CS 112 Introduction to Programming

Lecture #4:

Computer Hardware/Software;
Variables, C# Data Types, and IO

http://flint.cs.yale.edu/cs112/

Outline

- ➢ *Review*
- ❑ Computer hardware/software
- ❑ Variables and C# data types
- ❑ Input and output

Review

- Different programming language levels
  - Machine code, assembly code, intermediate language, high level language
  - A compiler translates a program from a higher level language to a lower level language
- C# compiler compiles a C# program to MSIL
- Structure of a C# program
  - A program consists of one or more classes
  - A class consists of one or more methods
  - A method consists of one or more statements
  - White space and comments, identifiers, keywords, namespace
- Our first C# programs

3

C# Program Structure (Console)

- **Program specifications (optional)**

```
//=========================================================
//
// File: HelloWorld.cs        CS112 Demo
//
// Author: Zhong Shao       Email: zhong.shao@yale.edu
//
// Classes: HelloWorld
// --------------------
// This program prints the string "Hello World!"
//
//=========================================================
```

- **Library imports (optional)**

```
using System;
```

- **Class and namespace definitions**

```
class HelloWorld
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

4

C# Program Structure (Window)

- Program specifications (optional)

```
//=======================================================
//
// File: HelloWorld.cs          CS112  Demo
//
// Author: Zhong Shao         Email: zhong.shao@yale.edu
//
// Classes: HelloWorld
// --------------------
// This program shows a message box.
//
//=======================================================
```

- Library imports (optional)

```
using System;
using System.Windows.Forms;
```

- Class and namespace definitions

```
class HelloWorld
{
    static void Main(string[] args)
    {
        MessageBox.Show("Hello World!");
    }
}
```

5

Outline

- Review
- *Computer hardware/software*
- Variables and C# data types
- Input and output

6

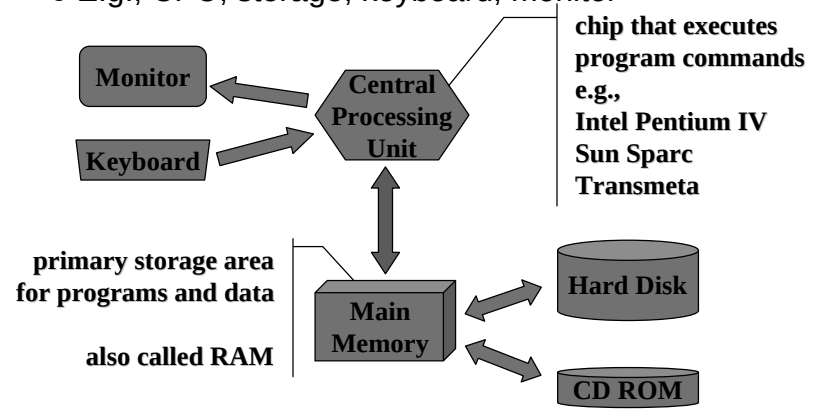
Outline

- Admin. and review
- *Computer hardware/software*
- Variables and C# data types

7

Computer Environment: Hardware

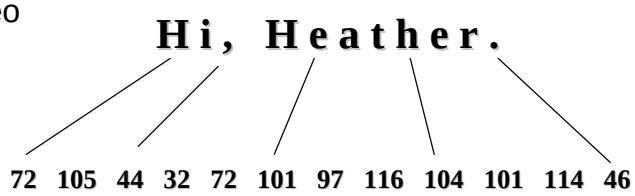
- Hardware
  - the physical, tangible parts of a computer
  - E.g., CPU, storage, keyboard, monitor



8

Storing Information

- Computers store all information digitally:
  - E.g. numbers, program instructions, text, audio, and video



- Information is stored in binary numbers
  - A single binary digit (0 or 1) is called a *bit*
  - A single bit can represent two possible states, like a light bulb that is either on (1) or off (0)
  - Combinations of bits are used to store values

9

Binary Bit Combinations

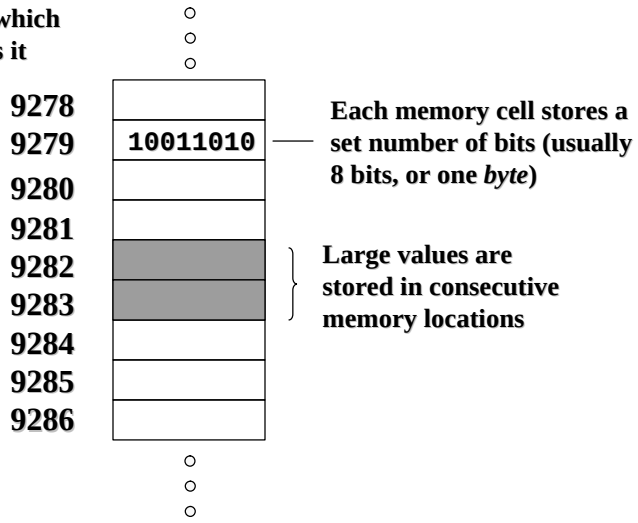
1 bit	2 bits	3 bits	4 bits	
0	00	000	0000	1000
1	01	001	0001	1001
	10	010	0010	1010
	11	011	0011	1011
		100	0100	1100
		101	0101	1101
		110	0110	1110
		111	0111	1111

Each additional bit doubles the number of possible combinations

10

Information is Stored in Memory

Each memory cell has a numeric *address*, which uniquely identifies it



11

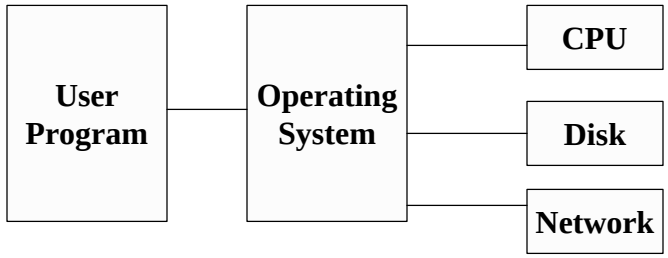
Computer Environment: Software

- Operating System
  - E.g., Linux, Mac OS X, Windows 2000, Windows XP
  - manages resources such as CPU, memory, and disk
  - controls all machine activities
- Application programs
  - generic term for any other kind of software
  - compiler, word processors, missile control systems, games

12

Operating System

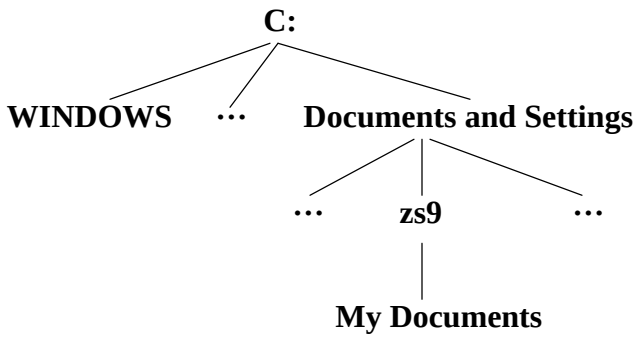
- What does an OS do?
  - hides low level details of bare machine
  - arbitrates competing resource demands
- Useful attributes
  - multi-user
  - multi-tasking



13

File System

- Hierarchical (directories and files)
- Filename: sequence of directory names ending with a file name



14

Some Useful Commands

- File system
  - mkdir as0 // creates a directory named as0
  - cd as0 // changes current directory to as0
  - cd .. // changes current directory one level up
  - dir // list the files of current directory
  - del <filename> // delete the file
  - Note 1: you can always do the above using Windows GUI
  - Note 2: you can even access the directory remotely by typing \\plucky.cs.yale.edu\zs9$\MyDocs in the Address field of your browser (replace zs9 with your net id)
- Editing
  - notepad <filename> // edit a file using notepad
    - Note: notepad insists on adding .txt after the file name. If you do not want the .txt suffix, choose “All Files” as “Save as type”
  - **scite** <filename> // edit file using SciTE, a code editor

15

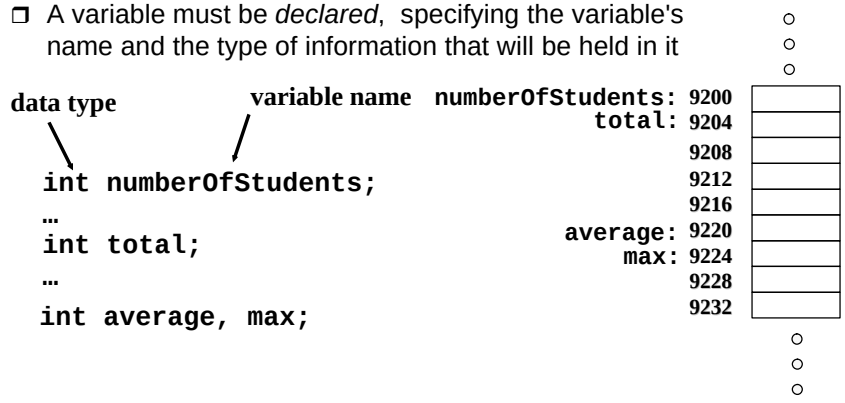
Outline

- Review
- Computer hardware/software
- *Variables and C# data types*
- Input and output

16

Variables

- A *variable* is a typed name for a location in memory
- A variable must be *declared*, specifying the variable's name and the type of information that will be held in it



Which ones are valid variable names?

myBigVar **VAR1** **_test** **@test**
99bottles **namespace** **It's-all-over**

17

Assignment

- An *assignment statement* changes the value of a variable
- The assignment operator is the = sign

```
int total;
…
total = 55;
```

- The value on the right is stored in the variable on the left
  - The value that was in `total` is overwritten
- You can only assign a value to a variable that is consistent with the variable's declared type (more later)
- You can declare and assign initial value to a variable at the same time, e.g.,

```
int total = 55;
```

18

Example

```
static void Main(string[] args)
{
   int total;

   total = 15;
   System.Console.Write("total = ");
   System.Console.WriteLine(total);

   total = 55 + 5;
   System.Console.Write("total = ");
   System.Console.WriteLine(total);
}
```

19

Constants

- A constant is similar to a variable except that it holds one value for its entire existence
- The compiler will issue an error if you try to change a constant
- In C#, we use the `constant` modifier to declare a constant

```
constant int numberOfStudents = 42;
```

- Why constants?
  - give names to otherwise unclear literal values
  - facilitate changes to the code
  - prevent inadvertent errors

20

C# Data Types

- There are 15 data types in C#
- Eight of them represent integers:
  - `byte`, `sbyte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`
- Two of them represent floating point numbers
  - `float`, `double`
- One of them represents decimals:
  - `decimal`
- One of them represents boolean values:
  - `bool`
- One of them represents characters:
  - `char`
- One of them represents strings:
  - `string`
- One of them represents objects:
  - `object`

21

Numeric Data Types

- The difference between the various numeric types is their size, and therefore the values they can store:

Type	Storage	Range
byte	**8 bits**	**0 - 255**
sbyte	**8 bits**	**-128 - 127**
short	**16 bits**	**-32,768 - 32767**
ushort	**16 bits**	**0 - 65537**
int	**32 bits**	**-2,147,483,648 – 2,147,483,647**
uint	**32 bits**	**0 – 4,294,967,295**
long	**64 bits**	**-9×10^{18} to 9×10^{18}**
ulong	**64 bits**	**$0 – 1.8 \times 10^{19}$**
decimal	**128 bits**	**$\pm 1.0 \times 10^{-28}$; $\pm 7.9 \times 10^{28}$ with 28-29 significant digits**
float	**32 bits**	**$\pm 1.5 \times 10^{-45}$; $\pm 3.4 \times 10^{38}$ with 7 significant digits**
double	**64 bits**	**$\pm 5.0 \times 10^{-324}$; $\pm 1.7 \times 10^{308}$ with 15-16 significant digits**

Question: you need a variable to represent world population. Which type do you use?

22

Examples of Numeric Variables

```
int x = 1;
short y = 10;
float pi = 3.14f;  // f denotes float
float f3 = 7E-02f; // 0.07
double d1 = 7E-100;
// use m to denote a decimal
decimal microsoftStockPrice = 28.38m;
```

Example: TestNumeric.cs

23

Boolean

- A `bool` value represents a true or false condition
- A boolean can also be used to represent any two states, such as a light bulb being on or off
- The reserved words `true` and `false` are the only valid values for a boolean type

```
bool doAgain = true;
```

24

Characters

- A `char` is a single character from the *a character set*
- A *character set* is an ordered list of characters; each character is given a unique number
- C# uses the Unicode character set, a superset of ASCII
  - Uses sixteen bits per character, allowing for 65,536 unique characters
  - It is an international character set, containing symbols and characters from many languages
  - Code chart can be found at:

  http://www.unicode.org/charts/
- Character literals are represented in a program by delimiting with single quotes, e.g.,

```
'a'  'X'  '7'  '$'  ','

char response = 'Y';
```

25

Common Escape Sequences

Escape sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. Any characters output after the carriage return overwrite the previous characters output on that line.
\'	Used to print a single quote
****	Backslash. Used to print a backslash character.
\"	Double quote. Used to print a double quote (**"**) character.

26

string

- A `string` represents a sequence of characters, e.g.,

```
string message = “Hello World”;
```

- Question: how to represent this string:

```
The double quotation mark is “
```

- Question: how to represent this string:

```
\\plucky.cs.yale.edu\zs9$\MyDocs
```

- Strings can be created with verbatim string literals by starting with @, e.g.,

```
string a2 = @“\\server\fileshare\Hello.cs”;
```

27

Outline

- Review
- Computer hardware/software
- Variables and C# data types
- *Input and output*

28

Data Input

- Console.ReadLine()
  - Used to get a value from the user input
  - Example

```
string myString = Console.ReadLine();
```

- Convert from string to the correct data type
  - Int32.Parse()
    - Used to convert a string argument to an integer
    - Allows math to be preformed once the string is converted
    - Example:

```
string myString = "1023";
int myInt = Int32.Parse( myString );
```

  - Double.Parse()
  - Single.Parse()

29

Output

- Console.WriteLine(variableName) will print the variable
- You can use the values of some variables at some positions of a string:

```
System.Console.WriteLine("{0} {1}.", iAmVar0, iAmVar1);
```

- You can control the output format by using the format specifiers:

```
float price = 2.5f;
System.Console.WriteLine("Price = {0:C}.", price);
```

```
Price = $2.50.
```

- For a complete list of format specifiers, see

http://msdn.microsoft.com/library/en-us/csref/html/vclrfFormattingNumericResultsTable.asp

Example: TaxAndTotal.cs

30