

Chapter 8: STRINGS

1) Fundamental Concepts.....	8-2
2) Aggregate Operations.....	8-4
3) String Functions.....	8-6
4) Exercises: Strings.....	8-8

Evolution

Fundamental Concepts

- In C, there is no fundamental data type named `string` and thus a string is kept as a character array.

Use the null character to mark the end of a string
`'\0'` (A BYTE OF ZERO BITS)

- A string constant is coded as a sequence of characters delimited by quote symbols.
 - ▶ Do not confuse the string `"A"` with the single character `'A'`
 - ▶ They are of different types and have different meanings to the C compiler.
- When the compiler sees a string constant, it places the string in memory and then places the `null` character behind them.
- If you need to process many strings (such as sorting them) you can use a two dimensional character array.

Fundamental Concepts

- String variables are one dimensional character arrays.

```
#define MAX    100
char string[MAX];
char word[MAX * 2];
char line[MAX + 1];
```

- String constants are enclosed in "quotes".

<u>STRING</u>	<u>LENGTH</u>
"a\nstring"	8
"a string\n"	9
"a"	1
" "	0

- The compiler stores a string as:

```
printf("MICHAEL");
```

```
-----
| 'M' | 'I' | 'C' | 'H' | 'A' | 'E' | 'L' | '\0' |
-----
```

- If several strings need to be processed, the following can be used:

```
#define NO_OF_STRINGS 10
#define LENGTH 20
char lines[NO_OF_STRINGS][LENGTH + 1];
```

Aggregate Operations

- An array is treated by the compiler as the address of the place in memory where the array is kept.
- A subscripted reference is resolved by knowing two facts:
 - ▶ Address of the beginning of the array
 - ▶ $\text{Offset} = \text{subscript} * \text{sizeof}(\text{item})$

```
char line[100] = "some stuff.....";  
line[6] = 'A';
```

The last statement is a reference to the 6th byte of memory away from the base address of `data`.

- Because array names are addresses in memory, aggregate operations are not allowed. Instead, functions must be written to:
 - ▶ Copy a string
 - ▶ Compare two strings
 - ▶ etc.
- Common string functions are in the C Standard Library.

Aggregate Operations

- Aggregate operations on arrays are not allowed.

```
line = word; /* Try to copy a string */
```

Compiler error: Can't change the beginning address of an array.

- Comparing strings:

```
if( line == word ) /* Wrong way */  
do something;
```

They can never be equal. Addresses are compared here.

- Must write functions to do each of these. Fortunately they already exist in the C standard library.

String Functions

- Some of the standard string functions and their prototypes are now shown. Note how a prototype serves as a documentation aid.

```
/* Copy source to target */
void strcpy(char target[ ], char source[ ]);

/* Compare two strings */
int strcmp(char string1[ ], char string2[ ]);

/* Compute length of string */
int strlen(char string[ ]);
```

- `strcpy` is invoked with:

```
strcpy(line, word); /* line <- word */
```

- `strcmp` is invoked with:

```
if (strcmp(line, word) == 0) {
    code for equality
}
else {
    code for inequality
}
```

- Character string constants are similar to character arrays.

```
strcpy(line, "copy me to line");

if ( strcmp(line, "quit") == 0) {
    code for equality
}
```

String Functions

```
int strlen(char s[ ]) /* length of string */
{
    int len = 0;
    while (s[len] != '\0') /* End of string? */
        len++;           /* No */
    return(len);         /* Yes */
}

void strcpy(char to[ ], char from[ ])
{
    int i = 0;
    while( from[i] != '\0') {
        to[i] = from[i];
        i++;
    }
    to[i] = from[i];
}

/* Another version of string copy */
void strcpy(char to[ ], char from[ ])
{
    int i = 0;
    while((to[i] = from[i]) != '\0')
        i++;
}

/* Still another version */
void strcpy(char to[ ], char from[ ])
{
    int i = 0;
    while(to[i] = from[i]) /* End of string? */
        i++;
}
```

Exercises: Strings

Note that all the exercises in this lab depend upon strings ending with the `'\0'` character.

1. Write the function `reverse` which reverses a string. The prototype for `reverse` is:

```
void reverse(char string[]);

DRIVER PROGRAM:
#define MAX 100
void reverse(char string[]); /* Prototype */
main()
{
    char line[MAX];
    gets(line);             /* Get it */
    reverse(line);         /* Reverse it */
    printf("%s\n", line);  /* Print it */
}
void reverse(char s[])
{ /* You fill in
  this part
*/
}
```

2. Write the function `changecase` which reverses the case of any alphabetic characters in a string. The other characters stay the same. The prototype is:

```
void changecase(char string[]);
```

3. Write the function `strcmp` which compares two strings. In order to distinguish between the equality and inequality cases, `strcmp` must return an `int`. The prototype is:

```
int strcmp(char string1[], char string2[]);
```

4. Write a function `palindrome` which detects whether a string is the same forward or backward. The prototype is:

```
int palindrome(char string[]);
```

Note that the `palindrome` function needs only to invoke some of the string functions mentioned in this chapter.