**Unit - IV** [ 12 Hours ]

Visual C++ Programming: Objects-Classes-VC++Components – Resources-Event Handling – Menus – Dialog Boxes – Importing VBX Controls – Files – MFC File Handling – Document View Architecture – Serialization.

Encapsulation

* Binding of data and functions into a single unit (called class) is known as encapsulation.
* Data and code are safe from outside interference and misuse.
* Data and code are combined to form a “ Black Box”.
* Data and code linked together and object is created.
* Object is a device that support encapsulation.
* While creating class, compiler will not allocate any memory space for the variables. While creating the objects compiler allocate separate memory location for each variables for each objects.
* Default protection level is private.
* Private members cannot access outside the class; Keyword: private
* Protected members are accessible in its class and derived class; Keyword: protected
* Public members are accessible inside and outside the class; Keyword: public

Class

* Class identifies a set of similar objects
* Class is an implementation of abstract data type
* Class: contains member variables and member functions
* The objects with the same data structure (attributes) and behavior (operations) are grouped into a class
* Class is a template that group data and operations
* Class is an abstraction of the real world entities with similar properties
* A class definition generates a user defined type. It holds characteristics of the objects, in terms of members

Objects

* Any noun can be represented as an object
* Objects are the basic run time entities in an object – oriented system.
* While executing programs, the objects interact by sending messages to one another
* Each object contains data and code to manipulate the data.
* Some objects may correspond to real – world entities such as students, employees, bank accounts, inventory items etc.
* Every object will have data structures called attributes & behavior called operations
* Creating new data types using encapsulated items is known as data abstraction.
* Data types created by the data abstraction process are known as Abstract Data Types (ADTs).
* Classes use the concept of data abstraction; they are known as ADT.
* Usage of data abstraction results in optimal, more readable, and flexible programs.
* Reusable software components that model items in the real world.
* Meaningful software units are Date objects, time objects, paycheck objects, invoice objects, audio objects, video objects, file objects, record objects, etc.
* Very reusable
* More understandable, better organized, and easier to maintain than procedural programming
* Helpful in modular design.
* Objects are usually referred to by references, which are aliases for an object
* Object is an instance of a class
* The declaration Car c1; means c1 is an object of type Car.
* *Objects serve the following purposes:*
  + - *Understanding of the real world and a practical base for designers*
    - *Decomposition of a problem into objects depends on the nature of the problem*

Inheritance

* It is a process by which one object can acquire the properties of another.
* It supports the concept of hierarchical classification.
* Inheritance provides the idea of reusability
* When a child class is derived from parent class that process is known as inheritance; parent class is known as base or super class; child class is known as derived or sub class
* Different types are : Single, multi level, multiple, hybrid, hierarchical etc
* Deriving a new class from an older one is called inheritance (or derivation)
* A class is derived from one base class is called single inheritance
* If a class derived from one or more base class is called multiple inheritance
* One class may be inherited by more than one class is known as hierarchical inheritance
* A class deriving from another derived class is known as multi-level inheritance
* A combination of several inheritance form hybrid inheritance

Benefits of inheritance

* We can reuse the code; so reliability will increase; and decreased maintenance cost
* Code sharing can occur at several levels. For e.g.:- at a higher level, individual or group users can use the same classes. At a lower level, code can be shared by 2 or more classes within a project
* Inheritance will permit the construction of reusable software components
* We can develop software systems more quickly and easily.

Polymorphism

* Polymorphism means ‘one interface multiple methods’.
* An operation may exhibit different behaviors in different instances.
* The behavior depends up on the types of the data and the number of the data used in the operation
* Polymorphism allowing member functions defined in classes to be overridden with member functions having same names, but different implementations in derived classes.
* C++ distinguishes between static type of a reference and dynamic type of the object it refers to at a given point
* If the function is non virtual, the call will be statically bounded to the member function of the reference ‘s class at compile time.
* If the function is virtual, it will be called dynamically at run time

Introduction of VC++

* Developed by Microsoft Corporation.
* The product has the features like interacting with user and distributing applications on the internet.
* VC++ supports adding of browser right into the application with distributed database support
* VC++ to build 32bit windows applications, including database applications, internet applications and applications with ActiveX technology
* Visual C++ used as an IDE
* It has tools for coding and debugging visual codes
* VC++ available with various editions.

1. Learning (Standard Edition) [ATL(ActiveX Template Library),AppWizard,MFC data binding]
2. Professional Edition [OLE(object linking and embedding)-DB,MTS,SMS ]
3. Enterprise Edition [ all support]

ActiveX technology

* ActiveX is a Microsoft-created technology
* It enables different software applications to share information and functionality.
* ActiveX only works with Microsoft applications like Word, Excel, Internet Explorer and PowerPoint
* It works only on Windows operating system.
* Programmer can develop C-language Windows programs using only the Win32 API using Visual C++.
* Programmers can use many Visual C++ tools: resource editors, to make low-level Win32

ActiveX Template Library (ATL)

* Visual C++ includes the ActiveX Template Library (ATL)
* It is used to develop ActiveX controls for the Internet
* ATL programming is neither Win32 C-language programming nor MFC programming.
* Formerly called ActiveX Template Library) is a Microsoft program library (set of predefined routines) for creating Active Server Page (ASP) code and other ActiveX program components
* MFC library programming interface: Use of direct Win32 calls in your MFC library programs

MFC library programming

* MFC Library classes used to create
* Graphical user interface elements (windows, frames, menus, tool bars, status bars, and so forth)
* Building interfaces to databases, for handling events such as messages from other applications
* Handling keyboard and mouse input, and for creating ActiveX controls.

Features of Visual C++

* Visual C++ used as an IDE
* It has tools for coding and debugging visual codes
* Main Features are smart pointers
* Expression parsing
* Exception handling
* New containers
* Polymorphism
* Garbage collection

### Difference between VC++ and C++

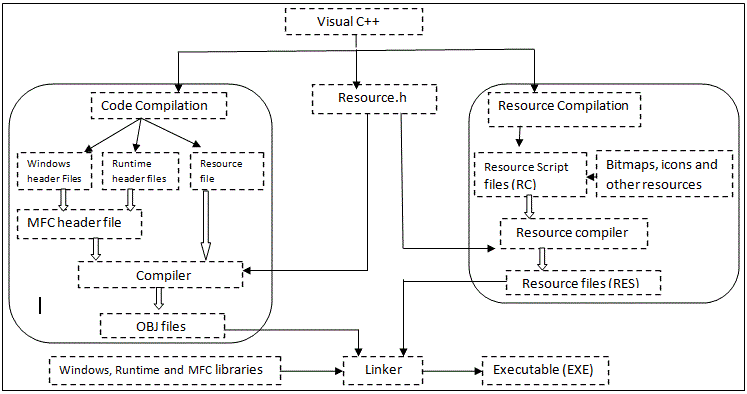
|  |  |
| --- | --- |
| **VC++** | **C++** |
| Microsoft Visual C++ (also known as VC++ or MSVC) is an integrated development environment (or IDE) product that was engineered by Microsoft for the C, C++, and C++/CLI programming languages. | it is a general-purpose programming language. It is also developed from the original C programming language. |
| VC++ includes tools that are used specifically for developing and debugging C++ code –most notably that code which is written for the Microsoft Windows API, the DirectX API, and the Microsoft .NET Framework. | it is a statically typed, free-form, multi-paradigm and a compiled programming language. |
| There are both 32 bit and 16 bit versions of VC++, both with multiple permutations therein. | its programming language is widely considered to be an intermediate-level language, as it made up of both high-level and low-level language features. |

**Visual C++ Components**

Microsoft Visual C++ has two complete windows application development system in one product

* 1. To develop C language windows program using only the win32 API
  2. By using many Visual C++ tools, including the resource editors, that make low level Win32 programming easier

**Architecture of Visual C++**



**Code compilation**:

Enter your code in one window, click on the compile button or menu to compile your code, and click on another button or menu to build your executable. When user creating a program it builds three types of Header files

1. Windows header files: Windows.h is a Windows-specific header file for the C/C++ programming language ( it contains declarations of all functions in the Windows API, macros used by Windows programmers, all the data types used by functions and subsystems)
2. Runtime header files
3. Resource file: A resource file is a text file with the extension .rc. The file can use single-byte, double-byte, or Unicode characters. The syntax and semantics for the RC preprocessor are similar to those of the Microsoft C/C++ compiler.

**MFC header file**: MFC library made of data types, variables, constants, functions, and classes used to create applications for the Microsoft Windows operating systems. An application is made of two objects: a control and a host object on which the control is positioned.

**Compiler:** it translates the C++ source code into object code.  Object code consists of machine instructions and information on how to load the program into memory prior to execution.

**OBJ files:** Object code is stored in a separate file with the extension .obj. The object file contains only the translation of the source code.

All the libraries (i.e. iostream) that are necessary to execute your program must be connected to the object code. (A library is a collection of code that has been programmed and translated by someone else, ready for you to use in your program.)

**Linker:** A special program called the linker takes your object file and the necessary parts from the included libraries and builds an executable file.

**Executable files:** file with the extension .exe.  It contains all the machine code necessary to run your program. You can run your program independently by clicking the file icon in Windows.

**Resource compilation:** The Microsoft Windows Resource Compiler (RC) is a tool used in building Windows-based applications, describes how to create a resource-definition (script) file.

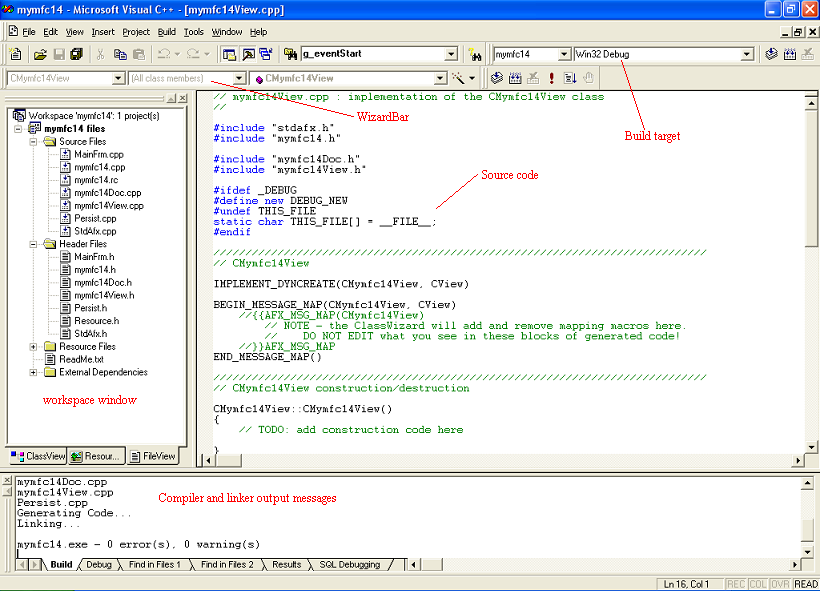
Resource file script: A resource script (.RC) file contains statements that define all of the items that will be included in the compiled binary resource file. For resources that exist in a separate file, such as icons and cursors, the resource script file simply names the resource and the file that contains it.

Some of the important VC++ components are

* The project
* Resource editors: Workspace Resource View
* The C/C++ compiler
* The source code editor
* The resource compiler
* The linker
* The debugger
* AppWizard
* Class Wizard

**The Project**

* A project is a collection of interrelated source files that are compiled and linked to make up an executable windows based program
* A separate text format WORKSPACE FILE has an entry for each project in the workspace.
* Possible to add multiple projects in a single workspace



**The Resource Editors: Workspace Resource View**

* Click the ResourceView tab in the Workspace window.
* It is used for editing the resource such as dialog box editing
* The window hosts a resource editor appropriate for resource type
* the window can host a WYSIWYG editor for menus and a powerful graphical editor for dialog boxes, includes tools for editing icons bitmaps and strings
* project has one text format resource script file (.RC) that describes project menu, dialog, string and accelerator resources
* The RC file also has # include statements to bring in resources from other subdirectories. These resource includes bitmap (.BMP) , icon (.ICO) files and common resources like error message strings

**The C/C++ Compiler**

* The visual C++ compiler can process both C and C++ source code using the extension .c and .cpp
* The templates, exceptions, and runtime type identification (RTTI) are fully supported in visual C++
* The C++ STL are included in visual C++, but not integrated into MFC library

**The Source Code Editor**

* Editor supports dynamic syntax coloring, auto tabbing, keyboard bindings
* The Visual C++ Editor has additional AutoComplete feature for better coding

**The Resource Compiler**

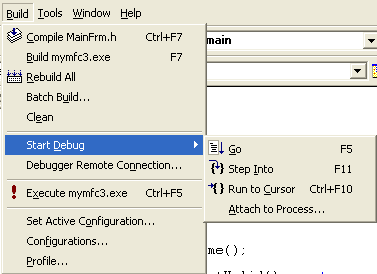
* It reads an ASCII resource script (RC) file from the resource editors and writes a binary RES file for the linker

**The Linker**

* The linker reads the .OBJ and .RES files produced by the C/C++ compiler and the resource compiler, and it accesses .LIB files for MFC code, runtime library code, and Windows code and then writes the project's EXE file.
* An incremental link option minimizes the execution time when only minor changes have been made to the source files.
* The MFC header files contain #pragma statements (special compiler directives) that specify the required library files, so no need to tell the linker explicitly which libraries to read.

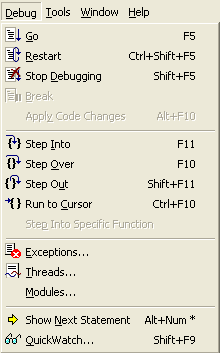
**The Debugger**

* Debugger is used to remove the bugs
* User can insert and remove breakpoints and control single-step execution.
* The Variables and Watch windows can expand an object pointer to show all data members of the derived class and base classes.
* If programmer positions the cursor on a simple variable, the debugger shows you its value in a little window.
* To debug a program, you must build the program with the compiler and linker options set to generate debugging information.
* The debug session can be invoked through the **Build** menu and **Start Debug** sub menu shown below.



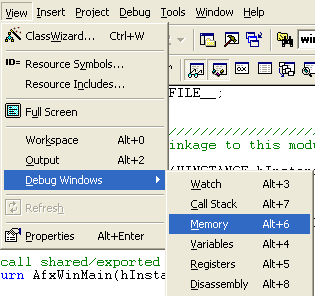
Visual C++ Debugging menu.

After the debug session been invoked, a new dynamic **Debug** menu will be displayed as shown below.



Visual C++ debug options.

Then programmer can view more information by the **View** menu and **Debug Windows** sub menu.



Visual C++ Debug window options

Visual C++ 6.0 adds a new twist to debugging with the **Edit & Continue** feature.

Edit And Continue for debug an application, change the application, and then continue debugging with the new code.

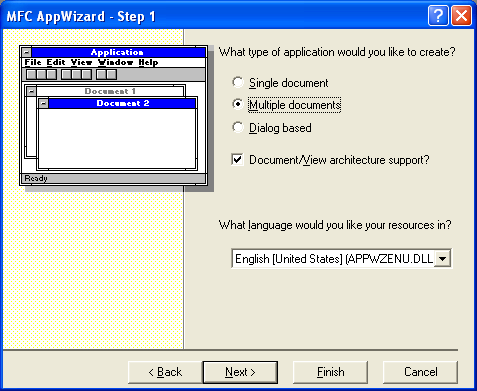
This feature dramatically reduces the amount of time you spend debugging because you no longer have to manually leave the debugger, recompile, and then debug again.

Here, simply edit any code while in the debugger and then hit the continue button

Visual C++ 6.0 automatically compiles the changes and restarts the debugger for you.

**AppWizard**

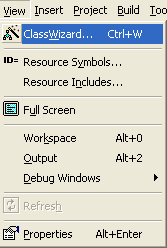
* AppWizard is a code generator that creates a working skeleton of a Windows application with features, class names, and source code filenames that you specify through dialog boxes.



* AppWizard code is minimalist code; the functionality is inside the application framework base classes.
* AppWizard gets you started quickly with a new application.
* Advanced developers can build custom AppWizards.

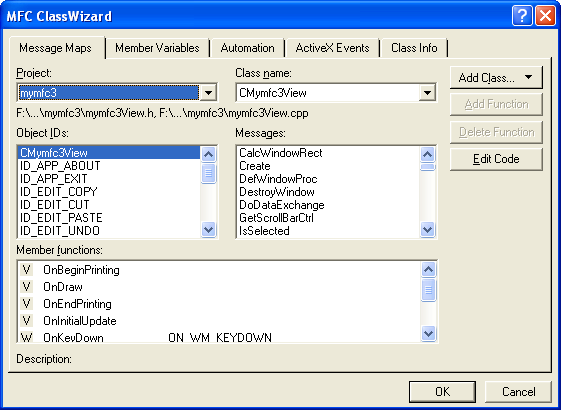
**ClassWizard**

* ClassWizard is a program (implemented as a DLL) that's accessible from Visual C++'s View menu.
* ClassWizard helps in maintaining Visual C++ class code.
* It can be access through the **View** menu and **ClassWizard…** sub menu.



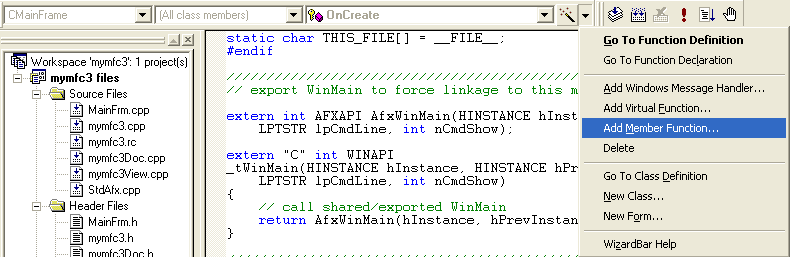
Invoking the ClassWizard through the **View** menu.

* The following is the MFC ClassWizard dialog box.



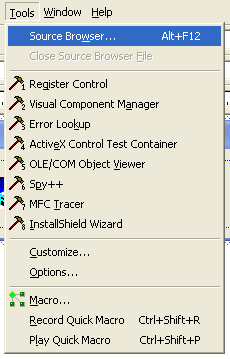
ClassWizard dialog

* When a new class, a new virtual function, or a new message-handler function (Message maps) is needed, the ClassWizard writes the prototypes, the function bodies, and necessary code to link the Windows message to the function.
* ClassWizard can update class code that are written to avoid the maintenance problems common to ordinary code generators.
* Some ClassWizard features are available from Visual C++'s WizardBar toolbar, shown below.



**The Source Browser**

* The Visual C++ Source Browser allows to edit an application from the class or function viewpoint instead of from the file viewpoint.

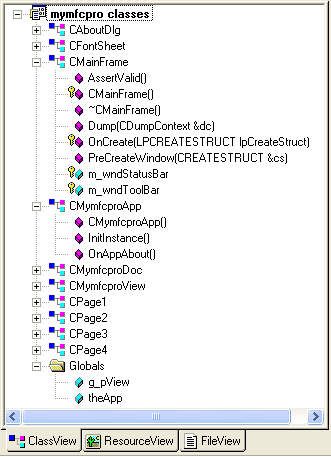


Visual C++ Source Browser.

 The browser has the following viewing modes:

* **Definitions and References**: Select any function, variable, type, macro, or class and then see where it's defined and used in your project.
* **Call Graph/Callers Graph**: For a selected function, you'll see a graphical representation of the functions it calls or the functions that call it.
* **Derived Classes and Members/Base Classes and Members**: These are graphical class hierarchy diagrams. For a selected class, you see the derived classes or the base classes plus members.
* **File Outline**: For a selected file, the classes, functions, and data members appear together with the places in which they're defined and used in your project.

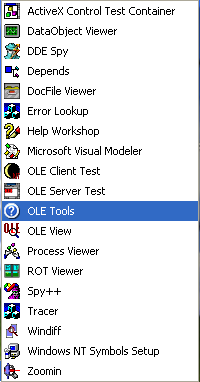
Visual C++ has a **ClassView** option allows to view all the classes in your project, showing member functions and data members. Double-click on an element can view the source code immediately.



Visual C++ ClassView.

**Windows Diagnostic Tools**

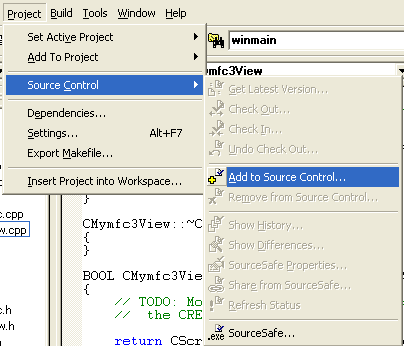
* Visual C++ 6.0 contains a number of useful diagnostic tools. For example, SPY++ gives you a tree view of your system's processes, threads, and windows.



* **PVIEW** (PVIEW95 for Windows 95) useful for killing error processes that aren't visible from the Windows 95 task list.

**Source Code Control**

* During development of Visual C++ 5.0, Microsoft bought the rights to an established source code control product named **SourceSafe**.

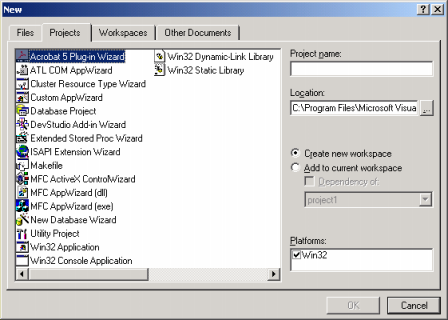


**Visual C++ Source Code control**

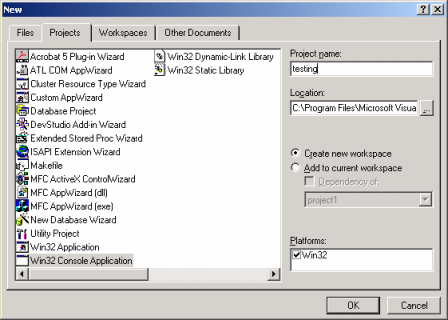
* This product has since been included in the Enterprise Edition of Visual C++ and Visual Studio Enterprise, and it is integrated into Visual C++ so that it is possible to coordinate large software projects.

**SAMPLE Visual C++ Application**

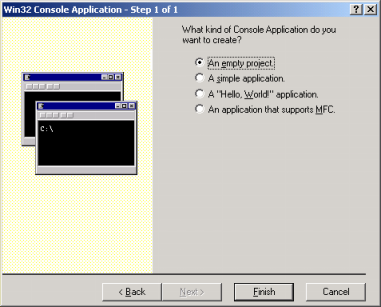
From the Visual C++ IDE, select**File** menu then**New** sub menu.  The following dialog box appears.



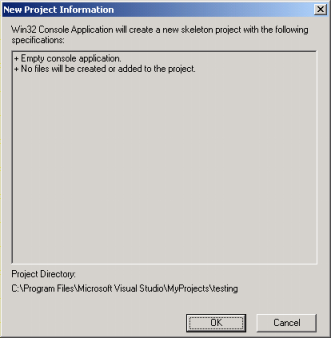
1. Then in the Projects pane, select the **Win32 Console Application**.  In Project name, type your project name (for example, **testing** in this case).  Select the **Create new workspace** radio button.  Check the **Win32** check box for the **Platforms:**.  Then click the **OK** button.



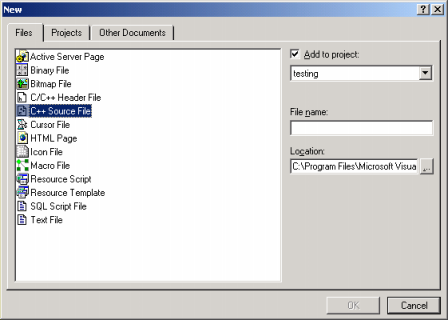
 The following dialog box appears.  Select **An empty project** radio button.  Then click**Finish** button.



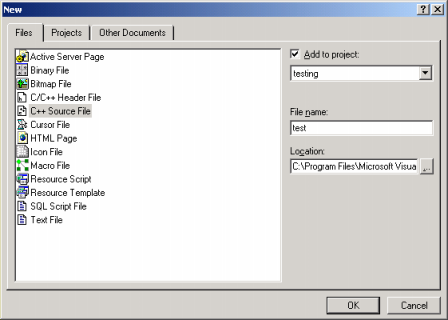
The following summary dialog box appears.  Just click the **OK** button. The following summary dialog box appears.  Just click the **OK** button.



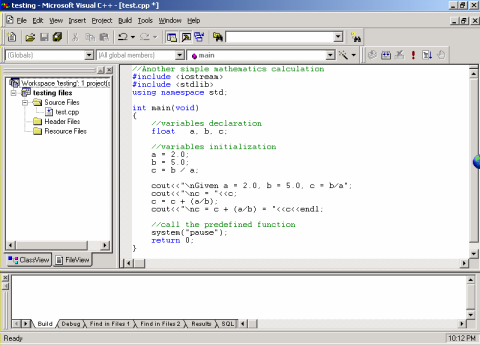
Next we are going to add item in our project.  Select **Project** menu → then **Add To Project** sub menu → then **New...** sub menu. The following dialog box appears.



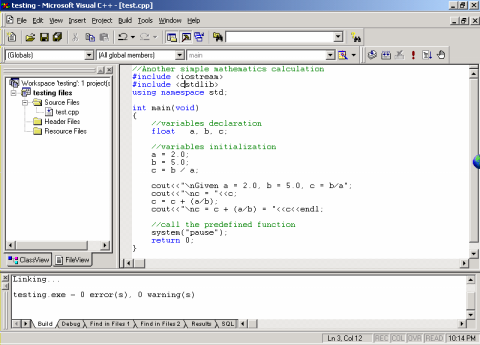
1. In building real applications there are a lot of resources involved as can be seen on the left pane.  Select the **C++ Source File**.  In the **File name:** type the source file name (for example, **testing**).  The file extension will be added automatically and it is .cpp.  Make sure the **Add to project:** tick box is ticked.  For the **Location**, up to you where to put the file by clicking the **…**, to browse your desired path.  It should be in the same path as your project.  Then click the **OK** button.



Next, type or copy source code example and paste into the editor for the source file on the right window, the editor.

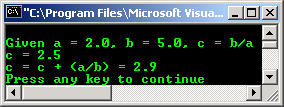


Next we are ready to build (compile and link). Select **Build** menu → **Build *filename.exe*** or press F7 or click the https://www.tenouk.com/Visualc_files/visualcpluspluscompiler008.png icon.  Any error or warning will be displayed in the output window at the bottom.  Before you can execute there must be no error.



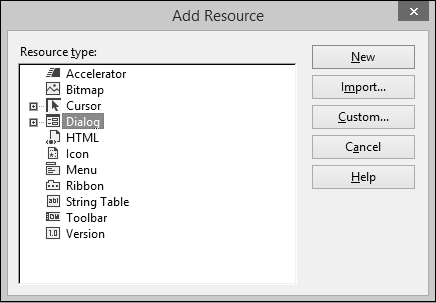
Finally executing our first program.  Select the **Build** menu → then, **Execute*filename.exe*** menu or press Ctrl+F5 or click https://www.tenouk.com/Visualc_files/visualcpluspluscompiler010.png icon.

If there is no error, the following console should be output.  If there are errors, you have to correct the errors before re building and re executing the program.  Any errors and/or warning will be displayed in the output window at the bottom.  By double clicking the error or warning, the error or warning location in the source codes will be highlighted.



**Resources**

* A resource is a text file.
* It allows the compiler to manage objects such as pictures, sounds, mouse cursors, dialog boxes, etc.
* Microsoft Visual Studio makes creating a resource file : do not have to use an external application to create or configure a resource file.
* Following are some important features related to resources.
* Resources are interface elements that provide information to the user.
* Bitmaps, icons, toolbars, and cursors are all resources.
* Some resources can be manipulated to perform an action such as selecting from a menu or entering data in dialog box.
* An application can use various resources that behave independently of each other, these resources are grouped into a text file that has the \*.rc extension.
* Most resources are created by selecting the desired one from the Add Resource dialog box.



Resource Identifiers

* An identifier is a symbol/constant integer
* Identifier name starts with ID.
* To make identifier name recognizable to MFC function, use a macro called MAKEINTRESOURCE

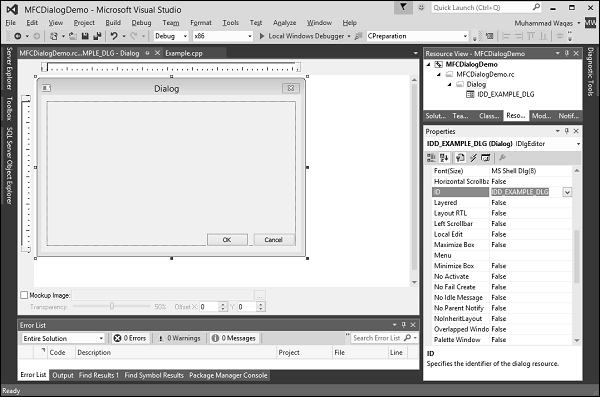
SYNTAX: LPTSTR MAKEINTRESOURCE(Word Identifier);

* It consists of two parts − a text string (symbol name) mapped to an integer value (symbol value).

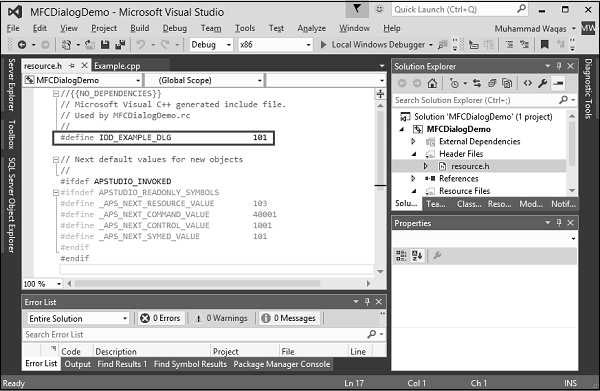
Resource editors

* Symbols provide a descriptive way of referring to resources and user-interface objects
* When you create a new resource or resource object, the resource editors provide a default name for the resource, for example, IDC\_DIALOG1, and assign a value to it.
* The name-plus-value definition is stored in the Resource.h file.

Step 1 – Eg : Created a dialog box and its ID is IDD\_EXAMPLE\_DLG.



**Step 2** − Go to the Solution Explorer, see the resource.h file under Header Files. Continue by opening this file in editor and see the dialog box identifier and its integer value as well.



* Different types of Ids in VC++
* The MFC ID naming defines different prefixes for different resource types

|  |  |
| --- | --- |
| Object ID | Description |
| IDR\_ | Multiple resource types (used for menus, accelerators) |
| IDD\_ | Dialog template resource (IDD\_DIALOG1) |
| IDC\_ | Cursor resources |
| IDI\_ | Icon resources |
| IDB\_ | Bitmap resources |
| IDS\_ | String resources |

**Menus**

Three types of menus in windows application

System menu, popup menu and drop down menu

System menu: it commands for restoring, moving, size, minimising, maximising, and closing window. Menu displays when mouse right click title bar/ clicking left top corner icon.

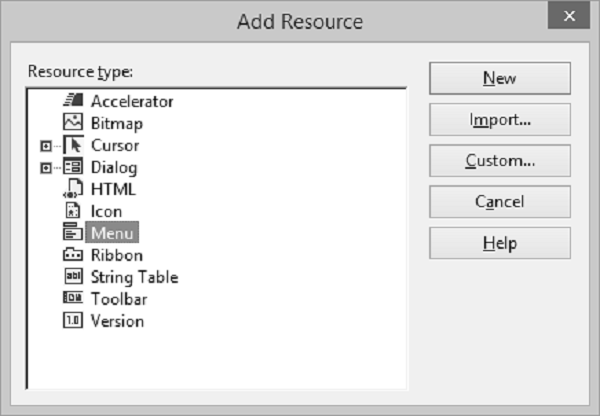
Popup menu: mouse right click and menu will popup anywhere on the screen

Drop down menu: by clicking the menu bar, choose the appropriate menu item from top of the window.

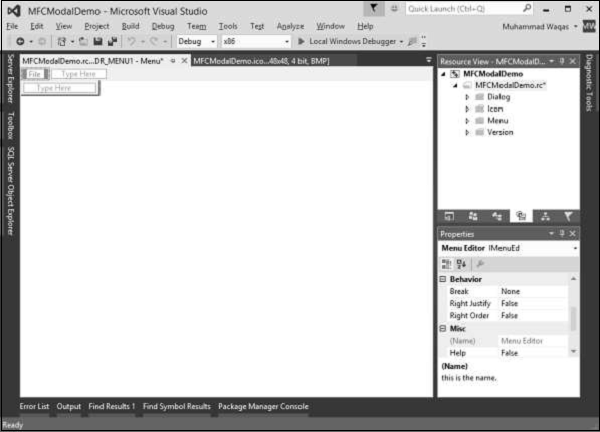
Menu editor is used to create and edit menus.

To create a menu, follow the steps given below:

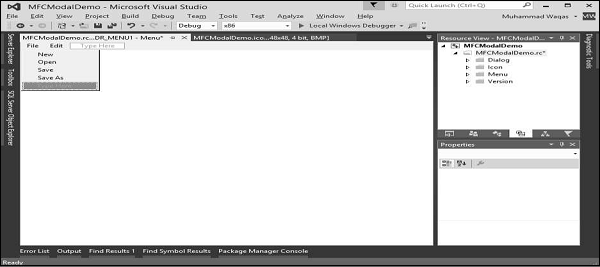
**Step 1** − Right-click on your project and select Add → Resources. You will see the Add Resources dialog box.



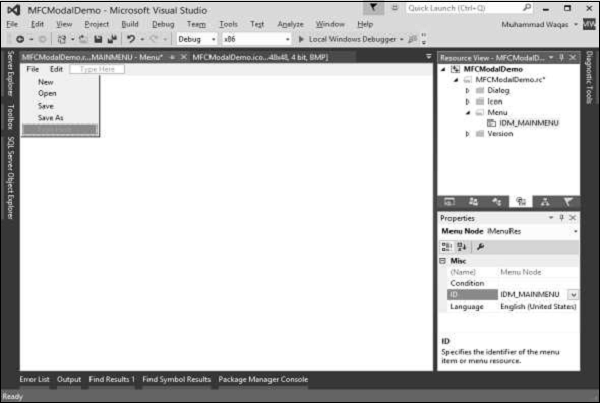
**Step 2** − Select Menu and click New. You will see the rectangle that contains "Type Here" on the menu bar.



**Step 3** − Write some menu options like File, Edit, etc. as shown in the following snapshot.

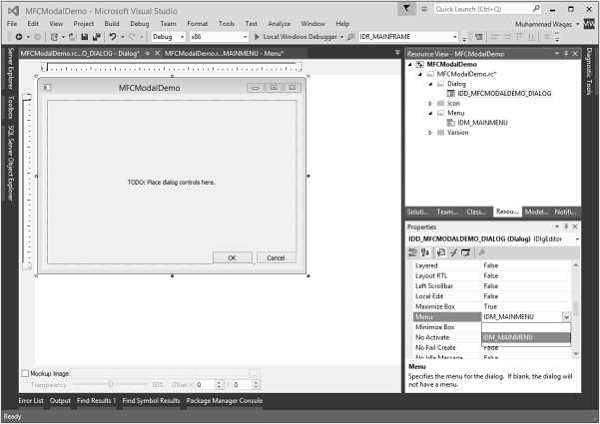


**Step 4** − If you expand the Menu folder in Resource View, you will see the Menu identifier IDR\_MENU1. Right-click on this identifier and change it to IDM\_MAINMENU.



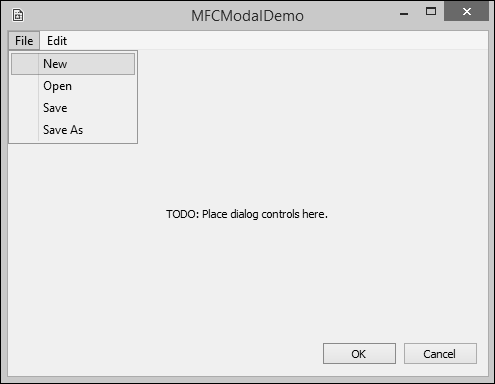
**Step 5** − Save all the changes.

**Step 6** − We need to attach this menu to our dialog box. Expand your Dialog folder in Solution Explorer and double click on the dialog box identifier.



**Step 7** − You will see the menu field in the Properties. Select the Menu identifier from the dropdown as shown above.

**Step 8** − Run this application and you will see the following dialog box which also contains menu options.



* VBX controls

**Windows controls** are objects that users can interact with to enter or manipulate data. They commonly appear in dialog boxes or on toolbars. There are various types of controls −

* A **text based control** which is used to display text to the user or request text from the user.
* A **list based control** displays a list of items.
* A **progress based control** is used to show the progress of an action.
* A **static control** can be used to show colors, a picture or something that does not regularly fit in the above categories.

A static control is an object that displays information to the user without his or her direct intervention. It can be used to show colors, a geometric shape, or a picture such as an icon, a bitmap, or an animation. Following are the static controls −

* Static Text
* Picture Control
* Group Box

A **bitmap button** displays a picture or a picture and text on its face. This is usually intended to make the button a little explicit. A bitmap button is created using the **CBitmapButton class**, which is derived from CButton.

A **command button** is an enhanced version of the regular button. It displays a green arrow icon on the left, followed by a caption in regular size. Under the main caption, it can display another smaller caption that serves as a hint to provide more information.

A **static control** displays a text string, box, rectangle, icon, cursor, bitmap, or enhanced metafile. It is represented by **CStatic class**. It can be used to label, box, or separate other controls. A static control normally takes no input and provides no output.

A **list box** displays a list of items, such as filenames, that the user can view and select. A List box is represented by **CListBox class**. In a single-selection list box, the user can select only one item. In a multiple-selection list box, a range of items can be selected. When the user selects an item, it is highlighted and the list box sends a notification message to the parent window.

A **combo box** consists of a list box combined with either a static control or edit control. it is represented by **CComboBox class**. The list-box portion of the control may be displayed at all times or may only drop down when the user selects the drop-down arrow next to the control.

A **radio button** is a control that appears as a dot surrounded by a round box. In reality, a radio button is accompanied by one or more other radio buttons that appear and behave as a group.

A checkbox is a Windows control that allows the user to set or change the value of an item as true or false.

An **Image List** is a collection of same-sized images, each of which can be referred to by its zero-based index. Image lists are used to efficiently manage large sets of icons or bitmaps. Image lists are represented by **CImageList class**.

An **Edit Box** is a rectangular child window in which the user can enter text. It is represented by **CEdit class**

A **Rich Edit** Control is a window in which the user can enter and edit text. The text can be assigned character and paragraph formatting, and can include embedded OLE objects. It is represented by **CRichEditCtrl class**.

A **group box** is a static control used to set a visible or programmatic group of controls. The control is a rectangle that groups other controls together.

A **Spin Button** Control (also known as an up-down control) is a pair of arrow buttons that the user can click to increment or decrement a value, such as a scroll position or a number displayed in a companion control. it is represented by **CSpinButtonCtrl class**.

**Updown Controls**

A **progress bar control** is a window that an application can use to indicate the progress of a lengthy operation. It consists of a rectangle that is gradually filled, from left to right, with the system highlight color as an operation progresses. It is represented by **CProgressCtrl class**.

A **progress bars** is a window that an application can use to indicate the progress of a operation

A **timer** is a non-spatial object that uses recurring lapses of time from a computer or fromyour application. To work, every lapse of period, the control sends a message to the operating system. Unlike most other controls, the MFC timer has neither a button to represent it nor a class. To create a timer, you simply call the CWnd::SetTimer() method. This function call creates a timer for your application. Like the other controls, a timer uses an identifier.

The date and time picker control (**CDateTimeCtrl**) implements an intuitive and recognizable method of entering or selecting a specific date. The main interface of the control is similar in functionality to a combo box. However, if the user expands the control, a month calendar control appears (by default), allowing the user to specify a particular date. When a date is chosen, the month calendar control automatically disappears.

If you need to display a picture for your application, Visual C++ provides a special control for that purpose.

The **Image editor** has an extensive set of tools for creating and editing images, as wellas features to help you create toolbar bitmaps. In addition to bitmaps, icons, and cursors, you can edit images in GIF or JPEG format using commands on the Image menu and tools on the Image Editor Toolbar.

A **Slider Control** (also known as a trackbar) is a window containing a slider and optional tick marks. When the user moves the slider, using either the mouse or the direction keys, the control sends notification messages to indicate the change. There are two types of sliders − horizontal and vertical. It is represented by **CSliderCtrl class**.

A **scrollbar** is a graphical control element with which continuous text, pictures or anything else can be scrolled in two directions along a control by clicking an arrow. This control can assume one of two directions − horizontal or vertical. It is represented by **CScrollBar** class.

A **Tree View Control** is a window that displays a hierarchical list of items, such as the headings in a document, the entries in an index, or the files and directories on a disk. Each item consists of a label and an optional bitmapped image, and each item can have a list of sub items associated with it. By clicking an item, the user can expand and collapse the associated list of sub items. It is represented by **CTreeCtrl** class.

List Control: Encapsulates the functionality of a List View Control, which displays a collection of items each consisting of an icon (from an image list) and a label. It is represented by **CListCtrl** class. A list control consists of using one of four views to display a list of items.

**Microsoft Foundations Class Library[MFC]**

* MFC is a collection of C++ classes for Microsoft Windows Operating System
* All classes are connected with hierarchical inheritance
* It simplifies the writing of Windows applications
* It is an Microsoft Win 32 Application framework for writing C/C++/VC++ applications using a collection of classes written in C++
* In Windows SDK (software development kit), 80 to 90 lines of code are used to create a empty window; using MFC only few lines of code are used to get a window
* We can develop applications more quickly and easily using MFC programming (using classes )
* Using these classes, create objects like windows, dialog boxes, menus and window controls
* Main categories of Classes are : Applications, Windows, Menus, Dialog boxes, Document and views, Controls, Graphics, Archival and file, and other support classes

**CObject Class**

* In MFC class hierarchy, almost all classes are derived from CObject class.
* CObject is the base class of most of the MFC classes
* Many diverse classes in MFC share the common functionality through CObject class

**The Application Class (CWinApp)**

* All MFC program start with an application object
* Application object is derived from MFC’s CWinApp class
* It provides initialization, runtime, and ending services for windows program
* Here CWinApp registers, creates and shows an application’s main window; sets application’s messaging loop running (application interact with user and windows); deletes the application when the user is finished.
* Additional services of CwinApp are: providing access to application’s command line, handling file activity, processing windows messaging, and detecting when windows is idle
* Every MFC program instantiates CWinApp derived object

**Window Classes [CWnd]**

* It is used to create customized main window
* MFC features different types of windows: Frame windows, view windows, MDI windows, Dialog boxes etc. All these window classes are derived from the **Cwnd** base class, so other derived classes holds the basic functionalities available in CWnd class
* Functionality includes: initialization, positioning, sizing, painting, coordinate mapping, scrolling, message handling etc.

**Frame Windows [CFrameWnd]**

* CFrameWnd class for Frame windows
* It supply all the functionality of the CWnd class to handle document – view, menu bar, control bar, accelerators, etc
* CFrameWnd class contains only two public data members:

m\_bAutoMenuEnable: controls menu items are enabled or disabled automatically

rectDefault: controls the window’s starting size, position etc

* Single document interface (SDI) application is derived from the CFrameWnd class
* Frame window has complete window controls like minimize, maximize/restore, close buttons, application menu bar, tool bar, and status bar
* Using MFC we can create any type of frame window using CFrameWnd class

**View Windows [CView]**

* MFC offers Document – View architecture.
* The CDocument class is responsible for storing a window’s data
* The CView class Displays the data in frame window’s client area

It enables the user to manipulate data

Printing the data

* A document can have more than one view
* MFC derives many view classes from general CView class like: CCtrlview, CEditView, CListView, CRichEditView, CTreeView, CScrollView, CFormView, CDaoRecordView,CRecordView

**MDI Windows [CMDIFrameWnd]**

* Multiple document Interface (MDI) enables the user to open more than one document simultaneously
* MFC provides CMDIFrameWnd class for extra functionality needed by MDI frame windows
* MDIChildWindow class manages the main and childwindow menu bars, forwarding messages to MDI child, arranging MDI child windows etc

**Dialog Boxes [CDialog]**

* Dialog boxes retrieve information from the user
* MFC has a basic dialog box class, CDialog which handles all the details of constructing, executing, and closing a dialog box
* It is used for transferring data to and from the dialog box
* These dialog classes includes: CColorDialog, CFileDialog, CFindReplaceDialog, CFontDialog, and CPrintDialog
* h
* g