"In the brain, the greater the challenge, the greater the changes - up to a point. Recent studies have shown that learning a new skill is much more effective at triggering structural changes in the brain than simply continuing to practice a skill that one has already learned. On the other hand, pushing too hard for too long can lead to burnout and ineffective learning. The brain, like the body, changes most quickly in that sweet spot where it is pushed outside - but not too far outside - its comfort zone.

-- <u>Peak: Secrets from the New Science of Expertise</u> by Ericsson and Pool

Bits, Bytes, and Integers

CSCI 237: Computer Organization 3rd Lecture, Wednesday, September 11

Kelly Shaw

Administrative Details

- Lab #1 checkpoint (Parts 0 and 1) due today at 11pm
 - Any questions?
- Read CSAPP 2.1-2.2
- Practice Problems #1 posted
- Colloquium talk this Friday at 2:35pm in Wege
 - Bryan Perozzi (Google Research, NYC)

Last Class

printf

- Compilation of C programs
- C syntax and examples
 - Primitive types
 - Operators
 - Arrays
 - Command line arguments

Today's Plan

- C syntax and examples
 - Reading inputs from stdin
 - Control structures
- Terminology
- Representing information as bits
- Number representation
 - Conversion between decimal and binary
 - Conversion between hexadecimal, decimal, and binary
- Relationship between digits and range of unique values
- Bit-level manipulations

HelloWorld''' – Print sum of 2 numbers from command line

```
#include <stdio.h>
```

```
int main(int argc, char *argv[])
{
    int x = atoi(argv[1]);
    int y = atoi(argv[2]);
    printf("%d\n", (x+y));
```

```
return 0;
```

}

scanf

scanf(<formatted string>, ...)

- Reads from stdin
- 1st argument is formatted string using % (like printf)
- Remaining arguments are memory locations to store values into
 - scanf(``%d %f", addr_int_var, addr_float_var);
 - To get memory address of static variable, use reference operator &
- Returns number of values successfully read in
- Part of the stdio.h library
 - Insert #include <stdio.h> at top of C file
- If user enters Ctrl-d, scanf reads in nothing

K&R 158

HelloWorld'''' – Print sum of 2 numbers read from stdin

```
#include <stdio.h>
```

```
int main(int argc, char *argv[])
{
    int x = 0, y = 0;
```

```
scanf("%d %d", &x, &y);
printf("%d\n", (x+y));
```

```
return 0;
```

}

Relational Operators and Logical Operators

- **Boolean values:** true is 1, false is 0
- Relational Operators evaluate to 0 or 1

<, >, >=, <=, ==, !=</pre>

- Logical Operators evaluate to 0 or 1
 - AND : & &
 - OR : ||
 - NOT : !

Relational Operators and Logical Operators

- Boolean values: true is 1, false is 0
- Relational Operators evaluate to 0 or 1

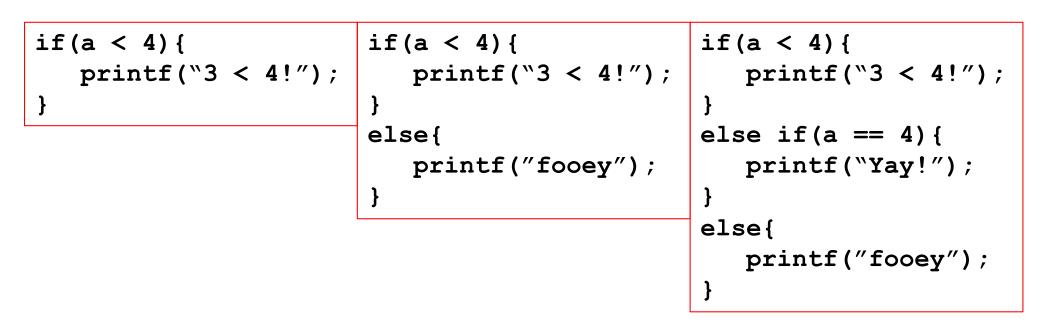
<, >, >=, <=, ==, !=</pre>

Logical Operators evaluate to 0 or 1

AND:&&
OR: ||
NOT: !
!x, !(!x), etc.

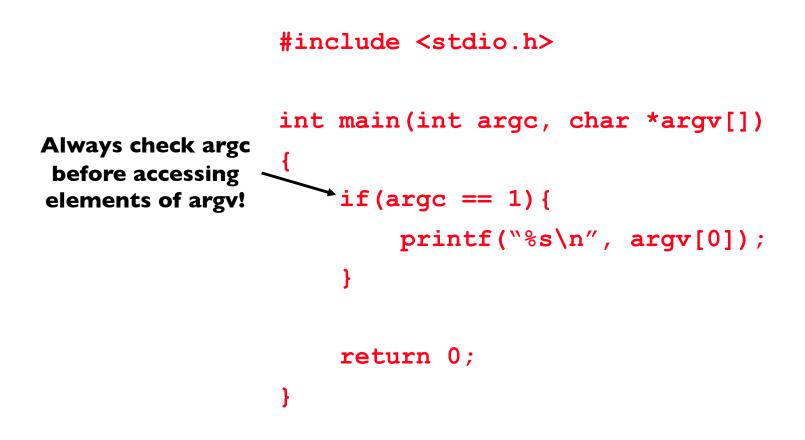
Note: Compound conditionals evaluate left to right and *short circuit*

Control: if/else if/else



- If condition evaluates to true, next statement (could be a set of statements inside brackets) executed
- Can have any number of "else if"

Command Line Argument: argv[0]



HelloWorld'''' – Print sum of 2 numbers read from stdin

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int x = 0, y = 0;
    if(scanf("%d %d", &x, &y) == 2){
      printf("%d\n", (x+y));
    }
    else{
       printf("Please enter 2 ints\n");
    }
    return 0;
}
```

HelloWorld'''' – Print sum of 2 numbers read from stdin

```
#include <stdio.h>
               int main(int argc, char *argv[])
                {
                    int x = 0, y = 0;
scanf returns #
                  ▶if(scanf("%d %d", &x, &y) == 2){
of items read in
                      printf("%d\n", (x+y));
                    }
                    else{
                       printf("Please enter 2 intsn'');
                    }
                    return 0;
```

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
  int num = 0;
  char str[10];
  for(int i = 0; i < 10; i++)</pre>
         str[i] = '\0';
  int val = scanf("%s %d", str, &num);
  printf("val %d str %s num %d\n", val, str, num);
  return 0;
}
```

Loops: for, while, do-while

for(initialization; condition; update){
 // body

while(condition){
 // body
 // often including update for check
}

do{

}

// body
// often including update for check
} while(condition);

HelloWorld'''' – Print sum of X numbers read from stdin

```
#include <stdio.h>
```

{

}

```
int main(int argc, char *argv[])
    int sum = 0, x = 0;
    // scanf returns -1 on Ctrl-D
    while(scanf("%d", &x) == 1){
      sum += x;
    }
```

```
printf("%d\n", sum);
```

```
return 0;
```

Bit Manipulations

■ Logical Operators (&&, ||, !) evaluate to 0 or 1

- -4 & -4 -4
- 1 & & 1 = 1
- 4 | | 0 = 1

Bitwise Operators (&, |, ~) evaluate on every bit
4 & 0 = 0 // AND
4 & 1 = 0 // AND
1 & 1 = 1 // AND
4 | 1 = 5 // OR
~0 = -1 // NEGATION

Some More C items

- Literals should have associated constant variable defined via #define
 - #define LENGTH 80
- The name of an array also corresponds to its address
 - char str[LENGTH];
 - int val;
 - scanf(``%d %s", &val, str);

Other C Notes

🛛 goto

- Used a lot in Linux kernel code, discouraged almost everywhere else
- "unconditional jump" to an instruction
- Makes code hard to read, but important concept
- switch
 - Similar to Java switch statement, except switch on integral types only
 - We will look at in more detail later, including jump tables (way cool!)
- struct
 - Define a new data type, composed of other data types
- typedef
 - Define a new name for an existing data type
- sizeof()
 - Indicates the number of bytes needed to store argument

Today's Plan

- C syntax and examples
 - Control structures
- Terminology
- Representing information as bits
- Number representation
 - Conversion between decimal and binary
 - Conversion between hexadecimal, decimal, and binary
- Relationship between digits and range of unique values
- Bit-level manipulations

Terminology

- Operating system: software layer between application and hardware that manages physical resources
 - Examples: macOS, Linux, Android, Windows 10
- Process: an actively running program or executable
- File: sequence of bytes
 - Every I/O device is modeled as a file
 - All input and output performed by reading and writing files