# Computer Programming

## Introduction

*Adapted from C++ for Everyone and Big C++ by Cay Horstmann*

# Objectives

- To learn about the architecture of computers
- To learn about machine languages and higher-level programming languages
- To become familiar with your compiler
- To compile and run your first C++ program
- To recognize syntax and logic errors
- To understand the notion of an algorithm
- To understand the activity of programming

# What Is a Computer? - Hardware

- Computers can handle repetitive chores without becoming bored or exhausted.
- A computer is programmable to handle different task - flexible:
  - Processing words
  - Playing Games
  - Browsing Internet
- Computer actions are composed of huge numbers of extremely primitive operations
- It executes these operations a great speed.

# What Is a Computer Program? - Software

- The programs the computer executes are called the ***software***.
- A computer program tells a computer, the sequence of steps that are needed to fulfill a task.

  A typical operation may be one of the following:
  - Add up these two numbers.
  - If this value is negative, continue the program at a certain instruction.

# What is Programming?

- The act of designing and implementing computer programs is called *programming.*
- A programmer designs and implements these programs.
- Most computer users are NOT computer programmers.
- Programming is not the only skill required to be a successful computer scientist.
- Early programming experiences are mundane and will not rival sophisticated software that you are familiar with.

# What is Programming?

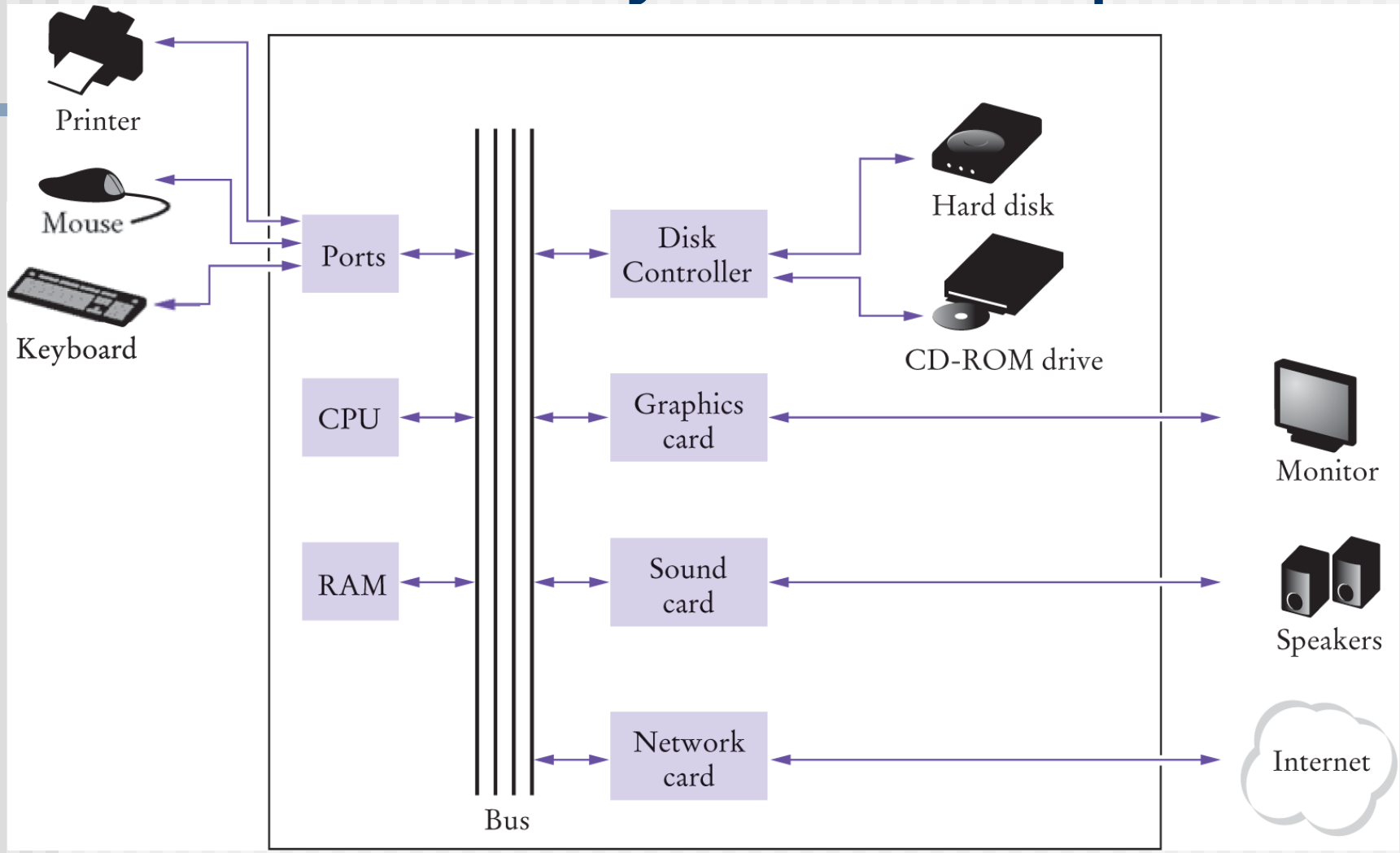
- For example, to tell the computer to write “Hello World!” on the screen, you will write a program.
- You will be creating *software:*

```
#include <iostream>

using namespace std;

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

# The Anatomy of a Computer



# Translating Human-Readable Programs to Machine Code (Machine Instructions)

- Extremely primitive:
- Encoded as numbers:
  - 161 40000 45 100 127 11280
    - Move the contents of memory location 40000 into CPU
    - If the value is > 100, continue with the instruction that is stored in memory location 111280
- Tedious and error prone to look up numeric codes and enter them manually.
- Each processor has its own set of machine instructions.

# Translating Human-Readable Programs to Machine Code (Assembler)

- Assigns short names to commands:
  - `mov 40000, %eax`
  - `sub 100, %eax`
  - `jg 11280`
- Makes reading easier for humans.
- Translated into machine instructions by another program.
- Still processor dependent

# Translating Human-Readable Programs to Machine Code (Higher-level Languages)

- Independent of the underlying hardware.
- Easiest for humans to read and write.

```
if (int_rate > 100)
   message_box("Interest rate error");
```

- Translated by compilers into machine instructions.

# 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)
- ANSI Standard C++ (1998)
- ANSI Standard C++ [revised] (2003)
- The present C++
  - a general-purpose language that is in widespread use for systems and embedded
  - the most commonly used language for developing system software such as databases and operating systems

# Your First Program

- At this point you should write your first C++ program: the classic program:  Hello World!
  - (everyone starts with this one)
- Its job is to write the words Hello World! on the screen.


```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7    cout << "Hello, World!" << endl;
8    return 0;
9 }
```

# Your First Program – Explained

- But first, some things about the code:

- C++
  - is *case sensitive.* Typing:

    ```
    int MAIN()
    ```

    will not work

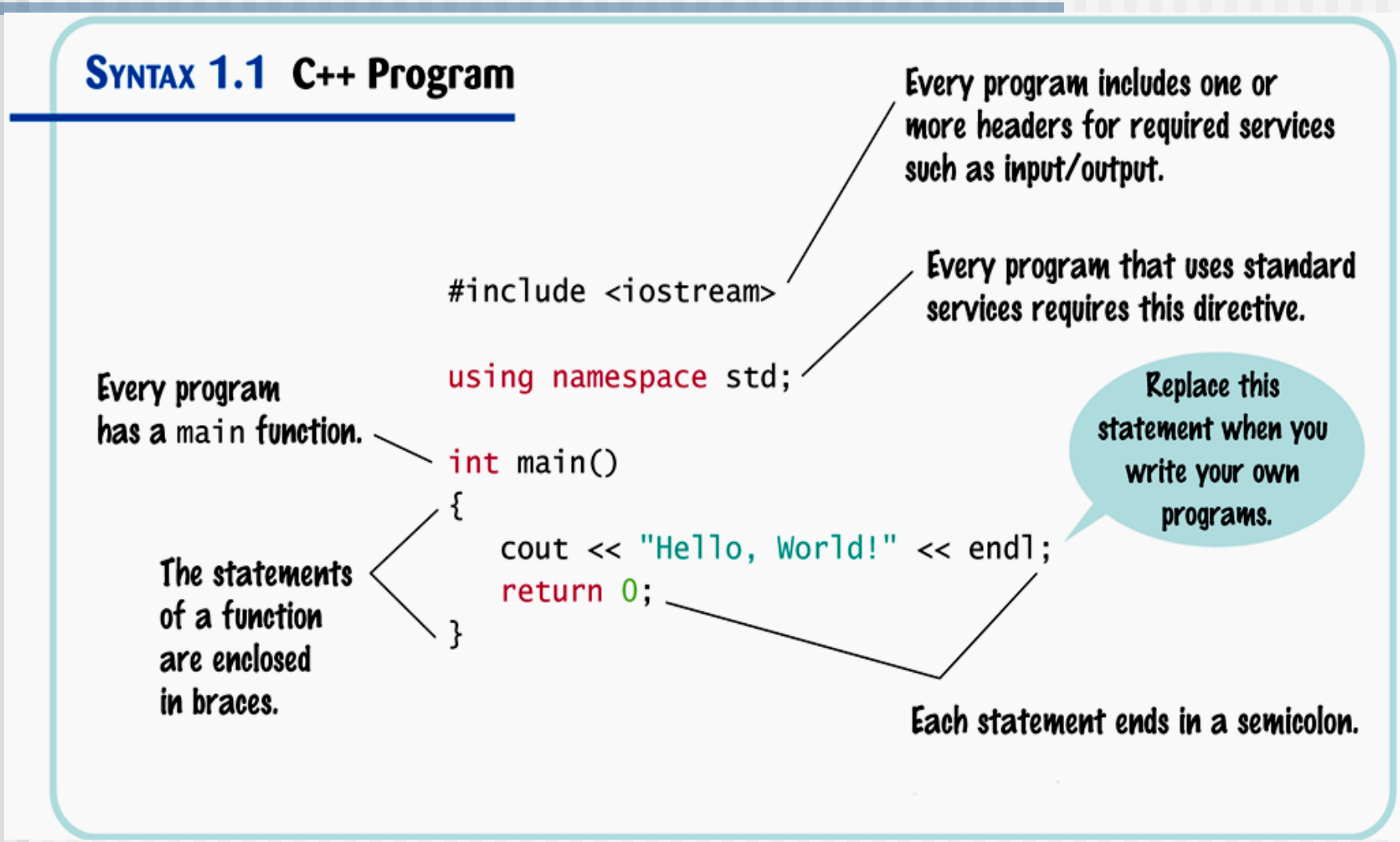
  - has *free-form layout*

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

    *will* work (but is practically impossible to read)

    A good program is readable.

# Your First Program – Explained



# The `main` Function

- A function is a collection of programming instructions that carry out a particular task.
- The construction

```
int main()
{
    ...
    return 0;
}
```

  defines a *function.*
  - The function's name is **`main`**.
  - The function "*returns*" the integer 0 (which indicates that the program finished successfully).

- Every C++ program must have one and only ONE **`main`** function.
- Most C++ programs contain other functions besides **`main`** (more about functions later).

# Your First Program – Explained

- `#include<iostream>` - read the file iostream that contains the definition for the stream input/output package.
- using `namespace std;` - all names in the program belong to the "standard namespace"

# Your First Program – Explained – The Output Statement

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

- Each C++ statement ends in a semicolon.
- **`"Hello World!"`** is called a *string*.
- The **`endl`** symbol denotes an *end of line* marker.
- OR:

```
cout << "Hello World!\n";
```

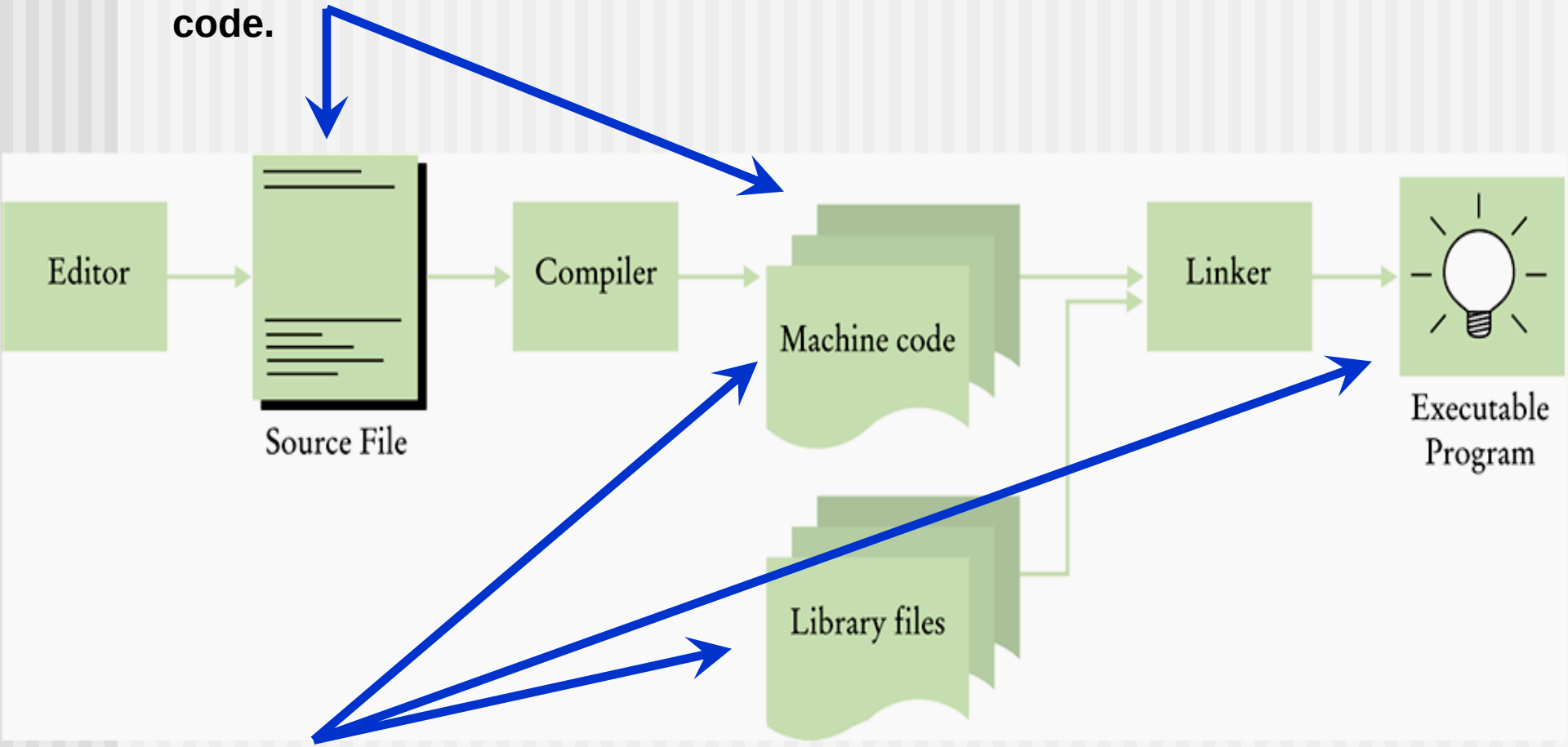
- The sequence of characters enclosed in quotations marks (`"Hello, World\n"`) is called a string.
  - Escape sequence  indicates a newline.

# Errors (Syntax and Logic Errors)

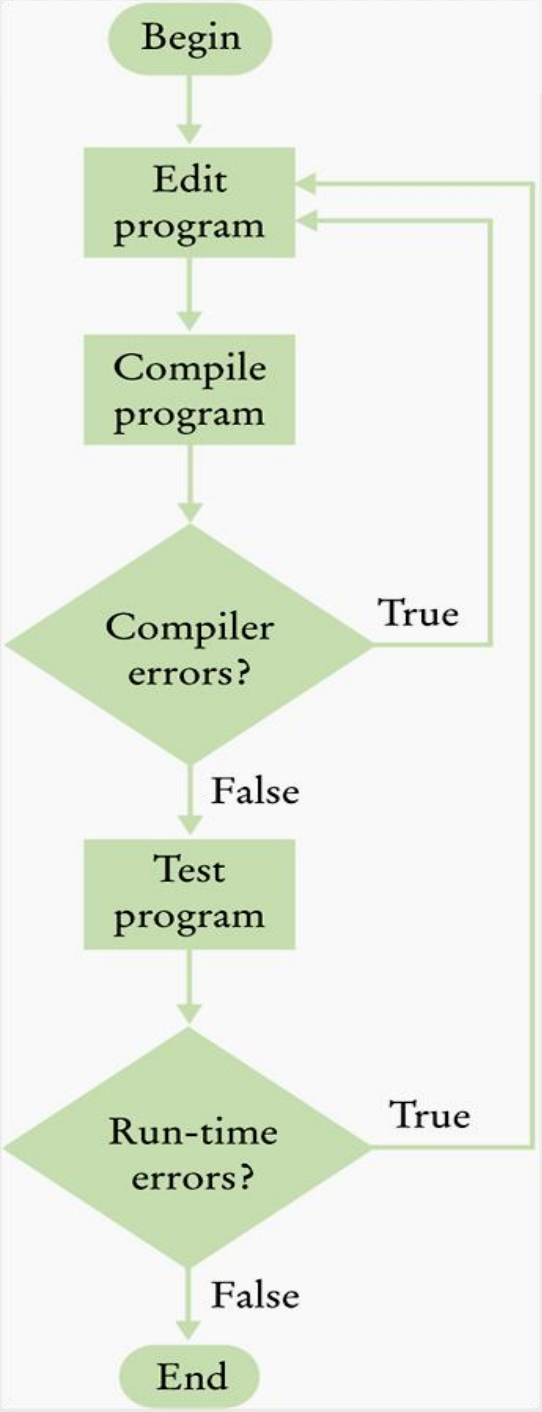
- Syntax error or compile-time error:
  - Something violates the language rules.
  - Compiler finds the error.
  - `cot << "Hello, World!";`
  - `cout << "Hello, World!\";`
- Logic error or run-time error:
  - Program doesn't do what it's supposed to do. The syntax is correct, but executes without performing the intended action.
  - Programmer must test and find the error.
  - `cout << "Hell, World\n";`
  - `cout << "Sum of num1 and num2:" << 5 - 6;`

# The Compilation Process

**The compiler translates C++ programs into machine code.**




**The linker combines machine code with library code into an executable program.**




# The Edit-Compile-Run Loop

This process reflects the way programmers work

**(shown as a *flowchart*)**

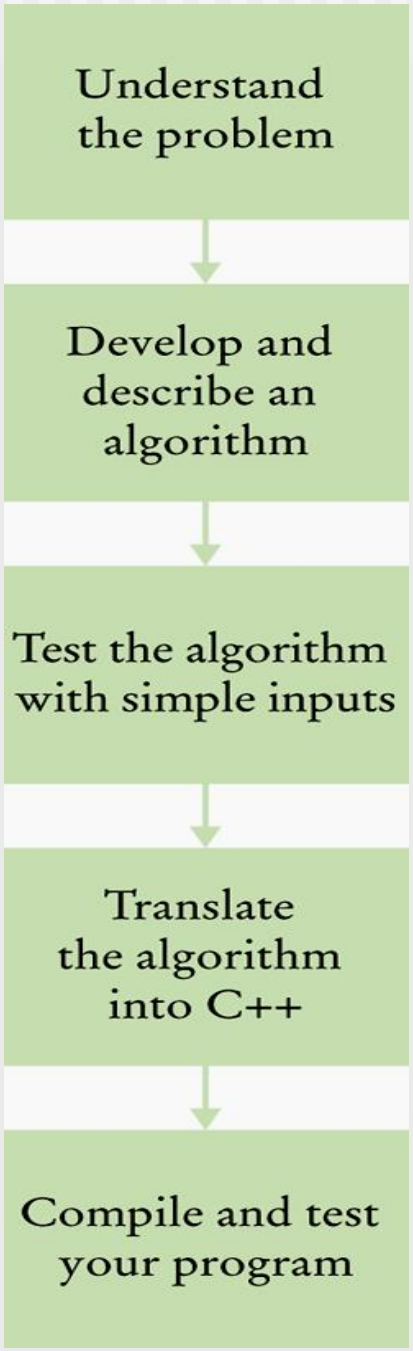
# Algorithms

A sequence of steps that is unambiguous, executable, and terminating is called an *algorithm*.

# Algorithms

Algorithmic cooker – using a recipe to cook a meal
An algorithm is a recipe for finding a solution

# The Software Development Process




For each problem the programmer goes through these steps

# Describing an Algorithm with Pseudocode

- An informal description
- Not in a language that a computer can understand, but easily translated into a high-level language (like C++).
- The method described in pseudocode must be
  - Unambiguous
    - There are precise instructions for what to do at each step
    - and where to go next.
  - Executable
    - Each step can be carried out in practice.
  - Terminating
    - It will eventually come to an end.

# Describing an Algorithm with Pseudocode

Consider this problem:

You put RM10,000 into a bank account that earns 5% interest per year. How many years does it take for the account balance to be double the original?

Determine the inputs and outputs.

In our sample problem, we have these inputs:

*Bank deposit :* RM10,000

*Interest rate : 5%*

We simply want to how many years does it take for the account balance to be double the original (10k x 2 = 20k).

That is the desired output.

# Describing an Algorithm with Pseudocode

**Step 1**: Set balance to 10,000 and year to 0

**Step 2**: Repeat steps 2a-2c while balance < 20,000

- Step 2a. Add one more than the preceding year's value
- Step 2c. Calculate the new balance value, the preceding balance value, multiplied by 1.05.

# Describing an Algorithm with Pseudocode

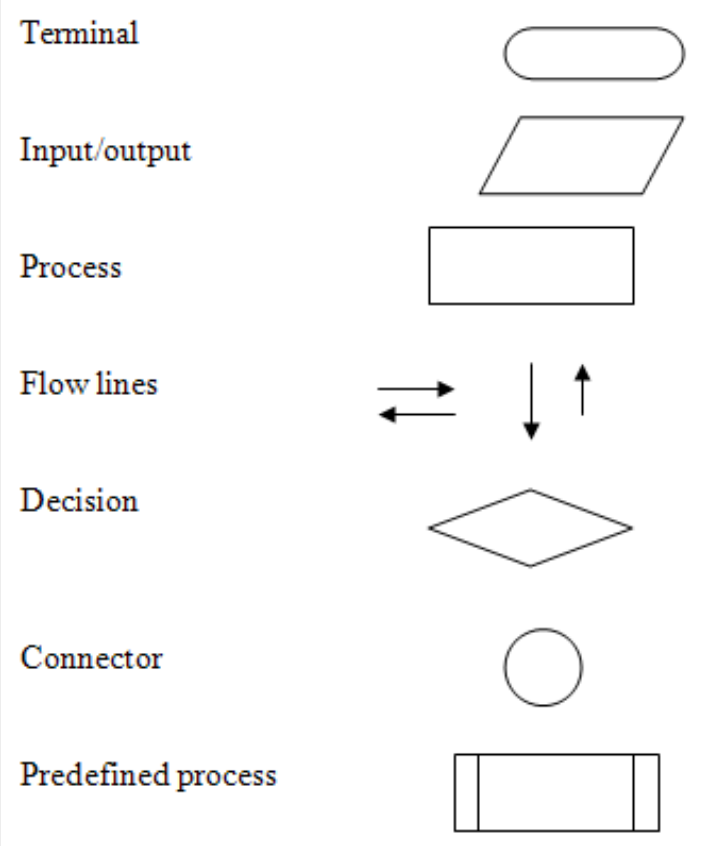
**Step 3:** Report the last value of the year in the repetition process. That value is the number of years required to double the investment

| After Year | Balance (RM) |
|---|---|
| 0 | 10,000 |
| 1 | 10,000.00 x 1.05 = 10,500.00 |
| 2 | 10,500.00 x 1.05 = 11,025.00 |
| 3 | 11,025.00 x 1.05 = 11,576.25 |
| 4 | 11,576.25 x 1.05 = 12,155.06 |

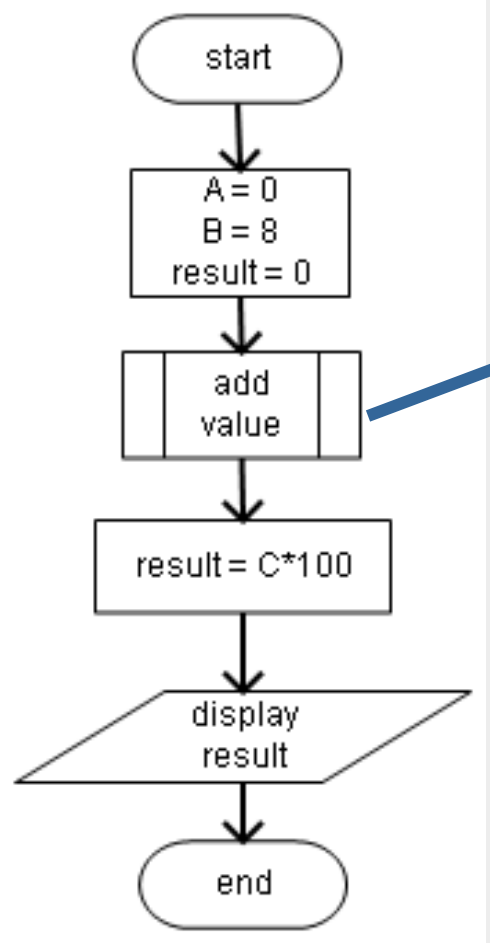
After year 14 , Balance = RM 19799.32

After year 15 , Balance = RM 20789.28

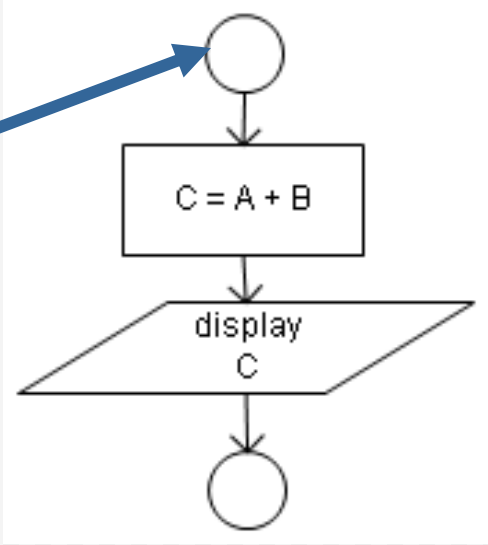
# Describing an Algorithm with Flowchart



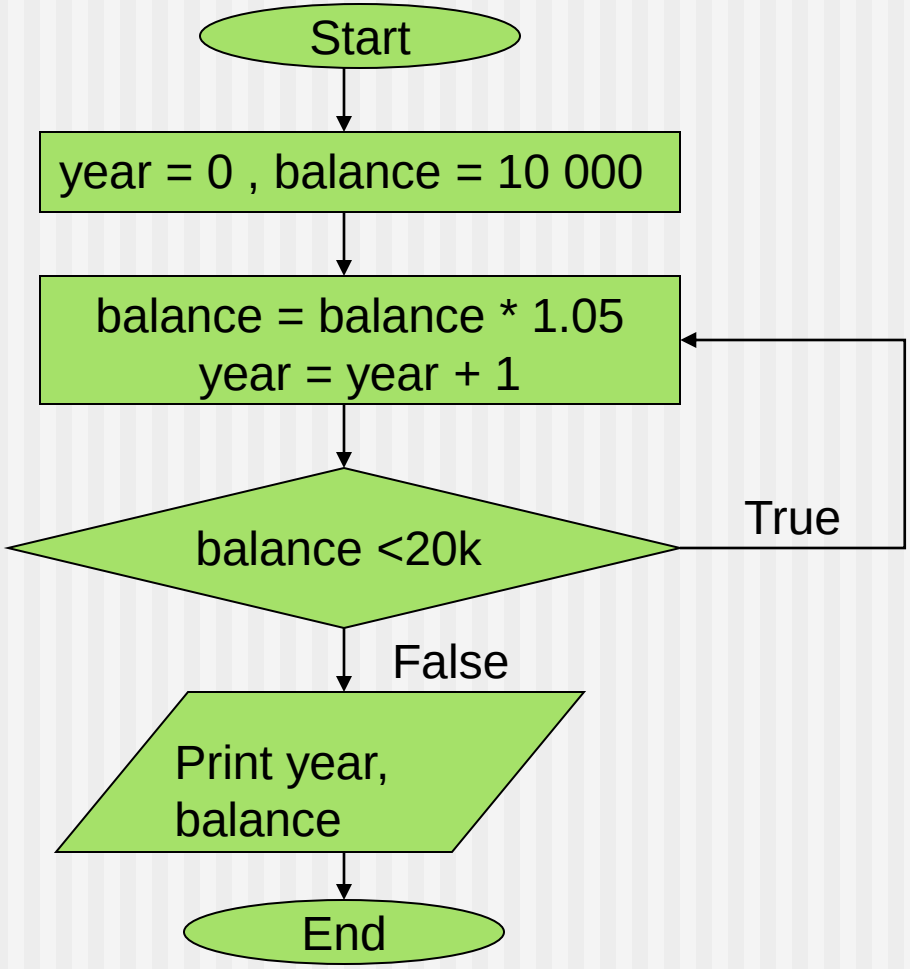
# Flowchart - Example




**Sub-process/function (Add Value)**



# Describing an Algorithm with Flowchart



# Summary

1. **A computer program is a sequence of instructions and decisions.**
2. **Programming is the act of designing and implementing computer programs.**
3. **Computer programs are stored as machine instructions in a code that depends on the processor type.**
4. **High-level programming languages are independent of the processor.**
5. **A syntax error is a part of a program that does not conform to the rules of the programming language while logic errors are errors in a program that executes without performing the intended action.**
6. **The compiler translates C++ programs into machine code.**

# Summary

9. **The linker combines machine code with library code into an executable program.**
10. **An algorithm for solving a problem is a sequence of steps that is unambiguous, executable, and terminating.**
11. **Algorithm can be described using pseudocode and flowchart.**

# References


- Big C++ 2nd Edition, Cay S. Horstmann, Timothy A. Budd, Wiley.
- C++ for Everyone, 1st Edition, Cay S. Horstmann,Wiley