C++ Basics

Lecture 2

Outline

1. Some Background
2. Hello World!
3. Output
4. Variables
5. Sequential Execution
6. Input
7. Comments

Origins of C++

- Bjarne Stroustrup (AT&T Bell Labs) developed C++ in the early 1980's to be like the C language, but better
- C is a *high-level* language, but just barely
  - High-level: makes it easier to write complex code
  - Low-level: fast, can access memory/hardware directly
- C++ introduces object-oriented programming and evolves new features regularly (latest was C++11)
- Most C programs are valid C++ programs (but typically not the reverse)

C++

- Like most programming languages, C++ has a fixed **syntax**
  - Syntax: a “grammar” that distinguishes well-formed statements from those that are not
  - Special **keywords** and characters tell the computer what to do
- C++ forces you to think like a processor: step-by-step and logically

Hello World

http://www.roesler-ac.de/wolfram/hello.htm

```
#include <iostream>
using namespace std;

int main()
{
   cout << “hello world\n”;

   return 0;
}
```

Allows program to send output to screen and get input from keyboard

“main” is where the program starts

Print the words “hello world” to the screen followed by a new line

Tells the computer it has reached the end of the program

Visual Studio

Create an Empty Project

- New Project
  - Visual C++ in left pane
  - Win32 Project in right pane
  - Enter a project name (e.g. HelloWorld)

- Application Wizard
  - Console Application
  - Empty Project
  - Finish

Visual Studio

Add a Source File

- Right click “Source Files”
  - Add -> New Item
- C++ File (.cpp)
  - Enter a name
  - Add
- Ready to enter source code!

Visual Studio

Enter Source Code

```
#include <iostream>
using namespace std;

int main()
{
   cout << “hello world\n”;

   return 0;
}
```

Visual Studio

Build and Run

- First Build your project
  - Build menu -> Build Solution
  - Performs compilation and linking
  - Check the output window for errors!

- Now run!
  - Debug menu -> Start without Debugging
  - You will see a command window in which you can enter input, see output

Visual Studio

Tips and Tricks

- Save often!
- To enable line numbers…
  - Tools menu -> Options
  - Text Editor -> All Languages -> Line Numbers

Output

- `cout` is the C++ representation of the screen (command window)
- Syntax:
  - `cout << “WORDS”;`
- You can string together multiple outputs
  - `cout << “WORD1” << “WORD2 WORDS3”;`

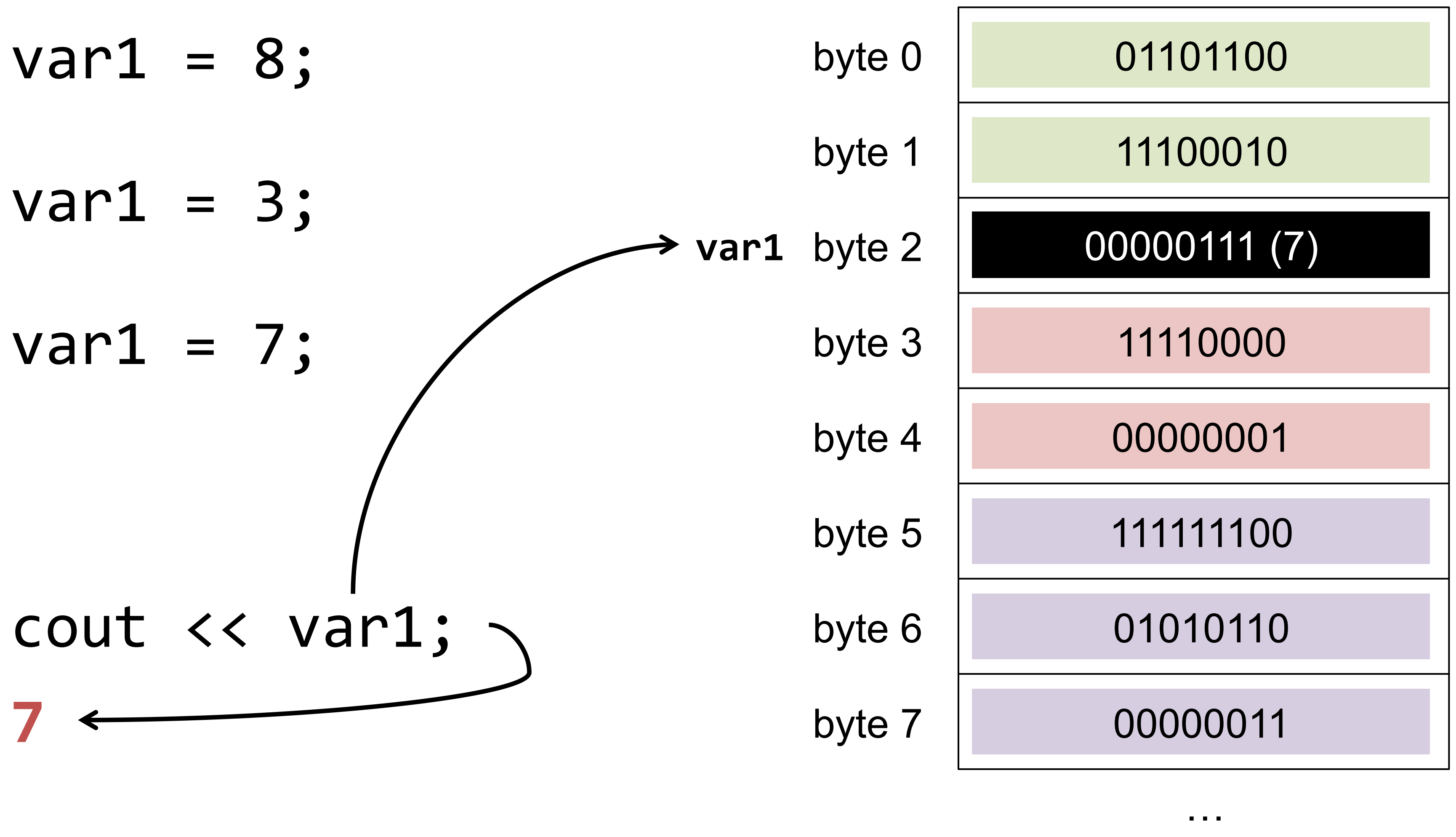
Special Characters

- Insert special characters by inserting a backslash in front of some characters
  - New line: `\n`
  - Horizontal tab: `\t`
  - Backslash: `\\`
- Instead of using `\n` to insert a new line, you can also use `endl`
  - `cout << “hello world” << endl;`

Variables

- The first fundamental concept in programming is that of a **variable**
- A variable is a named piece of memory
  - The **value** of the variable is what data is in its memory location
- A variable's value is initially garbage (i.e. whatever happens to be at the memory location)
  - You should always set an initial value - **initialization**
  - You can then get/change this value throughout the duration of the program

Variables Conceptually



```
var1 = 8;

var1 = 3;

var1 = 7;



cout << var1;
7
```

	byte 0	01101100
	byte 1	11100010
var1	byte 2	00000111 (7)
	byte 3	11110000
	byte 4	00000001
	byte 5	111111100
	byte 6	01010110
	byte 7	00000011

…

Variable Names

- In C++, variable names:
  - Must start with either a letter (uppercase or lowercase) or an underscore
  - Must contain only letters, digits, and underscores
  - Are case sensitive

- Valid names:
  - `count`, `x`, `user_input2`, `hit_points`
- Invalid ~~names~~:
  - `42`, `5x`, `$change`, `file.cpp`, `a-b`

Variable Declaration

- Every variable must be **declared**
  - Syntax: `TYPE NAME;`

- The `TYPE` tells C++ (1) how much memory is needed to store the variable and (2) how to interpret values in that space

- Common types:
  - `int`: integer (whole number), +/-
  - `double`: numbers with fractional component
  - `bool`: Boolean value (true or false)
  - `char`: single character

- Examples:
  - `int count;`
  - `double average;`
  - `char first_initial;`

Variable Initialization

- You can initialize a variable when you declare it or afterwards
  - During declaration: TYPE NAME = VALUE;
  - After declaration: NAME = VALUE;
- Examples:
  - `int count;`
  - `count = 0;`
  - `double average = 4.000;`
  - `char first_initial = ‘n’;`

Printing Variables

- `cout` is used to print the current value of a variable

- If a value is given in quotes, it is printed out literally; otherwise it is assumed to be a variable name
  - `cout << “name = “ << name << “\n”;`

Sequential Execution

- C++ programs start executing with the first line after the `{` after the `main()` line
- Each line is executed in order, one after the other, until you get to the `return` line
- We'll soon see how to affect this linear execution order, but even then programs execute one line at a time
- You have learn to think one statement at a time
  - This is one of the single most important skills you can have as a novice programmer

Input

- Output is sending information *to* the user; Input is getting information *from* the user
  - Input/Output is often abbreviated I/O

- In C++, the most common way to get input from the user/keyboard is with `cin`
  - Syntax: `cin >> VARIABLE;`

- Note that the double arrow (>>) is pointing towards the variable name to signify data being put into the variable

I/O Example

```
#include <iostream>
using namespace std;

int main()
{
    int age;

    cout << "Enter your age: ";
    cin >> age;

    cout << "You are ";
    cout << age;
    cout << " years old\n";

    return 0;
}
```

`cin` Notes

- When your program executes a `cin` statement, it pauses execution and waits for the user to input something on the keyboard
- All input is automatically separated by whitespaces (spaces, new lines, tabs)
- In other words, it won't continue executing your program after a `cin` statement until a non-whitespace value is entered
- Multiple input values can be separated on the same line by whitespaces

Comments in C++

In C++ source code, you can (and will!) include comments that explain in plain English what is happening in the code.

There are two types of comments in C++

1. Single-line comments start with `//`
   – Everything after the `//` until the end of the line is ignored by C++
2. Multi-line comments start with `/*` and end with `*/`
   – Everything between the `/*` and the `*/` is ignored by C++

Commenting Example

```
#include <iostream>
using namespace std;

/* This is a simple C++ program to demonstrate
   the usage of cout, cin, and comments. */

int main()
{
     // declare an integer variable named age
     int age;

     // read an age value from the user
     cout << "Enter your age: ";
     cin >> age;

     // print out the user's age
     cout << "You are ";
     cout << age;
     cout << " years old\n";

     return 0;
}
```

Wrap Up

- C++ programs are executed one statement at a time, starting after `main()` and going down from there
- `cout` is used to print values to the screen
  - Use double quotes to print literal words
  - Don't use quotes to print variables
- `cin` is used to read values from the user
- You should always include comments to explain your code for others who might read it (like the instructor…)