CSE 142
Programming I

Strings

Characters

- Recall that the char data type stores only one character
  - ➤ Good: 'y', '4', '#'
  - ➤ Bad: 'hello', 'Isaac', 'Mr. Wipple'

- But we really want to do this!!!

Strings

- A string is just a bunch of characters

- How might we think of storing those characters?

Strings in C

- Some languages have special data structures to deal with strings

- C doesn't—we just use arrays

- What are the benefits/drawbacks of this?

Arrays as Strings

- Strings are almost just arrays of characters

- There is a convention that we store strings so that there is a '\0' after the last character
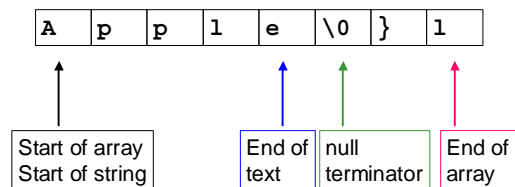 - ➤ Why?

Null Termination

- A possible string:

A	p	p	l	e	\0	}	l

Start of array
Start of string

End of text

null terminator

End of array

1

Strings as Arrays

- To fit a string with 6 characters, how large must the array be?

- What length strings can this array hold?
  - Where might the null be?

String Initialization

```
char pet[4] = {'p', 'i', 'g', '\0'};

char pet[4];
pet[0] = 'p';  pet[1] = 'i';
pet[2] = 'g';  pet[3] = '\0';

char pet[5] = "pig";

char pet[] = "pig";
```

This is No Good!

```
char pet[4];
pet = "lamb";
```

- Why?
  - Think arrays!

Things You Can't Do

- You can't use = to assign strings

- You can't use == to compare strings

- You can't have a string as a return type

- Why?
  - Think arrays!

String Output

- You can print a string using the %s flag:

```
char foo[] = "Ralph";
…
printf("%s is NOT a TA!\n", foo);
```

String Input

- You can also use %s to read a string
  - (but be careful!)

```
char name[100];
...
scanf("%s", name);
```

NO & — why?

Standard String Library

- So, there are all of these things we'd like to do with strings, but we can't

- In steps string.h and the string library

How to copy a string:

```
char a[10] = "lamprey";
char b[10];
int i = 0;

while (a[i] != '\0'){
  b[i] = a[i];
  i++;
}
```

How to Copy a String

```
#include <string.h>

char a[10] = "lamprey";
char b[10];

strcpy(b, a);
```

Caveat!

- ALWAYS make sure that the destination string is large enough to hold what you're copying in!

- What happens when we copy between
  ```
  char foo[] = "bigbigbig";
  char bar[] = "little";
  ```

String Length

```
/* find # chars *before* the \0 */
int strlen(char s[]){
  int size = 0;
  while (s[size] != '0')
    size++;

  return n;
}
```

Length Examples

```
char foo[] = "fruit";
char bar[] = "loop";
char baz[10] = "yum";

l1 = strlen(foo);
l2 = strlen(bar);
l3 = strlen(baz);
strcpy(foo, bar);
l4 = strlen(foo);
```

String Concatenation

- strcat appends one string to another

```
char foo[20] = "lamb";
char bar[] = "chop";

strcat(foo, bar);
```

Comparing Strings

- strcat compares two strings

```
char foo[] = "apple";
char bar[] = "orange";

strcmp(foo, bar);
```

- What does strcat return?

Recap: String Library

- C has many built in string functions
  - strcpy
  - strlen
  - strcat
  - srtcmp
  - etc.
- Make sure to #include <string.h> to use them!

Exercise:

- Write a function that reverses the letters in a string

```
Apple  ⟶  elppA
```