UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #25 – Strings

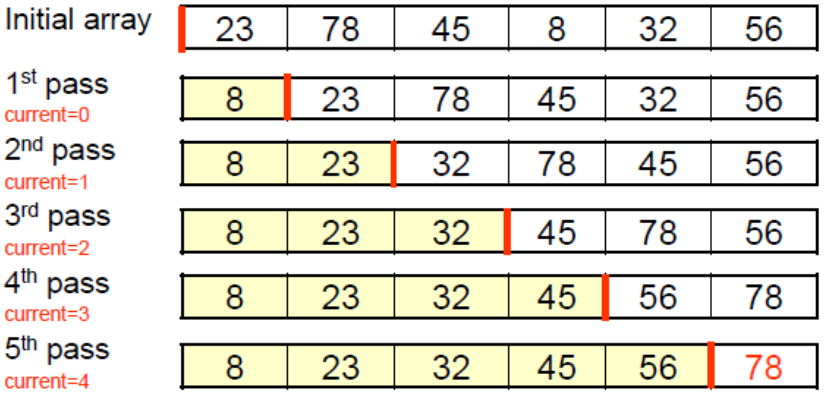
Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

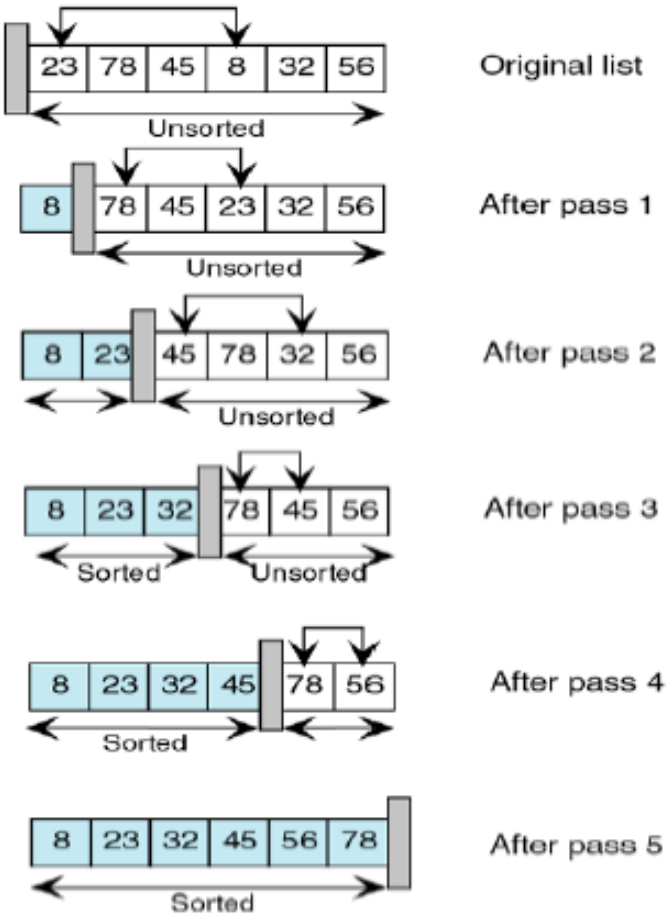
Administrative Issues

- Lab 11 due **5pm, Wednesday, April 12**
- **No Classes on Monday, April 17 (Patriot's Day Holiday); No Lab in the week of April 17**
- Exam #3 on **Friday, April 21**
  - Review session on Wednesday, April 19

Review of Lecture #24

- Sorting problem is a problem to sort/arrange a sequence of numbers into non-decreasing or non-increasing order
- Bubble sort works by repeatedly comparing adjacent elements and swapping adjacent elements that are out of order
- Selection sort works by repeatedly selecting the smallest/largest remaining element





Before:

- Each array element contains a *numerical* value: int, float, double,…!

```
int myarray[10]={1,2,3};

float yourarray[3]={1.2, 2.3, 3.4};
```

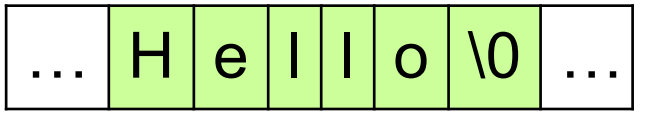
This lecture

Will examine arrays containing a single character in each element, and the last element is the null character (\0)

strings

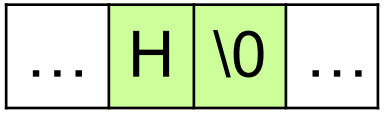
Strings in C

- A string is a series of characters treated as a unit.
- In C, a string is a variable-length array that is DELIMITED BY THE NULL CHARACTER (\0).
- Examples:
  - String “Hello”:



6 bytes

  - String “H”:



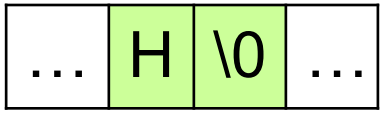
2 bytes

Why a null character is needed at the end of a string?

- A string is not a data type but a data structure (an array)
- It's a variable-length structure
- There is a need to identify the logical end of the data within the physical structure
- C uses '\0' as the end-of-string marker

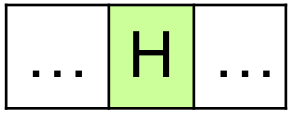
String Literals vs Character Literals

- A string literal is a sequence of characters enclosed in double quotes
  - Example: "H"



2 bytes

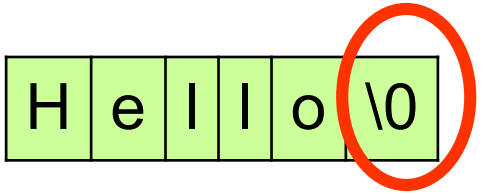
- A character literal is enclosed in single quotes!
  - Example: 'H'



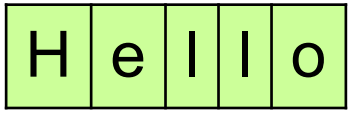
1 byte

Difference between strings and character arrays

- “Hello”



- char a[5]={‘H’, ‘e’, ‘l’, ‘l’, ‘o’};



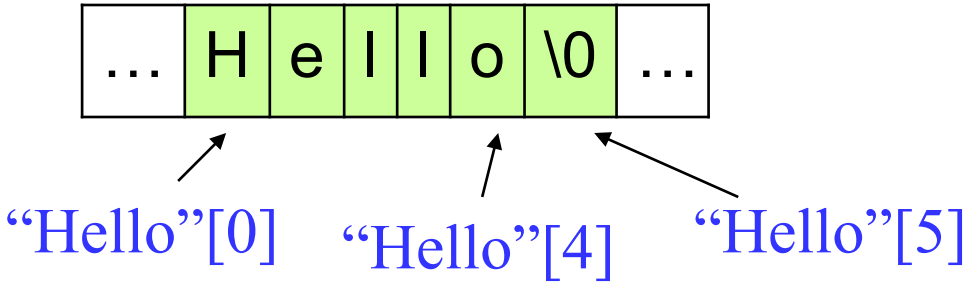
Note!

- In defining an array to store a string, you must provide enough room for the data and the delimiter!

- The storage structure must be 1 byte larger than the maximum data size.

Referencing String Literals

- A string literal is an array of characters
- Array name indicates the address of the first element of the array
- So, string itself is a pointer to the first element of the string
- Example: “Hello”



Exercises (1)

- What is the output of the following program?

```
#include "stdio.h"
int main(void)
{
        printf("%c\n%c\n", "Hello"[1], "Hello"[4]);
        return 0;
}
```

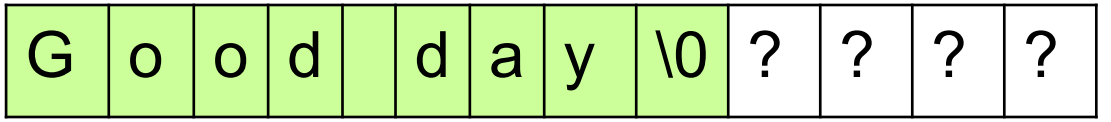
Ways to initialize a string?

Initializing Strings (1)

Example: to assign “Good day” to an array:

char a[13] = “Good day”;

- The compiler will create an array of 13 bytes
- The first nine spaces are taken up by the string characters and the null character.



- What about the rest? (hint: what is the value of any array element when only a portion of the array is specified by initialization?)

They are initialized to 0.

Exercises (2-1)

- What is the output of the following program?

```
#include "stdio.h"
void main(void)
{
	char a[13] = "Good day";
	for (int i = 0; i < 13; i++)
	{
		if (i <= 8)
			printf("%c\n", a[i]);
		else
			printf("%d\n", a[i]);
	}
}
```

Exercises (2-2)

- What is the output of the following program after changing “%d” to “%c” in the second printf()?

```
#include "stdio.h"
void main(void)
{
        char a[13] = "Good day";
        for (int i = 0; i < 13; i++)
        {
                if (i <= 8)
                   printf("%c\n", a[i]);
                else
                   printf("%c\n", a[i]);
        }
}
```

Exercises (2-3)

- What is the output of the following program after changing “%c” to “%d” in the first printf() as shown below?

```
#include "stdio.h"
void main(void)
{
        char a[13] = "Good day";
        for (int i = 0; i < 13; i++)
        {
                if (i <= 8)
                    printf("%d\n", a[i]);
                else
                    printf("%d\n", a[i]);
        }
}
```

Review Question

- What is the output of the following program?

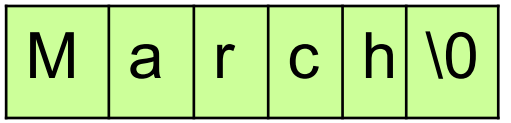
```
#include "stdio.h"
void main(void)
{
        char month[10] = "March";
        printf("%c\n%c\n", month[1], month[4]);
        printf("%c\n%d\n", month[8], month[9]);

}
```

Initialize Strings (2)

char month[6] = {‘M’, ‘a’,‘r’, ‘c’, ‘h’, ‘\0’};

- Initialize a string as an array of characters
- Need to ensure that the null character is at the end of the string



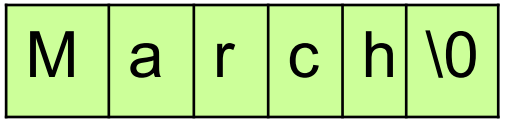
- It’s tedious to code!

Initializing Strings (3)

Example: to assign “March” to a string:

char month[] = “March”;

- The compiler will create an array of 6 bytes
- The six spaces are taken up by the string characters March and the null character.



- What if we store “December” in it later?

We could overrun the array and destroy whatever came after the array

Note!

char month[] = “March”;

- The above is a Dangerous way to initialize a string, because C creates an array with 6 elements and if we assign a bigger string to month, the program will crash.

Is there a safe and flexible way to initialize a string?

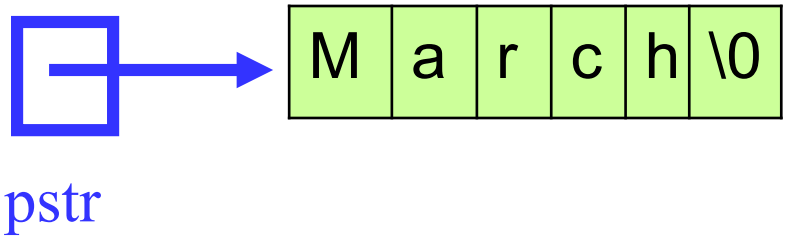
Using Pointers

- A pointer is a derived data type; a type built from one of the standard types
- The value of a pointer is any of the addresses available in computer for storing or accessing data
- To declare a pointer variable, use * in the declaration

Initializing Strings (4)

char *pstr="March";

- Assign a string literal to a character pointer



pstr is declared as a single pointer variable, used to hold the address of a character, here, the first character in the string "March"

Summary of Lecture #25

- In C, a string is a variable-length array that is DELIMITED BY THE NULL CHARACTER (\0).
- Four ways to initialize a string

char month[10] = “March”;

M	a	r	c	h	\0	0	0	0	0

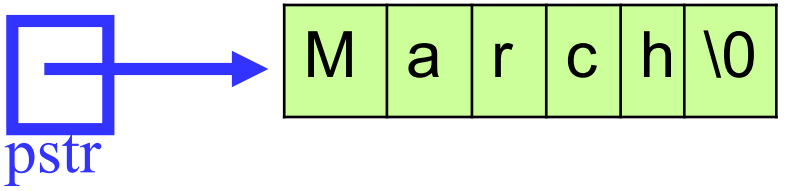
char month[] = “March”;

M	a	r	c	h	\0

char month[6] = {‘M’,‘a’, ‘r’,‘c’, ‘h’, ‘\0’};

M	a	r	c	h	\0

char *pstr=“March”;



Things To Do

- Review lecture notes
- Prepare for Exam#3

Next Topic

- Pointers