COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 11 - Class 1: Strings Part I

Administrivia

- SPA 3 now due on Weds.
- SPA 4 and LPA 2 will come out today, will be due on November, 8th, and November 15th respectively.
- Exam grades will be released on Weds.
- Mid-Semester Feedback Survey will be posted today.
  - Please complete to count as a quiz grade.

Today's Agenda

1. Introduction to Strings!

Strings

Character Arrays (Strings)

- Strings are an array of characters.
- Each element of the array stores a character.
- The last character of the string array is the null character `\0`.
- `\0` helps the compiler know when it reaches the end of a string for certain function operations.
- Everything after `\0` in the character array is known as garbage values.

Character Arrays Declaration

- Declaring the string has the exact same procedures as declaring an array of ints and doubles.
- All you have to place is the data type char.
- Example:

```
char word[5];
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	word[4] = ???
AA3	word[3] = ???
AA2	word[2] = ???
AA1	word[1] = ???
AA0	word[0] = ???

Character Arrays Declaration

- Strings have some unique syntaxes that allow for a proper declaration and initialization statement.
- C allows strings to be fully typed when be declared as long as the assignment operator is used.
- Double quotes are used to incorporate multiple characters.

```
char word[10] = "Pikachu";
```

Stack Space	
AA9	word[9] = ???
AA8	word[8] = ???
AA7	word[7] = ‘\0’
AA6	word[6] = ‘u’
AA5	word[5] = ‘h’
AA4	word[4] = ‘c’
AA3	word[3] = ‘a’
AA2	word[2] = ‘k’
AA1	word[1] = ‘i’
AA0	word[0] = ‘P’

Character Arrays Declaration

- Strings also have the initializer list like the ones we saw when working integers and doubles.
- They really don't need to be used, but you should know they exist.
- It's just extra time-consuming typing.
- You also MUST include the null character!

```
char word3[10] = {'P','i','k','a','c','h','u','\0'};
```

Char Array Declaration and Initialization

- you may think we can separate the declaration and initialization statements for strings, however we can't. It can only

- `word` is it expects an addr

There is another way we can separate the declaration and initialization statement. It involves the use of a string library function called `strcpy`. We see this very soon!

```
word = "Pikachu";
```

Char Array Declaration and Initialization

- you may think we can separate the declaration and initialization statements for strings, however we can't. It can only
- `word` is it expects an addr

There is another way we can separate the declaration and initialization statement. It involves the use of a string library function called `strcpy`. We see this very soon!

```
word = "Pikachu";
```

Reading Input into Arrays (Wrong!)

```c
int num[2];
int num2[2];
int mynum[2];

printf("Enter: ");
scanf("%d", num);
printf("Enter: ");
scanf("%d", num2);
printf("Enter: ");
scanf("%d", mynum);

for(int i = 0; i < 2; i++){

printf("num[%d] = %d\n", i, num[i]);
printf("num2[%d] = %d\n", i, num2[i]);
printf("mynum[%d] = %d\n", i, mynum[i]);

}
```

This will result in garbage being saved to the array after each first slot.

Reading Input into Arrays

```c
#include<stdio.h>

void readInArray(int arr[], int size);

int main(void){

int arr[2];

readInArray(arr, 2);

}

void readInArray(int arr[], int size) {
    int i;
    printf("Enter your list of numbers: ");
    for (i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
        printf("%d\n", arr[i]);
    }
}
```

To read in values properly, create a for loop, and iterate through each element in the array.

Input/Output with `printf` and `scanf`

- In past classes we have seen input and output instructions dealing with ints, doubles, floats, and chars.
- C allows programmers to write instructions for output and input with strings specifically.
- C has a special placeholder `%s` that only deals with strings.
- Think about it. Why does C need a placeholder for strings?

```
char word2[10] = "Pikachu";
printf("The pokemon is %s\n", word);
```

Collecting a String with `scanf`

- Collecting input for a string follows very similar procedures as collecting other data types.
- Two Differences:
  - Placeholder %s
  - No address operator (&)

```c
char pokemon[10];

scanf('%s', pokemon);

printf("the pokemon is %s\n", pokemon);
```

Reading Strings into Arrays

```c
char word[8];
printf("Enter: ");
scanf("%s", word);
printf("word = %s\n", word);
```

Reading Strings into Arrays

```c
char word[8];
printf("Enter: ");
scanf("%s", word);
printf("word = %s\n", word);
```

Why aren’t garbage values being displayed?

The null character!

Placeholder Values so Far!

Format Specifier	Data Type	Description	Syntax
%d	int	To print the integer value	printf("%d",<int_variable>);
%f	float	To print the floating number	printf("%f",<float_variable>);
%lf	double	To print the double precision floating number or long float	printf("%lf",<double_variable>);
%c	char	To print the character value	printf("%c",<char_variable>);
%p	pointer	Print a memory address	printf(“%p”,<pointer_variable>);
%s	string	To print a string value	printf(“%s",<string_variable>);

Demo

Acknowledgements

Slides adapted from Dr. Andrew Steinberg's COP 3223H course