CS31 INTRODUCTION TO COMPUTER SCIENCE

FALL 2015

DISCUSSION SESSION 1

Noor Abani

INTRODUCING OURSELVES

- My name: Noor Abani
- Email: noorabani@cs.ucla.edu
- Office hours:
  - Time: Wed 9:30 – 10:30, 1:30 – 2:30 Thurs:11:30 – 12:30
  - Location: BH 2432
- My role as a TA:
  - Answer any questions you might have (Office hours and discussion session)
  - Give you this discussion session
  - Grade your homework (style not correctness)

WHAT IS COVERED IN A DISCUSSION SESSION

- Review and practice everything that was covered during the lectures in the previous week
- Answer any questions that you could not ask during the lecture
- A high level review of the topics of the week with more focus on coding examples that illustrate those topics
- Practice by hand – You writing some code

DEVELOPMENT ENVIRONMENT

- A collection of your chosen tools, software components, compilers, editors, etc. that you use to develop software
- **Compilation** is the process by which you translate your human-readable source code into machine-executable machine code
- The compilers for this course:
  - On Mac: The compiler comes with Apple Xcode
  - On Linux: g++
  - On Windows: Visual C++ with Microsoft Visual Studio

MICROSOFT VISUAL STUDIO - WINDOWS

1. Start up the Visual C++ IDE (Integrated Development Environment)
2. FILE / New / Project
3. In the New Project dialog, select General in the left panel and then Empty Project in the middle panel
    a) Enter the name of the project
    b) Change the location to a convenient one
4. Select PROJECT / Properties. In the Property Pages dialog, in the left panel, select Configuration Properties / Linker / System. In the right panel, select SubSystem, and in the drop down list to its right, Console (/SUBSYSTEM:CONSOLE). Click OK.

MICROSOFT VISUAL STUDIO – WINDOWS - 2

5. Select PROJECT / Add New Item. In the Add New Item dialog, select C++ File (.cpp) in the middle panel. Enter a source file name, such as "hello", in the Name text box below. Click Add.
6. Edit the hello.cpp file in the window that appears
7. From the Debug menu, select Start Without Debugging. OR Ctrl + F5.
   a) If your program has any compile-time or link errors, a dialog box will appear telling you there were build errors.
   b) If your program runs, its output will appear in a new console window that pops up.

XCODE - MAC

1. Start up the Xcode IDE
2. Dismiss the welcome window.
3. Select File / New Project. You will be asked to choose a template for your new project. In the left frame, select OS X / Application, and in the right frame, Command Line Tool. Click Next.
4. Pick a name for your project as the Product Name, such as "hello". For Type, select C++. For Company Identifier, anything will do. Click Next.
5. Select the folder in which you want your project folder to be created. Click Create.

XCODE – MAC - 2

6. In the left frame, click once on the name main.cpp, and rename the file to hello.cpp
7. Once you've clicked outside the text box to effect the name change, double-click on the name to open the file, which already has some sample code in it. Replace that code with your own.
8. From the main menu, select View / Debug Area / Activate Console.
9. In the upper left corner, click the Run right-facing triangle to build and run your program.
    a) If your program has any compilation or link errors, you'll be notified. Fix the errors. Go back to step 7.
    b) If your program built successfully, your output will appear in the console frame.

BINARY REPRESENTATION

- A binary number is a number that includes only ones and zeroes.
- The number could be of any length
- Examples of binary numbers:
  - 0, 1 , 01010, 101110011
- Another name for binary is base-2 (just like decimal number are base- 10)
- Every Binary number has a corresponding Decimal value (and vice versa)

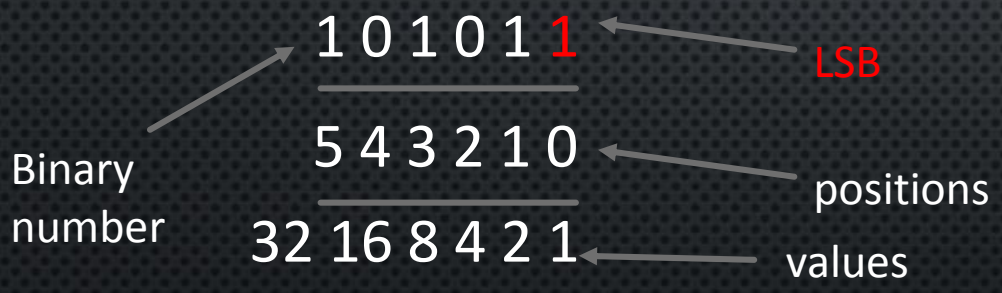
Examples:

Binary	Decimal
1	1
110	6
1011011	91
1111	15

HOW TO CONVERT FROM BINARY TO DECIMAL

- The numbers in a binary number have positions starting from position 0 for the right-most bit (also called the least significant bit - LS

- The value at position i is 2^i



- The decimal number is the sum of $digit_i \times value_i$ For all I
  - i.e. 1 0 1 0 1 1 in binary is equivalent to

$$1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 = 43$$

GENERAL BINARY TO DECIMAL CONVERSION FORMULA

- Binary number: $b_n\ b_{n-1} \ldots b_2\ b_1$
- Equivalent decimal number = $(b_n \times 2^n) + (b_{n-1} \times 2^{n-1}) + \ldots + (b_2 \times 2^1) + (b_1 \times 2^0)$

- Some examples:
  - $1111 = (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 8 + 4 + 2 + 1 = 15$
  - $11110 = (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 16 + 8 + 4 + 2 = 30$
  - $1000001 = (1 \times 2^6) + (0 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

    $= 64 + 1 = 65$

ASCII

- Why ASCII?
  - Computers can manage internally only 0s (zeros) and 1s (ones)
  - computer can express any numerical value as its binary translation
  - But it is not as evident for letters and other non-numeric characters
  - computers use *ASCII tables*, which are tables or lists that contain all the letters in the roman alphabet plus some additional characters
- ASCII is the abbreviation for **A**merican **S**tandard **C**ode for **I**nformation **E**xchange
- It is a character encoding scheme
- Encodes 128 characters into 7-bit binary integers
- For example, the ASCII code for the capital letter "A" is always represented by the order number 65, which is easily representable using 0s and 1s in binary: 65 expressed as a binary number is 1000001.

HOW TO READ THAT ASCII CODE TABLE?

ASCII Code Table

b_7 b_6 b_5					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FC	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

1. Read: $b_7b_6b_5b_4b_3b_2b_1$
2. Binary to decimal conversion

- Example:
  Character ‘c’
  Binary: 1100011
  Decimal: 99

ASCII CHARACTER SET

32		56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	"	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60	<	84	T	108	l
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	'	63	?	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	|
53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g		

GENERAL FORM OF A C++ PROGRAM

```
#include directives
int main()
{
    constant declarations
    variable declarations
    executable statements
}
```

FIRST C++ PROGRAM

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello World!" << endl;
}
```

EXPLAINNG THE CODE

`#include <iostream>:` In order to use the I/O library that we want to print out a message, we need to make sure we have the proper tools. The #include directive tells the preprocessor to treat the contents of a specified file as if those contents had appeared in the source program at the point where the directive appears." So, when we say #include <iostream>, it's as though the entire iostream file with all of its tools were written at the top of the page.

`Using namespace std:` A namespace allows you to group entities like functions, variables, and other references under a name

`int main(){ :` We are declaring a function with name main that is expected to return a value of type int.

`cout << "Hello World!" << endl; :` We are directing the string “Hello World!” to the standard output stream with a new line attached at the end. The new line, designated endl, moves the print cursor to the next line, like hitting "Enter" on your keyboard.

`} :` The closing curly-bracket for the main method that says, "Here's the end of this method block (the code that's in the method)."

NAMESPACE EXAMPLE

```
#include <iostream>
 using namespace std;
 namespace imANamespace {
    int x = 1;
 }
 namespace soAmI {
    int x = 2;
 }
 int main () {
    cout << imANamespace::x << endl;
    cout << soAmI::x << endl;
 }
```

COMPILATION/SYNTAX ERRORS

- Errors in which the programmer has violated a portion of the language syntax (the language structure). These will prevent the code from compiling.
- Some common syntax errors:
  - Missing semicolons at ends of statements
  - Missing brackets around blocks
  - Missing namespace or #include definitions
  - Misspelled variables or names

RUNTIME/LOGIC ERRORS

- Errors that might compile successfully, but encounter an error during runtime that either causes the program to break or produces unexpected results
- Common runtime errors:
  - Division by 0
  - Overflow (e.g. trying to hold a really big number in an int variable that exceeds its bounds)

BEST PRACTICES

- Comment your code: Comments are parts of the source code that are included mostly for human readability and information, but are ignored by the compiler, and subsequently, during runtime.
  - // for one sentence
  - /* for multiple
    sentences */
- Incremental development: Add bits of code at a time, then compile, and run. You will isolate bugs more quickly and understand what your code is doing.