Basis and Practice in Programming

Chapter 11: Character Strings and String Functions

Prof. Tamer ABUHMED
College of Software

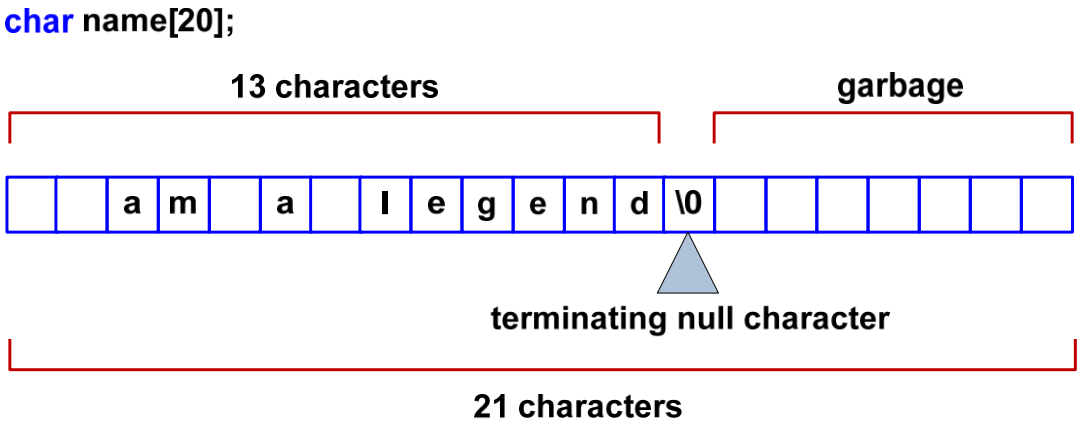
성균관대학교
SUNG KYUN KWAN UNIVERSITY(SKKU)

Class Objectives

- **Explain How to Handle Character Strings**
- **Explain How to define a String in a Program**
- **Again, Explain the Difference bet. Pointer and Array**
- **Explain How to Input a String**
- **Explain How to Output a String**

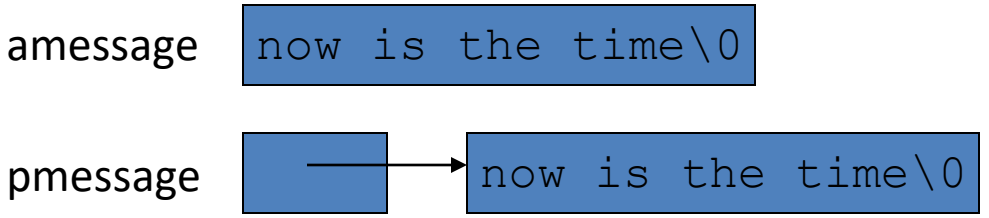
Handling Character Strings

- **Character String**
  - A character string is a char array terminated with a null character (\0).
    - What we studied regarding pointers and arrays still applies.
  - **Functions**
    - To print a char on the screen, we use putchar()
      - To print a string, we use puts()
    - To read a char, we use getchar()
      - To read a string, we use gets()



Character pointers

```
char amessage[] = "now is the time"; /* an array */
char *pmessage = "now is the time"; /* a pointer */
```



- `amessage` is an array, just big enough to hold the sequence of characters and '\0' that initializes it. Individual characters within the array may be changed but amessage will always refer to the same storage.
- `pmessage` is a pointer, initialized to point to a string constant; the pointer may subsequently be modified to point elsewhere, but the result is undefined if you try to modify the string contents.

Precedence of operators

- **`*p++`** increments p after fetching the character that p points to

```
char *p="hello" ;
printf("%c",*p++); // displays h
```

- **`*++p`** increments p before fetching the character that p points to

```
char *p="hello" ;
printf("%c",*++p); // displays e
```

Handling Character Strings – contd.

```c
# include <stdio.h>

# define MSG1 "A man with courage is a majority"
# define LIM 4

void main(void)
{
   int ii = 0;  char name[10];
   /* an array of 20 char */
   char m1[20] = "You are a student.";
   /* an array w.o. defining size */
   char m2[] = "You are 20 years old.";
   /* an array of 4 pointers */
   const char* hobbies[LIM] = { "Ski", "Programming",
          "Watching TV", "Doing nothing!"};
   printf("What is your name? ");
   fgets(name,sizeof(name), stdin);
   printf("Hello %s ! ", name);
   puts(m1);
   puts(m2);
   printf("Your favorite quote is ");
   puts(MSG1);
   printf("Your hobbies are:\n");
   for (ii = 0; ii < LIM; ii++)
     puts(hobbies[ii]);

}
```

Output

```
What is your name? Huey [Enter]
Hello Huey! You are a student.
You are 20 years old.
Your favorite quote is A man with co
    urage is a majority
Your hobbies are:
Ski
Programming
Watching TV
Doing nothing!
```

Defining Strings Within a Program

- **Character String Constants**
  - **String constant is anything enclosed in double quotation marks.**
  - **Examples**
    - **char greeting[50] = "Hello" " Anyeoung" " Salam.";**
    - **char greeting[50] = "Hello Annyeung Salam.";**
    - **char m1[5] = {'A', 'B', 'C', 'D', 'E'};**
      - ***m1 := m1[0]**
      - ***m1 := 'A'**
      - **m1 := &m1[0]**
    - **char a[] = "Today is Friday";**
    - **char *b = "Today is Friday"; //same as precedent one**
    - **char a[]; //Wrong: unknown size**

Defining Strings Within a Program – contd.

- **Difference bet. Array and Pointer**
  - **char** *a = "I am free";
  - **char** b[] = "I am here";

```
for(i = 0; i < 9; i++)
    putchar(a[i]);
```

```
for(i = 0; i < 9; i++)
    putchar(b[i]);
```

```
for(i = 0; i < 9; i++)
    putchar(*(a + i));
```

```
for(i = 0; i < 9; i++)
    putchar((*(b + i));
```

```
while(*(a) != '\0')
     putchar(*(a++));
```

```
while(*(b) != '\0')
     putchar(*(b++));
```

```
a = b;
```

```
b = a;
```

Defining Strings Within a Program – contd.

- **Arrays of Character Strings**
  - **Array of char pointers**

```
Const char *hobbies[LIM] = {"Ski",
                            "Programming",
                            "Watching TV",
                            "Doing nothing!"};
```

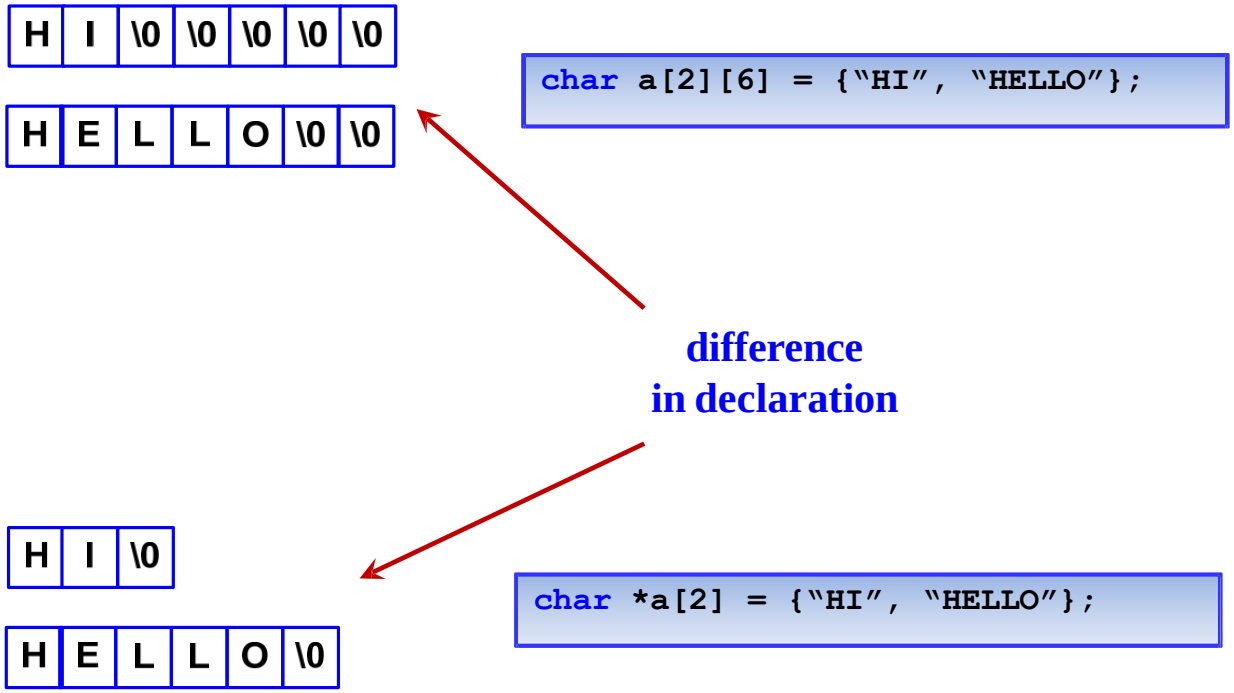
    - **Note that**
      - **hobbies[0] := hobbies := &hobbies[0][0]** **//address**
      - ***hobbies[0] := 'S', *hobbies[1] == 'P'**
      - ****hobbies == 'S'**

  - **Two-dimensional array**

```
Char hobbies[LIM][LEN] = {"Ski",
                          "Programming",
                          "Watching TV",
                          "Doing nothing!"};
```

Defining Strings Within a Program – contd.

- **Storage**



String Input

- **scanf()**
  - **It stops at the first space it encounters.**
  - **Example:**
    - **char name[30];**
    - **scanf("%s", name);**
    - **If you enters: Huey Freeman**
      - **name = "Huey"**
  - **You can force scanf() to stop before the space as follows.**
    - **scanf("%3s", name);**
    - **If user enters: Huey Freeman**
      - **name = "Hue"**
- **fgets()**
  - **Stops at the newline [enter].**

String Output

Example

```
/* put_out.c -- using puts() */
#include <stdio.h>
#define DEF "I am a #defined string."
void main(void)
{
    char str1[80] = "An array was initialized to me.";
    const char * str2 = "A pointer was initialized to me.";

    puts("I'm an argument to puts().");
    puts(DEF);
    puts(str1);
    puts(str2);
    puts(&str1[5]);   // starts at &str1[5]
    puts(str2+4);     // starts at str2+4
}
```

Output

```
I'm an argument to puts().
I am a #defined string.
An array was initialized to me.
A pointer was initialized to me.
ray was initialized to me.
inter was initialized to me.
```

String Output – contd.

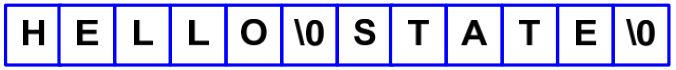
- **Do-It-Yourself Option**
  - **int put2(const char * string)**
    - **Prints the string and returns its length.**

```
/* put2.c -- prints  a string and counts characters */
#include <stdio.h>
int put2(const char  * string)
{
     int count = 0;
     while (*string)           /* common idiom */
     {
         putchar(*string++);
         count++;
     }
     putchar('\n');           /* newline not counted */

     return(count);
}
```

String Functions

- **Functions**
  - **strlen()**
    - **Finds the length of a string.**



      - **The length = 5!**

  - **strcat()**
    - **Concatenates two strings.**
    - **Example:**

```
char str1[30] = "My country is ";
char str2[10] = "Korea";
strcat(str1, str2);
```

      - » **str1 == "My country is Korea"**
      - » **Str2 == "Korea"**

String Functions – contd.

- Functions – contd.
  - strncat()
    - Specify the maximum number of characters to append.
    - Note that strlen() does not check if str2 can fir in str1.
    - Example

```
strncat(str1, str2, 10);
```

  - strcmp()
    - Compares two strings.
    - Example

```
int y = 0;          char str1[] = "Hello";    char str2[] = "Ciao"
y = strcomp(str1, str1);        // y = 0      same
y = strcomp(str1, str2);        // y = 1
y = strcomp(str2, str1);        //y = -1
```

String Functions – contd.

- strncat (*s1, s2, n*)
  - Copies *s2* to the *end* of *s1* until either the null character is reached or *n* characters have been copied, whichever occurs first. Returns *s1*.
- strncmp (*s1, s2, n*)
  - Performs the same function as strcmp, except that at most *n* characters from the strings are compared.
- strncpy (*s1, s2, n*)
  - Copies *s2* to *s1* until either the null character is reached or *n* characters have been copied, whichever occurs first. Returns *s1*.
- strchr (*s, c*)
  - Searches the string *s* for the last occurrence of the character *c*. If found, a pointer to the character in *s* is returned; otherwise, the null pointer is returned.
- strstr (*s1, s2*)
  - Searches the string *s1* for the first occurrence of the string *s2*. If found, a pointer to the start of where *s2* is located inside *s1* is returned; otherwise, if *s2* is not located inside *s1*, the null pointer is returned.

String to number: conversion functions

- atoi(s) converts string s to a type int value and returns it. The function converts characters until it encounters something that is not part of an integer.
- atof() converts a string to a type double value and returns it
- atol() converts a string to a type long value and returns it
- All of these functions are in <stdlib.h> header file

```
// Using atoi
#include <stdio.h>
#include <stdlib.h>
int main (void) {
    printf ("%i\n", atoi("245"));
    printf ("%i\n", atoi("100") + 25);
    printf ("%i\n", atoi("13x5"));
    return 0;
}
```

Example: array of structures

Example: a dictionary program

```
#include <string.h>
struct entry
{
char word[15];
char definition[50];
};

struct entry dictionary[100] =
    { { "aardvark", "a burrowing African mammal" },
    { "abyss", "a bottomless pit" },
    { "acumen", "mentally sharp; keen" },
    { "addle", "to become confused" },
    { "aerie", "a high nest" },
    { "affix", "to append; attach" },
    { "agar", "a jelly made from seaweed" },
    { "ahoy", "a nautical call of greeting" },
    { "aigrette", "an ornamental cluster of feathers" },
    { "ajar", "partially opened" } };
```

Example: dictionary continued

```
int lookup (const struct entry dictionary[],
        const char search[], const int entries);

int main (void)
{
    char word[10];
    int entries = 10;
    int entry;
    printf ("Enter word: ");
    scanf ("%14s", word);
    entry = lookup (dictionary, word, entries);
    if ( entry != -1 )
        printf ("%s\n", dictionary[entry].definition);
    else
        printf ("The word %s is not in my dictionary.\n", word);
    return 0;
}
```

Searching in array

```
// function to look up a word inside a dictionary
int lookup (const struct entry dictionary[],
       const char search[],const int entries)
{
int i;
for ( i = 0; i < entries; ++i )
       if ( strcmp(search, dictionary[i].word)==0 )
              return i;
return -1;
}
```

Summary & Discussion

- **Explained How to Handle Character Strings**
- **Explained How to define a String in a Program**
- **Again, Explained the Difference bet. Pointer and Array**
- **Explained How to Input a String**
- **Explained How to Output a String**