Computer Programming

EC2010

Dr. J. Jananie, Senior Lecturer, Department of Computer Engineering, Faculty of Engineering, University of Jaffna.

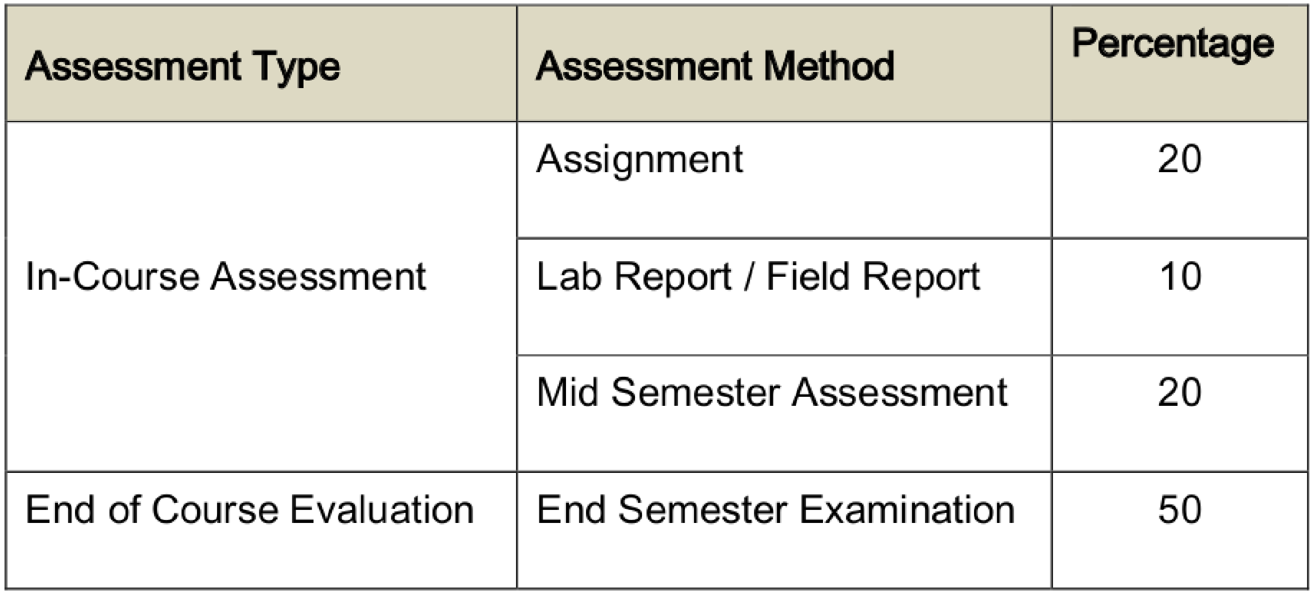
Information

Lecture:
Tuesdays: (10.10 a.m. – 12.00 p.m.) – 2 slots
Thursdays: (11.05 – 12 p.m.)

Textbook:

Paul Deitel and Harvey Deitel, C++ How to Program, 10th edition, Deitel & Associates, Inc., ISBN:9780134448237
Herbert Schildt, C++ the complete reference, 4th edition, McGraw-Hill Education, ISBN: 0072226803

Assessment/ Evaluation Details



Assessment Type	Assessment Method	Percentage
	Assignment	20
In-Course Assessment	Lab Report / Field Report	10
	Mid Semester Assessment	20
End of Course Evaluation	End Semester Examination	50

Assignment: Quiz (In class/ online), Lab exam, GUI project
Lab: Source code and lab report are to be submitted before leaving the labs.

Attendance

Lab- 100% (mandatory) – You have to talk to me if you are unable to attend for considerable reason. There are four groups and you should be able to join other three in this case, except you are in last group for that practical and reason for absence is unexpected.

Lectures- At least 80% (mandatory) if < 80%, will not be allowed to sit for the exams, no excuses (no medical certificates, etc.)

Slides/materials Will be uploaded to "MS Teams".

– Team code: **3yswsil**

Chapter 01: Program Structure

Basic structure of programming languages; Compilers and Interpreters; Using library functions for input/ output;

ILO: Describe various program building blocks

Acknowledged: C++ for Everyone *by C Horstmann by John Wiley & Sons.*

What Is Programming?

A computer program tells a computer, in minute detail, the sequence of steps that are needed to fulfill a task.

Hardware:

The physical computer and peripheral devices are collectively called the hardware.

Software:

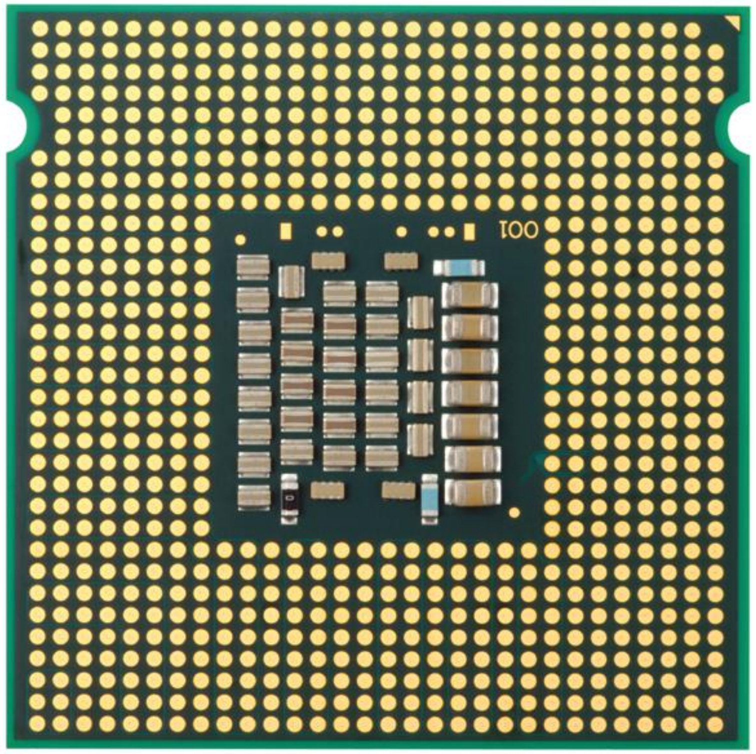
The programs the computer executes are called the software.

What Is Programming?

Programming is the act of designing and implementing computer programs.

The Anatomy of a Computer – The CPU

- The CPU (central processing unit)
  - heart of the computer
  - executes one operation at a time
  - performs program control and data processing



The Anatomy of a Computer – The CPU

- The computer stores data and programs in memory
  - Primary memory - memory chips
    - Random access memory (RAM) (read-write memory)
    - Read-only memory (ROM)
  - Secondary storage devices
    - disk drives
    - CDs

What is a programming language?

A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks.

The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal.

Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

High-level language - designed to be easy for human beings to write programs in and to be easy for human beings to read.

E.g. C++, Java

Assembly/low-level language - A symbolic representation of machine instructions.

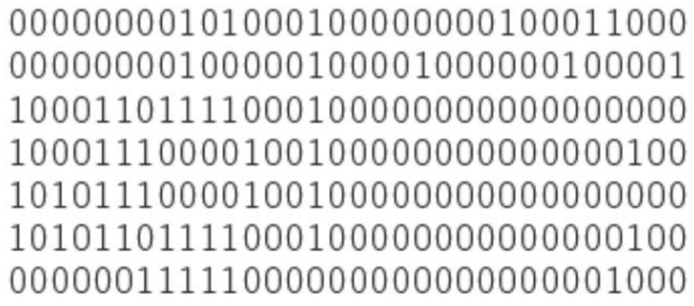
E.g. MIPS

lw – load word

sw – store word

jr – jump register

Machine language - A binary representation of machine instruction



```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

High-Level Languages and the Compiler

- High-level languages like C++ are independent of the processor type and hardware. They will work equally well:
  - on an Intel Pentium and a processor
  - in a cell phone
- The compiler
  - a special computer program, that translates the higher-level description (a program) into machine instructions for a particular processor.
- Low-level language: the machine code for a specific CPU
  - the compiler-generated machine instructions are different, but the programmer who uses the compiler need not worry about these differences.

Compiler - are used to convert high level languages (like C, C++) into assembly/machine code.

Example : gcc , Microsoft Visual Studio

Assembler - Assembler are used to convert assembly language code into machine code.

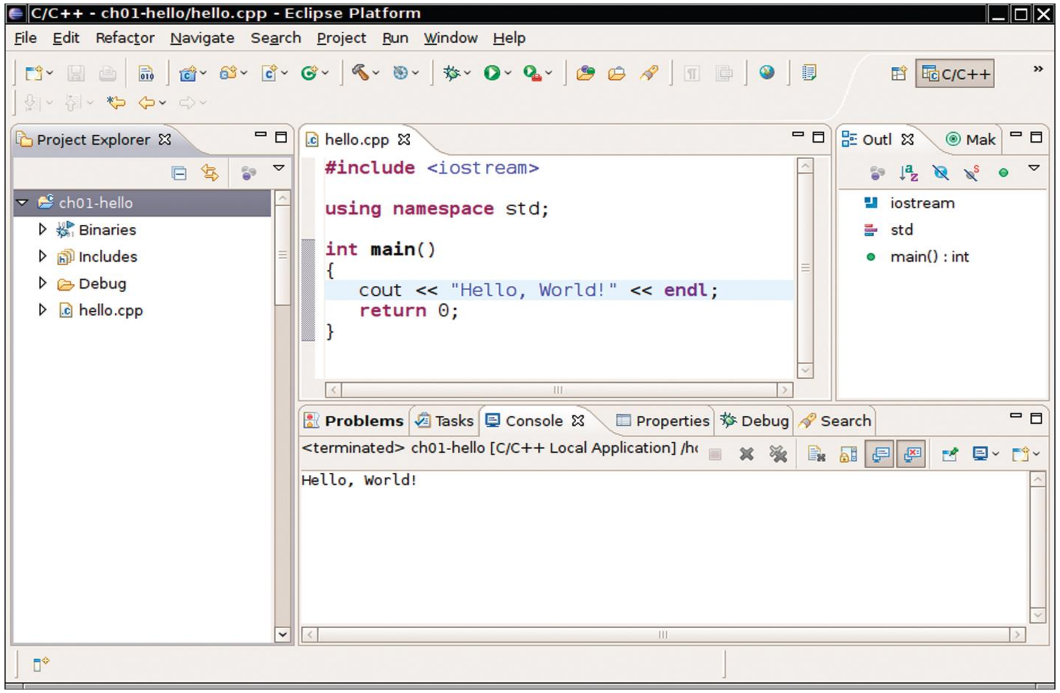
Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, JavaScript use interpreters.	Programming language like C, C++ use compilers.

The Evolution of C++

- Ancient history (pre 1972)
- C (1972)
- ANSI Standard C (1989)
- Meanwhile, Bjarne Stroustrup of AT&T adds features of the language Simula (an object-oriented language designed for carrying out simulations) to C resulting in:
- C++ (1985)

Becoming Familiar with Your Programming Environment

- You will need to know how to log in (if needed),
- and, how to start your C++ development environment. An IDE (integrated development environment) is where you will most likely work.

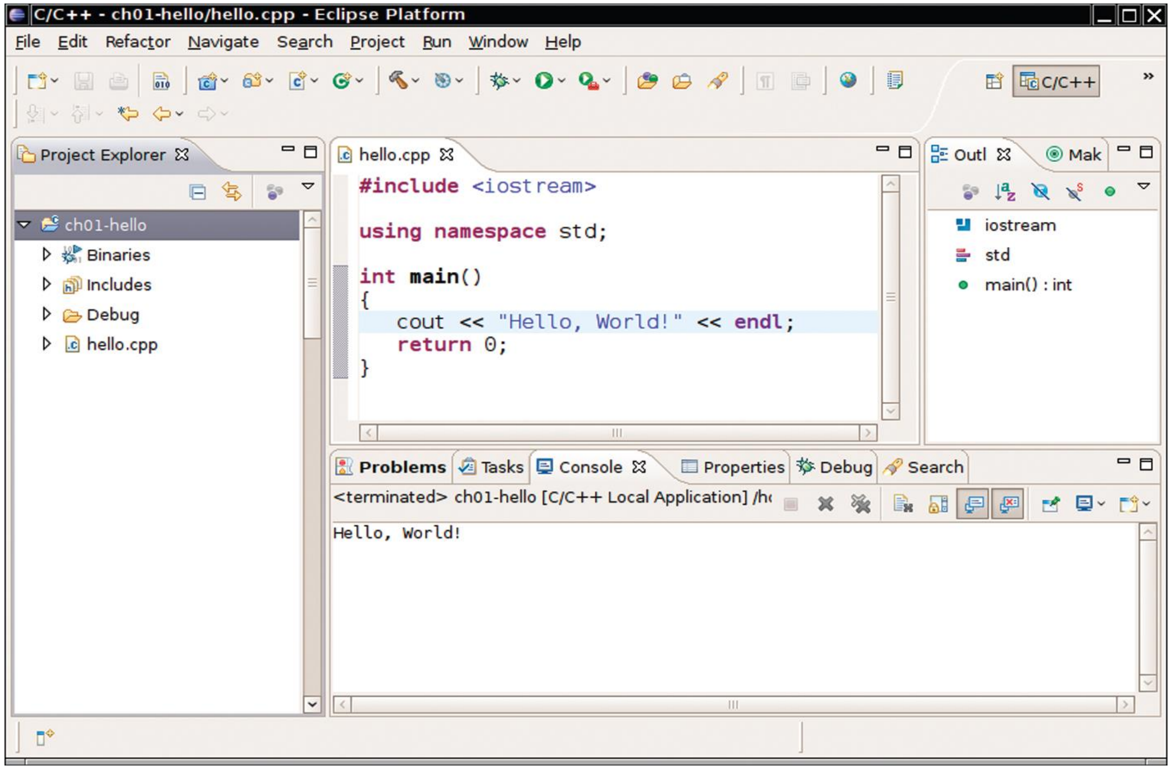


```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

```
<terminated> ch01-hello [C/C++ Local Application] /h
Hello, World!
```

Becoming Familiar with Your Programming Environment

- You will become a typist because you will use an editor to type your C++ programs into the IDE. Your program is called a source file.

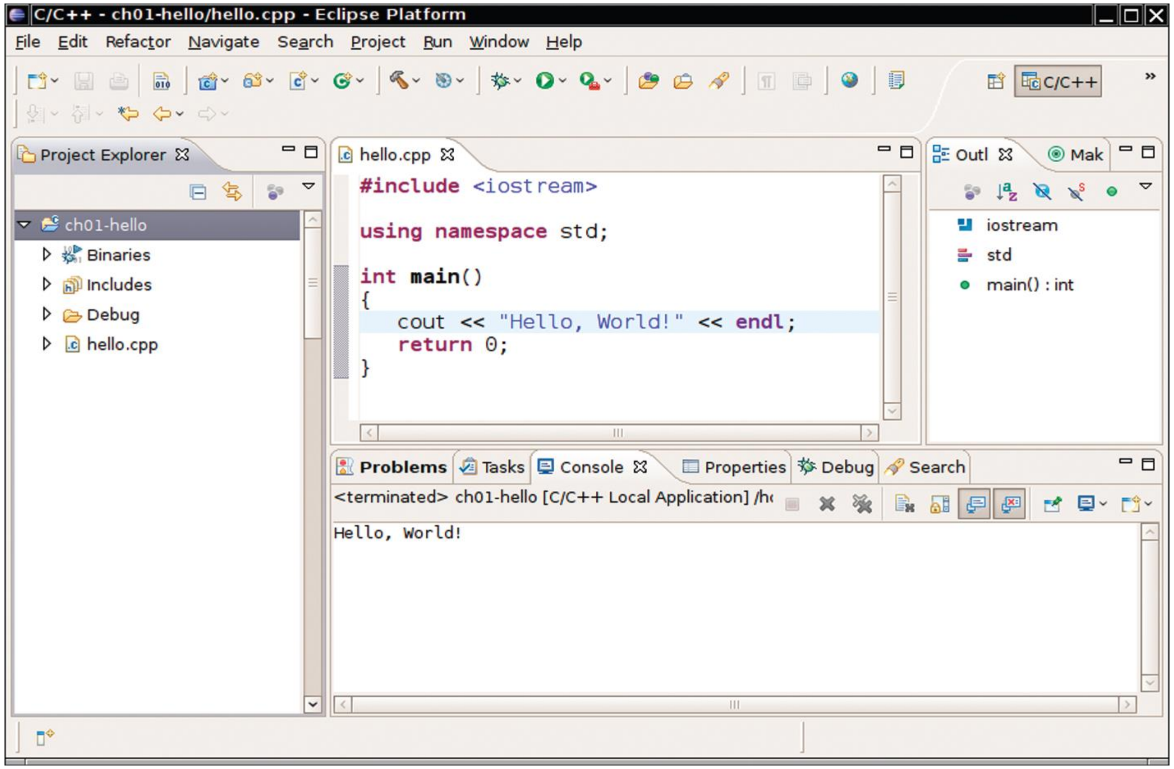

C/C++ - ch01-hello/hello.cpp - Eclipse Platform
File Edit Refactor Navigate Search Project Run Window Help
C/C++
Project Explorer
ch01-hello
Binaries
Includes
Debug
hello.cpp
hello.cpp
Outl
Mak
iostream
std
main() : int
Problems Tasks Console Properties Debug Search

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

```
<terminated> ch01-hello [C/C++ Local Application] /h
Hello, World!
```

Becoming Familiar with Your Programming Environment

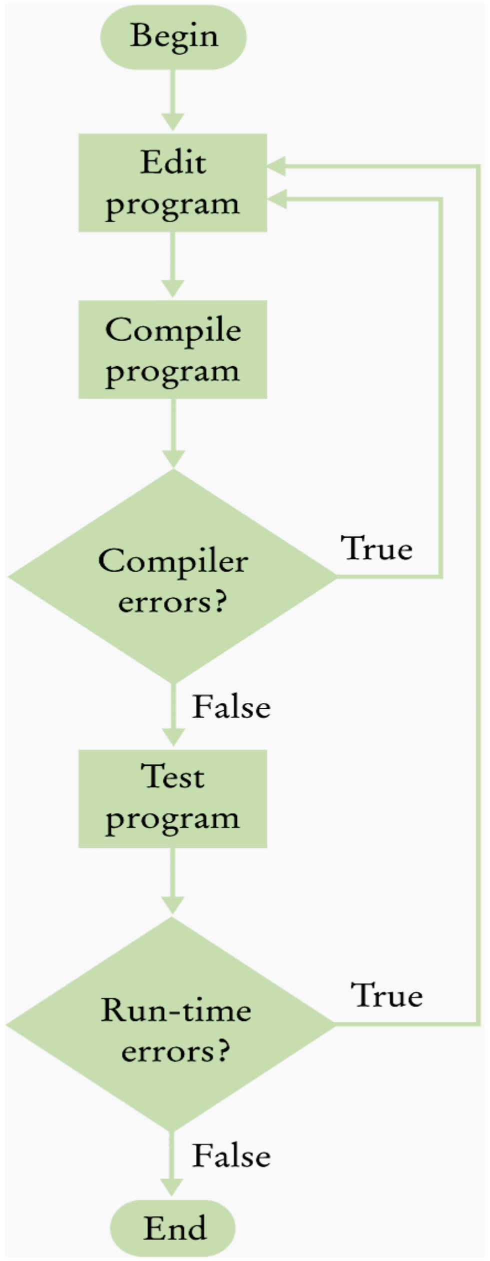
- You will need to learn how to compile and run your program in the IDE.



```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello, World!" << endl;
    return 0;
}
```

<terminated> ch01-hello [C/C++ Local Application] /h

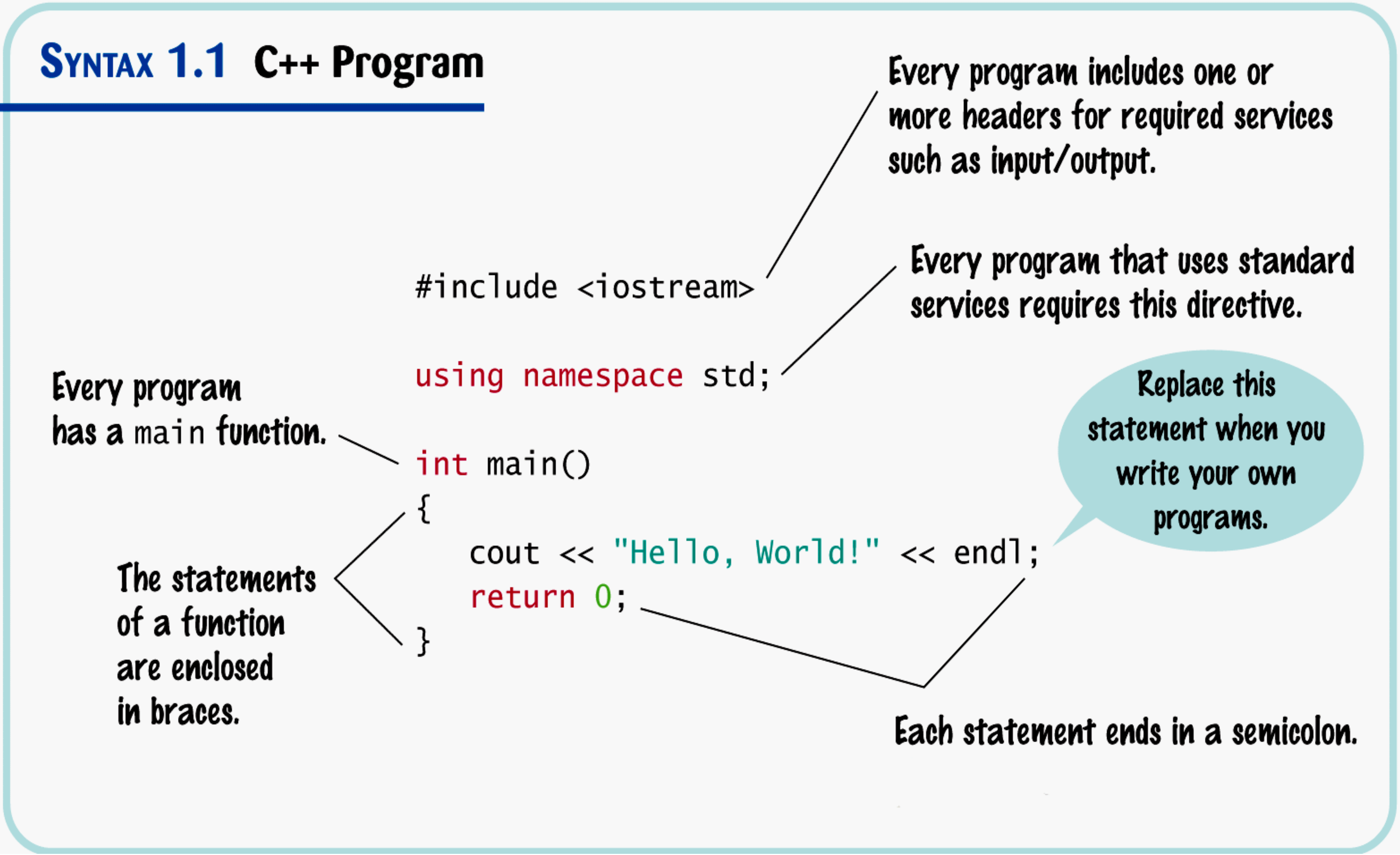
```
Hello, World!
```



This process reflects the way programmers work

(shown as a flowchart)

Analyzing Your First Program



```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Syntax 1.1 C++ Program
Every program includes one or more headers for required services such as input/output.
Every program that uses standard services requires this directive.
Every program has a main function.
Replace this statement when you write your own programs.
The statements of a function are enclosed in braces.
Each statement ends in a semicolon.

Analyzing Your First Program

- The first line tells the compiler to include a service for “stream input/output”. Later you will learn more about this but, for now, just know it is needed to write on the screen.

ch01/hello.cpp

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Analyzing Your First Program

- The second line tells the compiler to use the “standard namespace”. Again

ch01/hello.cpp

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Analyzing Your First Program

- The next set of code defines a function.
  The name of this function is main.

ch01/hello.cpp

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Analyzing Your First Program

- The main function "returns" an "integer" (that is, a whole number without a fractional part, called int in C++)
  with value 0. This value indicates that the program finished successfully.

ch01/hello.cpp

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Analyzing Your First Program

- To show output on the screen, we use an entity called cout.
- What you want seen on the screen is “sent” to the cout entity using the << operator (sometimes called the insertion operator): << "Hello, World!"

ch01/hello.cpp

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Analyzing Your First Program

- You can display more than one thing by re-using the << operator: << "Hello, World!" << endl;

ch01/hello.cpp

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Analyzing Your First Program

The output statement

cout << "Hello World!" << endl; is an output statement.

Analyzing Your First Program

```
cout << "Hello World!" << endl;
```

- "Hello World!" is called a string.
- You must put those double-quotes around strings.

- The endl symbol denotes an end of line marker which causes the cursor to move to the next screen row.

Analyzing Your First Program

- Each statement in C++ ends in a semicolon.

ch01/hello.cpp

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Common Error – Omitting Semicolons

Common error

Omitting a semicolon

Oh No!

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl
   return 0;
}
```

Errors

Without that semicolon you actually wrote:

```
cout << "Hello, World!" << endl return 0;
}
```

which thoroughly confuses the compiler!

This is a compile-time error or syntax error.

A syntax error is a part of a program that does not conform to the rules of the programming language.

Errors

Suppose you (accidentally of course) wrote:

```
cot << "Hello World!" << endl;
```

- This will cause a compile-time error and the compiler will complain that it has no clue what you mean by cot.
  The exact wording of the error message is dependent on the compiler, but it might be something like
  “Undefined symbol cot”.

Errors

C++

```
int main(){cout<<"Hello, World!"<<endl;return 0;}
```

– will work (but is practically impossible to read)

A good program is readable.

Errors

Some kinds of run-time errors are so severe that they generate an exception: a signal from the processor that aborts the program with an error message.

For example, if your program includes the statement

```
cout << 1 / 0;
```

your program may terminate with a “divide by zero” exception.

Errors

C++

– is case sensitive. Typing:

```
int Main()
```

will compile but will not link.

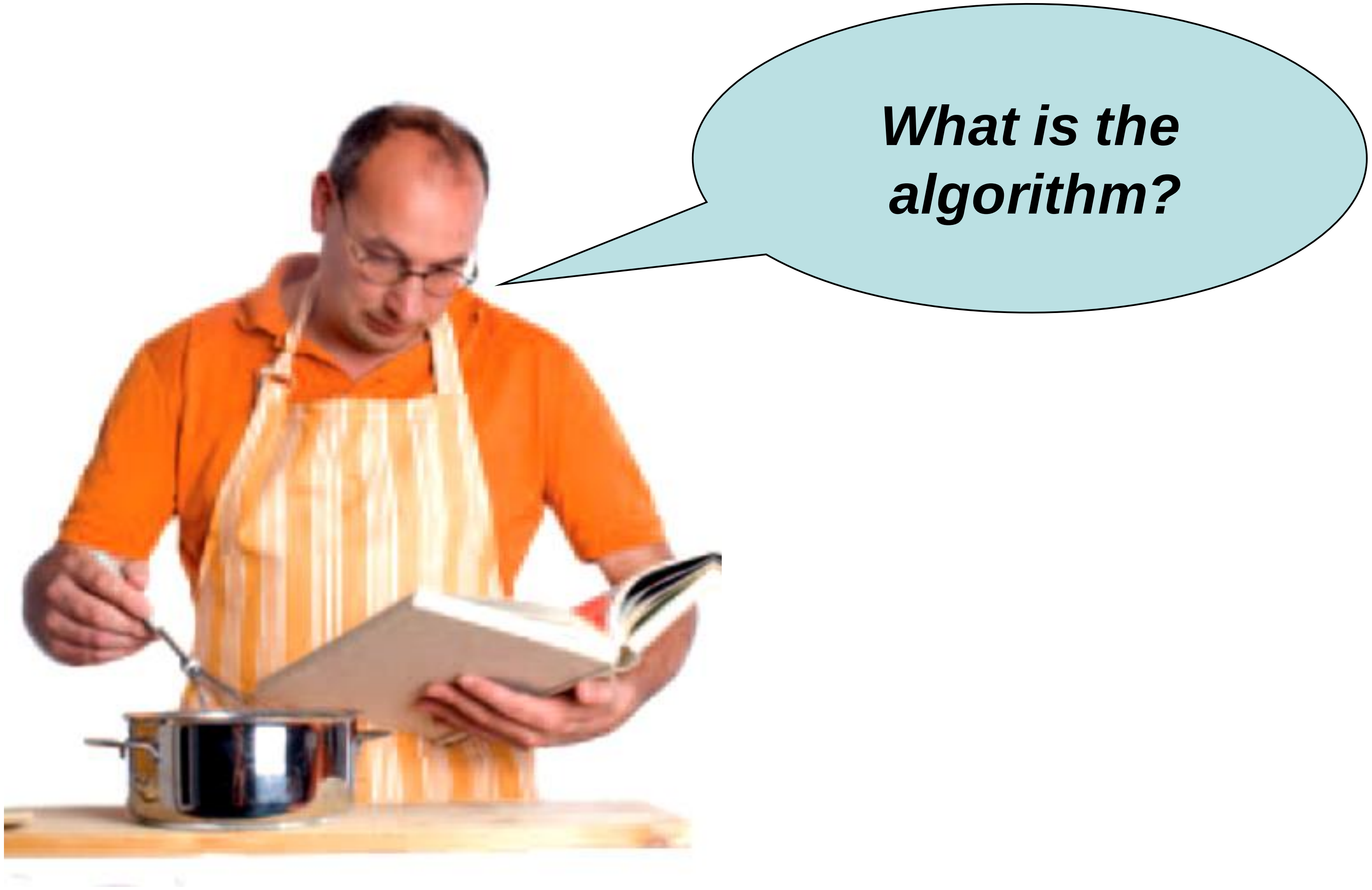
A link-time error occurs here when the linker cannot find the main function – because you did not define a function named main. (Main is fine as a name but it is not the same as main and there has to be one main somewhere.)

Describing an Algorithm with Pseudocode

Pseudocode

- An informal description
- Not in a language that a computer can understand, but easily translated into a high-level language (like C++).

Algorithms



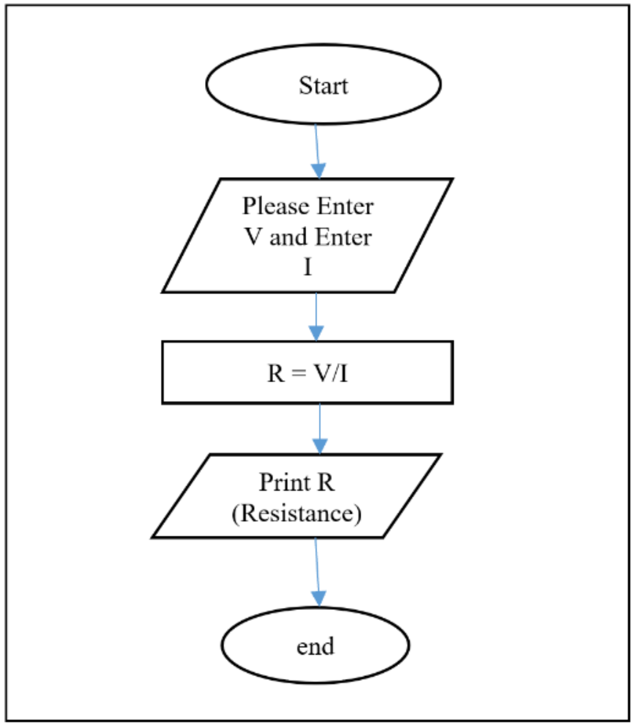
Describing an Algorithm with Pseudocode

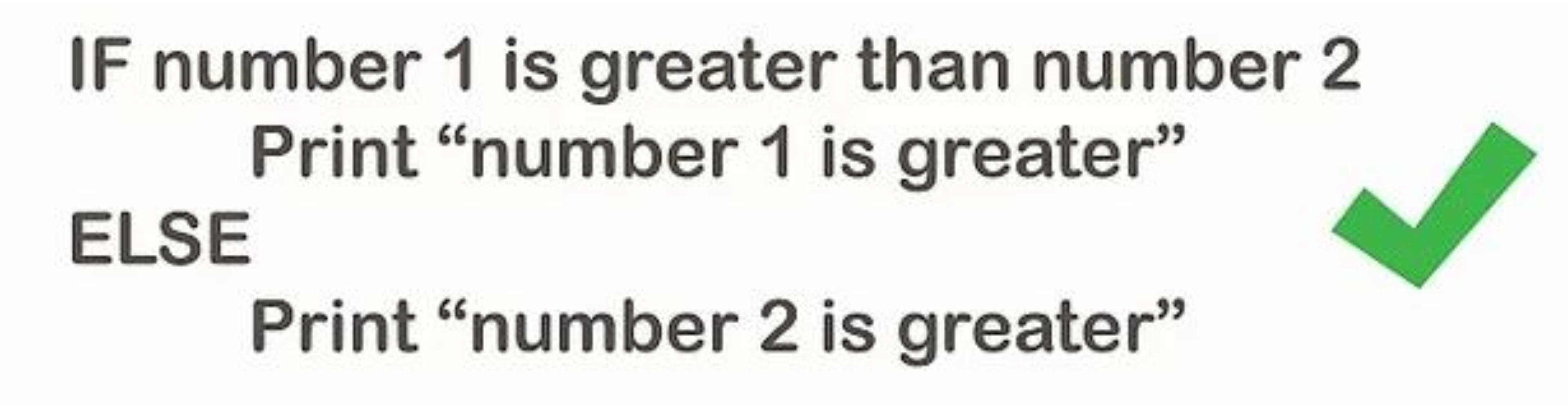
Example 1: Develop a pseudocode and a flowchart to ask the user to enter the voltage (V) and Current (I), then calculate and displays the resistance R = V/I.

Pseudocode:

```
Start
        PRINT "Please enter the Voltage"

        GET V

        PRINT "Please enter the Current"

        GET I

        R = V/I

        PRINT "Resistance = "

Stop
```

Flowchart:





```
IF number 1 is greater than number 2
        Print "number 1 is greater"
ELSE
        Print "number 2 is greater"
```

```
if “1”
     print response
          “Dandy!”

if “2”
     print response
          “Fantastic!”

if “3”
     print response
          “Lighten up, buttercup!”
```

✓

Exams

Lab exam: week 11/12/13 in lab time

GUI simple project: 3-15 weeks

Quiz: 6/7 week

Tutorial: Total 5, frequently