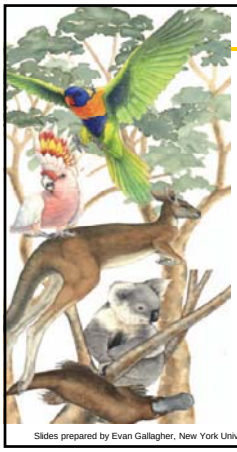


Chapter One: Introduction

Slides prepared by Evan Gallagher, New York University

Chapter Goals

- 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

MSOffice1

What Is a Computer?

Yes, ALL that is ONE computer!

MSOffice7

What Is a Computer?

Not quite a laptop.

MSOffice6

What Is a Computer?

The *E*lectronic *N*umerical *I*ntegrator *A*nd *C*omputer
(The ENIAC)

What Is a Computer? The Hardware

- The computer itself is a machine that stores data (numbers, words, pictures), interacts with devices (the monitor, the sound system, the printer), and executes programs.
- The physical computer and peripheral devices are collectively called the ***hardware***.

Slide 3

MSOffice1 In the chapter, the ENIAC comes after the discussion of hardware & software? OK as is?
, 10/20/2008

Slide 4

MSOffice7 In the chapter, the ENIAC comes after the discussion of hardware & software? OK as is?
, 10/20/2008

Slide 5

MSOffice6 In the chapter, the ENIAC comes after the discussion of hardware & software? OK as is?
, 10/20/2008

What Is a Computer Program? The Software

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

  A typical operation may be one of the following:
  - Put a red dot at this screen position.
  - 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.*
- In this book, you will learn how to program a computer — that is, how to direct the computer to execute tasks.

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:*

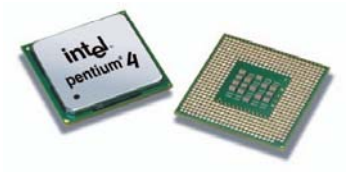
```
#include <iostream>

using namespace std;

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


The Anatomy of a Computer – 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 CPU
  - carries out arithmetic operations such as addition, subtraction, multiplication, and division
  - it fetches data from external memory or devices and stores data back.
- All data must travel through the CPU whenever it is moved from one location to another.

The Anatomy of a Computer – Storage Devices

- The computer stores data and programs in memory
  - Primary memory - memory chips
    - Random access memory (RAM) (read-write memory) MSOffice2
    - Read-only memory (ROM)
  - Secondary storage devices
    - disk drives
    - CDs

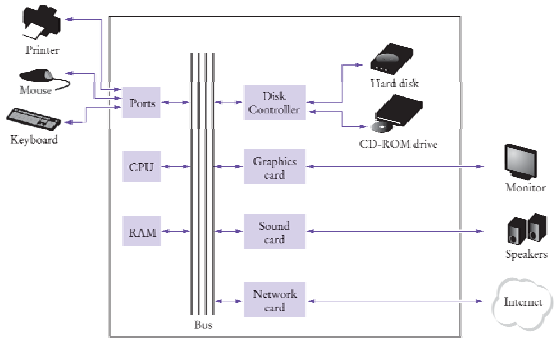
Slide 12

MSOffice2 This font was smaller than the rest on the page; OK as changed?
, 10/20/2008

The Anatomy of a Computer – Peripheral Devices

- The *user* is the human using a program that a programmer wrote.
- The computer transmits information (called *output*) to the user through a display screen, speakers, and printers.
- The user can enter information (called *input*) for the computer by using a keyboard or a pointing device such as a mouse.

The Anatomy of a Computer – Schematic Design



Machine Code and Programming Languages

- Computer programs are stored as machine instructions in a code that depends on the processor type.
- A typical sequence of machine instructions is
  - 1. Move the contents of memory location 40000 into the CPU.
  - 2. If that value is > 100, continue with the instruction that is stored in memory location 11280.

Machine Code and Programming Languages

- Machine instructions are encoded as numbers so that they can be stored in memory.
- On a Pentium processor, this sequence of instructions from the previous slide is encoded as the sequence of numbers

```
161 40000 45 100 127 11280
```

- On a processor from a different manufacturer, the encoding would be different.

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.

The Evolution of C++

- Ancient history (pre 1972)
- C (1972)
- ANSI Standard C (1989) MSOffice
- 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++ MSOffice4
  - 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

... the future: another Standard (2010?)

Slide 18

MSOffice3 If you want to indicate that between 1972 and 1989, while the ANSI Standard C was being finalized, Stroustrup was also working on Simula and C which resulted in C++ in 1985, you might be better off using side-by-side bullets.

At the very least, I would not include C++ (1985) on its own. The timeline aspect is lost.
, 10/20/2008

MSOffice4 Is there a year for the present C++?
, 10/20/2008

Getting Familiar with Your Computer – What You Must Know

- How to log in (if needed).
- Where your C++ compiler is and how to use it.
  You will probably want to work in an
  IDE (integrated development environment).
- Your file system: files, folders, and extensions.
  C++ program files typically have the extension: .cpp
  (or .cc or even .C).

Getting Familiar with Your Computer – What You Must Know

and

1. HOW TO BACK UP YOUR WORK.
2. HOW TO SAVE YOUR WORK
   (same thing as backing up your work).
3. HOW TO SAVE YOUR WORK BY BACKING IT UP
   (repeated so you will know this is very important).

Getting Familiar with Your Computer – What You Must Know

And, finally…

Getting Familiar with Your Computer – What You Must Know

Back up your work

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.

ch01/hello.cpp

```
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

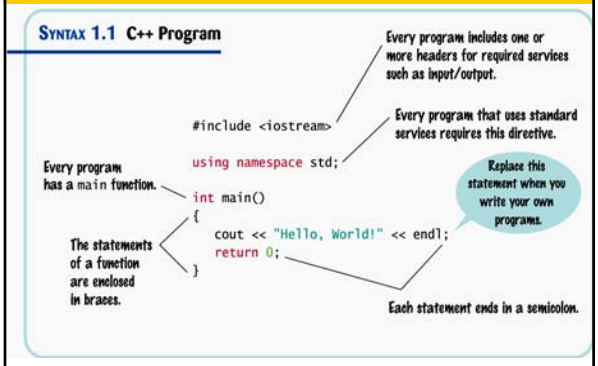
```
int main()
{
   ...
   return 0;
}
```

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

Your First Program – Explained – The `main` Function

- 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



Your First Program – Explained – The Output Statement

The output statement

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

is an *output statement*.

Your First Program – Explained – The Output Statement

The output statement

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

- To display values on the screen, you send them to an entity called **`cout`**.

Your First Program – Explained – The Output Statement

The output statement

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

- To display values on the screen, you send them to an entity called **`cout`**.
- The **`<<`** operator denotes the "send to" command.

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.

Your First Program – Explained – ERRORS!

Common error

Omitting a semicolon (or two)

Oh No!

```
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 – Syntax Errors

Without that semicolon you actually wrote:

```
7  cout << "Hello, World!" << endl return 0;
8  }
```

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.

Your First Program – Explained – How Many Errors?

- The compiler will not stop compiling, and will most likely list lots and lots of errors that are caused by the first one it encountered.
- You should fix only those error messages that make sense to you, starting with the first one, and then recompile (after SAVING, of course!).

Your First Program – Explained – Run-time Errors

Consider this:

```
cout << "Hollo, World!" << endl;
```

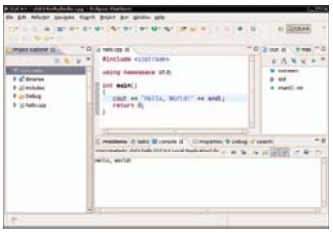
- *Logic errors* or *run-time errors* are errors in a program that compiles (the syntax is correct), but executes without performing the intended action.

not really an error

- The programmer is responsible for inspecting and testing the program to guard against logic errors.

The Compilation Process

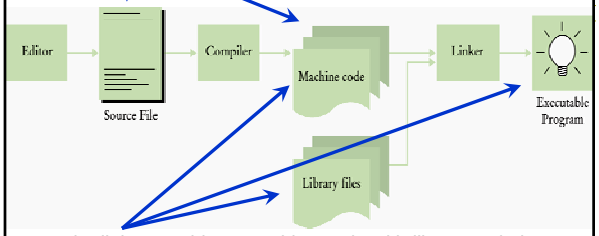
- A C++ IDE in action



- What goes on behind the scenes?

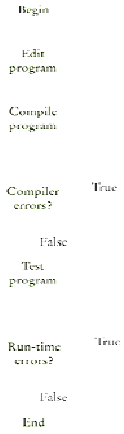
The Compilation Process

- 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



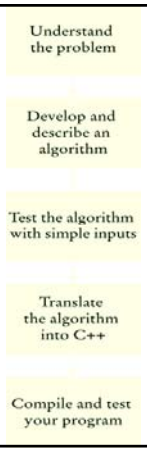
This process reflects the way programmers work

(shown as a *flowchart*)

Algorithms

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

The Software Development Process



For each problem the programmer goes through these steps

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++).

Describing an Algorithm with Pseudocode

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.

Slide 37

MSOffice5 Will the entire slide show (the Editor and Executable Program screens are ON the edge).
, 10/20/2008

Describing an Algorithm with Pseudocode

Consider this problem:

- You have the choice of buying two cars.
- One is more fuel efficient than the other, but also more expensive.
- You know the price and fuel efficiency (in miles per gallon, mpg) of both cars.
- You plan to keep the car for ten years.
- Assume a price of gas is $4 per gallon and usage of 15,000 miles per year.
- You will pay cash for the car and not worry about financing costs.

Which car is the better deal?

Describing an Algorithm with Pseudocode

Step 1 Determine the inputs and outputs.

In our sample problem, we have these inputs:

- *purchase price1* and *fuel efficiency1*
  the price and fuel efficiency (in mpg) of the first car
- *purchase price2* and *fuel efficiency2*
  the price and fuel efficiency of the second car

We simply want to know which car is the better buy.
That is the desired output.

Describing an Algorithm with Pseudocode

Step 2 Break down the problem into smaller tasks.

What will we do for ***each*** car?

1. The total cost for a car is
   purchase price + operating cost
2. We assume a constant usage and gas price for ten years, so the operating cost depends on the cost of driving the car for one year.
   The operating cost is
   10 x annual fuel cost
3. The annual fuel cost is
   price per gallon x annual fuel consumed
4. The annual fuel consumed is
   annual miles driven / fuel efficiency

Describing an Algorithm with Pseudocode

Step 3 Describe each subtask in pseudocode.

You will need to arrange the steps so that any intermediate values are computed before they are needed in other computations.

For each car, compute the total cost as follows:

```
annual fuel consumed = annual miles driven / fuel efficiency
annual fuel cost = price per gallon x annual fuel consumed
operating cost = 10 x annual fuel cost
total cost = purchase price + operating cost
```

If total cost1 < total cost2

```
Choose car1
```

Else

```
Choose car2
```


Describing an Algorithm with Pseudocode

Step 4 Test your pseudocode by working a problem.

Use these sample values:

- Car 1: $25,000, 50 miles/gallon
- Car 2: $20,000, 30 miles/gallon

FIRST CAR:
annual fuel consumed = 1500 / 50 = 300
annual fuel cost = 4 x 300 = 1200
operating cost = 10 x 1200 = 12000
total cost = 25000 + 12000 = 37000

SECOND CAR:
(let's assume you can do the math) total cost = 40000

If total cost1 < total cost2 …

The algorithm says: choose the FIRST CAR

Chapter Summary

1. Computers execute very basic operations in rapid succession.
2. A computer program is a sequence of instructions and decisions.
3. Programming is the act of designing and implementing computer programs.
4. The central processing unit (CPU) performs program control and data processing.
5. Storage devices include random-access memory (RAM) and secondary storage.
6. Computer programs are stored as machine instructions in a code that depends on the processor type.
7. High-level programming languages are independent of the processor.
8. C++ is a general-purpose language that is in widespread use for systems and embedded programming.
9. Set aside some time to become familiar with the computer system and the C++ compiler that you will use for your class work.

Chapter Summary

10. Develop a strategy for keeping backup copies of your work before disaster strikes.
11. Every C++ program contains `#include` directives and a function called `main`.
12. Use `cout` and the `<<` operator to display values on the screen.
13. Enclose text strings in quotation marks.
14. Use `+` to add two numbers and `*` to multiply two numbers
15. Send `endl` to `cout` to end a line of displayed output.
16. End each statement with a semicolon.
17. A syntax error is a part of a program that does not conform to the rules of the programming language.
18. Logic errors are errors in a program that executes without performing the intended action.
19. The programmer is responsible for inspecting and testing the program to guard against logic errors.

Chapter Summary

20. The compiler translates C++ programs into machine code.
21. The linker combines machine code with library code into an executable program.
22. Pseudocode is an informal description of a sequence of steps for solving a problem.
23. An algorithm for solving a problem is a sequence of steps that is unambiguous, executable, and terminating.