CS19001: Programming and Data Structures Laboratory

Aritra Hazra;
CSE, IIT Kharagpur

http://cse.iitkgp.ac.in/~aritrah/course/lab/PDS/Autumn2019/

13-Sep-2019

Table of Contents

Characters

Declaration and Initialization

```
char ch = 'a';   OR   char ch; ch = 'a';
```

ASCII Values of Characters

Every character has an integer ASCII value and you can get that by printing it in integer format.

```
char ch = 'a';
printf(''%d'',ch);
   // prints ASCII value (97) of 'a'
```

Let us not memorize the ASCII values (of a-z, A-Z and 0-9). It can easily be assigned to any integer and can be found/operated. Moreover, integers and characters are inter-operable.

```
int x = 'A';
printf(''%c'',x+3);
   // prints the character 'D'
```

Character manipulations

Example: Simple text encryption

Caesar cipher is a simple technique of encryption of plain text byreplacing every character in the plain text by a character fixed number of positions down the list of the alphabet. The last characters are folded back to the beginning. The numerical digits and all other characters will remain unchanged.

Shift: 5		Shift: 2	
Original	Encrypted	Original	Encrypted
'A'	'F'	'a'	'c'
'B'	'G'	'b'	'd'
⋮	⋮	⋮	⋮
'Y'	'D'	'y'	'a'
'Z'	'E'	'z'	'b'

Let us program to read a text stream and will encrypt the English alphabets, [A - Z] and [a - z], using Caesar cipher. The value of shift should be within $1 - 10$ and will be decided by the `rand()` function.

C-Program: Simple text encryption

```c
#include <stdio.h>
#include <stdlib.h> // for rand()
#include <ctype.h> // for isalpha()
int main()
{
  char c, shift;
  // generating random shift
  shift = (char)(rand()%10 + 1);
  while((c = getchar()) != EOF) {
    if(isalpha(c)) { // checking for alphabets
      if(isupper(c)) // upper-case alphabet
        putchar((c-'A'+shift)%26+'A');
      else // lower-case alphabet
        putchar((c-'a'+shift)%26+'a');
    }
    else putchar(c); // other characters unchanged
  }
  putchar('\n');
  return 0;
}
```

Table of Contents

Strings

In C, a string is defined to be a null-terminated character array. The null character '\0' is used to indicate the end of the string.

```
int main ()
{
   char greet[3]={'H','i','\0'};
   printf("Greeting message: %s\n",greet);
   return 0;
}
```

Variation in initialization

```
char c[]="abcd";
char c[5]="abcd";
char c[]={'a','b','c','d','\0'};
char c[5]={'a','b','c','d','\0'};
```

Reading string from terminal

```
#include <stdio.h>
int main(){
    char name[20];
    printf("Enter name: ");
    scanf("%s",name);
    printf("Your name is %s.",name);
    return 0;
}
```

Enter name: Dennis Ritchie
Your name is Dennis.

- scanf() function takes only string before the white space.

Reading a line of text

```
int main(){
    char name[30],ch;
    int i=0;
    printf("Enter name: ");
    while(ch!='\n')
    {// terminates if user hit enter
        ch=getchar();
        name[i]=ch;
        i++;
    }// inserting null character at end
    name[i]='\0';
    printf("Name: %s",name);
    return 0;
}
```

Better method

```
int main(){
    char name[30];
    printf("Enter name: ");
    gets(name);
    //Function to read string from user.
    printf("Name: ");
    puts(name);
    //Function to display string.
    return 0;
}
```

Enter name: Dennis Ritchie
Name: Dennis Ritchie

Passing Strings to Functions

```
void Display(char ch[]);
int main(){
    char c[50];
    printf("Enter string: ");
    gets(c);
    Display(c);
    // Passing string c to function.
    return 0;
}
void Display(char ch[]){
    printf("String Output: ");
    puts(ch);
}
```

Library functions

```
#include <stdio.h>
#include <string.h>
int main ()
{
    char str1[12] = "Hello";
    char str2[12] = "World";
    char str3[12];
    int  len ;
    strcpy(str3, str1);
    printf("strcpy(str3,str1): %s\n",str3);
    strcat( str1, str2);
    printf("strcat(str1,str2): %s\n",str1);
    len = strlen(str1);
    printf("strlen(str1) :  %d\n", len );
    return 0;
}
```

Result

strcpy(str3, str1) : Hello
strcat(str1, str2): HelloWorld
strlen(str1) : 10

- do not forget to include string.h

A bit more about string manipulation

- `int strcmp (char s[ ], char t[ ])`:
  Returns 0 if the two strings are identical, a negative value if s is lexicographically smaller than t (i.e., if s comes before t in the standard dictionary order), and a positive value if s is lexicographically larger than t. Comparison is done with respect to ASCII values (A - 65, a - 95)
- `int strlen (char s[ ])`:
  Returns the length (the number of characters before the first null character) of the string s.

Thank You