

Computer Science & Engineering 150A

Problem Solving Using Computers – Laboratory

Lecture 14 - String

Shuai Xie

(Adapted from Derrick Stolee, Lin Liu & Shuai Xie)

Spring 2010

Announcement: CodeLab assignment

CSCE150A

Assignment

As Function
Argument

Multidimensional
Array

String

Manipulating
Strings

- The 7th CodeLab assignment

"CodeLab Assignment-7"

Due: 23:59 Apr 25

Array as Function Argument

CSCE150A

Assignment

As Function
ArgumentMultidimensional
Array

String

Manipulating
Strings

- Entire arrays as function arguments.

```
void func(int array[]);    -- Not so good
```

Put array size as another arguments.

```
void func(int array[], int size);    -- Good
```

- Unlike passing values, C sends addresses of arrays.
 - Thus, there is no need for a `&` in the function call.
 - It can change the value of array element.
 - To avoid changing array values: **const int x[3];**
- In C, it is illegal for a function's return type to be an array;

Array as Function Argument: Example

CSCE150A

Assignment

As Function
ArgumentMultidimensional
Array

String

Manipulating
Strings

```
#include <stdio.h>

void bar(int temp[], int size);

int main(void)
{
    int foo[] = {1,2,3,4,5,6,7,8,9,10}, i;
    bar(foo,10);
    for (i = 0; i < 10; i++)
        printf("%d\n", foo[i]);
    return 0;
}

void bar(int temp[], int size) {
    int i;
    for (i = 0; i < size; i++)
        temp[i] *= temp[i];
}
```

Multidimensional Array declaration and initialization

CSCE150A

Assignment

As Function
Argument

Multidimensional
Array

String

Manipulating
Strings

- Example: `int x[10][10], y[3][2][4];`
- Initialization
`int x[][3]={ {1,2,3},{4,5,6},{7,8,9} } ;`
- Accessing: `x[0][1], y[2][1][0];`
- Nested for loop to access the array.
- Iterating:


```
for ( i = 0; i < sizeX; i++ )
    for ( j = 0; j < sizeY; j++ ) {
        name[i][j] = 0;
    }
```

- Unfortunately C language doesn't support directly use two-dimensional array as function argument

```
int func(int arr[][],int x, int y);
```

Wrong!

- You must give out all the dimension size except the first dimension

```
int func(int arr[][10],int x);
```

OK!

Here, x is the size of the first dimension

- Try unix command `ls -a > 1.txt`
- To test our codes easily or other reason, we can use redirect input.
Store all the input data into a file.
Use unix redirection command "<"
- Example: `./GetAverage < Score.txt`
- For redirect input, your program will use `scanf` as normal, but the program will get input data from the file instead of keyboard, save time for programmer to input data

- A list of characters is a *string*.
- We use arrays of type `char`:

```
char s[] = "My string!";
```

'M'	'y'	' '	's'	't'	'r'	'i'	'n'	'g'	'!'	'\0'
77	121	32	115	116	114	105	110	103	33	0

- The final `'\0'` character signifies the end of the string.
- *Note: there may be more entries in the array, but they will be ignored!*

Dealing with Character Values

CSCE150A

Assignment

As Function
Argument

Multidimensional
Array

String

Manipulating
Strings

- Each char is essentially a number, but gets translated into a letter with the ASCII table.
- <http://www.asciitable.com>
- String examples
 - "UNL", "C Program", "15.21"

Dealing with Character Values

CSCE150A

Assignment

As Function
ArgumentMultidimensional
Array

String

Manipulating
Strings

- Upper-case letters 'A' through 'Z' appear as contiguous numbers, in order.
- Lower-case letters 'a' through 'z' appear as contiguous numbers, in order.
- You can treat a char type (including string index) and char literal (such as 'a') as numbers!
 - You can use comparisons: == != <= >=
 - You can use operations: + - / *

Dealing with Literal Strings

CSCE150A

Assignment

As Function
ArgumentMultidimensional
Array

String

Manipulating
Strings

- We have seen string literals before, using double quotes ("")
- Examples:
 - `printf("Hello world!");`
- These strings cannot be changed after compile time, since they are never stored into an array.

Storing Strings

CSCE150A

Assignment

As Function
Argument

Multidimensional
Array

String

Manipulating
Strings

- You can use static arrays to store strings.

```
char string1[] = "Length determined by string";
char string2[100] = "Length determined by 100";
```

- Then, you can output strings with printf:

```
printf("My string is:  %s\n",string1);
printf("My string is:  %s\n",string2);
```

Pulling Characters

CSCE150A

Assignment

As Function
ArgumentMultidimensional
Array

String

Manipulating
Strings

- We can access individual characters by treating the string as an array.

```
char string[] = "What is the word?";  
printf("Starts with %c\n", string[0]);
```

Using String Library

CSCE150A

Assignment

As Function
ArgumentMultidimensional
Array

String

Manipulating
Strings

- There are a lot of built-in functions in `string.h`
`#include <string.h>`
- Key pattern: first parameter will store the result.
`strlen(char*)`
`strcpy(char*, char*)`
`strcat(char*, char*)`
`strncpy(char*, char*, int)`
`strncat(char*, char*, int)`
`strncmp(char*, char*)`
- More available in textbook(Page 455).