To:
Tenouk

# C LAB WORKSHEET 3_1
# Building, Running and Debugging C programs 2

In this page, we will:

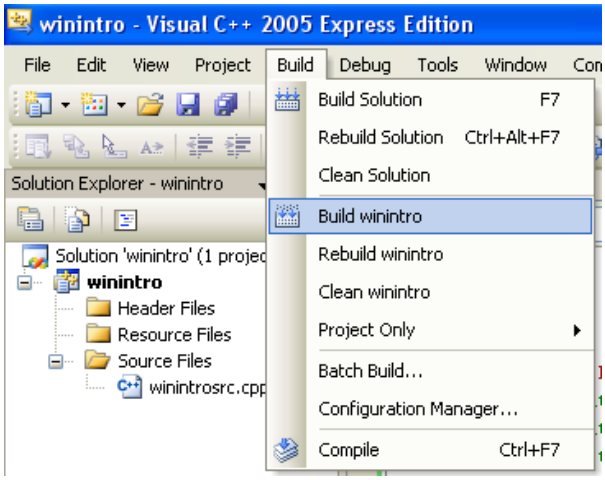
1. Building and running our first C program.
2. Debugging the C program.
3. Some information about managed, unmanaged code and .Net framework.

**Building and Running Our C Project**

The following steps show how build and run our C program.

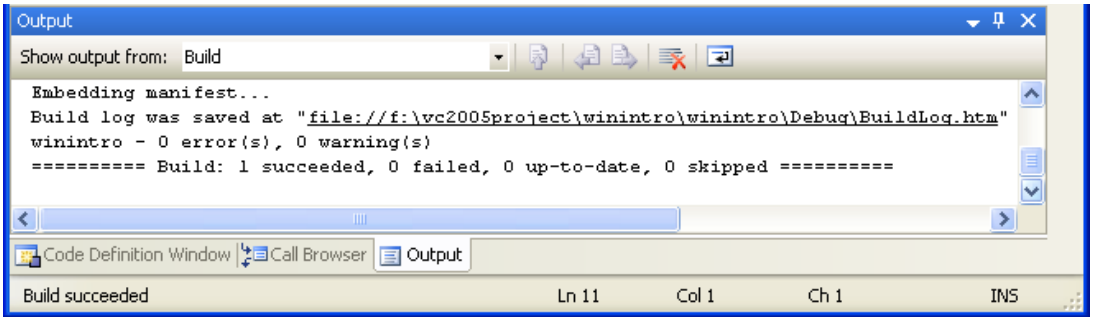
1. Now we are going to build (compile and link) and run (Debug) our C program. Edit the previous code as shown below.

```
#include <stdio.h>

void main(void)
{
    printf("HELLO WoRlD!!!\n");
    // statement 2;
    // statement 3;
    // more C statements as
needed...
}
```

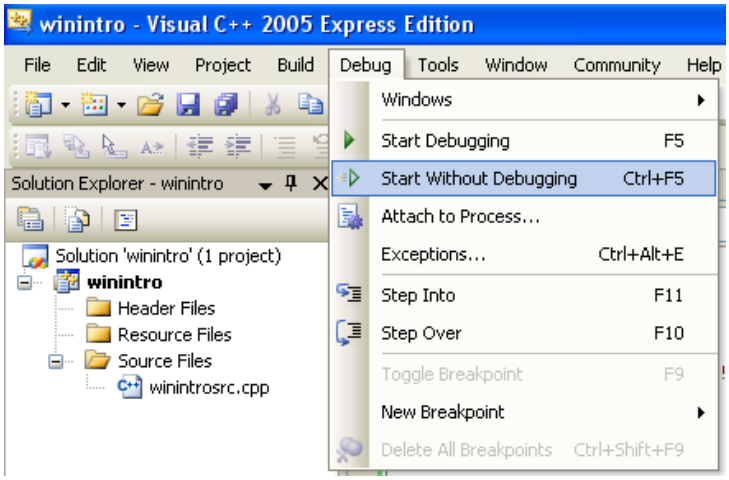



2. Next, select **Build ⟶ Build *your_project_name*** menu.

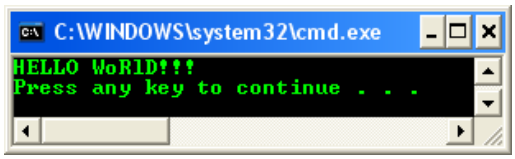
   Notice the **Output** windows at the bottom as shown below.




3. If there is no error(s), select **Debug ⟶ Start Without Debugging** menu.



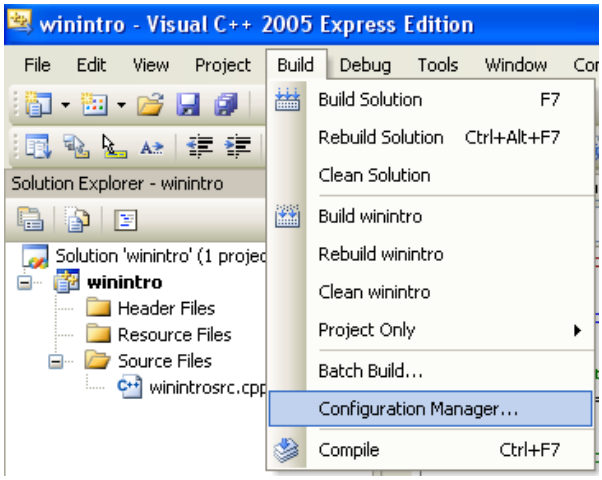
4. Well, you should see the output as shown on the right. Congratulation!



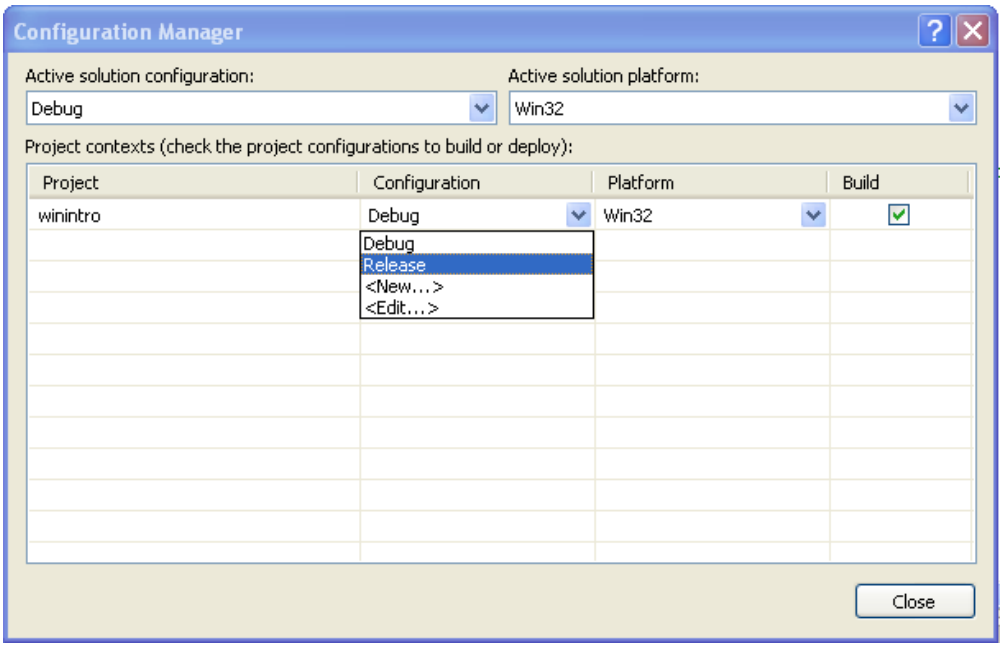

**Debugging Your Codes**

Most of your C/C++ program development will be done in Debug mode. For Alfa, Beta or Release version, you may build your program in Release mode that may include other processes such as code optimization and program's size reduction. Then, you may need programs such as Install Shield to create the deployment (installation) package, media and distribution.

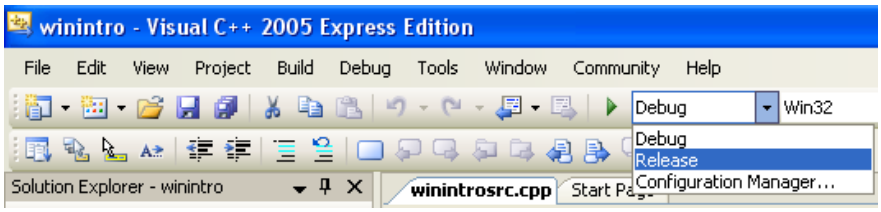
1. Select **Build** ⟶ **Configuration Manager** menu.



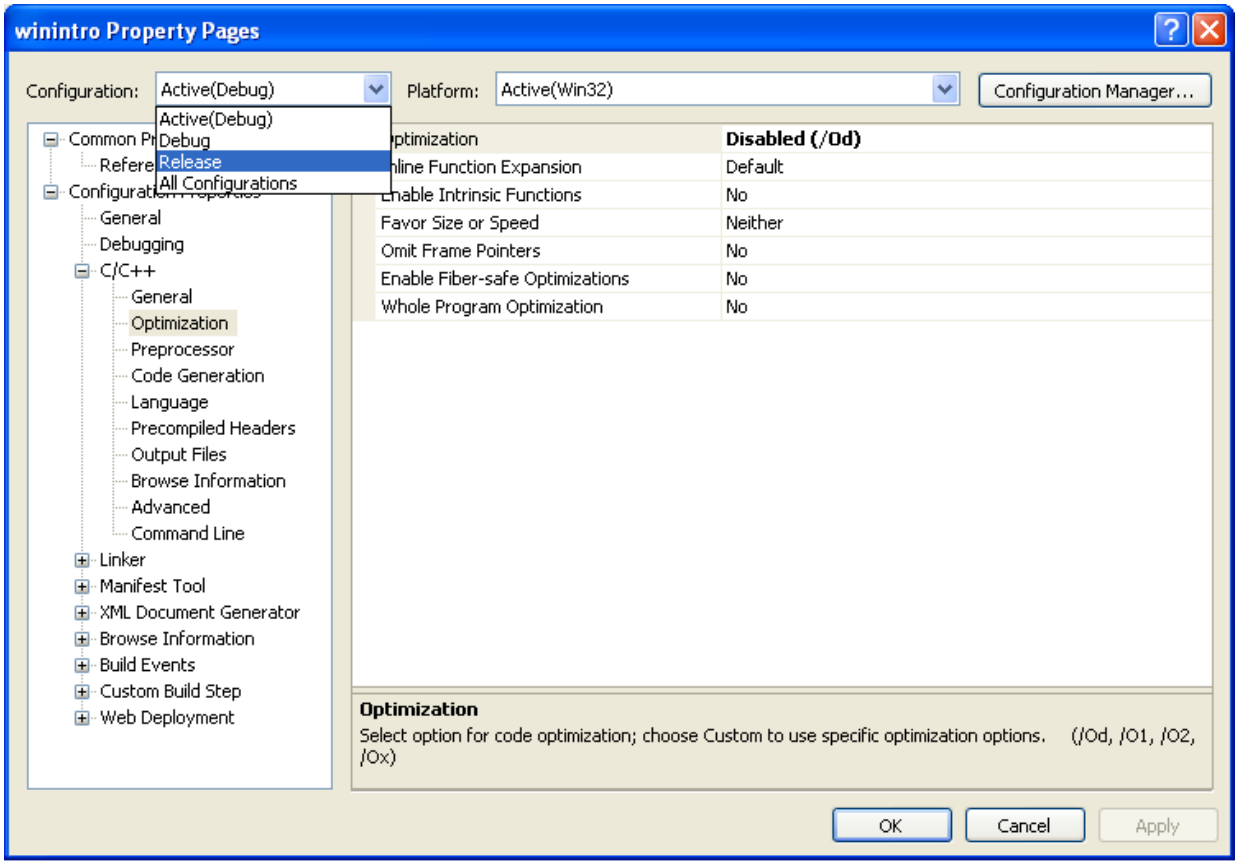

2. Through the **Configuration Manager**, you can build your project for the **Debug**, **Release** or **Customized** version.




3. You can also change the configuration through the shortcut as shown below.




4. Other options can be changed/enabled/disabled through the project property form for those configurations as shown below.




5. Stick to the **Debug** mode and let go through the debug process, though our code doesn't have any bug for the moment.
6. Edit the previous source code as shown on the right.

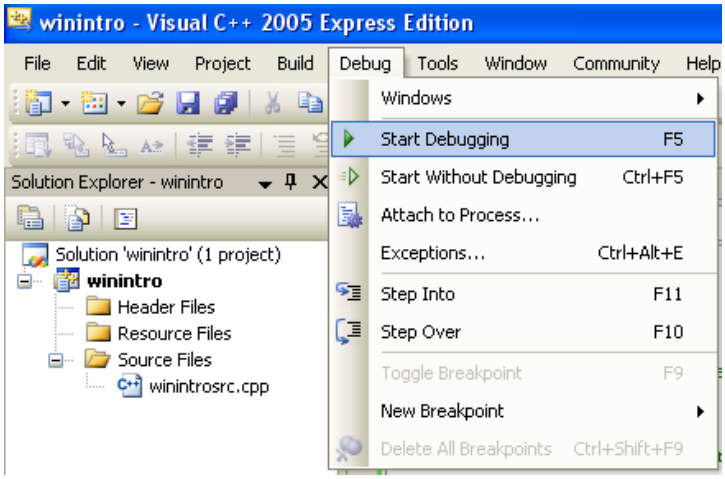
```
#include <stdio.h>

void main(void)
{
    // declare x, y, z as integer data type and initialize y and z variables
    int x, y=20, z=30;
    // put some string on the standard output, console/terminal/screen
    printf("HELLO WoRlD!!!\n");
    printf("We put a simple math operation and let debug this program.\n");
    // a very simple math operation
    x = y + z;
    // print the total to the standard output...
    printf("The total of x = %d\n", (y+z));
    // statement 2;
    // statement 3;
    // more C statements as needed...
}
```

7. You can use the following debug types.

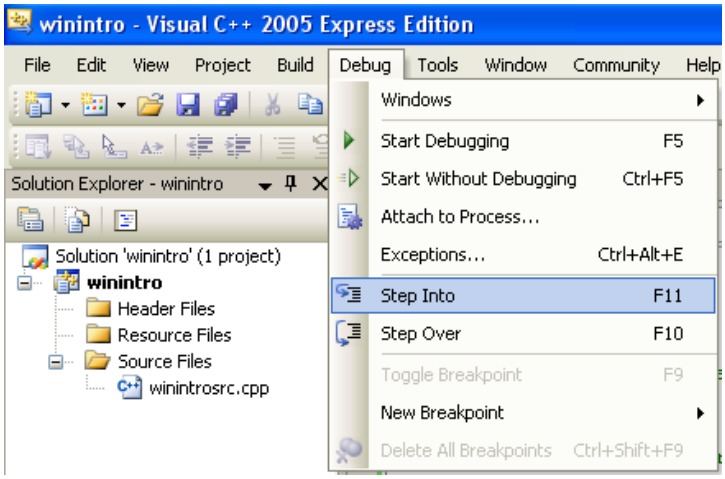
| Debug option | Description |
|---|---|
| Step Into (F11) | Executes code one statement at a time, following execution into function calls. |
| Step Over (F10) | Executes the next line of code but does not follow execution through any function calls. |
| Step Out (SHIFT + F11) | Executes the remaining lines of a function in which the current execution point lies. |
| Stop Debugging (SHIFT + F5) | Stop the debug process. |

Table 2.

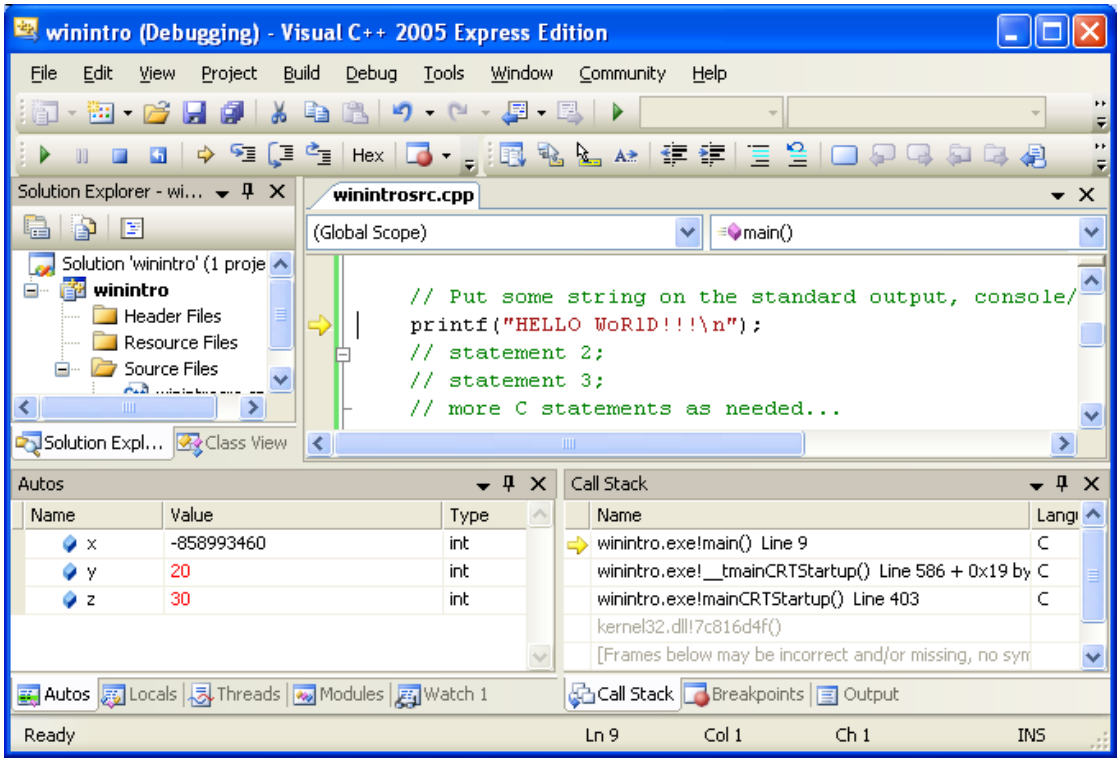
8. Select **Debug** ⟶ **Start Debugging** menu. By default, this menu will debug your code until the end of file.




9. Next, select **Debug ⟶ Step Into** menu. This menu will debug your code line by line.

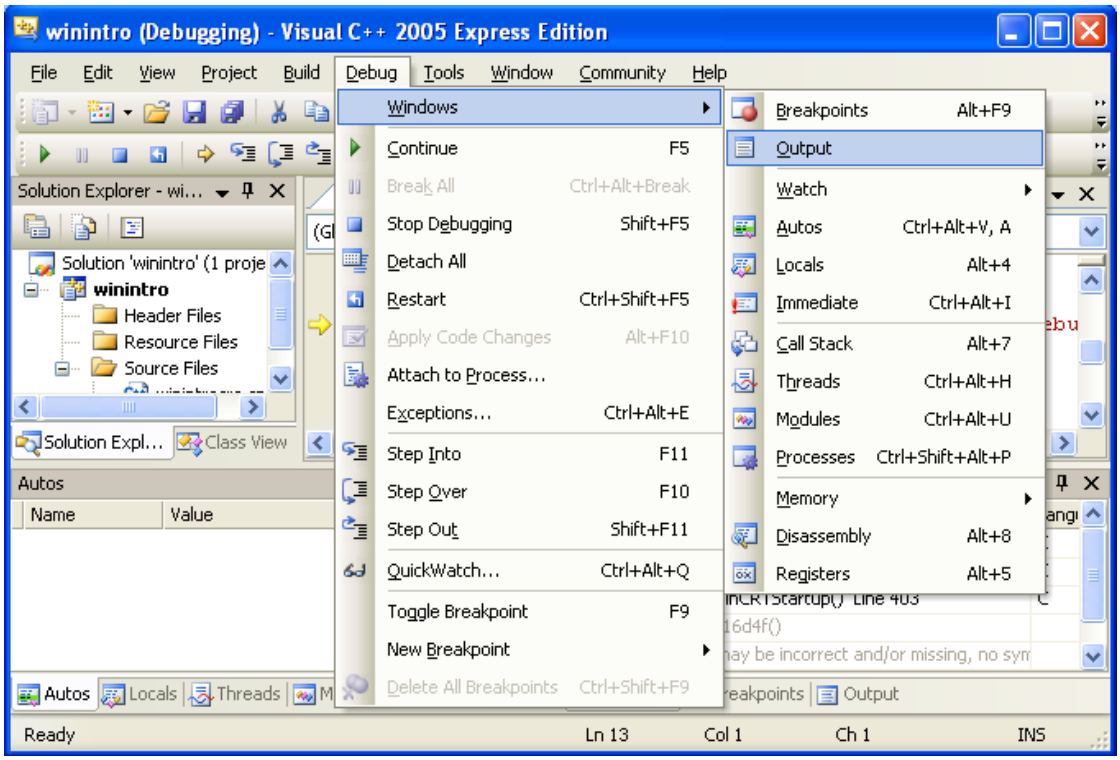



10. Press **F11** to go to the next line of code as shown below.

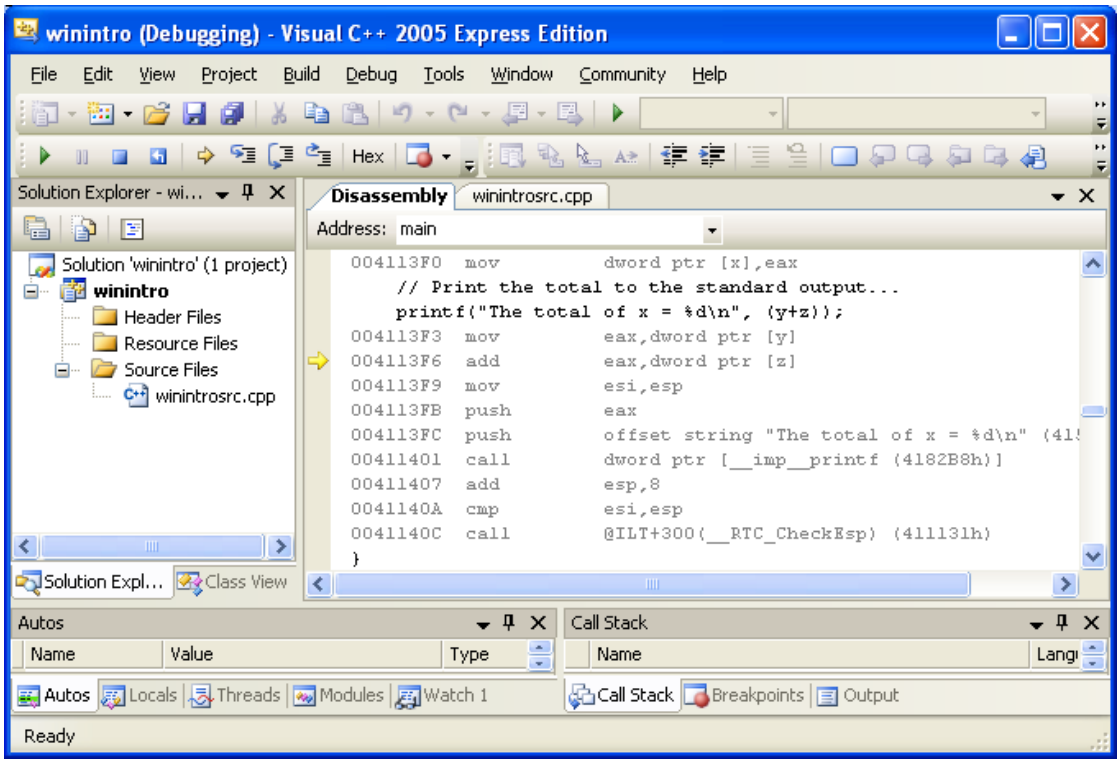



11. Notice that the output console (command prompt window) already there.

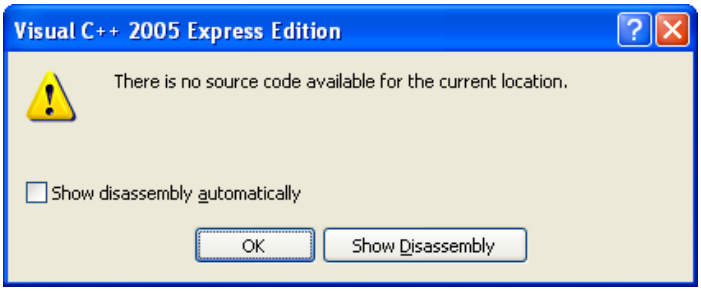
12. During the debug process you can view or check a lot more information about your program execution by selecting the **Debug** ⟶ **Windows** menu as shown below.



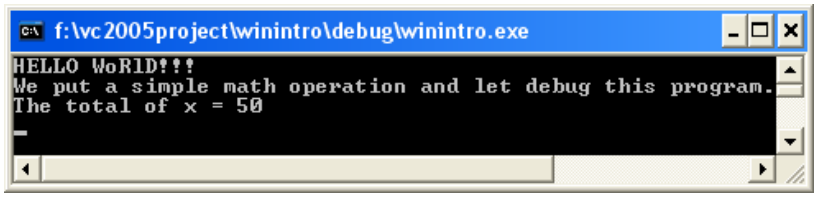

13. The following Figure shows the Assembly code of your program when you select the **Disassembly** submenu.



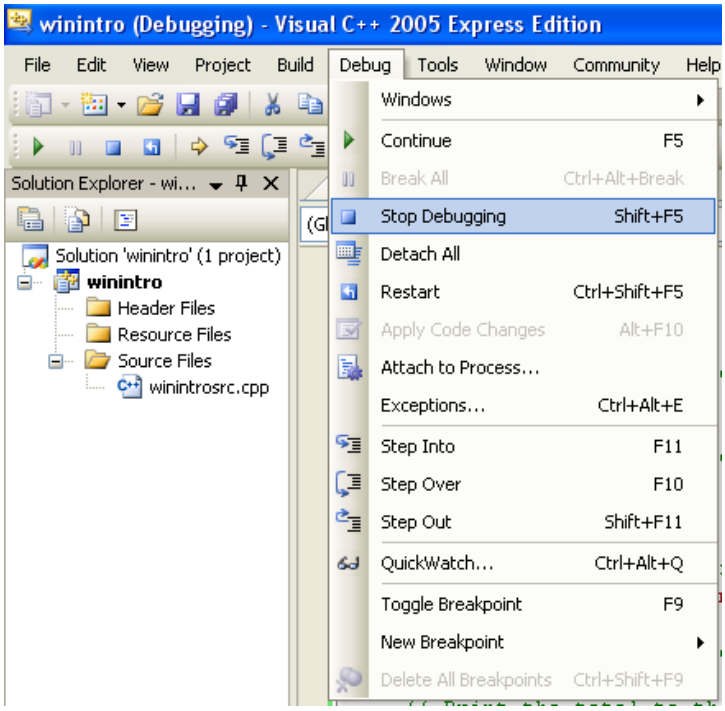

14. If the following form shown when you finish debugging until the end of file, just click the **OK** button.



15. You should see the program output as shown below.

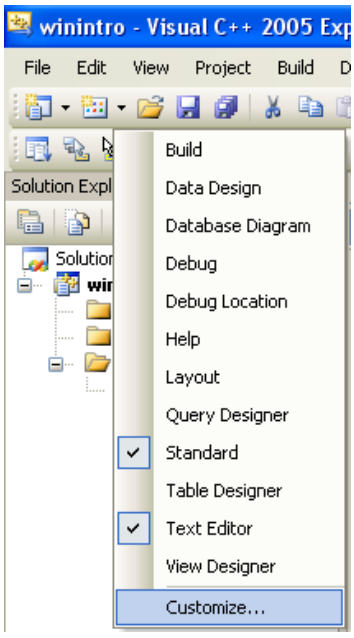



16. You can stop the debugging process anytime by using the following menu.

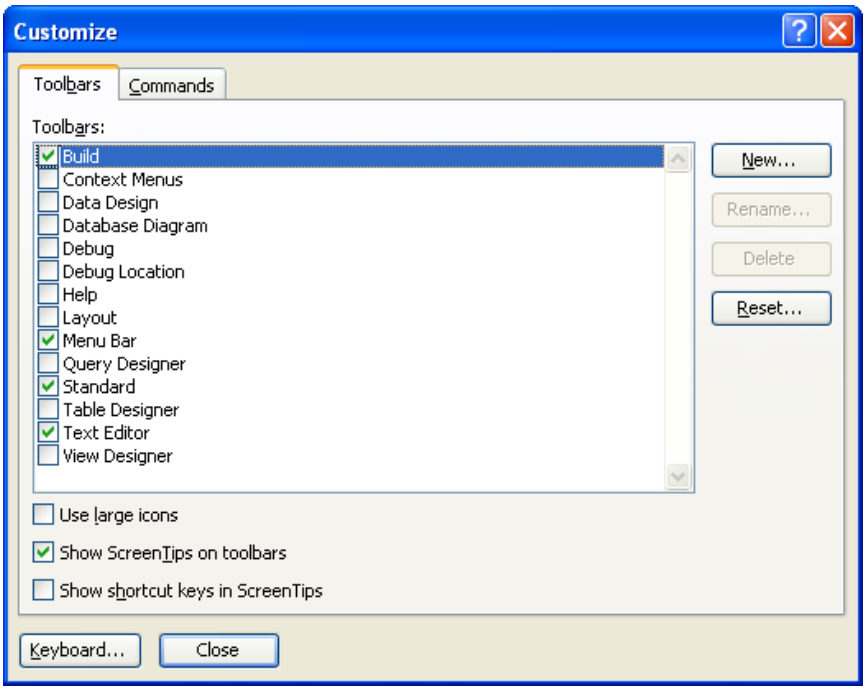



17. Don't forget that, similar to other Windows applications, you can use the shortcut keys or the button in the toolbar. The following is the Debug toolbar.

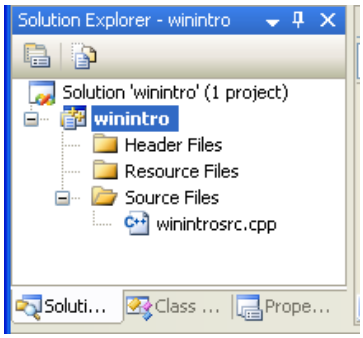

18. You can add or remove those buttons and/or toolbar groups by right clicking anywhere on the toolbar area. You can select the toolbar group or refine more buttons selection through the **Customize** menu as shown below.



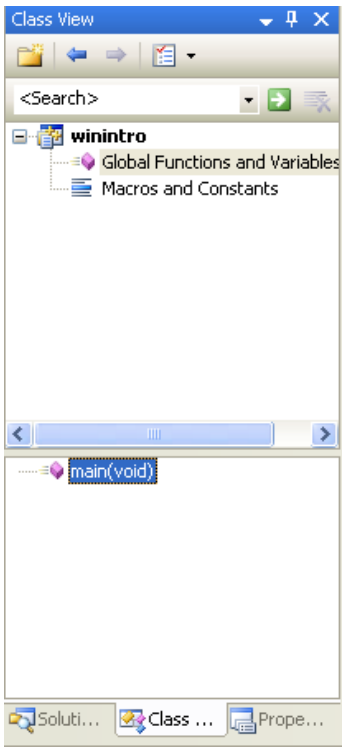
19. The following Figure shows the available toolbars. You can select the individual icon through the **Commands** tab.



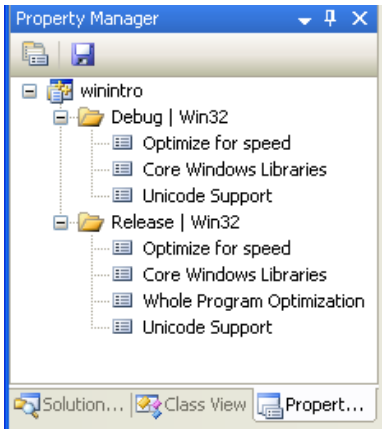

20. Let browse other things in our VC++ 2005 EE IDE. The following Figure shows the **Solution Explorer** window. You can see all the project resources here such as source, header, bitmap and icon files. By double clicking the file, it will be opened on the left window, ready for editing.



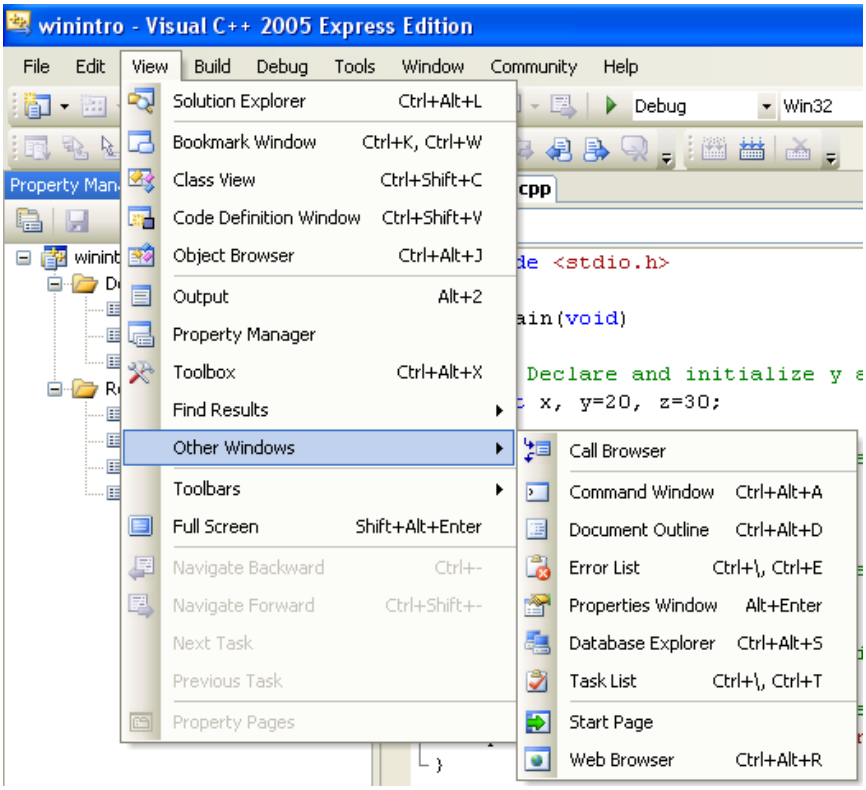

21. The following shows the **Class View** window. If we do object oriented or Microsoft Foundation Class (MFC) project, we can see many classes, micros and constants here. It is similar to the previous version of the VC++ IDE.



22. The following is a **Property Manager** window, similar to VC++ 2003 but not available in Tenouk's VC++ 6.0 copy. It resembles the Visual Basic programming environment. It just another way to access the project property page as you seen previously. Keep in mind that for object oriented or MFC programming, there will be other views.




23. Other VC++ windows can be accessed through the **View** ⟶ **Other Windows** menu.




**Questions:**

If you can answer the following questions, you should already understand this module. Please answer it and submit to your instructor at the end of the class.

1. List down the steps to create your VC++ empty Win32 Console Application project. **Ans:** As in the notes.
2. List down the steps to add a C/C++ source file to your project in #1. **Ans:** As in the notes.
3. List down the steps to build and run (in debug mode) your C/C++ program. **Ans:** As in the notes.
4. List down the steps to debug your program. **Ans:** As in the notes.
5. Why you need to debug your program? **Ans:** To find and rectify any compile time, run-time (linking as well) and logical errors.
6. During the debug process, gives five information that you can view or check. **Ans:** threads, modules, locals, call stack and registers.
7. How your VC++ compiler differentiate between C and C++ codes? **Ans:** Through the "Compile as C Code (/TC)" for C and "Compile as C++ Code (/TP)" project's settings. Another one is the source code file extension: .c for C and .cpp for C++.
8. What is the difference between managed and unmanaged code? **Ans:** a managed code is run within the runtime engine such as CLR of the .NET Framework, cannot run without it whereas unmanaged code runs by itself, launched from the OS, share other OS routines.
9. Give examples of the Programming Language that categorized as unmanaged and managed codes. **Ans:** Native C is unmanaged and C++ .NET, VB .NET and C# are managed code examples.