C++

Lab 1

Introduction

Required terminology and general information for this chapter:

Program – a series of instructions for a computer to execute.

Programming languages: Machine, Assembly, and High Level

How the first computers were programmed? By manually connecting wires.

> **Machine language**: programs and data were entered into the computer using zeros and ones. All operation codes (**opcodes**) and **operands** were entered in binary numbers. Machine language is the machine "understandable" language. It does not matter what other languages are used to program, eventually all programs and data will have to be translated into a "machine executable" format.
>
> **Assembly Language**: An easier programming language than machine language, where programs are written using familiar symbols (for example A for add). The operands could be in **binary**, **hexadecima**l or decimal format. For every machine language code, there needs to be one assembly language code. The program was not shorter, just easier to write. Once the program was written in assembly language, it had to be compiled to produce the machine executable code in order to run the program. The compiler used is called an **ASSEMBLER**.
>
> **High Level Language**: Included are languages such as Fortran (Formula Translator), COBOL (Common Business Oriented Language), BASIC (Beginner's All purpose Symbolic Instruction Code), Pascal, C, Ada, and many more. Programs are much shorter than lower level languages. Each instruction is converted to several machine executable codes. This conversion is done in

three steps: **preprocessing, compiling and linki**ng. An original program written using an EDITOR in a high level language is known as the **Source Code**. A compiled program is called the **Object Code** and the linked program is called the **Executable Code**. Examples of files created in each stage are: **Carpet.CPP, Carpet.OBJ, and Carpet.EXE**

Syntax: Rules for constructing a legal sentence in a programming language. If a program has syntax errors, it will not compile. Therefore, it produces **compile time errors**.

Semantics: Rules that determine the meaning of instructions written in a programming language.

Runtime and Logical errors: A runtime error occurs when the program cannot execute what is being asked to do. For example, open a file when it does not exist. Logical errors are most difficult to fix; it is caused by faulty logic by the programmer. For example, suppose you wanted write a program to give discount if purchase is greater than $100.00, instead you programmed to give discount if purchase is less than $100.00.

C++: The C (Dennis Ritchie) language is a modification of another language called BCPL (Ken Thompson). C language was written for programming under the Unix operating system environment which continues to be a very popular operating system for universities and businesses. C++ (Bjarne Stroustrup) is an object oriented version of C.

Constant : a data item that does not change during the execution of the program. A constant could be a named constant or a literal constant. Example of a literal constant is given in the following program. Example of a named constant is:

const float TAX_RATE = 8.025; //TAX_RATE is a named constant // 8.025 is a literal constant

First C++ program: Call up the C++ programming environment from your windows desktop and type the following program in. Compile, and run it.

If you do not know how to write the source code and compile, go to the end of this chapter where I explain it in the appendix.

Program 1-1

```
/*********************************

          Say Hello program

          By Dr. John Abraham

          Created for 1380 students

Teaching objective: program structure

*********************************/

#include <iostream>                    //this is preprocessor directive

using namespace std;  //tells the compiler certain objects such as cout are contained in the standard namespace

int main ()             //this is the main function

{

cout << "See mom! I wrote my first C++ program\n";

getchar(); //wait for the enter key to be pressed. Try the program without this line.

return 0;

}
```

Program run 1

```
See mom! I wrote my first C++ program
```

Save your program to your floppy or thumb drive by clicking File, Save Hello.cpp as (the name will be whatever you named it. I named it Hello), and give it a name like a:assignment1.cpp.

A program in C++ is a collection of one or more functions. This program has only one function named main. The function *main* is a required function in all programs. If there are several functions in a program, each function carries out a different task. The *main* function will call other functions into action.

Description of the program: Lines beginning with /* and ending with */ are comments, C++ ignores these lines. These comments are used for internal documentation. // may be used for commenting one line.

Preprocessor library: C language is a small language and does not have inherent input/output support. Input/output support is provided by modules (library). This program outputs one line of text to the monitor. The monitor is the standard output device and the keyboard is the standard input device. A pound sign # indicates that the remainder of that line is an instruction (directive) to the compiler. #include <iostream.h> tells the compiler to add the code found in that file to the beginning of this program. This code handles all character stream inputs and outputs. Without this you cannot read from the keyboard or display anything to the monitor.

A function: int main () is a function. A function receives one or more parameters (arguments) from the calling module and returns none or one result to the calling module. The word **int** before the function name (main) indicates that the function will return an integer to the calling module. The calling module of the main is the operating system. This program returns a zero to the operating system, indicating normal execution. The **()** after the function name indicates that this function does not receive any parameters from the calling module. Later we will see variations to this.

Begin and End of a block: The beginning of a body or compound statement is indicated by **{** and the ending is indicated by **}**.

Statement separator: A statement is separated from another statement by placing a semicolon between them. End of the line does not indicate end of a statement.

cout and cin: cout displays (prints) to an output device such as a monitor, and cin receives input from an input device such as a keyboard. The << or >> indicates the direction of flow of the data. In this program, `"See mom! I` `wrote my first C++ program\n,` is the data (in this case the data is a literal) that flows into the output device. Notice that the string literal is enclosed in double quotes. The last two characters in the literal **\n** makes the cursor to go to a new line. The backslash is used as a symbol for escape. An escape sequence is used to control output devices. We will see additional escape sequences later.

The return statement: return 0, returns zero to the calling module through the main. In this case, the operating system is given the value 0, indicating that the program had a normal execution. Perhaps, another procedure would have returned a result of a calculation to the calling module.

Program 1-1A

```
/*********************************

        Say Hello program

        By Dr. John Abraham

        Created for 1380 students

*********************************/

#include <iostream>      //this is preprocessor directive

using namespace std.

int main () //this is the main function

{

cout << "See mom! I wrote my first C++ program\n";

getchar(); //wait for the enter key to be pressed

return 0;

}
```

Arithmetic in C++

When we write code in C++ to do calculations, it is important to remember that results of integer and integer calculations may be different than real and real calculations. We also need to know how mixed number calculations will be carried out. C++ will allow you to assign a number with decimal to an integer, however, the fractional part will be discarded. Program 1-2 explores various arithmetic calculations.

Program 1-2

```
/*****************************************
          c++ Arithmetic
          By Dr. John Abraham
          Created for 1380 students
   Instructional objective: Arithmetic
*****************************************/

#include <iostream>   //this is preprocessor directive

using namespace std;        //using directive

int main ()           //this is the main function
{
 int i,j,k,l, m,n;
 float a,b,c;
 //integer operations
 cout << "INTEGER OPERATIONS\n \n";
 i = 6 + 3;
 l = 6.5;
 m = 3.5;
 j = l + m;
 k = 10 /3;
 n = 10 % 3;
 cout << "6 + 3 = " << i << "\n";
 cout << "l = 6.5, m = 3.5 --------->l + m = " << j << "\n";
 cout << "10 / 3 = " << k << "\n";
 cout << "10 % 3 = " << n << "\n";

 //real and mixed operations
 cout << "\nREAL AND MIXED OPERATIONS \n \n";
 a = 10 / 3;
 b = 10.0 / 3.0;
 c = 10.0 /3;
 cout << "10 / 3 = " << a << "\n";
 cout << "10.0 / 3.0 = " << b << "\n";
 cout << "10.0 / 3 = " << c << "\n";
 getchar();

return 0;
}
```

```
INTEGER OPERATIONS

6 + 3 = 9

l = 6.5, m = 3.5 --------->l + m = 9

10 / 3 = 3

10 % 3 = 1

REAL AND MIXED OPERATIONS

10 / 3 = 3

10.0 / 3.0 = 3.33333

10.0 / 3 = 3.33333
```

If l=6.5 and m =3.5 then l+m should be 10, why is it 9? We assigned these numbers to integer variables, which discards the fractional part leaving 6 and 3, which give a total of 9. How about 10/9 yielding 3? This is called integer division. The next problem 10 % 3 gives a result of 1, which is the remainder of the integer division (also known as the modulus). In the next problem even though we assigned the result of 10/3 to a real variable (a), the variable only received the result of an integer division. The result of 10.0/3.0 is 3.333333; here both numbers are real numbers (float). However the last problem 10.0/3 also gives 3.33333, why? In a mixed operation like this the integer is converted to float first, then the operation is carried out.

Homework

The terminology given at the beginning of this chapter is very important. You must learn it thoroughly. Make q-cards and memorize the terms.

Write this program over and over again until you do not have to look at the notes. You should not continue to the next chapter until you mastered this chapter thoroughly.

This homework may appear surprisingly easy to you. Don't be fooled. Many students do not finish the course because they do not spend much time with the first two chapters.

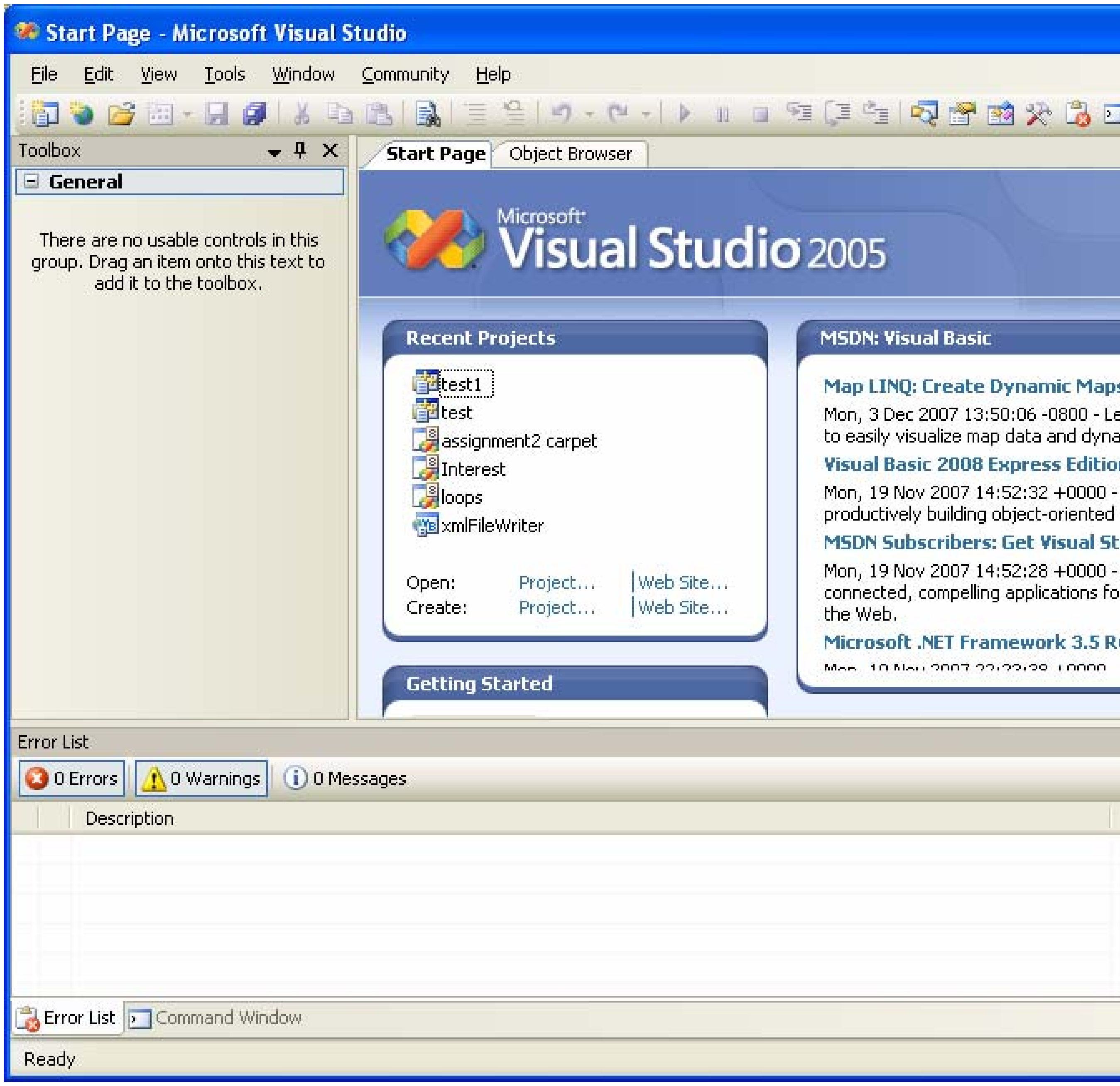
Write a program to determine the number of thousands, hundreds, tens, and ones in a given number. Hint: use integer division and modulus. Example of a program run:

```
In 8532 there are

        8 thousands

        5 hundreds

        3 tens

        2 ones.
```

Appendix

How to launch Microsoft visual studio, write a program, compile and run.

1. Find the icon for Microsoft visual studio 2005 and click it to launch the program.
2. The start page looks like this:



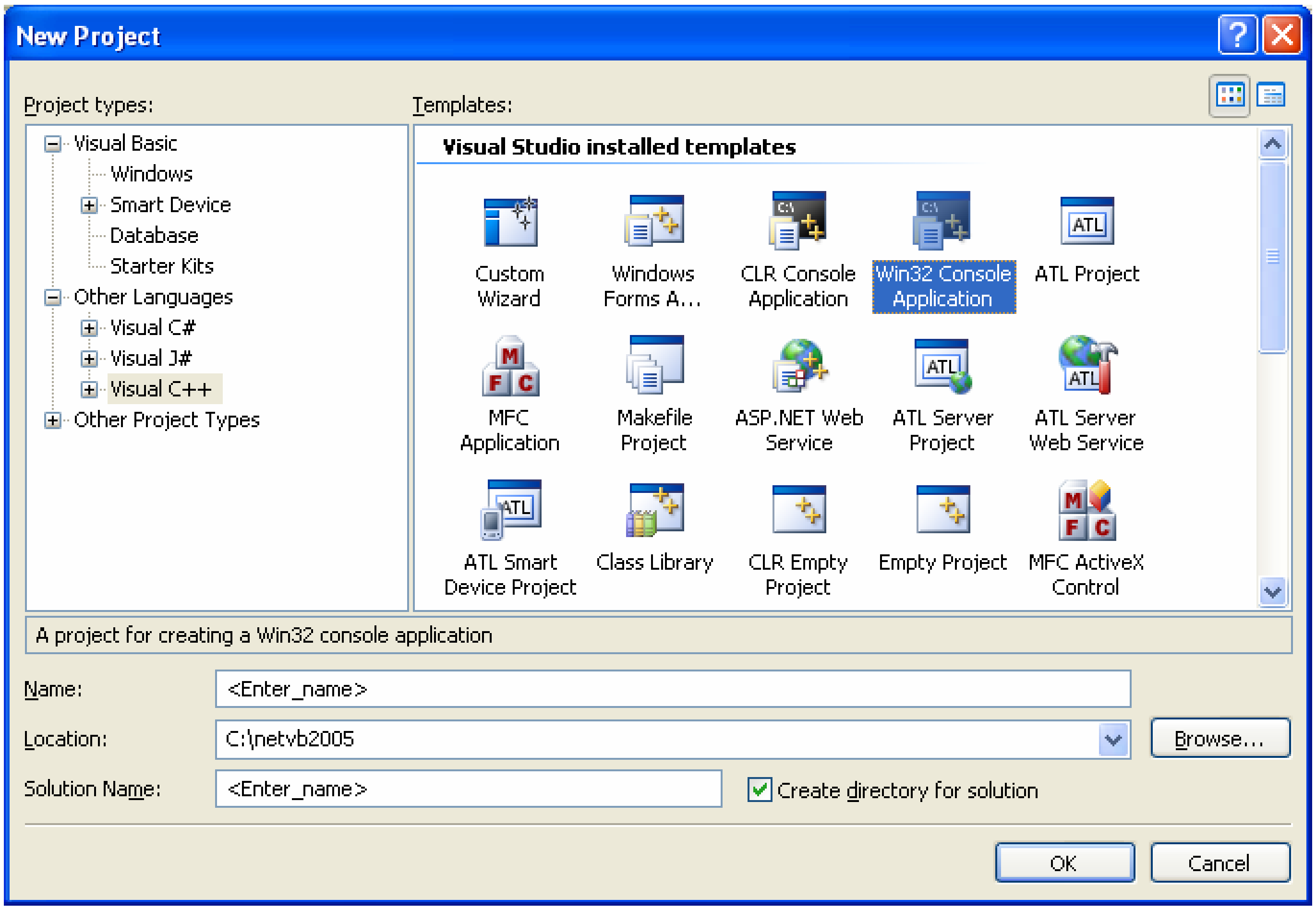
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

test1
test
assignment2 carpet
Interest
loops
xmlFileWriter

Open: Project... | Web Site...
Create: Project... | Web Site...

Map LINQ: Create Dynamic Maps
Mon, 3 Dec 2007 13:50:06 -0800 - Le
to easily visualize map data and dyna
Visual Basic 2008 Express Editio
Mon, 19 Nov 2007 14:52:32 +0000 -
productively building object-oriented
MSDN Subscribers: Get Visual St
Mon, 19 Nov 2007 14:52:28 +0000 -
connected, compelling applications fo
the Web.
Microsoft .NET Framework 3.5 R

3. You will see Open and Create. Click on Create Project. On the left side you will see the languages, if C++ is not shown click on other languages. Click on Visual C++ and Win 32 console application. Enter a name such as Hello Program. Pay attention to the location and subdirectory. This is where all your programs will be kept.


New Project

Project types:

- Visual Basic
  - Windows
  - Smart Device
  - Database
  - Starter Kits
- Other Languages
  - Visual C#
  - Visual J#
  - Visual C++
- Other Project Types

Templates:

Visual Studio installed templates

Custom Wizard, Windows Forms A..., CLR Console Application, Win32 Console Application, ATL Project, MFC Application, Makefile Project, ASP.NET Web Service, ATL Server Project, ATL Server Web Service, ATL Smart Device Project, Class Library, CLR Empty Project, Empty Project, MFC ActiveX Control

A project for creating a Win32 console application

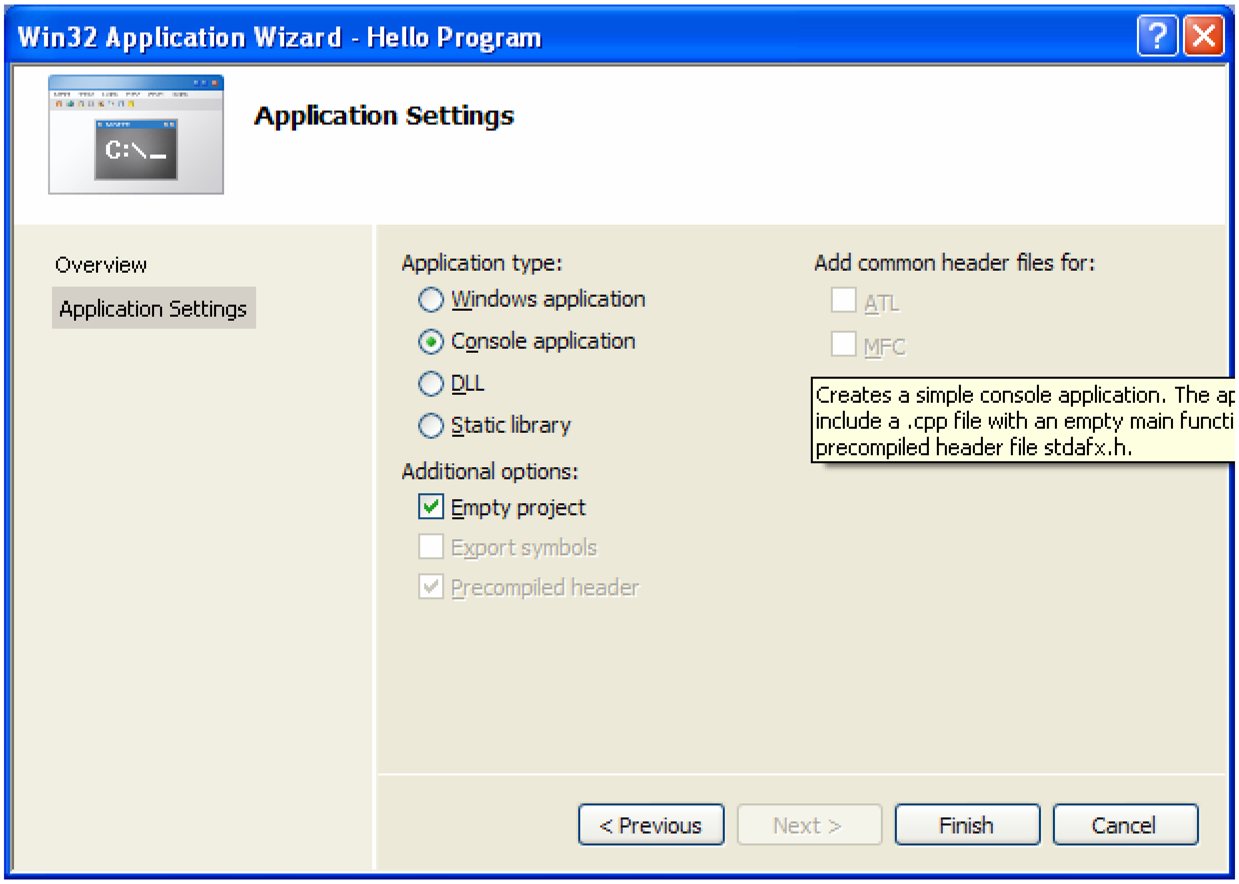
Name: <Enter_name>

Location: C:\netvb2005

Solution Name: <Enter_name>

[✓] Create directory for solution

4. Click OK, then click next. In the following screen click empty project and press finish.



Application type:

- ( ) Windows application
- (●) Console application
- ( ) DLL
- ( ) Static library

Additional options:

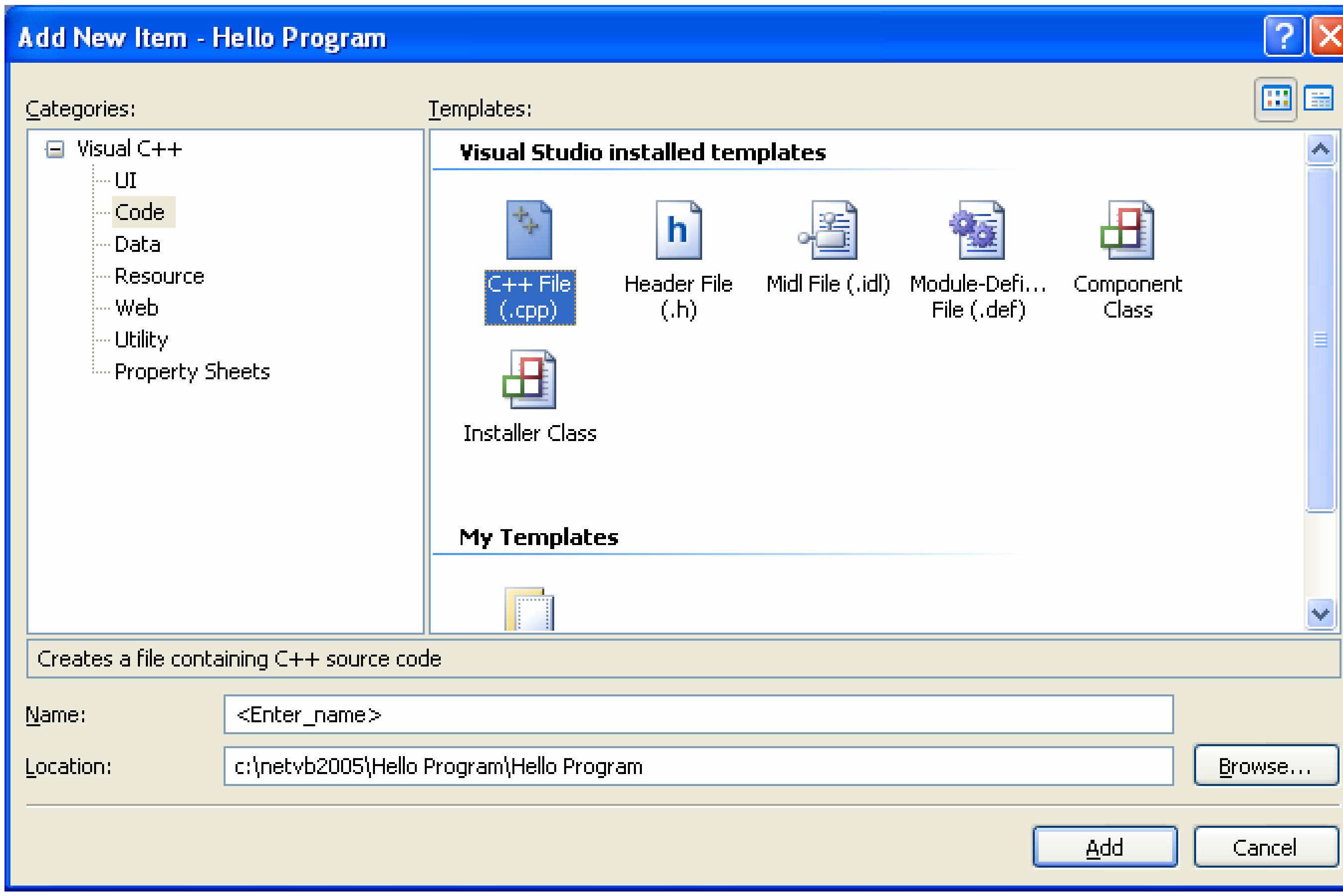
- [✓] Empty project
- [ ] Export symbols
- [✓] Precompiled header

Add common header files for:

- [ ] ATL
- [ ] MFC

Creates a simple console application. The ap
include a .cpp file with an empty main functi
precompiled header file stdafx.h.

5. **This step is important: right click on source files** in the solution explorer and click to **add** new item. If the solution explorer does not show up you need to click on the solution explorer icon (the one with the magnifying lens).



Creates a file containing C++ source code

Name: <Enter_name>

Location: c:\netvb2005\Hello Program\Hello Program

6. Click on code on the left side and click on C++ file. Give it a name and click Add.
7. Type in the program.