Department of Information Systems and Technologies

CTIS151 – Introduction to Programming

SPRING 2023 - 2024

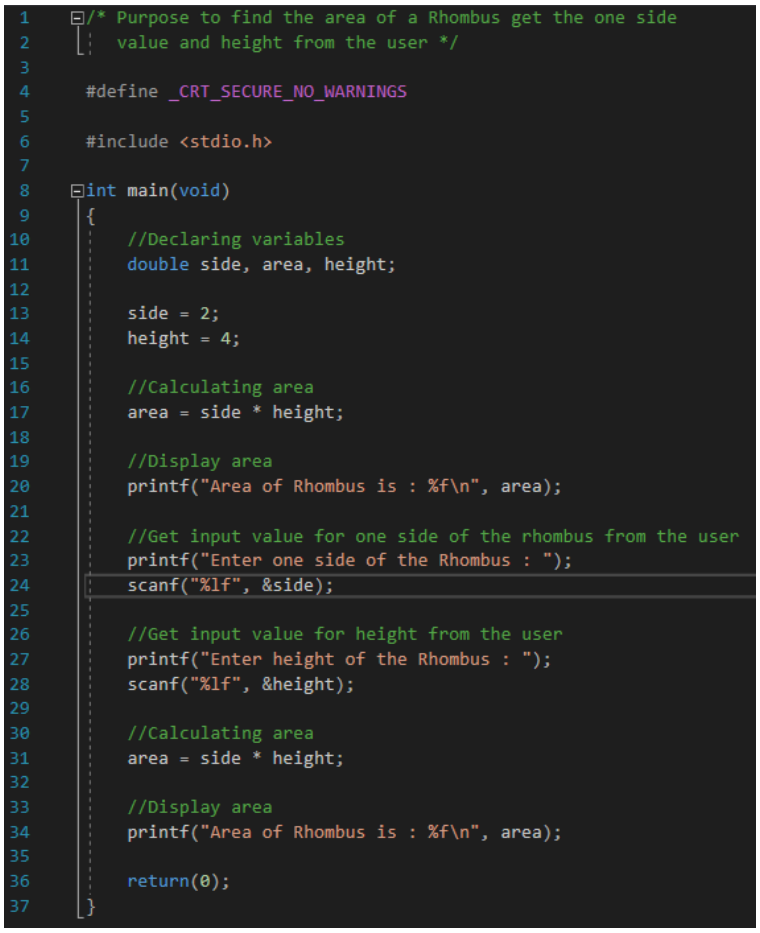
Lab Guide #2

OBJECTIVE : To get acquainted with Microsoft Visual Studio. You will learn how to:
1. Code a given program using the editor.
2. Compile, build and run your program.
3. Debug your program.

Instructors : Burcu LİMAN
Assistants : Engin Zafer KIRAÇBEDEL, Sıla YAPICI

Q1.

1. Open Visual Studio application wizard as you learned in the last lab hour.
   - Open a new Windows Desktop Wizard, give **LG2_Q1** to the *Project Name* and **LG2_Sols** to the solution name
   - Select the destination folder as **D:\CTIS151\Lab2**
2. Open a new C++ Source File and give File Name as Q1.cpp.
3. Type the following source program exactly as you see below. Omitting, the line numbers.



```
/* Purpose to find the area of a Rhombus get the one side
   value and height from the user */

#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main(void)
{
    //Declaring variables
    double side, area, height;

    side = 2;
    height = 4;

    //Calculating area
    area = side * height;

    //Display area
    printf("Area of Rhombus is : %f\n", area);

    //Get input value for one side of the rhombus from the user
    printf("Enter one side of the Rhombus : ");
    scanf("%lf", &side);

    //Get input value for height from the user
    printf("Enter height of the Rhombus : ");
    scanf("%lf", &height);

    //Calculating area
    area = side * height;

    //Display area
    printf("Area of Rhombus is : %f\n", area);

    return(0);
}
```

Project Name: LG2_Q1
File Name: Q1.cpp

<u>Example Run</u>:

```
Area of Rhombus is: 8.000000
Enter one side of the Rhombus: 5
Enter height of the Rhombus: 8
Area of Rhombus is: 40.000000
```

LINE 1,2:

```
/* Purpose to find the area of a Rhombus get the one side
   value and height from the user */
```

LINE 10, 16, 19, 22, 26, 30, 33:

```
//Declaring variables
//Calculating area
//Display area
//Get input value for one side of the rhombus from the user
//Get input value for height from the user
//Calculating area
//Display area
```

As you can see in the lines there are two types of comments in C, Block Comments and In-Line Comments. The characters **/*** start a comment, and the characters ***/** end the comment.

Block Comments

Block comments can span several lines.

```
/*
   This is a block comment. The body of the
   comment is placed over several lines. The
   comment ends when the terminating characters
   are provided.
*/
```

In-Line Comments

An in-line comment is a comment that is on a single line.

```
 // This is an in-line comment
```

You can also use **/*** and ***/** characters for in-line comments such as:

```
/* This is an in-line comment */
```

Nested comments are not allowed in C. The following comment statements causes an error.

```
 /*
    Start of a block comment.
    /* An illegal nested comment */
    End of a block comment.
*/
```

Line 6: `#include <stdio.h>`

As part of compilation, the C compiler runs a program called the C preprocessor. The preprocessor is able to add and remove code from your source file. In this case, the directive "**#include**" tells the preprocessor to include code from the file **stdio.h**. This file contains declarations for functions that the program needs to use. A declaration for the **printf** function is in this file.

`#define PI 3.14`

This statement defines the variable **PI** as a constant and holds the value **3.14** as its content.

Line 8: `int main(void)`

This statement declares the main function. A C program can contain many functions but must always have one main function. A function is a self-contained module of code that can accomplish some task. Functions will be examined in a later lab guide.

Line 9: {

This opening brace denotes the start of the program.

Line 11: `double side, area, height;`

The program requires variables in which to store the values for **side,** **area** and **height**. The declaration, **double side, area** and **height;** provides a temporary storage called **side,** **area** and **height** that has a data type of double (float number).

Line 13,14: `side = 2;`
`height = 4;`

These lines are for assigning a value to variables **`side = 2;`** and **`height = 4;`** means variable side **holds the value 2 and height holds the value 4 as its content**.

Line 23, 24: `printf("Enter one side of the Rhombus : ");`
`scanf("%lf", &side);`

printf is a function from the standard C library that is used to print strings to the standard output, normally your screen. The compiler links code from these standard libraries to the code you have written to produce the final executable. **printf** statement is used to prompt the user to enter r.

scanf function is used to read a double value from the user into the variable **side**.

Line 31: `area = side * height;`

This line is to calculate the area and to hold this result in **`area`** variable's content. This means;

`area = 2 * 4;` so, the content of **`area`** will be 8.000000

Line 34: `printf("Area of Rhombus is : %f\n", area);`

The "**\n**" is a special format modifier that tells the **printf** to put a line feed at the end of the line. Thus, the string of the second printf will print on the next line.

Line 36: return(0);

The **int** keyword from the "**`int main(void)`**" statement, means that the program should return an integer (whole number) to the operating system, signifying *success* or *failure*. So in this statement a **return** value of **zero** indicates *success*.

Line 37: }

This closing brace denotes the end of the program.

4. After you type your program don't forget to save the file again. Now you are going to compile your program to see if there are any bugs, then build your program before running your program. (See Figure 1)

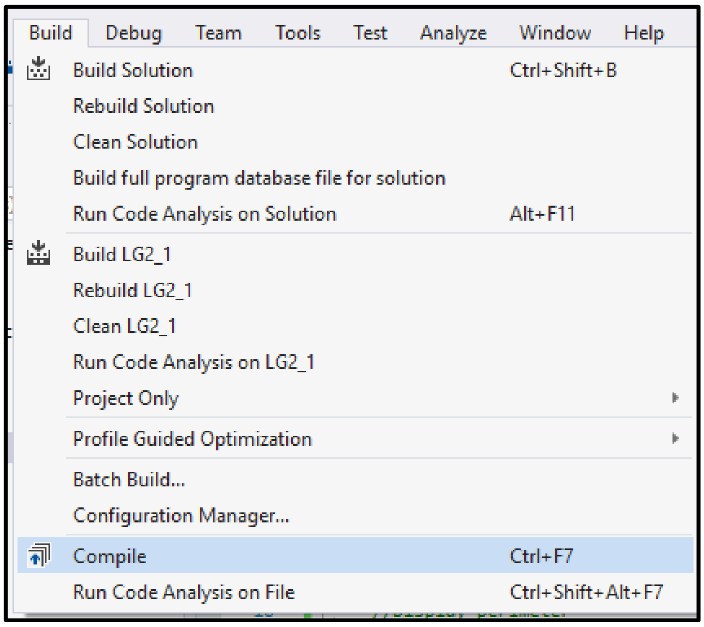
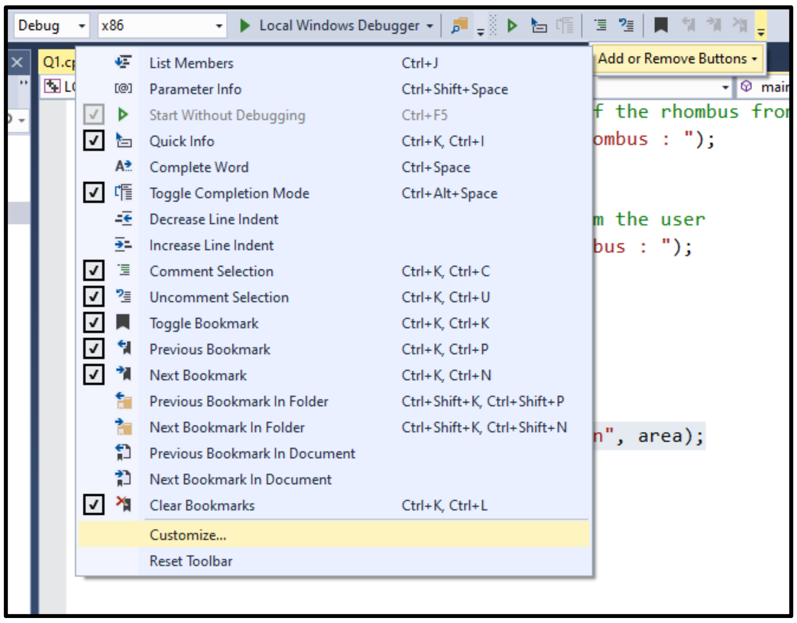
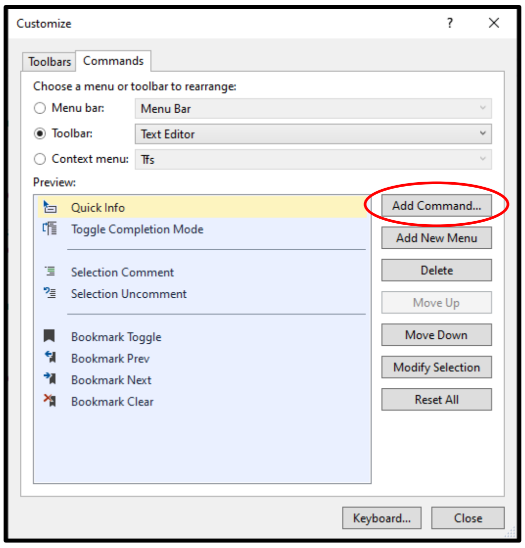

Build
Debug
Team
Tools
Test
Analyze
Window
Help
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Build full program database file for solution
Run Code Analysis on Solution Alt+F11
Build LG2_1
Rebuild LG2_1
Clean LG2_1
Run Code Analysis on LG2_1
Project Only
Profile Guided Optimization
Batch Build...
Configuration Manager...
Compile Ctrl+F7
Run Code Analysis on File Ctrl+Shift+Alt+F7

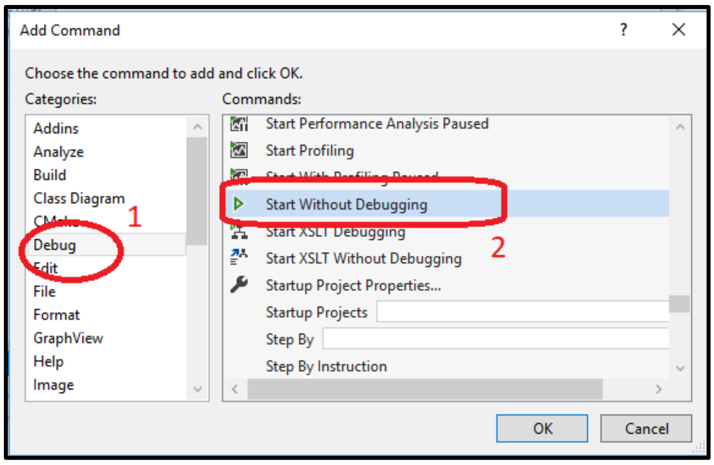
***Figure 1**: Compile and Build Q1.cpp file*

5. Now run your program by pressing **Ctrl + F5**. If you do not see the icon for "**Start without Debugging**" on the Debug menu, you can add it by clicking the arrow on the left of the menu, Add or Remove Buttons and Customize. Then click on Add Command button on Customize screen. Select **Debug** from left and **Start Without Debugging** from right and click OK. Now you have Start without Debugging button. You can run your code by clicking on it. (See Figure 2).



Debug
x86
Local Windows Debugger
Add or Remove Buttons
List Members Ctrl+J
Parameter Info Ctrl+Shift+Space
Start Without Debugging Ctrl+F5
Quick Info Ctrl+K, Ctrl+I
Complete Word Ctrl+Space
Toggle Completion Mode Ctrl+Alt+Space
Decrease Line Indent
Increase Line Indent
Comment Selection Ctrl+K, Ctrl+C
Uncomment Selection Ctrl+K, Ctrl+U
Toggle Bookmark Ctrl+K, Ctrl+K
Previous Bookmark Ctrl+K, Ctrl+P
Next Bookmark Ctrl+K, Ctrl+N
Previous Bookmark In Folder Ctrl+Shift+K, Ctrl+Shift+P
Next Bookmark In Folder Ctrl+Shift+K, Ctrl+Shift+N
Previous Bookmark In Document
Next Bookmark In Document
Clear Bookmarks Ctrl+K, Ctrl+L
Customize...
Reset Toolbar
f the rhombus fro
ombus : ");
m the user
bus : ");
n", area);





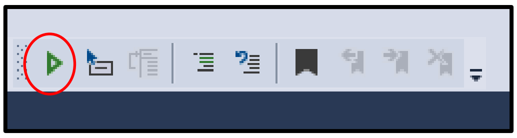


***Figure 2**: Start without debugging button*

6. After Compile and build operations, see what happens when you debug your program with F10 (Step Over) and how you will add your variables into **watch view ()** to see your variable's value. After you start Step Over debug (F10) you will see the watch window at the bottom of the screen.(See Figure 3)

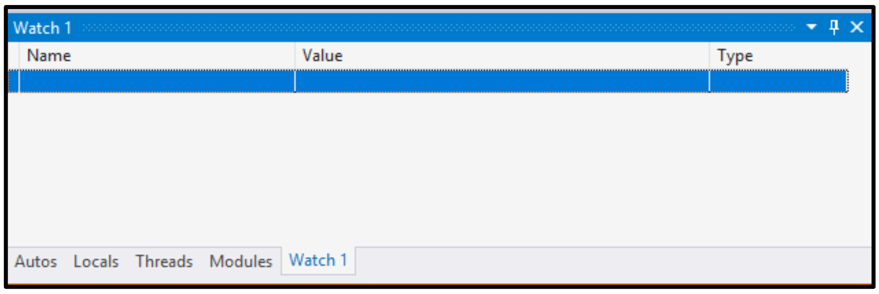


***Figure 3**: Watch window at the bottom of the screen*

If you don't see Watch pane, you can add it by clicking on the Debug menu, choosing Windows, then choosing Watch. (See Figure 4)

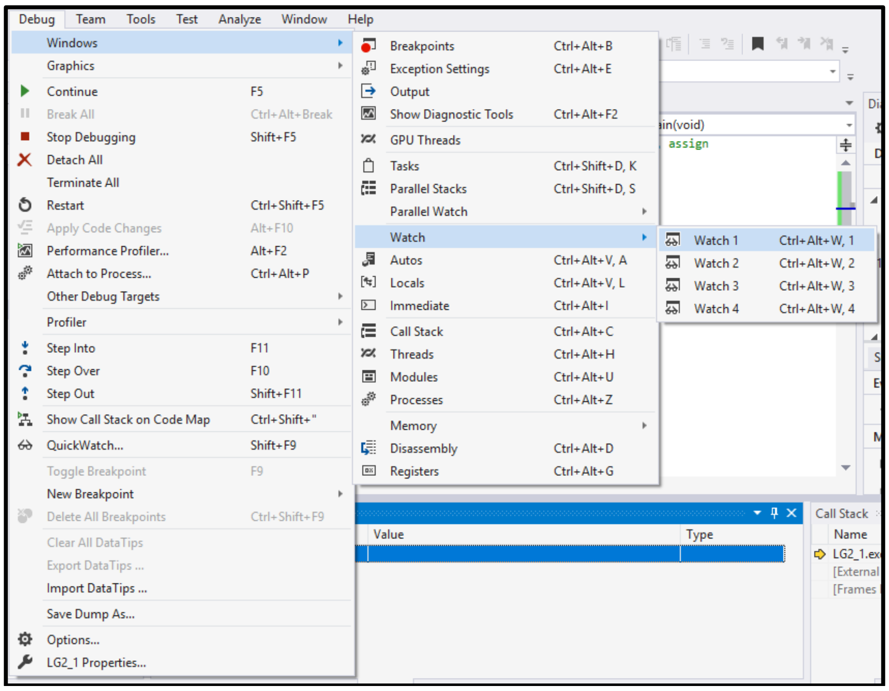
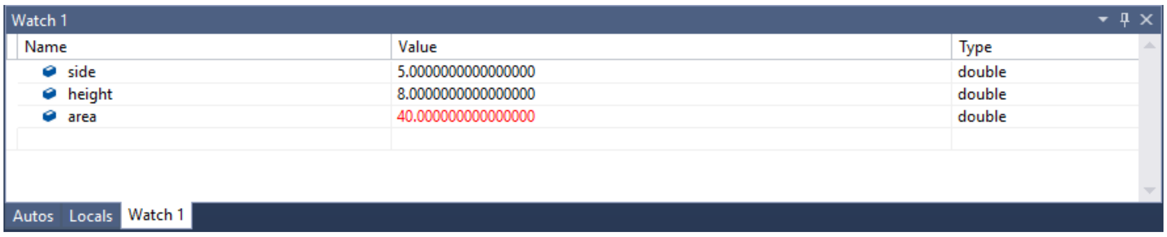


***Figure 4**: Add watch window*

7. After you see the watch window at the bottom, you have to write your variable names into name area of the window. The names must be exactly the same as variable names in order to be able to see the true values. After you write them continue with **F10** to Step Over debug. After you pass all the lines your variable's values will be seen in the watch window. (See Figure 5).



Name	Value	Type
side	5.0000000000000000	double
height	8.0000000000000000	double
area	40.000000000000000	double

***Figure 5**: Watch window after all steps are passed with F10 key*

8. When you come at the last bracket (}) this means you finished debugging so you have to end the debug operation. You have to choose **Debug Menu -> Stop Debugging** option or with **Shift + F5**. (See Figure 6).

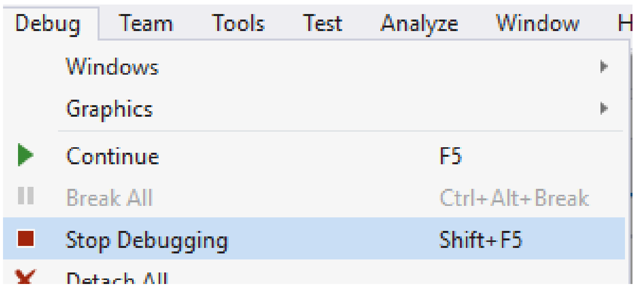
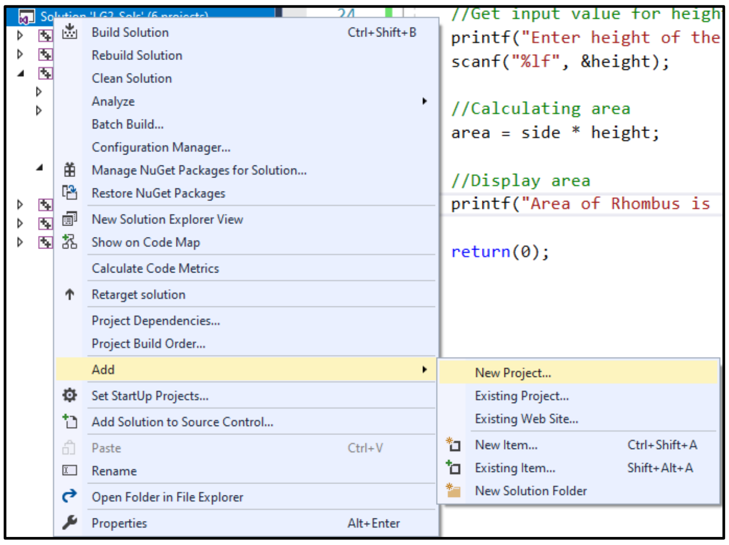


***Figure 6**: Stop debugging*

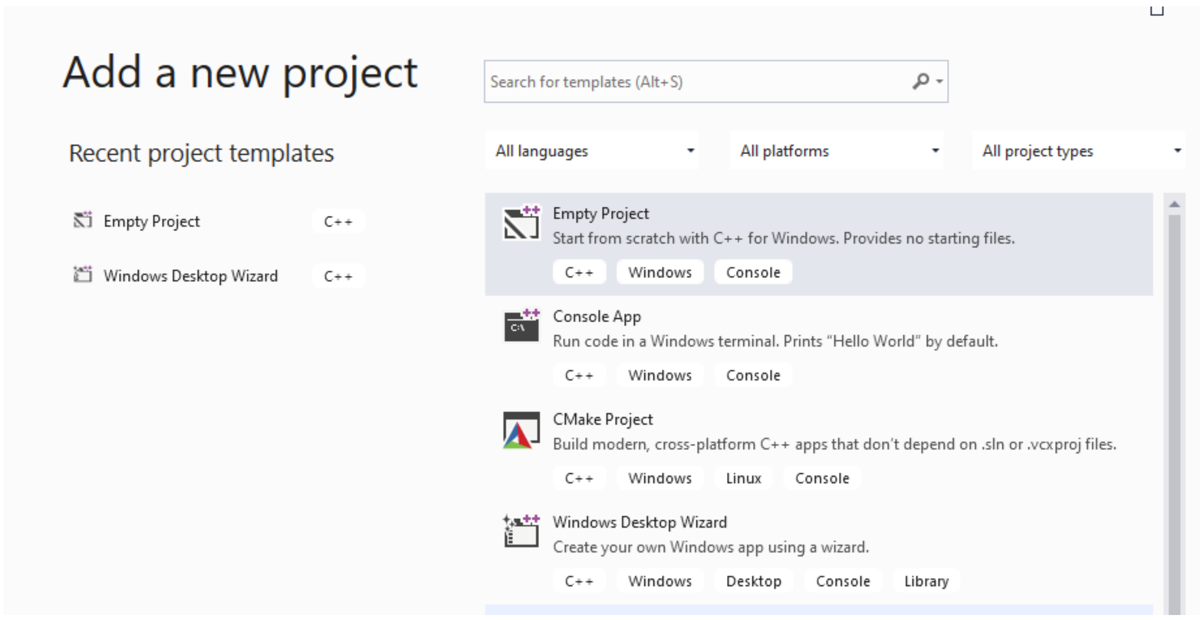
9. You just add a new project to your solution. Just right click on your solution then choose **Add -> New Project**. (See Figure 7).



```
//Get input value for heigh
printf("Enter height of the
scanf("%lf", &height);

//Calculating area
area = side * height;

//Display area
printf("Area of Rhombus is

return(0);
```

***Figure 7**: Add new project to the same solution*

After you click **Add New Project**, the New Project Pane will appear on the screen. (See Figure 8) Follow the same steps for adding a project and creating a source file. After finishing that, you will see your new project files under your current workspace.


Add a new project

Search for templates (Alt+S)

All languages | All platforms | All project types

Recent project templates

Empty Project C++

Windows Desktop Wizard C++

Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ Windows Console

Console App
Run code in a Windows terminal. Prints "Hello World" by default.
C++ Windows Console

CMake Project
Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.
C++ Windows Linux Console

Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library

***Figure 8**: Add new project*

Q2. Write the same C program, but this time you will write it for **volume** calculation of a ellipsoid.

$$V = \frac{4}{3} * PI * a * b * c$$

Make the necessary changes in variable names, in their values and correct the comments according to your own program. (You can give any names you want to your variables, but they must be reasonable and assign the values you want to your variables.)

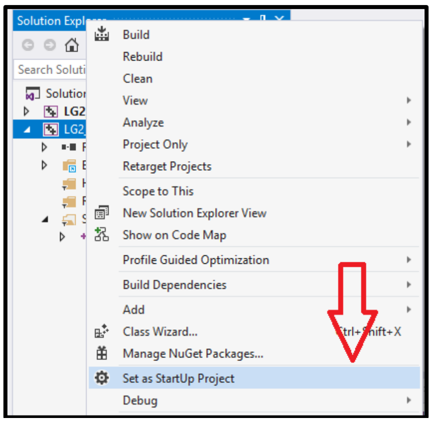
After you write the program please compile, build your program and do Step Over debugging as you have learned. Use watch window to see your variables and their values.

Example Run:

```
Enter first axis (a) of the Ellipsoid: 5
Enter second axis (b) of the Ellipsoid: 6
Enter third axis (c) of the Ellipsoid: 7
The Volume of Ellipsoid is: 879.200000
```

Project Name: LG2_Q2
File Name: Q2.cpp

*****HINT:** In the same solution, to run another project you should right click the other project's name then **Set as Startup Project**.*



Q3. Write the following code into a new project and Q3.cpp source file. Add your new project to the workspace as you learned earlier. Your project name and source file name are given at the end of the question on the right-hand side. In the code below, however, because that you cannot see the results of each calculation as they take place on the screen (as there are **no printf statements**), you should use the Step Over short key (F10) as you learned in the debugging section. Add all the variables to the watch window to observe the changes as calculations take place. Write each result next to each operation.

```c
#include <stdio.h>

int main(void)
{
       int megabyte = 379;
       double bit = 0, byte = 0, kilobyte = 0;


       //calculate the values of kilobyte, byte
         and bit
       kilobyte = 1024 * megabyte;

       //byte = 1024*1024*megabyte;
       byte = 1024 * kilobyte;

       //bit = 8*1024*1024*megabyte;
       bit = 8 * byte;

      return(0);
 }
```

Variable Name	Value
megabyte	
kilobyte	
byte	
bit	

Project Name: LG2_Q3
File Name: Q3.cpp

Q4. Write a C program that has a variable named **lira** (Turkish Lira) and has a value **675.4**. The program you write should convert this lira value first to **dollars,** then to **euros**. You can use the program above as an example to write your new program's statements. There will be NO output statement to see the results, so you will use **F10 (Step Over) debug** and **watch** window to see your variable's values.

Hint : 1 ₺ = 0,033 $
1 $ = 0,93 €

Project Name: LG2_Q4
File Name: Q4.cpp

ADDITIONAL QUESTIONS

AQ1. Write a C program that calculates the how old are you. There will be defined CURRENTYEAR as 2024.

Example Run:

```
Enter your birth year: 1980
Nice age: 44
```

Project Name: LG2_AQ1
File Name: AQ1.cpp

AQ2. Write a C program that calculates the following operation:

$$((\boldsymbol{a}+\boldsymbol{b})/\mathbf{2})-(\boldsymbol{b}.\boldsymbol{c})$$

a = 80.6, b = 8.4, c = 4

Your variable names and their values are given above. Write also the output statement to see the result on the screen. After you write the program please compile, build your program and do Step Over debugging as you learn. Use watch window to see your variables and their values.

Example Run:

```
Result = 10.90
```

Project Name: LG2_AQ2
File Name: AQ2.cpp

Debugger Short Keys:

Display documentation for the active window.	F1
Display a system menu for the application window.	ALT+SPACEBAR
Add and remove breakpoints on the current lines.	**F9**
Clear all breakpoints.	CTRL+SHIFT+F9
Disable breakpoint.	CTRL+F9
Display Breakpoints dialog box.	CTRL+B
Adds a watch on the currently selected word.	SHIFT+F9
End debugging session.	**SHIFT+F5**
Execute code one statement at a time, following execution into function calls (Step Into).	**F11**
Execute the next line of code but without following execution through any function calls (Step Over).	**F10**
Execute the remaining lines of a function in which the current execution point lies (Step Out).	SHIFT+F11
Restart a debugging session.	CTRL+SHIFT+F5
Resume execution of your code from the current statement to the selected statement (Run to Cursor).	CTRL+ F10
Run the application to the next break point.	**F5**
Set the next statement.	CTRL+SHIFT+ F10
Stop execution (Break).	CTRL+BREAK
Open watch window #x.	**CTRL+ALT+W, #{1, 2, 3}**
Open locals' window.	**CTRL+ALT+V, L**