CS 222: Structs and Strings

Chris Kauffman

Week 3-2

Today

Session 1

- HW 3 Overview/Questions
- Crash Course in `structs`
- Possibly discussion of Strings
- Exam problems review

Session 2

Exam 1

- Ch 1-6 (through arrays)

HW 3: Arrays and Structs

Problem 1: Trig Sig

Fill arrays with some values

Problem 2: DNA base counting

Iterate and count, return a `bpcount_t` struct

```
mytype_t mt = {.length=something, .width=something_else, ...};
return d;
```

Character comparisons are very useful; string functions not required but may be useful

```
if(x=='a'){...}
if(x=='B' || x=='b'){ ... }
```

Problem 3: Euclidean Distance

Two array args of `double`, compute the vector distance between them

```
double a[] = {2.3, 3.4, 4.5, 5.6}, b[] = {3.2, 4.3, 5.4, 6.5}
double result = distance(a,b,4); // should be 1.8 = sqrt(3.24)
```

struct As Function Args and Return Values

Both are readily done: colors.c

```
/* A simple struct for an RGB color */
typedef struct {
  double red;
  double green;
  double blue;
} color_t;
```

Returning an int vs struct

```
typedef struct { int a; double b;} mystruct;
```

Return an int

```
// return an int like this
int get_int(){
  int a = 22;
  return a;
}

// NOT like this
int get_int(){
  int a = 22;
  return int;
}

// and NOT like this
int get_int(){
  int a = 22;
  return int a;
}
```

Return a struct

```
// return a struct like this
mystruct get_struct(){
  mystruct s = {.a=1, b=2.3 };
  return s;
}

// NOT like this
mystruct get_struct(){
  mystruct s = {.a=1, .b=2.3 };
  return mystruct;
}

// and NOT like this
mystruct get_struct(){
  mystruct s = {.a=1, b=2.3 };
  return mystruct s;
}
```

Exercise: bluer(color1, color2)

- Write a function `bluer`
- Takes two `color_t` structs
- Determines which struct has a higher `blue` field
- Returns that struct

```
/* A simple struct for an RGB color */
typedef struct {
  double red, green, blue;
} color_t;

int main(){
  color_t c1 = {.red=0.5, .green=0.7, .blue=0.1};
  color_t c2 = {.red=0.6, .green=0.2, .blue=0.5};
  color_t r = bluer(c1,c2);  // should be same as c2 now
}
```

Reading Into Structs

Can read into parts of structs with `scanf()` style

```
int main(){
  printf("Enter the RGB values for the color:\n");
  color_t c;
  scanf("%lf %lf %lf", &c.red, &c.green, &c.blue);
  printf("Your color is R:%lf G:%lf B:%lf\n",
         c.red,c.green,c.blue);
  return 0;
}
```

In `read_color.c`

Strings

A string is just a character array. They occupy a funny spot in C.

- Standard array syntax works
  - `char c[6]; c[0] = 'H';`
- Have a special initialization syntax

  ```
  char c[6] = "Hello"; // Why 6??
  ```
- `printf` and `scanf` know about them
  - But not about other aggregate types
  - `printf("%s\n",c);`
- Null termination convention: strings end with the character

  ```
  '\0'
  ```

  called the *null character* (ASCII code 0)

A Warning

Arrays of `char` have funky exceptions to the initialization rules

```
/* Demonstration of some char array initializations,
   the infamous strings */
int main(){
  char ca1[16] =
    {'H','i',' ','m','o','m','\0'}; // Win
  char ca2[16] = "Hi mom";          // Win
  char ca3[16] = {"Hi mom"};        // Win
  char ca4[4] = "Hi mom";           // Fail
  char ca5[16];
  ca5 = "Hi mom";                   // Fail
  ca5[0] = 'H'; ca5[1] = 'i'; ca5[7] = ' \0';
  char ca6[16];
  ca6 = {"Hi mom"};                 // Fail

  char *cp  = "Hi mom";             // Win
  char ca[] = "Hi mom";             // Win
}
```

Character vs String Comparisons

Character comparison works just like numbers

```
char x='a', y='b', s[]="abc", t[]="abc";
int bool1 = x==y;         // T/F ?
int bool2 = x==s[0];      // T/F ?
int bool3 = y==s[0];      // T/F ?
int bool4 = x==s[1];      // T/F ?
int bool5 = y==s[1];      // T/F ?
int bool6 = s[0]==t[0];   // T/F ?
int bool6 = s[0]==t[1];   // T/F ?
```

String comparison involves many character comparisons (more in a moment)

String Library `<string.h>`

- Declare: `#include <string.h>`
- Define: Done for you, part of `libc`
  - Just like `printf`/`scanf` are always there

String Comparison

See `stringcompare.c`

- str1 = `str2 ?`   (= doesn't work)
- `int b = strcmp(str1,str2);`
- WARNING string comparison defies C convention
  - Why?

Practice Program

`wordguess.c`

- A mystery word called `answer`
- Repeated prompting to user for guess word
- Check if guess word is correct
- End game is guess is correct
- Otherwise, reveal progressive characters of `answer`

Write this program for me

Functions in `string.h`

See `stringlib.c`

- Length : `strlen()`
  - myint ← length(str)
  - `int l = strlen(str);`
- Copy : `strcpy()`
  - str1 ← str2
  - `strcpy(str1, str2);`
- Concatenation : `strcat()`
  - str1 ← str1 str2
  - `strcat(str1, str2);`

A few Character Functions

In `ctype.h`: can be useful for checking conditions

```
int isupper(char c);
int islower(char c);
int isspace(char c);
...

int toupper(int c);
int tolower(int c);
...
```

Not really needed for HW: just check specifically for characters with ==.

Relation of `*a` and `a[]`

What is `a` versus what is `c`?

```
int a[10];
char c[5];
```

- A memory address
- Access `a[4]` means `a + 4*sizeof(int)`
- Access `c[4]` means `c + 4*sizeof(char)`
- Next week explicitly deal with memory locations
  - `int *ap;` a pointer to memory which contains `int`s
  - `char *cp;` a pointer to memory which contains `char`s

Review Time

Questions ore topics to review before the exam