Week 6 - Monday

CS222

Last time

- What did we talk about last time?
- Pointers
- Passing values by reference
- Lab 5

Questions?

Project 3

Pass pointer example

- Let's write a function that takes a pointer to a **`char`**
- If the **`char`** is an upper case letter, we change it to lower case
- Otherwise, we do nothing
  - Remember that most **`char`** values are **not** letters!
- Prototype:

```
void makeLower(char* letter);
```

Pointers to Pointers

Pointers to pointers

- Just as we can declare a pointer that points at a particular data type, we can declare a pointer to a pointer
- Simply add another star

```
int value = 5;
int* pointer;
int** amazingPointer;
pointer = &value;
amazingPointer = &pointer;
```

Why would we want to do that?

- Well, a pointer to a pointer (**) lets you change the value of the pointer in a function
- Doing so can be useful for linked lists or other situations where you need to change a pointer
- Pointers to pointers are also used to keep track of dynamically allocated 2D arrays

What's the limit?

- Can you have a pointer to a pointer to a pointer to a pointer… ?

```
int********** madness;
```

- Absolutely!
- The C standard mandates a minimum of 12 modifiers to a declaration
- Most implementations of **gcc** allow for tens of thousands of stars
- There is no reason to do this, however

Quotes

Three Star Programmer

A rating system for C-programmers. The more indirect your pointers are (i.e. the more "" before your variables), the higher your reputation will be. No-star C-programmers are virtually non-existent, as virtually all non-trivial programs require use of pointers. Most are one-star programmers. In the old times (well, I'm young, so these look like old times to me at least), one would occasionally find a piece of code done by a three-star programmer and shiver with awe.*

Some people even claimed they'd seen three-star code with function pointers involved, on more than one level of indirection. Sounded as real as UFOs to me.

Just to be clear: Being called a ThreeStarProgrammer is usually ***not*** *a compliment.*

From C2.com

Command Line Arguments

Strings

- Before we get into command line arguments, remember the definition of a string
  - An array of **`char`** values
  - Terminated with the null character
- Since we usually don't know how much memory is allocated for a string (and since they are easier to manipulate than an array), a string is often referred to as a **`char*`**
- Remember, the only real difference between a **`char*`** and a **`char`** array is that you can't change where the **`char`** array is pointing

Command line arguments

- Did you ever wonder how you might write a program that takes command line arguments?
- Consider the following, which all have command line arguments:

```
ls -al
chmod a+x thing.exe
diff file1.txt file2.txt
gcc program.c -o output
```

Getting command line arguments

- Command line arguments do **not** come from **`stdin`**
- You can't read them with **`getchar()`** or other input functions
- They are passed directly into your program
- But how?!

You have to change `main()`

- To get the command line values, use the following definition for **`main()`**

```
int main(int argc, char** argv)
{

  return 0;
}
```

- Is that even allowed?
  - Yes.
- You can name the parameters whatever you want, but **`argc`** and **`argv`** are traditional
  - **`argc`** is the number of arguments (argument count)
  - **`argv`** are the actual arguments (argument values) as strings

Example

- The following code prints out all the command line arguments in order on separate lines

```
int main(int argc, char** argv)
{
  int i = 0;
  for( i = 0; i < argc; i++ )
     printf("%s\n", argv[i] );

  return 0;
}
```

- Since **argv** is a **char****, dereferencing once (using array brackets), gives a **char***, otherwise known as a string

Command line example

- Let's write a program that
  - Expects exactly one command line flag
  - If the flag is:
    - `-y` Print **`"yak"`**
    - `-c` Print **`"cormorant"`**
    - `-t` Print **`"Tasmanian devil"`**
  - For any other argument, we should print **`"Unknown animal"`**
  - If there is not exactly one command line argument (after the program name), print:

`"Usage: program [-y | -c | -t ]"`

Quiz

Upcoming

Next time…

- Review
- Lab 6

Reminders

- Keep reading K&R chapter 5
- Start working on Project 3
- **Lab 6 is Wednesday**
- **Exam 1 is Friday**