
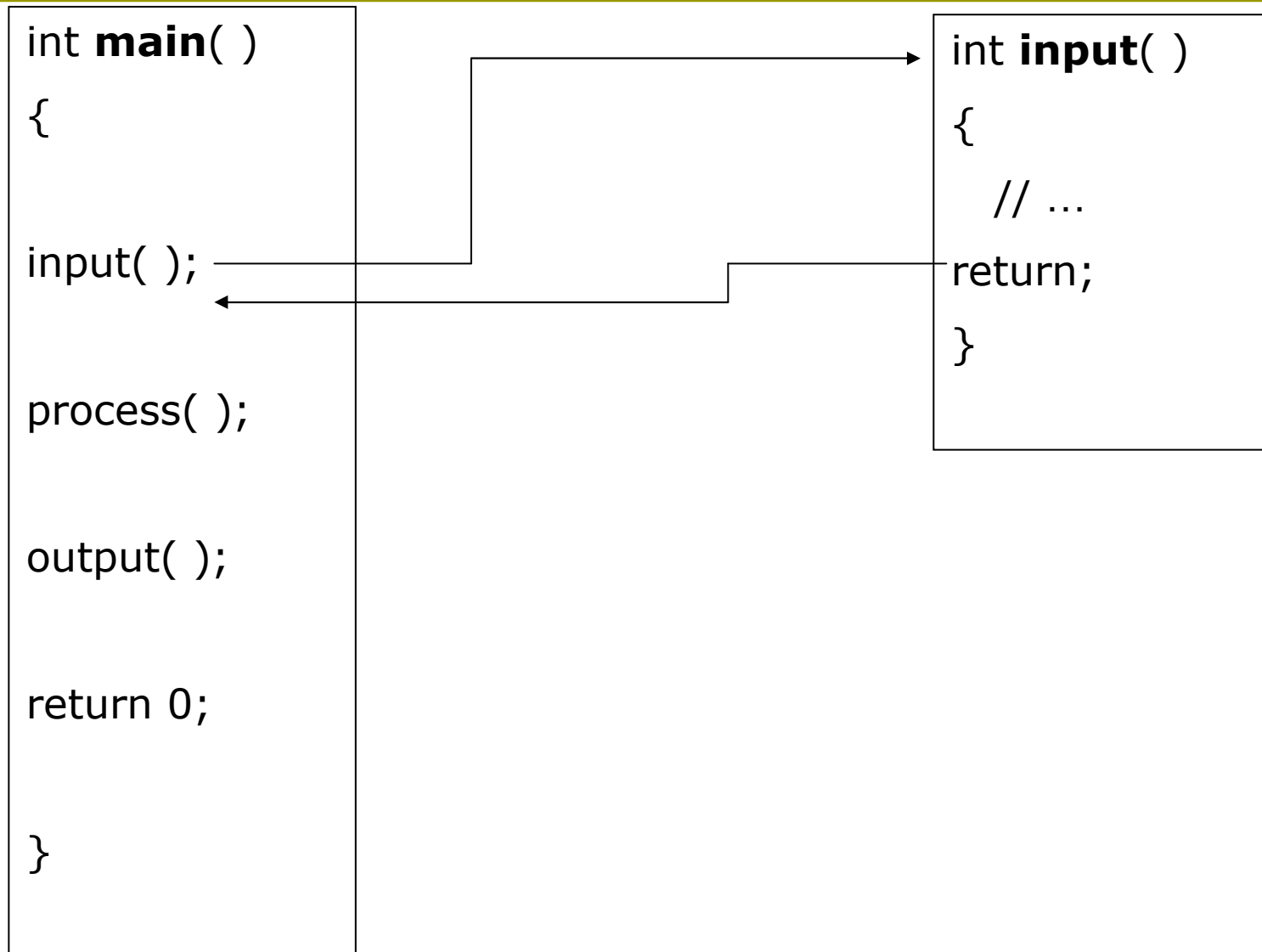


Chapter 2

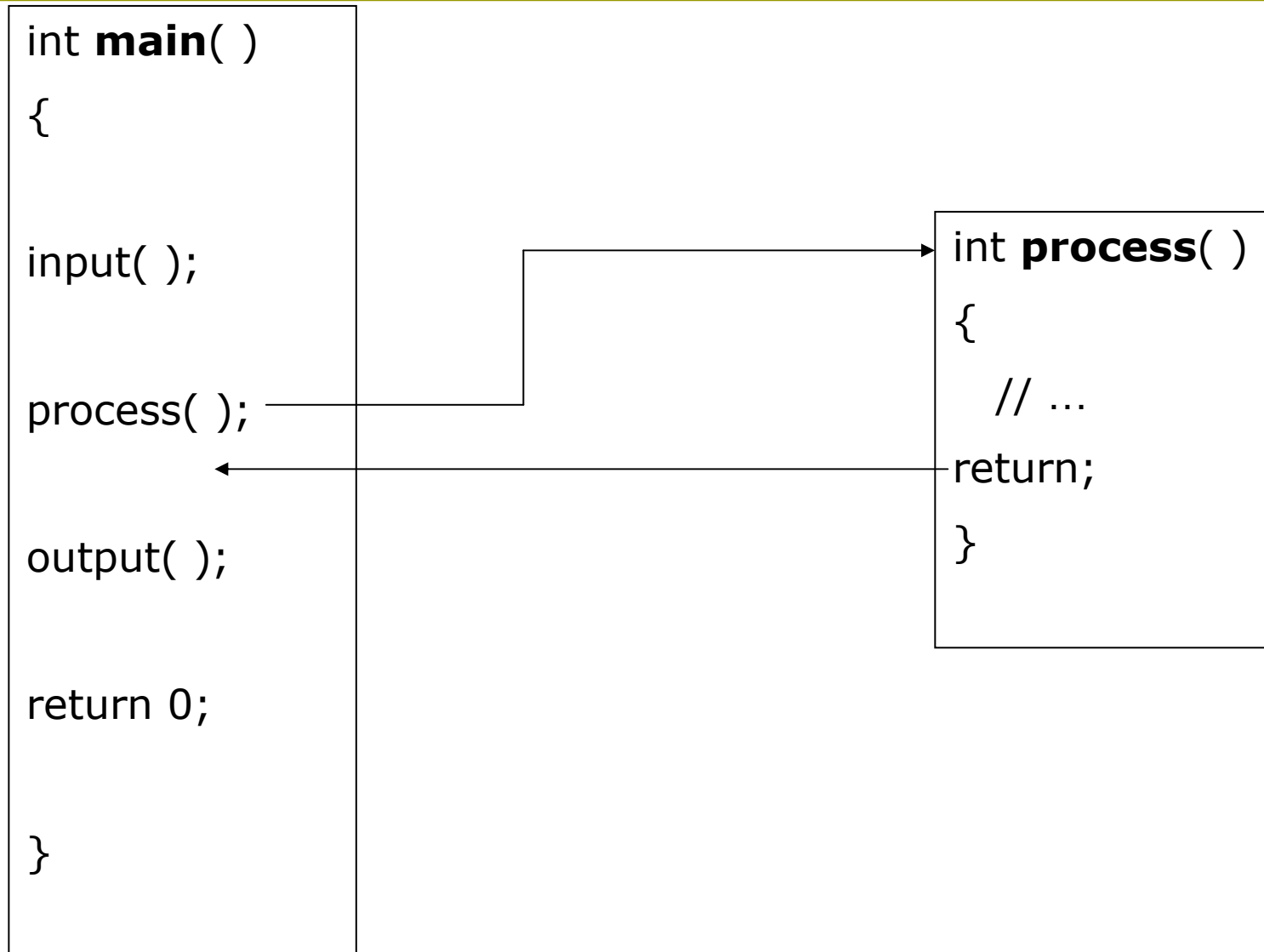


Data, Variables, and Calculations

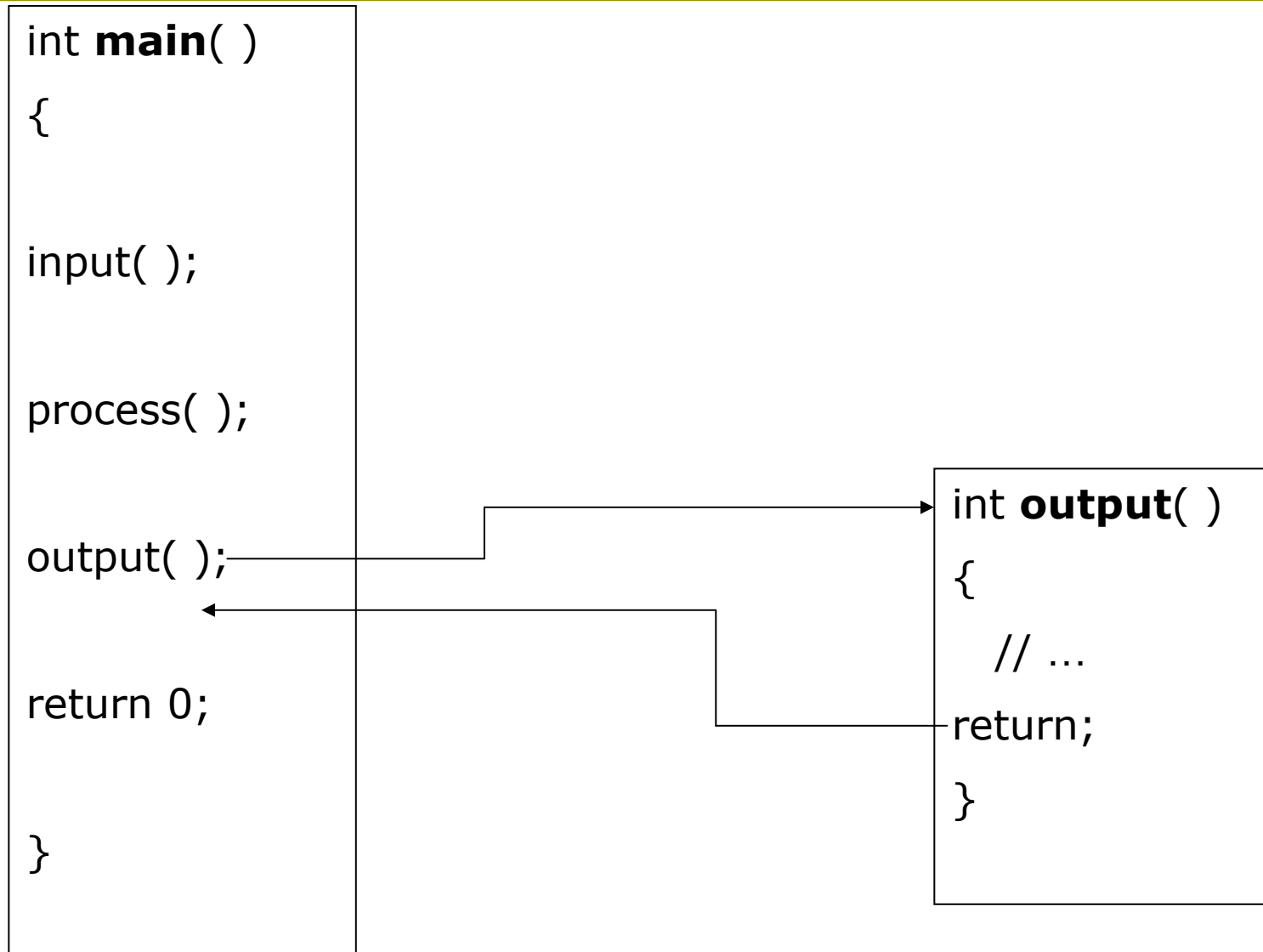
The Structure of a C++ Program



The Structure of a C++ Program



The Structure of a C++ Program

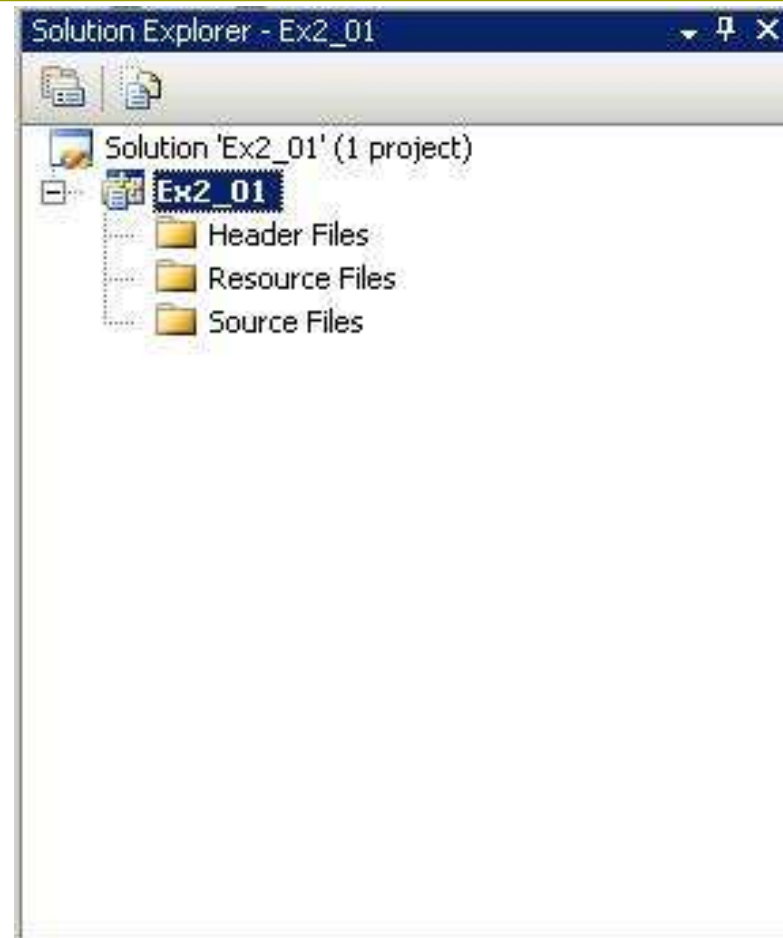


main()

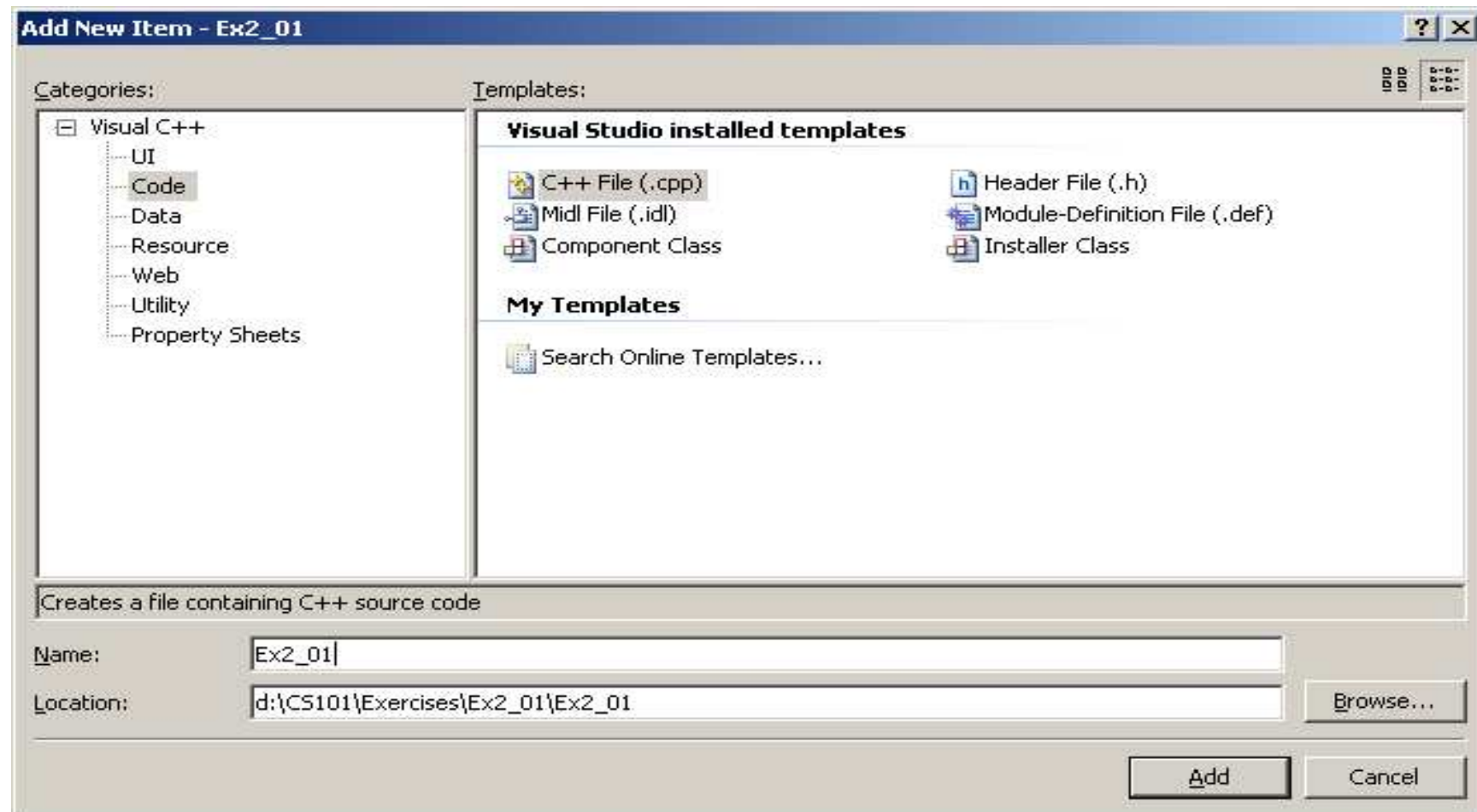
- ❑ Every ANSI/ISO standard C++ program contains the function main().
- ❑ A Program in C++ consists of one or more functions.
- ❑ A function is simple a self-contained block of code with a unique name.
 - You can invoke a function by its name.
- ❑ The principal advantage of having a program broken up into functions is that you can write and test each piece separately.
 - Re-use

Ex2_01

- Start a new Win32 Console Project
 - Ctrl+Shift+N
 - Choose Empty project
 - Right-click Source Files and Add > New Item
- Choose category Code and template C++ file (.cpp).



Add New Item



Syntax of C Language

```
// Ex2_01.cpp  
// Simple calculation  
#include <iostream>
```

Comments
begin with //

```
using std::cout;  
using std::endl;
```

A statement
block is
enclosed by
braces.

```
int main()  
{
```

Whitespace

```
    int a, b, c;
```

```
    a = 10;
```

```
    b = 20;
```

```
    c = a + b;
```

```
    cout << "The summation a + b = ";
```

```
    cout << c;
```

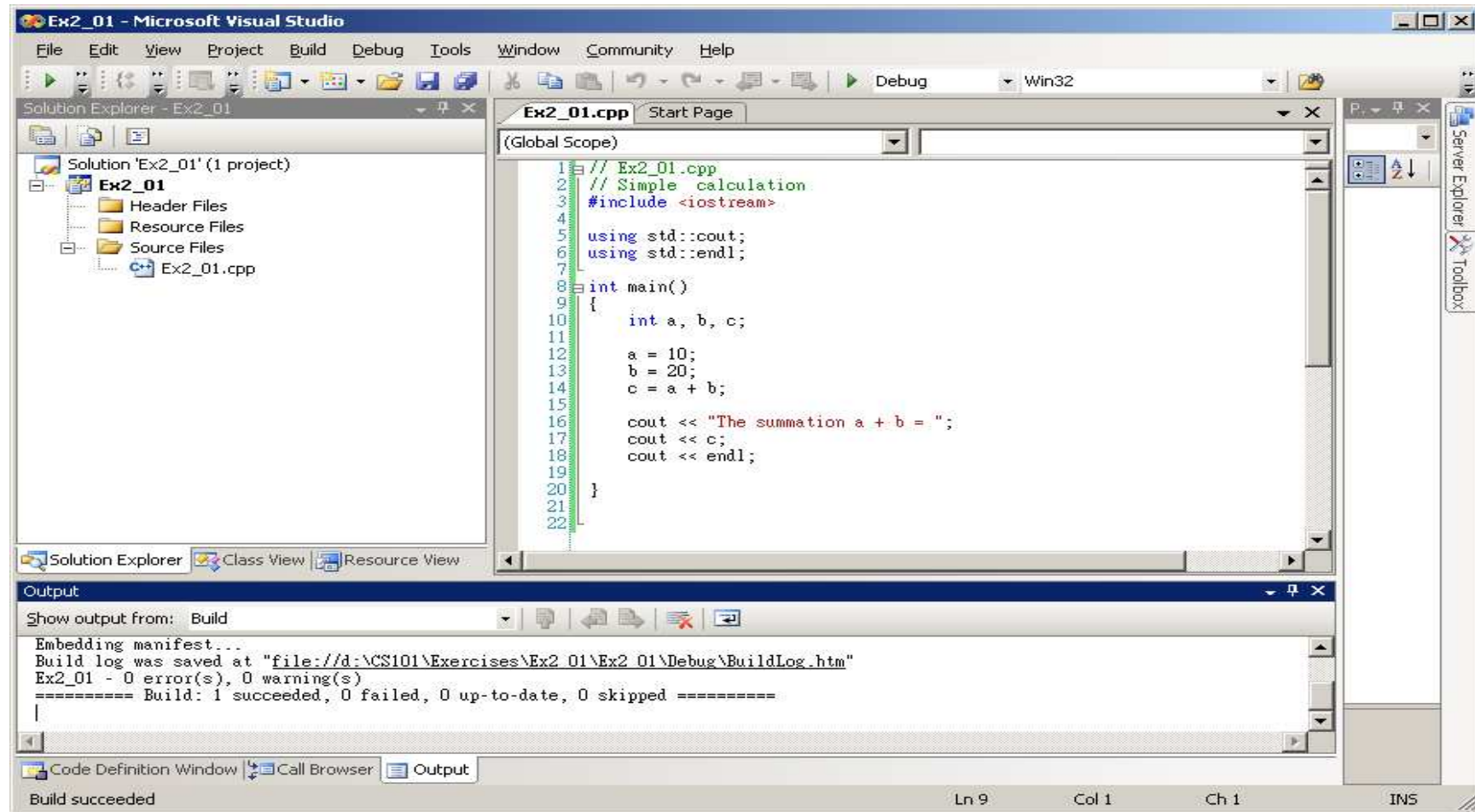
```
    cout << endl;
```

Each statement
ends with a
semicolon.

```
}
```

End-of-Line

<F7> to Build



Naming Variables

- ❑ Variable names can include the letters A-Z, a-z, the digits 0-9, and the underscore character (`_`).
 - Variable names cannot begin with digits.
 - Avoid naming a variable name to begin with an underscore (`_this`, `_that`), because it may conflict with standard system variables.
- ❑ Variable names are case-sensitive.
- ❑ Convention in C++
 - Classes names begin with a capital letter.
 - Variable names begin with a lowercase letter.

Naming Variables (2)

- ❑ In Visual C++ 2008, variable names can be up to 2048 characters long.
 - number_of_students
 - strips_per_roll
- ❑ However, not all compilers support such long names.
 - It's a good idea to limit names to a maximum of 31 characters.
- ❑ Illegal variable names:
 - 8_Ball, 7UP
 - Hash!, Mary-Ann

Declaring Variables

- `int value;`
 - This declares a variable with the name `value` that can store integers.
- `int i, j, k;`
 - A single declaration can specify the names of several variables.
 - However, it is better to declare each variable in a single line. (Why?)
- `int value = 10;`
 - When you declare a variable, you can also assign an initial value to it.

Integer Types & Character Types

□ Integer Types

- `int` // 4 bytes
- `short` // 2 bytes
- `long` // 4 bytes,
 - the same as `int` in Visual C++ 2008

□ Character Data Types

- `char letter = 'A';` // 1 byte
 - Single quote, not double quote (")
 - The ASCII code of the character is stored in that byte.
 - Therefore, it is equivalent to `char letter = 65;`

Integer Type Modifier

□ Examples:

- signed int; // equivalent to int
- signed short; // equivalent to short
 - Range: -32768 ~ 32767
- unsigned short;
 - Range: 0 ~ 65535
- signed char;
 - Range: -128 ~ 127
- unsigned char;
 - Range: 0 ~ 255

Boolean Type

□ Examples:

- `bool testResult;`
- `bool colorIsRed = true;`

□ In old C language, there is no `bool` data type.

- Variables of type `int` were used to represent logical values.
 - Zero for false; non-zero for true.
 - Symbols `TRUE` and `FALSE` are defined for this purpose.
 - Note that `TRUE` and `FALSE` are not C++ keywords.
 - Don't confuse `true` with `TRUE`.

Floating-Point Type

- A floating-point constant contains a decimal point, or an exponent, or both.
 - 112.5
 - 1.125E2 (1.125×10^2)

- Examples:
 - `double inch_to_cm = 2.54;`
 - 8 bytes
 - Ref. Chapter 3 of Forouzan:
 - 1 bit sign
 - 11 bit exponent
 - 52 bit mantissa
 - `float pi = 3.14159f;`
 - 4 bytes

Enumeration

- ❑ Declare an enumeration type `Week`, and the variable `thisWeek`;
 - `enum Week {Sun, Mon, Tue, Wed, Thu, Fri, Sat} thisWeek;`
- ❑ You may then assign one enumeration constant as the value to the variable `thisWeek`:
 - `thisWeek = Thu;`
- ❑ Actually, the first name in the list, `Sun`, will have the value 0, `Mon` will be 1, and so on.

The const Modifier

- ❑ `const float inch_to_cm = 2.54;`
 - If you accidentally wrote an incorrect statement which altered the value of `inch_to_cm`, the compiler will fail and complain.
 - Avoid using magic numbers like 2.54 in your program when the meaning is not obvious. Declare a constant for it.
- ❑ All the above data types can have const modifiers.
- ❑ Constant Expressions
 - `const float foot_to_cm = 12 * inch_to_cm;`

Basic Input/Output Operations

□ Input from the keyboard

- `cin >> num1 >> num2;`

□ Output to the command Line

- `cout << num1 << num2;`

- `cout << num1 << ' ' << num2;`

- `cout << setw(6) << num1 << setw(6) << num2;`

- `#include <iomanip>`

- Causes the next output value to have width of 6 spaces.

Escape Sequences

- An escape sequence starts with a backslash character, \.

- `cout << endl << "This is a book." ;`

- `cout << endl << "\tThis is a book." ;`

- Some useful escape sequences:

- `\a` alert with a beep

- `\n` newline

- `\b` backspace

- `\t` tab

- `\'` single quote

- `\"` double quote

- `\\` backslash

Assignment Statement

- *variable = expression ;*

- `c = a + b;`

- `q = 27 / 4; // the quotient is an integer`

- `r = 27 % 4; // remainder`

- Repeated assignment

- `a = b = 2;`

- Modifying a variable

- `d = a + b / c; // d = a + (b / c)`

- `count = count + 5;`

- `count += 5; // shorthand notation`

- `count *= 5; // count = count * 5`

- `a /= b + c; // a = a / (b + c)`

Increment Operators

- ❑ Frequently used in C++
- ❑ The following statements have exactly the same effect:
 - `count = count + 1;`
 - `count +=1;` // shorthand
 - `++count;` // unary operator
- ❑ Prefix form: increment before the value is used.
 - `int total, count = 1;`
 - `total = ++count + 6;` // count=2; total = 8
- ❑ Postfix form: increment after the value is used.
 - `total = count++ + 6;` // total = 7; count=2
 - `total = 6 + count++;`

Decrement Operators

- Unary operator to decrease the integer variable by 1.
 - `total = --count + 6;`
 - `total = 6 + count--;`
- Both increment and decrement operators are useful in **loops**, as we shall see in Chapter 3.

Comma Operator

- Specify several expressions in an assignment
 - `int num1;`
 - `int num2;`
 - `int num3;`
 - `int num4;`
 - `num4 = (num1=10, num2=20, num3=30);`

- Operator Precedence (see P.75)
- It is a good idea to insert parentheses to make sure.

Casting

- ❑ The conversion of a value from one type to another
 - Implicit cast
 - ❑ `int n;`
 - ❑ `float a = 7.0;`
 - ❑ `float b = 2.0;`
 - ❑ `float c = a / b;`
 - ❑ `n = c;`
 - The floating-point value will be rounded down to the nearest integer (3)
 - The compiler will issue a warning.
 - Explicit cast
 - ❑ `n = static_cast<int> (c);`
 - The compiler assumes you know what you are doing and will not issue a warning.
 - Old-style cast (not recommended)
 - ❑ `n = (int) c ;`

Bitwise Operators

- The bitwise operators are useful in programming hardware devices.
 - Review Chapter 4 of Forouzan.
 - & AND
 - | OR
 - ^ exclusive OR
 - ~ NOT
 - >> shift right
 - << shift left
- You may pack a set of on-off flags into a single variable.

Examples of Bitwise Operators

□ Bitwise AND

- `char letter = 0x41;`
- `char mask = 0x0F;`
- `letter = letter & mask;`

□ Bitwise Shift Operators

- `char j = 2; // 0000 0010`
- `j <<= 1; // 0000 0100`
- `j >>= 2; // 0000 0001`
- `J = -104; // 1001 1000`
- `J >>= 2; // 1110 0110 (=?)`

Storage Duration and Scope

□ Duration

- Automatic storage duration
- Static storage duration
- Dynamic storage duration (Chapter 4)

□ Scope

- The part of your program over which the variable name is valid.

Automatic Variables

- Automatic variables have **local scope (block scope)**.
 - Every time the block of statements containing a declaration for an automatic variable is executed, the variable is created anew.
 - If you specified an initial value for the automatic variable, it will be reinitialized each time it is created.
 - When an automatic variable dies, its memory on the stack will be freed for used by other automatic variables.

Ex2_07.cpp in P.87

- ❑ From the viewpoint of the outer block, the inner block just behaves like a single statement.
- ❑ The inner block also declares a variable named `count1`, so the variable `count1` declared in the outer block becomes hidden now.
- ❑ Other variables (`count3`) declared at the beginning of the outer scope are accessible from within the inner scope.
- ❑ After the brace ending the inner scope, `count2` and the inner `count1` cease to exist.
- ❑ Try to uncomment the line

```
// cout << count2 << endl;
```

to get an error.

Global Variables

- ❑ Variables declared outside of all blocks are called global variables and have **global namespace scope**.
- ❑ Global variables have **static storage duration** by default. It will exist from the start of execution of the program, until execution of the program ends.
 - If you do not specify an initial value for a global variable, it will be initialized with **0** by default.
 - On the contrary, automatic variables will not be initialized by default.
- ❑ Figure 2-13 shows an example that the **lifetime** and **scope** may be different (`value4`).

Class View Pane of IDE Window



- ❑ Do NOT declare all variables global!
- ❑ For a large program, there are many variables:
 - Accidental erroneous modification of a variable
 - Difficult to name all the variables consistently and uniquely
 - Memory occupied for the duration of program execution

Namespaces

- Namespace is a mechanism to prevent accidental naming clash.
 - The libraries supporting the CLR and Windows Forms use namespaces extensively.
 - The ANSI C++ standard library does, too.
- Every non-local variable or function must be qualified.

```
// Ex2_09.cpp
#include <iostream>

int value = 0;

int main()
{
    std::cout << "enter an integer: ";
    std::cin >> value;
    std::cout << "\nYou entered " << value
        << std::endl;
    return 0;
}
```

global namespace scope

Note the absence of using declarations for `cout` and `endl`

using Directive

- `using namespace std;`
 - This imports **all** the names from the `std` namespace
 - so that you don't need to qualifying the name with prefix `std::` in your program.
 - However, this negates the reason for using a namespace.
 - Only introduce the names that you use with “using declaration”:
 - `using std::cout;`
 - `using std::endl;`

Declaring a Namespace

```
// Ex2_10.cpp
// Declaring a namespace
#include <iostream>

namespace myStuff
{
    int value = 0;
}

int main()
{
    std::cout << "enter an integer: ";
    std::cin  >> myStuff::value;
    std::cout << "\nYou entered " << myStuff::value
              << std::endl;

    return 0;
}
```

using Directive

```
// Ex2_11.cpp
// using a using directive
#include <iostream>

namespace myStuff
{
    int value = 0;
}

using namespace myStuff;

int main()
{
    std::cout << "enter an integer: ";
    std::cin >> value;
    std::cout << "\nYou entered " << value
        << std::endl;
    return 0;
}
```

using Declaration

```
// Ex2_11a.cpp
// using a using declaration
#include <iostream>

namespace myStuff
{
    int value = 0;
}

using myStuff::value; // only important the variables you need

int main()
{
    std::cout << "enter an integer: ";
    std::cin >> value;
    std::cout << "\nYou entered " << value
                << std::endl;
    return 0;
}
```

CLI Specific

□ Fundamental Data Types

- `long long`
8bytes
- `unsigned long long`
8bytes
 - `long int` only occupies 4 bytes

□ Use `safe_cast` and not `static_cast` in your C++/CLI code.

□ Each ANSI fundamental type name maps to a **value class type** in the `System` namespace.

- See P.98
- It is suggested to write
 - `int count = 10;`
 - `double value = 2.5;`
- instead of
 - `System::Int32 count = 10;`
 - `System::Double value=2.5;`

C++/CLI Output to the Command Line

- Console – a class in the System namespace

- Write()

- WriteLine

- `Console::WriteLine(L"\n Orange");`

- Formatting the Output:

- `Console::WriteLine(L"Sum of {0} and {1} = {2}", i, j, i+j);`

- `Console::WriteLine(L"{2} = {0} + {1}", i, j, i+j);`

C++/CLI Input from the Keyboard

- ❑ `String^ line = Console::Readline();`
- ❑ `char ch = Console::Read();`
- ❑ `ConsoleKeyInfo keyPress = Console::ReadKey(true);`
 - `true` – hide the character
 - `false` – display the character
- ❑ When you press the button 'a' without Caps Lock:
 - `keyPress.KeyChar = 'a'`
 - `keyPress.Key = A`
- ❑ When you press the button '1' on the NumPad:
 - `keyPress.KeyChar = '1'`
 - `keyPress.Key = NumPad1`

怎麼辦?

- 考不好
- 考得**很**不好
- 怎樣才能考好?
- 我會被當掉嗎?
- 一定要念英文嗎?