовок головной функции программы
{
int a,b;              //объявление переменных a и b
cout<<"a=";           //вывод текста
cin>>a;               //ввод переменной a
cout<<"b=";
cin>>b;
int y=a*b;
cout<<"y="<<y;        //вывод текста и значения y
getch();              //ожидание нажатия любой клавиши
return 0;             /* функция main() возвращает вызывающему процессу (в роли которого
обычно выступает операционная система) значение 0. Для большинства операционных систем
нулевое значение, которое возвращает эта функция, свидетельствует о нормальном завершении
программы. Другие значения могут означать завершение программы в связи с какой-нибудь
ошибкой. Слово  r e t u r n относится к числу ключевых и используется для возврата значения из
функции. При нормальном завершении (т.е. без ошибок) все ваши программы должны возвращать
значение 0.*/
}
```

5. Запустите программу на выполнение (Команда главного меню Run\Run). Проанализируйте результат.

6. Заключите в комментарий строку программы //`#include <conio.h>`

Проанализируйте сообщение компилятора.

Выделите двойным щелчком имя функции getch(). Нажмите F1, чтобы вызвать справку.

Выполните двойной щелчок левой кнопкой мыши на поле слева от строки программы

`int y=a*b;`

7. Запустите программу на выполнение. В окне **Local Variables (локальные переменные)** проанализируйте значения переменных **a** и **b**.

Задание 3. Разработка программы вычисления результата деления двух чисел.

1. Скопируйте в буфер обмена текст программы из файла **1.cpp**.(выделить текст и **<Ctrl+C>**)
2. Закройте приложение и создайте новое с именем **2.**
3. В файл **2.cpp** вставьте из буфера обмена скопированный текст. Измените программу, чтобы в результате работы программы определялся:

− результат деления двух целых чисел (***int a,b;***)
− результат деления двух вещественных чисел (***float a,b;***)
4. Закройте приложение.
5. Откройте с помощью **Проводника** из папки **2** файл **1.exe**.

Задание 4. Разработка программы обмена местами двух целочисленных ячеек памяти без использования дополнительной памяти.

Предлагаемый алгоритм следующий:
a= a + b;
b= a - b;
a= a - b;

Программа должна запросить у пользователя два целых числа, затем выполнить алгоритм по шагам, показывая содержимое ячеек памяти до первого шага и после каждого шага. Во время выполнения данного алгоритма могут возникать целочисленные переполнения ячеек; нужно уметь правильно определить, где и почему они возникли.

***Задание 5. Разработка программы для реализации линейного вычислительного процесса* (по вариантам)**

Все переменные объявить как вещественные
float (или **double)**

Для ввода с клавиатуры вещественных переменных использовать точку
Например:
x=3,5
с клавиатуры вводить
3.5

Для использования математических функций
#include<math.h>

Для обозначения числа π
M_PI

Все аргументы в тригонометрических функциях задаются в *радианах*.

Математическая функция	**Функция библиотеки math.lib**	**Описание**
$\|x\|$	**abs(x)**	Вычисление абсолютного значения (**только для целых чисел!**)
$\|x\|$	**fabs(x)**	Вычисление абсолютного значения *x* (для вещественных чис.)
$\sqrt{x}$	**sqrt(x)**	Вычисление квадратного корня *x*
x^y	**pow(x, y)** **powl(x, y)**	Возведение *x* в степень *y*
sin(x)	**sin(x)**	Вычисление синуса *x*
sh(x)	**sinh(x)**	Вычисление синуса гиперболического *x*
cos(x)	**cos(x)**	Вычисление косинуса *x*

Математическая функция	**Функция библиотеки math.lib**	**Описание**
ch(x)	**cosh(x)**	Вычисление косинуса гиперболического *x*
tg(x)	**tan(x)**	Вычисление тангенса *x*
tgh(x)	**tanh(x)**	Вычисление тангенса гиперболического *x*
arccos(x)	**acos(x)**	Вычисление значения арккосинуса *x*
arctg(x)	**atan(x)**	Вычисление значения арктангенса *x*
arctg(x/y)	**atan2(x,y)**	Вычисление значения арктангенса двух ар-гументов *x* и *y*
e^x	**exp(x)**	Вычисление экспоненты числа *x*
ln(x)	**log(x)**	Вычисление натурального логарифма *x*
lg(x)	**log10(x)**	Вычисление десятичного логарифма *x*

Для получения справки о функции, выделить ее и нажать клавишу F1

Составить программу на языке С++ для расчета соотношения.
Исходные данные ввести с клавиатуры.

Вариант №1.

$$S=\frac{A^2+b*\cos(x)}{D^3+(A+D-b)},$$

где *A=D*x/b, b=x+D*

Вариант №2.

$$У=1+\frac{K^2}{2AB}\text{-B+DC},$$

где *A=x+sin(p), B=*e^K

Вариант №3.

$$Q=\frac{B^2}{KD}+BC^3,$$

где B=*cos(x), C=p-n.*

Вариант №4.

$$T=\cos(x)+\frac{A^2}{K-CD}\text{-B},$$

где *A=x-y,* $B=\sqrt{z}$

Вариант №5.

$$У=1{,}29+\frac{K}{A}+D^2,$$

где $A=|n+m|$, $D=tg(x)$

Вариант №6.

$$S=10{,}1+\frac{A}{C}+\frac{D}{K^2},$$

где $A=x+y$, $D=|C-A|$.

Вариант №7.

$$Y=0{,}78B+\frac{A^3}{KCD},$$

где $A=x-p$, $B=ln(h)$.

Вариант №8.

$$Y=(A+B)-\frac{C^2}{K},$$

где $A=lg(x)$, $B=x+e^d$.

Вариант №9.

$$Y=(A+B)^2-\frac{K}{CD},$$

где A=sin(x)-z, B=|p-x|.

Вариант №10.

$$Y=D^2+\frac{C^2}{0{,}75A}+B,$$

где A=ln(x)-k, $B=\sqrt{z}$.

Дополнительно:

1. Студент начал решать задачи данного урока программирования, когда электронные часы показывали h1 часов и min1 минут, а закончил, когда было h2 часов и min2 минут. Составьте программу, позволяющую определить, сколько времени студент решал эти задачи. (Будем считать, что задачи решались не дольше суток.)
2. Найти максимум и минимум двух натуральных чисел не используя алгоритма ветвления.
3. Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.

4. Определить время падения камня на поверхность земли с высоты h.
5. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
6. Треугольник задан координатами своих вершин. Найти: периметр треугольника; площадь треугольника.
7. Вычислить высоту треугольника, опущенную на сторону a, по известным значениям длин его сторон a, b, c.
8. Вычислить объем цилиндра с радиусом основания r и высотой h.
9. Определить расстояние, пройденное физическим телом за время t, если тело движется с постоянным ускорением a и имеет в начальный момент времени скорость V0
10. Вычислить площадь треугольника по формуле Герона, если заданы его стороны.
11. По данным сторонам прямоугольника вычислить его периметр, площадь и длину диагонали.

Доцент кафедры Информатики Жвакина А.В.