Lecture 8 Strings

* What is a string?
  + A string is a sequence of characters
  + In C, a string is a sequence of characters *terminated with the null character* stored in a char array
* A char array is the memory space to hold the string data
  + Each character is stored in an array position, one follows another
  + A string is represented by a char pointer pointing to the position of the first character
  + The end of string is marked by a null character ‘\0’ (0 in decimal)
* The length of a string is the number of non-null characters
* A string is accessed by a char pointer to its first character

Example

Char str[10]; //declare a char array of 10 bytes

Str[0] = ‘H’;

Str[1] = ‘e’;

Str[2] = ‘l’;

Str[3] = ‘l’;

Str[4] = ‘0’;

Printff(“%s”,str); //output: Hello

* The first five elements hold the characters h e l l o
* The sixth element str[5] holds the string terminator ‘\0’
* Array name str is a char pointer pointing to the first character
* The length of the string str is 5
* The length of the array str is 10
* The size of the array str is 10

**Other ways to declare arrays**

Char name[] = “cp264”;

Or

Char name[] = {‘c’.’p’,’2’,’6’,’4’,’\0’};

* The compiler allocated char array name of 6 elements and assigns the values above to the char array elements
* The length of the string that array *name* holds is 5
* The size of the array *name* is 6

**String operation by pointer**

* Since a string is stored in a char array, array pointer and operations apply to string data operations

Char name[20] = “cp264”;

Char \*p;

P = &name[0];

Printf(“%c\n”,\*p) //prints c

Printf(“%c\n”,\*(p+3)); //prints 6

Printf(“%s\n”,p+2); //prints 264

\*p = ‘C’;

\*(p+1) = ‘P’;

Printf(“%s\n”,p) prints CP264

Note: p is a char type pointer and char type has 1 byte, p+1 increase the value of p by 1

**String pointer**

* If a string is declared like char \*p = “cp264”; then string p is a read-only string, cannot change
  + \*(p+1) = ‘c’ is not allowed

**String traversal**

Void display\_string(char s[]){

Char \*p = s;

While(\*p!=’\0’){ printf(“%c”,\*p); p++;}

}

Char str[] = “cp264”;

Display\_string(str);

//prints cp264

* Since a string has a terminator, we don’t need to pass the length of a string to a function

**Array of strings**

* An array of strings is considered a 2D array
* Arr[i] is a pointer pointing to the first char of row i

**Command line arguments**

Main(int argc, char \*argv[])

* Parameter argc represents the number of command line arguments
* Parameter argv represents an array of char pointers, argv[i] points to the ith argument string

Example

a.out argument1 argument 2

argc will have value 3

argv[0] points to string “a.out”

argv[1] points to string “argument1”

argv[2] points to string “argument2”

**String operations**

1. Read string from stdin
2. Write string to stdout
3. Length of a string
4. Change case
5. Copy string
6. Concatenate string
7. Compare string
8. Reverse string

General string processing algorithm

Input: string str[] or char \*str

Step 1: set char \*ptr = str

Step 2:

While(\*ptr){ //traversal scan the string by loop

If pattern is matched

Take action

Ptr++

{

Step 3: stop

**Reading strings from stdio (keyboard)**

We need a char array to store a string for stdin input

Char str[100];

Stdio provides three functions to get input from the keyboard

* Scanf() – get formatted data, ex. Scanf(“%s”, str); prompts user to type a string and hit the enter key to terminate the input
* Gets() – get a string ex. Gets(str); prompts the user to type a string and hit enter key to terminate the input
* Getchar() – get and return a character from keyboard

Both scanf() and gets() functions insert the typed characters one after another into the array str starting from location str[0], and insert ‘\0’ at the end

Char name[20];

Scanf(“%s”, name);

>>cp264 2020 <enter>

Printf(“%s”, name); //output: cp264

OR

gets(name, sizeof(name), stdin);

>>cp264 2021 <enter>

Printf(“%s”, name); //output 2021

* Getchar() only reads and returns one character from i/o buffer at a time. Using getchar() repeatedly until a stop character is encountered, then insert the null at the end.

Int i=0;

Char ch = Getchar();

While(ch!=’\n’){

Str[i] = ch;

I++;

Ch=Getchar();

}

Str[i] = ‘\0’;

**Writing string to stdout**

* Stdio provides three functions to output to stdout (screen)
  + Printf() – formatted printing
  + Puts() – print a string
  + Putchar() – print a character

Example

Char str[] = “data structures”;

Printf(“%s”, str);

Puts(str);

Char \*p=str;

While(\*p) putchar(\*p++);

**Compute the length of a string**

* The length of a string is the number of non-null characters
* The length if “cp264” is 5, the length of “hello c” is 7

Int str\_length(char \*s){

If(s==NULL) return -1;

Int counter = 0;

While(\*s){ //pattern is \*s!=0

Counter++;

S++;

}

Return counter;

}

Char a[50] = “C programming language”;

Printf(“%d”, str\_length(a)); //output 22

**String copy**

Char a[] = “C programming language”;

Char c[30];

Int I;

For(I = 0; \*(a+i)!=’\0’;i++){

\*(c+i) = \*(a+i);

}

\*(c+i) = ‘\0’; //add NULL to end

Char \*p1, \*p2;

P1 = a;

P2 = c;

For(;\*p1!=’\0’;p1++, p2++){

\*p2 = \*p1;

}

\*p2 = ‘\0’ //add NULL to end

**Another way**

Void copy\_string(char \*from, char \*to){

While((\*to++=\*from++)!=’\0’);

}

**Converting characters of a string into Upper Case**

Void upper\_case(char \*s){

If(s==NULL) return;

While (\*s){

If(\*s>=’a’&&\*s<=’z’)

\*s -=32; //do the opposite to convert to lower case

S++

}

}

**Concatenate String**

Appending a source string to the end of a destination string is like copying the source string to the end position destination string.

The algorithm first traverses the destination string to the null position, then starts to copy character from the source string.

Void append\_string(char \*from, char \*to){

If(from==NULL || to== NULL) return;

While (\*to) to++; //traverse to the null position

While ((\*to++ = \*from++) !=’\0’); //copy string

}

**Comparing two string in dictionary/lexical order**

Input: Given two strings s1 and s2

Output: 0 if s1 and s2 are exactly the same ; //s1==s2

Otherwise, let I be first I, such that s1[i] != s2[i].

If (s1[i]>s2[i]) return 1; // s1>s2

Else return -1; //s1<s2

Code:

Int compare\_string(char \*s1, char \*s2){

While(\*s1||\*s2){

If(\*s1>\*s2)

Return 1;

If(\*s1 < \*s2)

Return -1;

S1++;

S2++;

}

Return 0;

}

**Reverse of a string**

* To reverse a string we just need to swap the first character with the last, second character with the second last character, and so on and so forth

Void reverse\_string(char \*s){

If(s==NULL) return;

Char \*p = s, t

While (\*p++); //traverse p to the null position

p--; //let p point to the last character

while(s<p) { //s and p point to symmetric positions

t = \*s //swap

\*s = \*p;

\*p = t;

S++;

p--;

}

}

**String library**

#include <string.h>

1. String copy function

* Char \*strcopy(char \*destination, char \*source);
  + Copy the source string to the destination location

1. Memory copy function

* Void \*memcpy(void \*destination, const void \*source, tize\_t n);
  + Copy **n** characters from the source location to the destination location

1. String length function

* Int strlen (cont char \*str);
  + Returns string length

1. Concatenate two strings

* Char \*strcat(char \*destination, char \*source);
  + Source is appended to destination

1. Compare two strings

* Int strcmp(const char \*first, const char \*second);
  + Returns -1 if first <second
  + 0 if first = second
  + 1 if first > second

1. 8Finds the first occurrence of a substring

* Char \*strstr(const char \*s1, const char \*s2)
  + Returns NULL pointer if s1 does not contain s2, else a pointer pointing to the first substring that matches s2;
  + Returns s1 if s2 is NULL