ENGINEERING PROGRAMMING I

Chapter 02

2017

Administrivia

- 1. We would need at least 5 volunteers who don't mind bringing your laptop to your lab session.
  - Because we found a couple of PCs in the lab are not working…☹
- 2. The next chapter you need to read is Ch.4

Chapter 2: Introduction to C++

Starting Out with C++
Early Objects
Seventh Edition

by Tony Gaddis, Judy Walters,
and Godfrey Muganda

Addison-Wesley
is an imprint of
PEARSON

Topics

- 2.1 The Parts of a C++ Program
- 2.2 The cout Object
- 2.3 The #include Directive
- 2.4 Standard and Prestandard C++
- 2.5 Variables, Constants, and the Assignment Statement
- 2.6 Identifiers
- 2.7 Integer Data Types
- 2.8 The char Data Type
- 2.9 The C++ string Class
- 2.10 Floating-Point Data Types
- 2.11 The bool Data Type
- 2.12 Determining the Size of a Data Type
- 2.13 More on Variable Assignments and Initialization
- 2.14 Scope
- 2.15 Arithmetic Operators
- 2.16 Comments

Anatomy of a simple C++ source code

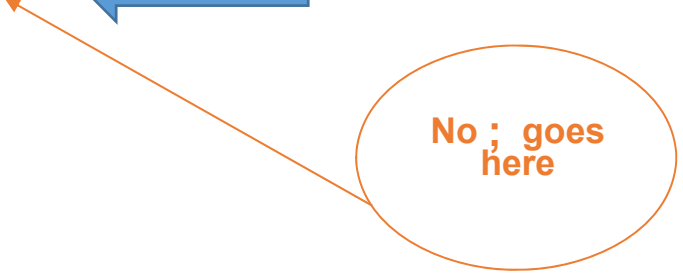
```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, there!";
   return 0;
}
```

// sample C++ program

- Comments
  - Portions of the code ignored by compiler
  - Allow programmers to leave some explanation about the program.
  - Start with // and continue until the end of the line

Anatomy of a simple C++ source code

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, there!“ <<endl;
   return 0;
}
```



#include <iostream>

- directive
  - When a compiler translates a source code, the compiler follows *directives* in the source code before it translates the rest of the code.
  - #include directive tells the compiler to use a library code.
  - <iostream> is the name of a standard library that allows us to use standard input and output devices, such as monitor console and keyboard.
  - E.g. In order to print text in a console, we use "cout".
  - Since cout is defined in iostream library, we include it.

Anatomy of a simple C++ source code

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, there!“ << endl;
   return 0;
}
```

using namespace std;

- Let's defer our discussion on this.
- For now, just memorize that we need this line.

Anatomy of a simple C++ source code

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, there!“ << endl;
   return 0;
}
```

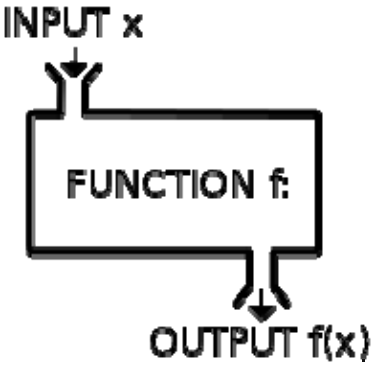
int main()

- main() is the name of the starting function in C++ programs.

- Definition of Function
  - A special relationship where each input has a single output.

- Example functions :

$$h(x) = ax + b$$
$$f(x) = h(x) + ax + b$$
$$f(x) = b$$
$$f() = b$$



- A function is to return a value.

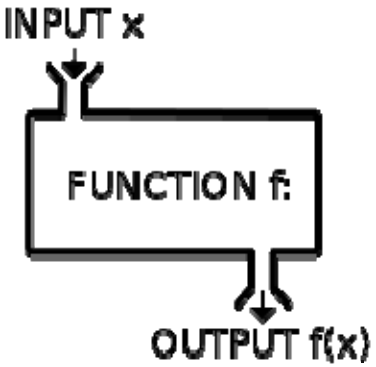
- main() is no exception.
  - int main() means, main() will return an integer value.

Anatomy of a simple C++ source code

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, there!“ << endl;
   return 0;
}
```

{ }

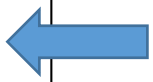
{ // beginning of a function



} // end of a function

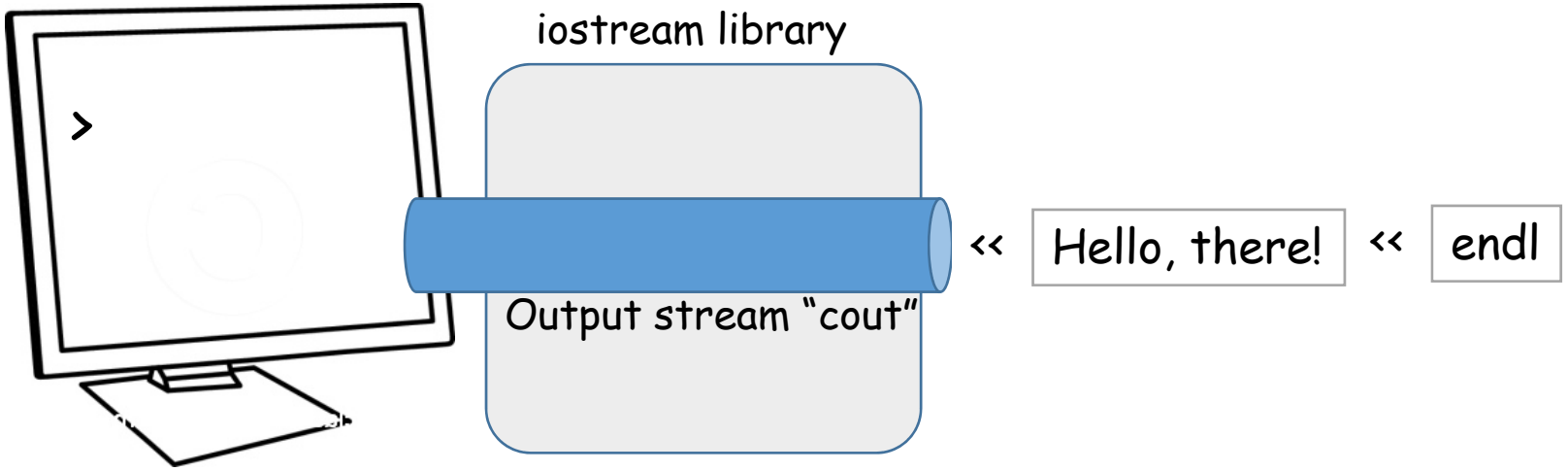
Anatomy of a simple C++ source code

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, there!“ << endl;
   return 0;
}
```



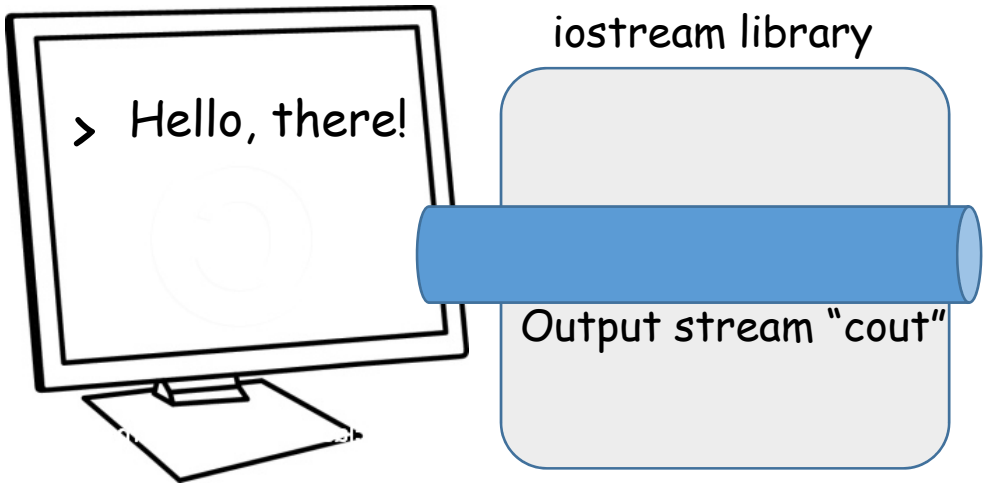
cout << "Hello, there!" << endl;

- cout
  - Standard output stream object defined in iostream library.
- <<
  - Stream insertion operator
- "Hello, there!"
  - Text that we'd like to display.
  - In Programming, we call it a text string, or a string.
- endl
  - end of line



cout << "Hello, there!" << endl;

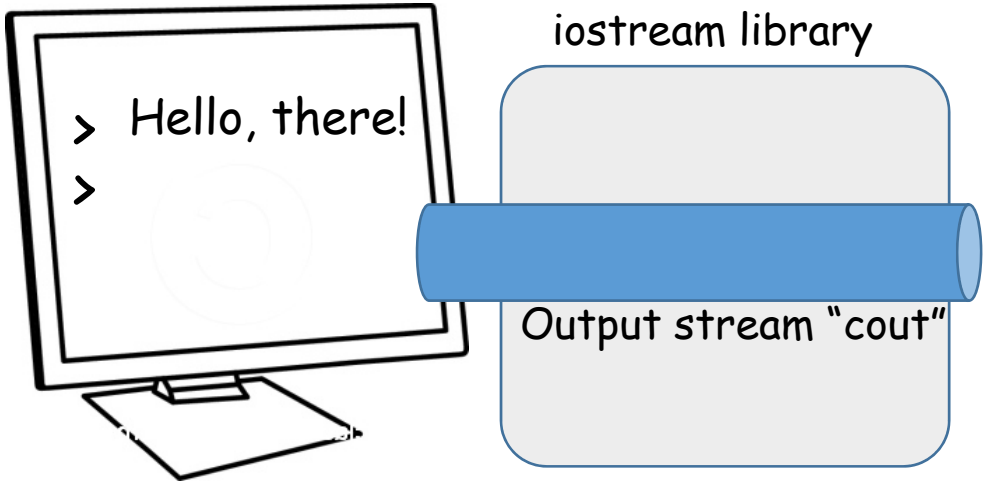
- cout
  - Standard output stream object defined in iostream library.
- <<
  - Stream insertion operator
- "Hello, there!"
  - Text that we'd like to display.
  - In Programming, we call it a text string, or a string.
- endl
  - end of line



Hello, there! << endl

cout << "Hello, there!" << endl;

- cout
  - Standard output stream object defined in iostream library.
- <<
  - Stream insertion operator
- "Hello, there!"
  - Text that we'd like to display.
  - In Programming, we call it a text string, or a string.
- endl
  - end of line



endl

cout != output of main()

- While main() is being executed, we can print out as many outputs as we can.
- But there can be only one return value from main().

Anatomy of a simple C++ source code

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, there!“ << endl;
   return 0;
}
```

return 0;

- return 0;
  - main() is supposed to return an integer value.
  - This is where we return the value.
  - Q: Who needs this value?
    A: Operating systems.

Special Characters

Character	Name	Description
`//`	Double Slash	Begins a comment
`#`	Pound Sign	Begins preprocessor directive
`< >`	Open, Close Brackets	Encloses filename used in `#include` directive
`( )`	Open, Close Parentheses	Used when naming function
`{ }`	Open, Close Braces	Encloses a group of statements
`" "`	Open, Close Quote Marks	Encloses string of characters
`;`	Semicolon	Ends a programming statement

Details

- C++ is case-sensitive. Uppercase and lowercase characters are different characters. '`Main`' is not the same as '`main`'.
- Every { must have a corresponding }, and vice-versa.

More about cout

- Displays information on computer screen
- Use << to send information to cout
  - `cout << "Hello, there!";`
- Can use << to send multiple items to cout
  - `cout << "Hello, " << "there!";`
  - Or
  - `cout << "Hello, ";`
  - `cout << "there!";`

Starting a New Line

- To get multiple lines of output on screen
  - Use endl

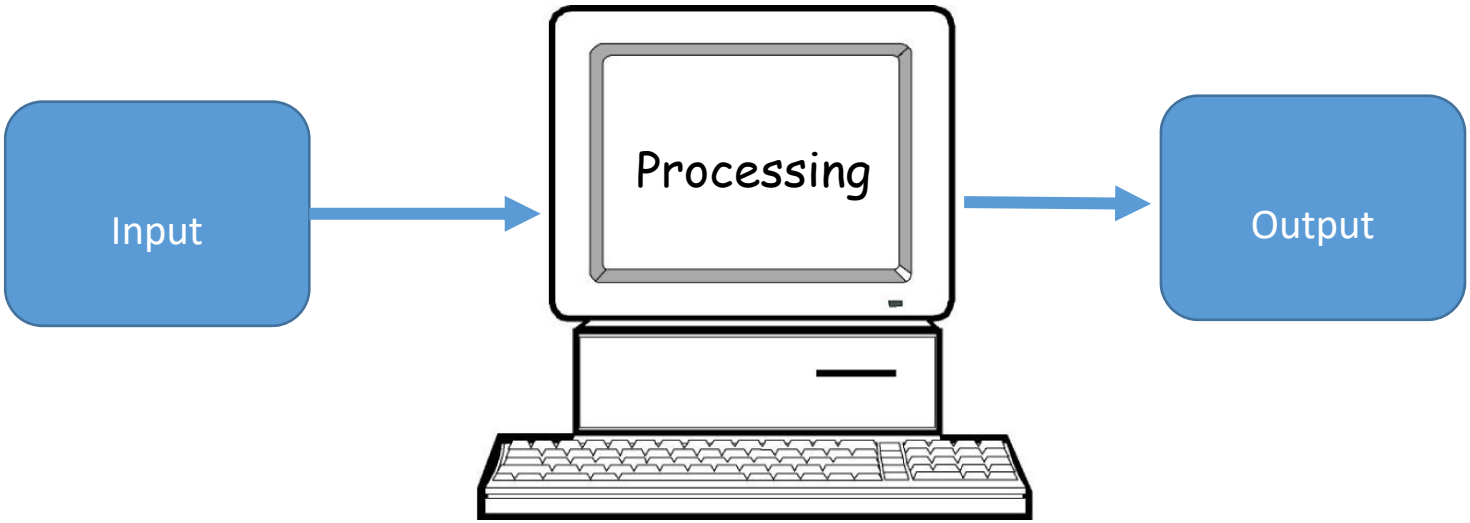
```
cout << "Hello, there!" << endl;
```

  - Use \n in an output string

```
cout << "Hello, there!\n";
```

Now, we know how to show outputs. But what about inputs?

- We need memory space to hold the input value.
  - So, let's learn variables first.



Program#1: Adding 2 numbers

- Alice: Hey Bob, I just learned how to add two numbers today.
- Bob: Cool!
- Alice : Give me the first number.
- Bob: 2.
- Alice : Ok, and give me the second number.
- Bob: 5.
- Alice : Ok, here's the answer: 2 + 5 = 7.
- Bob: Wow! You are amazing!

after Bob says "2", Alice has to keep this number in his mind.

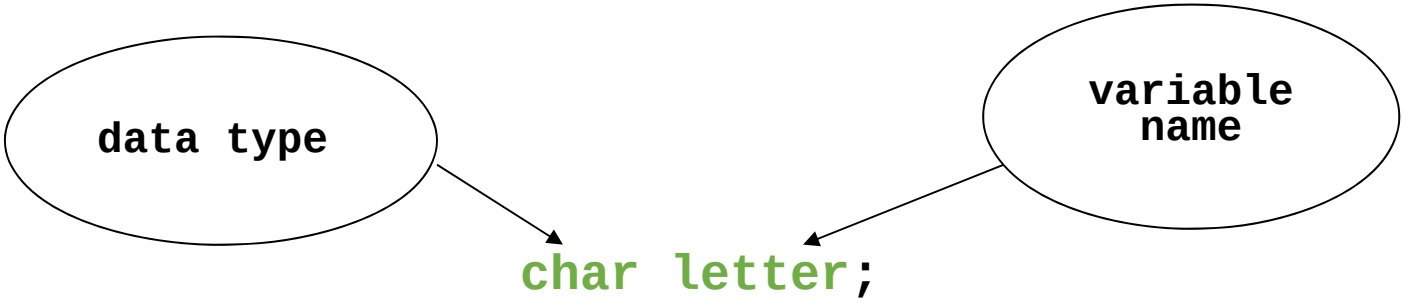
after Bob says "5", Alice also needs to keep this number in his mind.

Need Variables

First number: 2 **Second number:** 5 **Sum:** 7

Variable

- Variable
  - Has a name (id) and a type of value it can hold



  - is used to reference a location in memory where a value can be stored
  - Must be defined before it can be used
  - The value that is stored can be changed, i.e., it can "vary"
    - E.g.) $h(x) = ax + b$

      x is a variable while a and b are constants.

cin

- User input goes from keyboard to the input buffer, where it is stored as characters
- cin converts the data to the type that matches the variable
  - `int height;`
  - `cout << "How tall is the room? ";`
  - `cin  >> height;`

cin

- Can be used to input multiple values
  - `cin >> height >> width;`
- Multiple values from keyboard must be separated by spaces or [Enter]
- Must press [Enter] after typing last value
- Multiple values need not all be of the same type
- Order is important; first value entered is stored in first variable, etc.

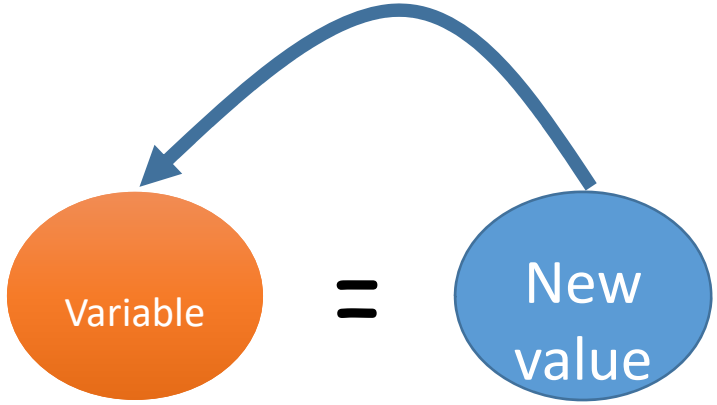
Again, Variables

- If a new value is stored in the variable, it replaces the previous value
- The previous value is overwritten and can no longer be retrieved

```
int age;
age = 17;      // age is 17
cout << age;   // Displays 17
age = 18;      // Now age is 18
cout << age;   // Displays 18
```

Assignment Statement

- Uses the = operator
- Has a single variable on the left side and a value on the right side
- Copies the value on the right into the variable on the left
  - `item = 12;`
  - `item = 1 + 2;`
  - `item = 2*item + 1;`



Constants

- Constant
  - Data item whose value does not change during program execution
  - Is also called a literal

```
'A'       // character constant
"Hello"   // string literal
12        // integer constant
3.14      // floating-point constant
```

Program#1: Adding 2 numbers

```
#include <iostream>
using namespace std;
int main()
{
 int first_number, second_number, total_number;

 cout << “Alice: Hey Bob, ";
 cout << “I just learned how to add two numbers today.\n”;
 cout << “Bob: Cool!\n”;
 cout << “Alice: Give me the first number.\n";
 cout << “Bob: ";
 cin >> first_number;
 cout << “Alice: Ok, and give me the second number.\n";
 cout << “Bob: ";
 cin >> second_number;
```

Program#1: Adding 2 numbers

```
 total_number = first_number + second_number;

 cout << “Alice: Ok, here’s the answer: ";
 cout << first_number << “ + “ << second_number;
 cout << “ = “ << total_number << “.\n”;
 cout << “Bob: Wow! You are amazing!\n”;

 return 0;
}
```

Identifiers

- Programmer-chosen names, such as variables
- Name should indicate the use of the identifier
- Cannot use C++ key words as identifiers
  - main, int, char, include, etc.

- Must begin with alphabetic character or _, followed by alphabetic, numeric, or _ .
  - Alphabetic character may be upper- or lowercase

Valid and Invalid Identifiers

IDENTIFIER	VALID?	REASON IF INVALID
`totalSales`	Yes	
`total_Sales`	Yes	
`total.Sales`	No	Cannot contain period
`4thQtrSales`	No	Cannot begin with digit
`totalSale$`	No	Cannot contain $

Integer Data Types

- Designed to hold whole numbers
- Can be signed or unsigned
  - 12 -6 +3

- Available in different sizes (i.e., number of bytes):

  `short, int, long`

- Size of short ≤ size of int ≤ size of long

Defining Variables

- Variables of the same type can be defined
  - In separate statements

```
int length;
int width;
```

  - In the same statement

```
int length,
    width;
```

- Variables of different types must be defined in separate statements

Integral Constants

- To store an integer constant in a long memory location, put '**L**' at the end of the number: `1234L`

- Constants that begin with '**0**' (zero) are octal, or base 8:
  - E.g.) `075`

- Constants that begin with '**0x**' are hexadecimal, or base 16:
  - E.g.) `0x75A`

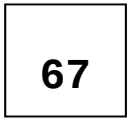
The char Data Type

- Used to hold single characters or very small integer values
- Usually occupies 1 byte of memory
- A numeric code representing the character is stored in memory

SOURCE CODE

```
char letter = 'C';
```

MEMORY



String Constant

- Can be stored a series of characters in consecutive memory locations

"Hello"

- Stored with the null terminator, \0, at end

- Is comprised of characters between the " "

H	e	l	l	o	\0

A character or a string constant?

- A character constant is a single character, enclosed in single quotes:

```
'C'
```

- A string constant is a sequence of characters enclosed in double quotes:

```
"Hello, there!"
```

- A single character in double quotes is a string constant, not a character constant:

```
"C"
```

The C++ string Class

- Must #include <string> to create and use string objects
- Can define string variables in programs

```
string name;
```

- Can assign values to string variables with the assignment operator

```
name = "George";
```

- Can display them with cout

```
cout << name;
```

Floating-point Data Types

- Designed to hold real numbers
  - `12.45` `-3.8`
- Stored in a form similar to scientific notation
- Numbers are all signed

- Available in different sizes (number of bytes):
  `float, double, and long double`
- Size of float ≤ size of double ≤ size of long double

Floating-point Constants

- Can be represented in
  - Fixed point (decimal) notation:

```
31.4159          0.0000625
```

  - E notation:

```
3.14159E1        6.25e-5
```

- Are **double** by default
- Can be forced to be float

```
3.14159F
```

or long double

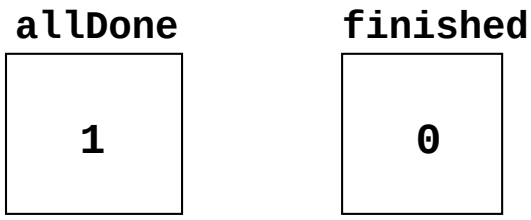
```
0.0000625L
```

Assigning Floating-point Values to Integer Variables

- If a floating-point value is assigned to an integer variable
  - The fractional part will be truncated.
  - The value is not rounded
  - `int rainfall = 3.88;`
  - `cout << rainfall;  // Displays 3`

The bool Data Type

- Represents values that are true or false
- bool values are stored as short integers
- false is represented by 0, true by 1
  - `bool allDone = true;`
  - `bool finished = false;`



Determining the Size of a Data Type

- The sizeof operator gives the size of any data type or variable
  - `double amount;`
  - `cout << "A float is stored in "`
  - `     << sizeof(float) << " bytes\n";`
  - `cout << "Variable amount is stored in "`
  - `     << sizeof(amount) << " bytes\n";`

More on Variable Assignments and Initialization

- Assigning a value to a variable
  - Assigns a value to a previously created variable
  - A single variable name must appear on left side of the = symbol

```
int size;
size = 5;     // legal
5 = size;     // not legal
```

Variable Assignment vs. Initialization

- Initializing a variable
  - Gives an initial value to a variable at the time it is created
  - Can initialize some or all variables of definition

```
int length = 12;
int width = 7, height = 5, area;
```

Scope

- The scope of a variable is that part of the program where the variable may be used
- A variable cannot be used before it is defined

```
int a;
cin >> a;   // legal
cin >> b;   // illegal, why?
int b;
```

Arithmetic Operators

- Used for performing numeric calculations
- C++ has unary, binary, and ternary operators
  - unary (1 operand) -5
  - binary (2 operands) 13 - 7
  - ternary (3 operands) exp1 ? exp2 : exp3

Binary Arithmetic Operators

SYMBOL	OPERATION	EXAMPLE	ans
+	addition	`ans = 7 + 3;`	10
-	subtraction	`ans = 7 - 3;`	4
*	multiplication	`ans = 7 * 3;`	21
/	division	`ans = 7 / 3;`	2
%	modulus	`ans = 7 % 3;`	1

/ Operator

- C++ division operator (/)performs integer division if both operands are integers
  - `cout << 13 / 5;    // displays 2`
  - `cout <<  2 / 4;    // displays 0`

- If either operand is floating-point, the result is floating-point
  - `cout << 13 / 5.0;  // displays 2.6`
  - `cout << 2.0 / 4;   // displays 0.5`

% Operator

- C++ modulus operator (%) computes the remainder resulting from integer division
  - `cout << 9 % 2;   // displays 1`

- % requires integers for both operands
  - `cout << 9 % 2.0; // error`

Comments

- Are used to document parts of a program
- Are written for persons reading the source code of the program
  - Indicate the purpose of the program
  - Describe the use of variables
  - Explain complex sections of code
- Are ignored by the compiler

Single-Line Comments

- Begin with // through to the end of line

```
int length = 12; // length in inches
int width = 15;  // width in inches
int area;        // calculated area
// Calculate rectangle area
area = length * width;
```

Multi-Line Comments

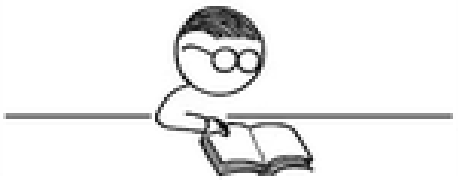
- Begin with /* and end with */
- Can span multiple lines

```
/*----------------------------
   Here's a multi-line comment
  ----------------------------*/
```

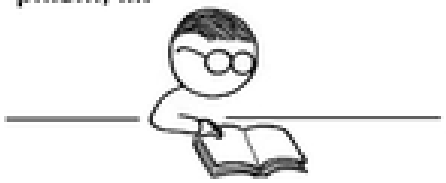
- Can also be used as single-line comments

```
int area;   /* Calculated area */
```

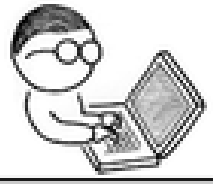
Days 1 - 10
Teach yourself variables, constants, arrays, strings, expressions, statements, functions,...



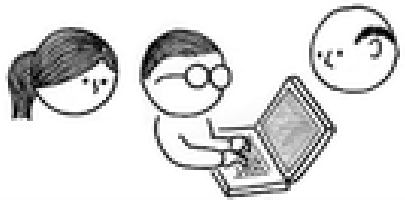
Days 11 - 21
Teach yourself program flow, pointers, references, classes, objects, inheritance, polymorphism, ...



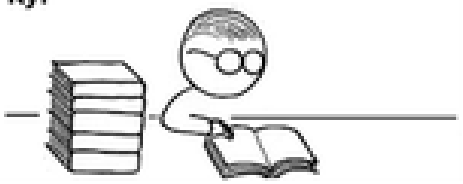
Days 22 - 697
Do a lot of recreational programming. Have fun hacking but remember to learn from your mistakes.



Days 698 - 3648
Interact with other programmers. Work on programming projects together. Learn from them.



Days 3649 - 7781
Teach yourself advanced theoretical physics and formulate a consistent theory of quantum gravity.

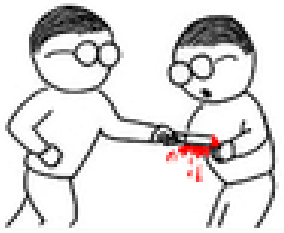


Days 7782 - 14611
Teach yourself biochemistry, molecular biology, genetics,...

Day 14611
Use knowledge of biology to make an age-reversing potion.

Day 14611
Use knowledge of physics to build flux capacitor and go back in time to day 21.

Day 21
Replace younger self.



As far as I know, this is the easiest way to "Teach Yourself C++ in 21 Days".