Computer Programming

An-Najah N. University
Computer Engineering Department
Luai Malhis, Ph.D,

Strings

String Definition

- Series of characters treated as single unit
- Can include letters, digits, special characters **+**, **-**, ***** and any character in the ASCII code
- String literal (string constants) Enclosed in double quotes, for example: **"Palestine"**
- Array of characters, ends with null character **'\0'**
- String name is a pointer that points to the first character of the string.
- String name can be static or dynamic as follows:

char ss[20] =“palestine”; // static pointer

char *ds = “palestine”; // dynamic pointer

String assignment

- Character array **char color[] = "blue";**

  Creates 5 element **char** array **color** last element is**'\0‘**

  Same as char color[5] =“blue”;

  Alternative for character array

  char color[] = {‘b’,‘l’,‘u’,‘e’,‘\0’ };

- Variable of type **char ***

  char *colorPtr = "blue";

  Creates pointer **colorPtr** to letter **b** in string **“blue”**

- **“blue”** is stored somewhere in memory.

String input output

- In addition to initializing string as one unit we can read and print a string as one unit:
- Reading strings: Read string content from Keyboard Given: **char word[20]; then cin >> word**;

  Reads characters until whitespace (BLANK, TAB, ENDLINE) is reached. Then appends **‘\0’**.

  Can read at most 19 characters.

- Printing strings: display string content on the screen

  cout << word ;

  prints characters until **'\0‘ is reached**

String Processing

- Given a string char * s = “ABCDE”;
- The statement cout << s; prints ABCDE
- The statement cout << s+2; prints CDE
- s = s+3; cout << s; is a valid operation since s is a dynamic pointer; prints DE on the screen
- for (;*s;s++) cout << s << “ “; prints:
  ABCDE BCDE CDE DE E
- for (;*s;s++) cout << *s << “ “; prints:
  A B C D E

String Processing 2

- Given a string char s[] = “ABCDE”;
- The statement cout << s; prints ABCDE
- The statement cout << s+2; prints CDE
- s = s+3; cout << s; is an in valid operation since s is a static pointer; cannot change s
- for (int i =0; s[i];i++) cout << s+i << “ “; prints:
  ABCDE BCDE CDE DE E
- for (int i =0; s[i];i++) cout << s[i] << “ “; prints:
  A B C D E

String Functions

- Set of build in function in C to manipulate strings. These functions are found in library <string.h>: must #include <string.h>
- Some of the most important functions are:
  - Copy one string to another
  - Compare two strings strings
  - Compute string length
  - Concatenate one string into another string
  - In the next few slides we will study these function
  - There are many more functions in string.h

String functions prototypes

`int strlen( char *s1);`	Returns the number of characters in string s without the null.
`char *strcpy(char *s1, char *s2 );`	Copies the string **`s2`** into the character array **`s1`**. The value of **`s1`** is returned.
`char *strcat( char *s1, char *s2 );`	Appends the string **`s2`** to the string **`s1`**. The first character of **`s2`** overwrites the terminating null character of **`s1`**. The value of **`s1`** is returned.
`int strcmp(char *s1, char *s2 );`	Compares the string **`s1`** with the string **`s2`**. The function returns a value of zero, less than zero or greater than zero if **`s1`** is equal to, less than or greater than **`s2`**, respectively.

String Length: strlen

`int strlen( char *s1);`

Returns the number of charters in the string without null.

Given char s1[10] =“ABCDEF”; char s2[]=“zyz”;

Cout<<strlen(s1); prints 6.

Cout << strlen(s2); prints 3.

Cout << strlen(s1+2); prints 4

Cout << streln(s2+streln(s2)); prints 0.

Cout << strlen(“12345”); 5

Cout << strlen(“”); prints 0

String Copy: strcpy

`char *strcpy( char *s1, char *s2 )`

Copies second argument into first argument. S1 must be large enough to store s2 with the null character.

Given char S1[10] = “ABCDEFG”; char S2 =“XYZ”; Then:

strcpy (s2,s1); is an invalid because s2 is too small for s1;

strcpy (s1,s2); “XYZ” is stored in s1 and s2 is not changed

strcpy(s1+2,s2); s1 becomes ABXYZ;

strcpy(s1,”123456789”); s1 becomes “123456789”

strcpy(s1+2,s2+2); s1 becomes ABZ

strcpy(s2+2,”LMN”); invalid operation

Strcpy (s1,strcpy(s2,”ABCD”); copies ABCD into s1 and s2

Concatenating strings: strcat

`char *strcat(char *s1, const char *s2)`

Appends second s2 to the end of s1. Must make sure s1 large enough to store all characters in s1, s2 and null.

returns pointer to s1.

Examples: Given char S1[10] = “ABCDEFG”;char S2 =“XYZ”;

strcat (s2,s1); is an invalid because s2 is too small for s2+s1;

strcat (s1,s2); “XYZ” is stored at the end of s1= ABCDEFGXYZ

strcat(s1+2,s2); s1 becomes ABCDEFGXYZ

strcat(s1+2,s2+2); s1 becomes ABCDEFGZ

strcat(s2,”LM”); is invalid operation becaue s2

strcat(“lm”,s2); is invalid operation because “lm” is constant.

stract (strcat(s1,”1”),”2”); s1 becomes ABCDEFG12

String Compare : strcmp

`int strcmp( char *s1, char *s2 )`

Characters represented as ASCII code.

Compares string character by character according to their ASCII code values. Example

S1 =“abc” s2 =“abcd”, s3 =“ABCDEF”, s4 =“123456”;

Returns Zero if the two strings are equal.

Returns Negative value if s1 is smaller than s2

Returns Positive value if s1 is greater than s2.

In examples above s1 < s2, s2 > s3, s4 <s1,s2, s3.

String compare continue

Given char s1[10] = “ABCD”; char s2[] =“ABM”; Then

strcmp(s1,s2); returns a value < 0; since C < M

strcmp(s2,s1); returns a value > 0; since M > C

strcmp(s2, “ab”); returns a value < 0 since A < a.

strcmp(s2,”ABM”); returns 0 since both strings are equal

strcmp(s1+2”,”CD”); returns 0 since both are equal

strcmp(“abc”,”a”) returns > 0 since b > null

strcmp(“abc”,strcpy(s1,”abc”); returns > 0;

strcmp(“s1+2,strcpy(s1,”M”); returns 0

String Function Examples 1

Write code to read 100 strings print the average string size. Assume max string size is 20.

```
char st[21]; int sumall =0;
for (int i =0; i < 100; i++) {
    cin >> st;
    sumall += strlren(st);
}
cout << “average string size is “<< sumall/100.0;
```


String Functions Example 2

Write code to keep reading strings until the string “finish” is entered print the largest entered string. Assume max string size is 20.

```
char st[21]; char maxst[21] =“”;
while(1) { cin >> st;
      if (strcmp(st,”finish”) ==0) break;
      if (strlen(st) > strlen(maxst))
          strcpy(maxst,st);    }
 cout << maxst;
```


String Functions Example 3

Write code to read 50 strings concatenate them into 1 string. Assume max string size is 20. The compute the strlen of the new string and the count of ‘a’ in the new string.

```
char st[21]; char all[50*20+1] =“”;
 for(int i =0; I < 50; i++) { cin >> st;  strcat(all,st)}
 cout << “the length of all is” << strlen(all);
 int counta =0;
 for (chat *p = all; *p; p++) if(*p == a) couta++;
```

String Functions Example 4

Write a function that takes string s1 and char c1 as parameters. Your function returns the number of times c1 is found in s1.

```
int find(char *s, char c1) {
        int count =0;
       while (*s) {
            if ( *s == c1)  count++;
            s++;  }
        return(count);
}
```


String Functions Example 5

Write function that takes s1, c1 and c2 as parameters your function returns a pointer to new allocated string that contains all characters between c1 and c2 inclusive.

```
char * extract(char *s, char c1, char c2) {
   char *p1 = s, *p2 = s, *p, *ns;
   while(*p1 != c1) p1++; while(*p2 != c2) p2++;
   ns = new char[p2 – p1 +2];
   for (char *t = ns, p = p1; p <=p2;) *t++ = *p++;
   return(ns);  }
```


Array of Strings 1

- We can allocate a matrix of characters and store each string in a given row of the matrix. Following is a code to read student names from the keyboard and store the names in a two dimensional array.

```
char students[40][15];
//40 students, Max length of a name is 14 letters
 for (int I =0; I < 40; I++) {
     cout  <<"Enter Student Name:";
     cin >> students[i];
  }
```

- // also we can print them one string per line

```
for (I =0; I <40; I++) cout << students[i] << endl;
```


Array of Strings 2

- Given char months[12][20] = {“January”, “February”, “March”, ...,”December”};

Print the number of months starts with M or J

Print the months that are larger than 5 characters

Print the number of months that end with y.

```
Int smj =0;  int ey =0;
for (int i = 0; i < 12; i++) {
   if (months[i][0]==‘M’||months[i][0]==‘J’) smj++;
   if (strlen(months[i]) > 5) cout<<months[i]<<endl;
   if (months[i][strlen(months[i]) -1] == y) ey++;}
```