Discussion Session
Week 4

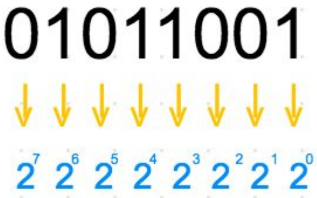
Exam #1 Review - Basics of Programming, Logic, and C++

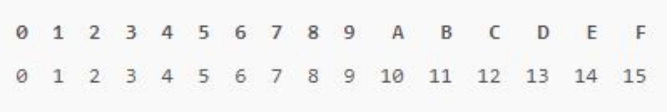
General Data Information

- Data is stored in bits and bytes
    - 1 bit is the smallest unit of data (0 or 1)
    - 4 bits = 1 nibble
    - 8 bits = 1 byte
    - 1024 bytes = 1 kilobyte
    - 1024 kilobytes = 1 megabyte

Representation of Numbers

- Decimal (base 10)
  - Numbers you’re familiar with
- Binary (base 2)
  - Powers of 2 and add
  - Can be x bits long, powers increase from right to left
- Hexadecimal (base 16)
  - Powers of 16 and add
  - 0-9, A-F
- Octal (base 8)
  - Powers of 8 and add



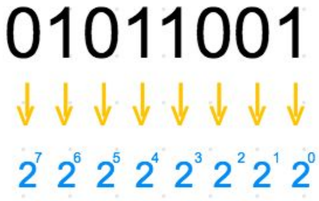


0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Binary to Decimal

Let’s convert 01011001 to a decimal

- Recall that binary is base 2 with a decreasing value from left to right



$$(0 * 2^7) + (1 * 2^6) + (0 * 2^5) + (1 * 2^4) + (1 * 2^3) + (0 * 2^2) + (0 * 2^1) + (1 * 2^0)$$

$$\downarrow$$

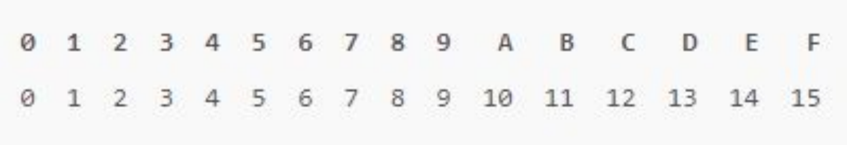
$$0 + 64 + 0 + 16 + 8 + 0 + 0 + 1$$

$$64 + 16 + 8 + 1 = 89$$

Hexadecimal Conversions

Converting 7A to a decimal

- Recall that hexadecimal numbers are in base 16
- 0-9, A-F



0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$(7 * 16^1) + (10 * 16^0)$

$112 + 10$

$= 122$

Base Conversions (From Decimal)

- To Binary
  - Divide the given number by 2, take the remainder, repeat
  - Write remainders backwards
- To Hex
  - Same process, but split it into parts
  - We can get binary for 3, 4, and A more easily than the combined “34A” string
- To Octal
  - Same process again, dividing by 8
- Simplest way from one base to another is to go through base 10

Decimal to binary

Let’s convert the decimal value 14

Value	Remainder
14 / 2 = 7	0
7 / 2 = 3	1
3 / 2 = 1	1
1 / 2 = 0	1

14 = 1110

Let’s convert the decimal value 21

Value	Remainder
21 / 2 = 10	1
10 / 2 = 5	0
5 / 2 = 2	1
2 / 2 = 1	0
1 / 2 = 0	1

21 = 10101

Hexadecimal to binary

Let’s convert the decimal value 3B

- Easiest way to do this is to convert each piece and push together

Value (B)	Remainder
11 / 2 = 5	1
5 / 2 = 2	1
2 / 2 = 1	0
1 / 2 = 0	1

Value (3)	Remainder
3 / 2 = 1	1
1 / 2 = 0	0

B = 1011

3 = 0011

3B = 00111011

Background on C++

- C++ is a compiled language
  - There are many compilers, g++ is a common one
  - Code is translated into machine language for you through the compiler
  - Any syntax errors will prevent successful compilation
- C++ is essentially C with libraries
  - Object oriented capabilities
  - Manual memory management
    - No garbage collection
- Everything in C++ can be boiled down to bits of information, and everything is treated as either true or false.

Basic C++ Programming

- All C++ programs require a main function in order to run
- Main (usually) returns an integer and (usually) takes in two parameters, argc and argv
- In general, the value returned from main indicates the error status of a program (0 means successful exit by standard, non 0 denotes unsuccessful)
- C++ is strictly typed
    - Types of variables and return types of functions must be stated explicitly unlike Python
- Lines of code are ended with semicolons
- Comments can be written like so:
    - //Single line comments
    - /*

      Multi Line comments

      */

C++ Data Types

- Numbers
    - int
    - Signed integer values, 32 bits
    - 1st bit is the “sign” bit
        - 2^n - 1 is the maximum *unsigned* value for n bits
        - 2^{n-1} - 1 is the maximum *signed* value for n bits
    - Modifiers
        - Long, short, unsigned
            - Change the max/min value, number of bits stored in an int
            - Unsigned long long is 64 bits
    - float/double
        - Decimal values (varying precision)

C++ Data Types (cont)

- char
    - Characters (‘h’, ‘e’, ‘l’, ‘l’, ‘o’, etc)
    - Really anything that can be found on the ASCII table
- bool
    - True/False
- void
    - Valueless
    - Used as return types for functions that do not return values, or for polymorphism

Variables and Functions

- Variables are a means of storing data
  - Syntax
    - dataType variableName = value;
- Functions are blocks of code that can be repeated by calling them
  - Syntax
    -
      ```
      functionReturnType functionName(paramOneType paramOne, paramTwoType paramTwo...)

      {

          //Function body

      }
      ```

Logic (Truth Tables)

- Logic is the basis of programming
    - ○ True can also be expressed as 1
    - ○ False can be expressed as 0

- Different types of logical operators
    - && (and)
    - | | (or)
    - ! (not)

P	Q	P && Q	P || Q
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

Extended Truth Table

P	Q	P && Q	P || Q	!P && Q	P && !Q	!P || Q	P || !Q	!!P && Q	P || !! Q
T	T	T	T	F	F	T	T	T	T
T	F	F	T	F	T	F	T	F	T
F	T	F	T	T	F	T	F	F	T
F	F	F	F	F	F	T	T	F	F

Let’s evaluate a boolean expression

```
a = 0, b = 1, c = 15, d = 5, e = 20

(!b && !!c) || ( d == e) || (!a && ((d+e) % 10 == 5));
```

Let’s evaluate a boolean expression

Note that this is not true && true, which is false

a = 0, b = 1, c = 15, e = 20

```
(!b && !!c) || (d == e) || (!a && ((d+e) % 10 == 5));
```

F || F = F

T && T = T

Break this down by itself before and-ing

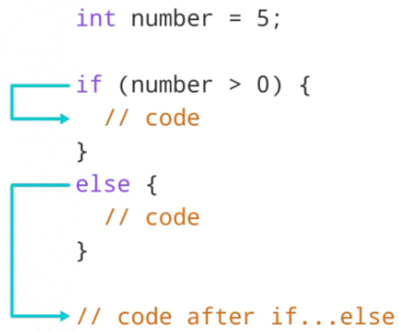
F || T = T

Other Operators

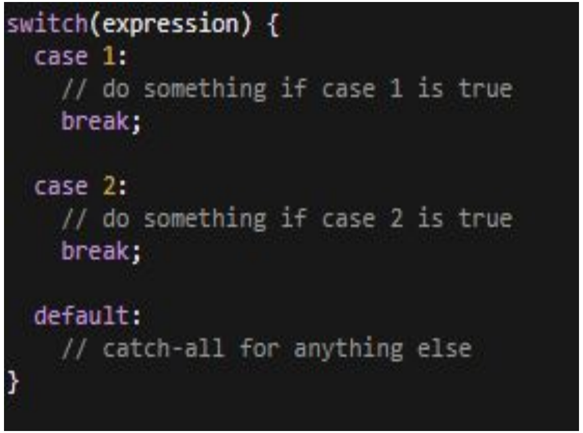
- +, -, *, /
  - Addition, subtraction, multiplication, division
- %
  - Modulo
- <, <=, >, >=
  - Less than, less than or equal to, greater than, greater than or equal to

Conditionals

- If else if else statements
  - The classic conditional branch
  - Traditional, ternary, one-liner
- Switch Statements
  - Used for many different cases of a condition
  - Must have “default” case and “breaks”



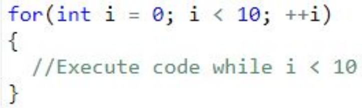
```
int number = 5;

if (number > 0) {
  // code
}
else {
  // code
}

// code after if...else
```



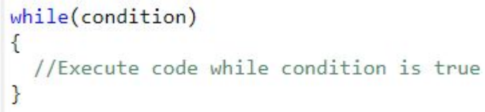
```
switch(expression) {
  case 1:
    // do something if case 1 is true
    break;

  case 2:
    // do something if case 2 is true
    break;

  default:
    // catch-all for anything else
}
```

Loops

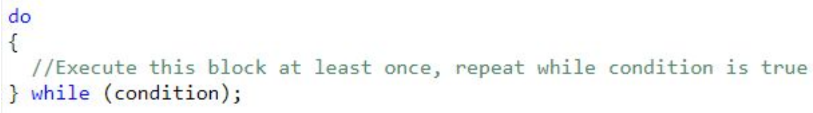
- For Loops
    - Pre-increment vs post-increment
- While loops
- Do While Loops
    - The body will always execute at least once



```
for(int i = 0; i < 10; ++i)
{
  //Execute code while i < 10
}
```



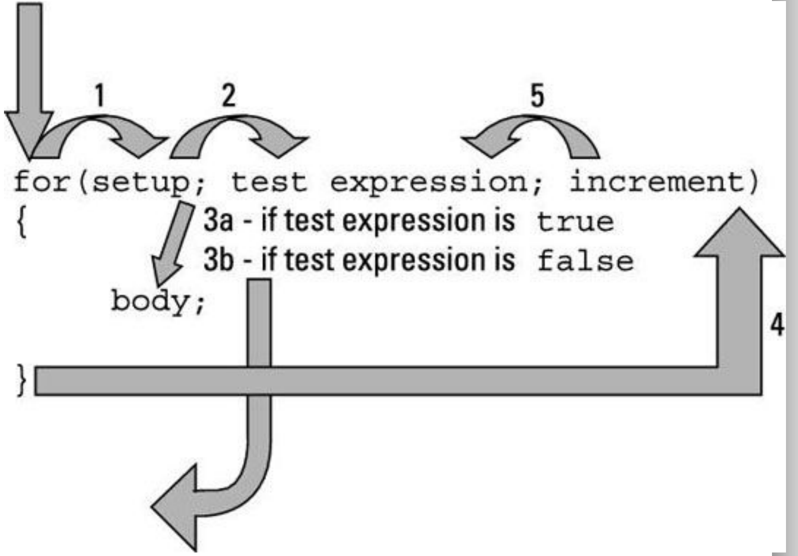
```
while(condition)
{
  //Execute code while condition is true
}
```



```
do
{
  //Execute this block at least once, repeat while condition is true
} while (condition);
```

For Loops Expansion

- More can be done with for loops
- “setup” , “test expression”, “increment” can have really any code there, but it is always executed in the given order


1
2
5
for(setup; test expression; increment)
{
3a - if test expression is true
3b - if test expression is false
body;
}
4

The "++" Part

- Libraries can be included into your files using #include
    - #include <libraryName>
    - #include "filename"

Tracing Code

- When tracing code, we go sequentially and change data as told to
- Recall your code tracing from 110
- http://pythontutor.com/visualize.html#mode=edit
  - PythonTutor is a great resource for practice in tracing code
  - Write some programs like we've done in the assignments and view the stack while it executes for some great exam prep