Week 6 - Friday

COMP 2400

Last time

- What did we talk about last time?
- Pointers
- Command line arguments

Questions?

Project 3

Exam 1 Post Mortem

Pointer problems

- Imagine the following declarations

```
int value = 10;
int *pointer = &value;
```

- What are the types of the following? (one of them is illegal syntax)
  1. `&value`
  2. `*pointer`
  3. `&pointer`
  4. `*value`
  5. `pointer[0]`
  6. `pointer + 4`
  7. `value + 4`
  8. `*(pointer + 4)`
  9. `*&value`

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);
```

Returning pointers

- Functions can return pointers
- If you get a pointer back, you can update the value that it points to
- Pointers can also be used to give you a different view into an array

```
char* moveForward(char* string) {
  return string + 1;
}
```

```
char* word = "pig feet";
while (*word) {
  printf ("%s\n", word);
  word = moveForward (word);
}
```

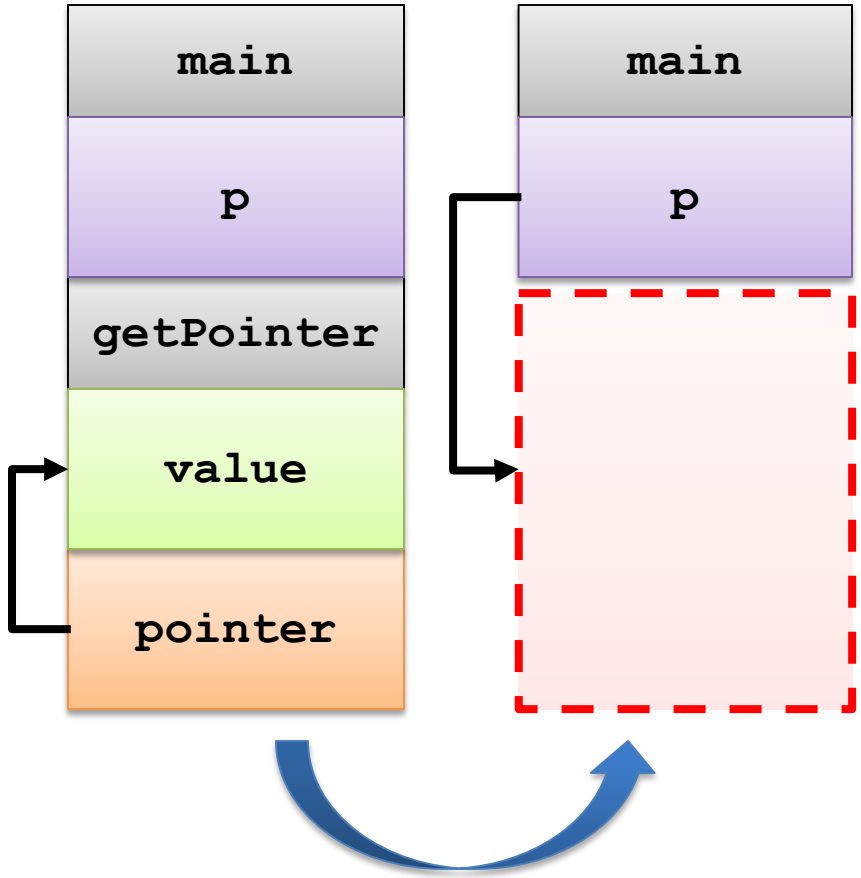
Pointer return problems

- Unfortunately, you can't return a pointer to a local variable
  - Well, you **can**, but it would be crazy
- It would be pointing to a value that is no longer on the stack
- Maybe it's still there…
- But the next time a function's called, it could be blown away

Stack visualization

```
int* getPointer()
{
  int value = 5;
  int* pointer = &value;
  return pointer;
}
```

```
int* p = getPointer();
```



After **return**

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 at least 12 modifiers are allowed for 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 Three-Star Programmer is usually ***not*** *a compliment.*

From C2.com

Ticket Out the Door

Upcoming

Next time…

- Input with **`scanf()`**
- Dynamic memory allocation

Reminders

- Keep reading K&R chapter 5
- Work on Project 3