

This is Week 5

Jason Hirschhorn

Fall, 2011

Agenda

- Announcements
- Review
 - Structs
 - Pointers
 - Stack vs. Heap
- Quiz 0
 - Details
 - Resources
 - OMG! How do I prepare?!
 - Questions
 - Practice Problems

Announcements

- Quiz 0!
 - Details + Resources in second half of section
 - (Not a factorial...just excited)
- Problem Set Feedback
 - pset3 – already sent out!
 - (Did you read through the comments?)
 - pset4 – back to regular, one-week turnaround

Review

Structs

- As seen in pset4 – Sudoku
- Allows you to create an object with different types inside it

```
struct {  
    char *level;  
    int board[9][9]  
    int number;  
    int top, let;  
    int y, x;  
} g;
```

Structs

Without typedef

```
struct student
{
    char *name;
    int class_year;
    char *house;
};
```

```
struct student djm;
```

With typedef

```
typedef struct
{
    char *name;
    int class_year;
    char *house;
} student;
```

```
student djm;
```

Structs

- Access fields with “.” or “->”
- “.” used for a normal instance of a struct

```
student djm;  
djm.house = “Mather”;
```
- “->” used for a pointer to a struct

```
student *djm = malloc(sizeof(student));  
djm->house = “Mather”
```

Pointers

- Declare a pointer

```
int x = 5;  
int *ptr;
```

- Get the address of a variable

```
ptr = &x;
```

- Go to the address and get/set the value (“dereference”)

```
printf(“%d”, *ptr); // prints 5
```

- Binky Pointer Fun

- <http://cslibrary.stanford.edu/104/>

Pointers

```
char x = 'a';
```

- What is x? What is &x? What is *x?

```
char *y = &x;
```

- What is y? What is &y? What is *y?

```
*y = 'b';
```

- What is x?

Stack vs. Heap

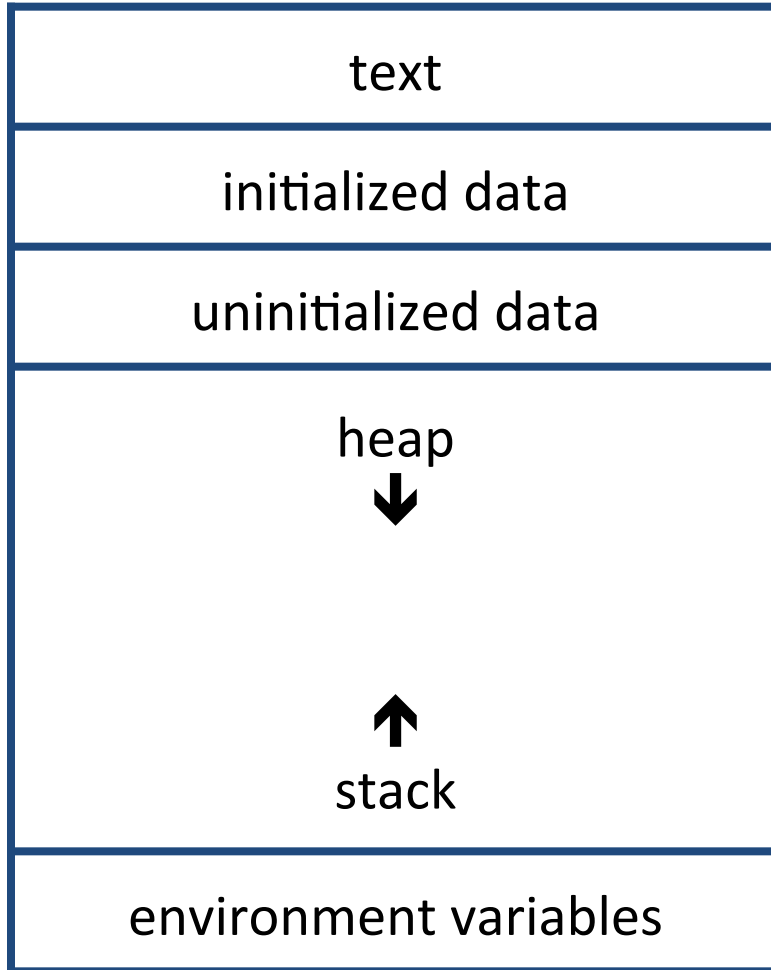
Stack

- Each function gets its own frame
 - Stores local variables
 - Becomes inaccessible when the function returns

Heap

- Dynamically allocated memory
 - Variables last as long as you want them to
- Reserve space with `malloc`
 - Returns a pointer
- Remember
 - Check that `malloc` doesn't return `NULL`
 - free everything you `malloc`

Stack vs. Heap



- Text – 0s and 1s that make up the program
- Initialized and uninitialized data – global variables
- Heap – dynamically allocated memory
- Stack – local variables
- Environment variables – special variables

Quiz 0

Details

- “About Quiz 0”
 - <https://www.cs50.net/quizzes/2011/fall/0/about0.pdf>

Details

- Covers weeks 0-5
- Wed, 10/12
- 75 min, 1-2:30pm
- Locations announced online by Tue, 10/11

Resources

- Course-Wide Review Session (Sun, 7pm, NW B103)
- Lecture slides, videos + transcripts, source code and scribe notes
 - <https://www.cs50.net/lectures/>
- Section slides and videos (shameless plug)
 - <https://www.cs50.net/sections/>
- Problem set specs and your code
 - <https://www.cs50.net/psets/>
- Past quizzes (N.B. course material changes, so you won't know – or be expected to know – everything on them)
 - <https://www.cs50.net/quizzes/>

OMG! How do I prepare?!

- Binary
 - Hexadecimal
 - ASCII
 - GCC
 - Compiling Errors
 - Segmentation Faults
 - Variables
 - Sizeof
 - Conditionals
 - Loops
 - Strings
 - Arrays
 - Libraries
 - Printf
 - Functions
 - Command Line Arguments
 - Scope
 - Memory
 - Stack
 - Heap
 - Recursion
 - Asymptotic Notation
 - Linear + Binary Search
 - Bubble + Selection
 - Sort
 - Merge Sort
 - Pointers
 - Pointer Arithmetic
 - Dynamic Memory Allocation
 - Structs
 - File I/O
 - GDB
 - Header files
 - Makefiles
- And more...

Breathe

- You are already prepared...but you should practice
- Make a **two-sided study guide**
 - Best way to review material
 - You can bring it to the quiz!
- Do the **practice quizzes**
- Look over **source code and your pset code**
 - Redo the problem set problems if time
- Write out code by hand
- And...

Questions?

Bueller?...Bueller?...Bueller?

Practice Problems

Floor

- floor of some real number x is the largest integer less than or equal to x
 - $\text{floor}(50.0) = \text{floor}(50.5) = \text{floor}(50.99) = 50$
- Implement floor
 - You may assume that x will be non-negative
 - You may not use any functions declared in `math.h`

```
int  
floor(float x)  
{
```

* Question 34 from 2010's Quiz 0

Floor

```
int  
floor(float x)  
{  
    return (int) x;  
}
```

Ceiling

- ceiling of some real number x is the smallest integer greater than or equal to x
 - $\text{ceiling}(49.01) = \text{ceiling}(50.0) = 50$
- Implement ceiling
 - You may assume that x will be non-negative
 - You may not use any functions declared in `math.h`
 - You may call `floor`

```
int  
ceiling(float x)  
{
```

* Question 35 from 2010's Quiz 0

Ceiling

```
int
ceiling(float x)
{
    if(x - (int) x > 0.0)
        return (int) (x + 1.0);
    else
        return (int) x;
}
```

Strlen

- According to its man page, `strlen` “calculates the length of the string `s`, not including the terminating `'\0'` character”
- Implement `strlen`
 - If `s` happens to be `NULL`, return 0

```
int  
strlen(char *s)  
{
```

* Question 30 from 2010's Quiz 0

Strlen

```
int
strlen(char *s)
{
    int n = 0;
    if(s == NULL)
        return 0;
    for(int i = 0; s[i] != '\0'; i++)
        n++;
    return n;
}
```


Ucwords

- ucwords returns a copy of s with the first letter of each word in the copy capitalized
 - Between each pair of words will be a single space
 - s will not begin or end with spaces
 - All characters in s will be alphabetical (or spaces)
- Implement ucwords
 - s may be NULL

char*

ucwords(const char *s)

{

* Question 27 from 2009's Quiz 0

Ucwords

```
char*
```

```
ucwords(const char *s)
```

```
{
```

```
    if(s == NULL)
```

```
        return NULL;
```

```
    int len = strlen(s);
```

```
    char *t = malloc((len + 1) * sizeof(char));
```

```
    if (t == NULL)
```

```
        return NULL;
```

Ucwords

```
bool first = true;
for(int i = 0; i <= len; i++) {
    if(first) {
        t[i] = toupper(s[i]);
        first = false;
    } else {
        t[i] = s[i];
        if(s[i] == ' ')
            first = true;
    }
}
return t;
}
```

That was Week 5

<http://www.youtube.com/watch?v=qybUFnY7Y8w>