CSE 115L: Programming Language I Lab (Section: 06)

Spring 2020

Lab-09 (Strings)

Strings: Strings are actually one-dimensional arrays of characters terminated by a null character '\0'.

Declaration & Initialization of strings

Strings are declared in C in a similar manner as arrays. Only difference is that, strings are of char type:

char s[5];

In C strings can be initialized in many ways:

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

a	b	c	d	\0

When, compiler encounters strings, it appends null character at the end of string

(String declaration, input and output)

```
#include<stdio.h>

int main(void)
{
	char str[10];
	char name[20];
	int i;
	//Taking Inputs with Loop
	for(i=0; i<5; i++)
	{
	fflush(stdin);
	printf("Enter character:");
	scanf("%c",&str[i]);
	}
	//printf("%s",str);
	//output using loop
	for(i=0; i<10; i++)
	{
	printf("%c", str[i]);
	}
	//Taking Inputs without loops
	printf("\nEnter string:");
	scanf("%s", name);
	printf("%s", name);
	fflush(stdin);
	//Taking string with space in between using gets &
puts
	printf("\nEnter string2:");
	gets(name);
	//fgets(name, 20, stdin);
	puts(name);
	printf("%s",name);

	return 0;
}
```

Example: Passing string to a function

```
#include <stdio.h>
void displayString(char str[]);

int main(void)
{
        char str[50];
        printf("Enter string: ");
        fgets(str, sizeof(str), stdin);
        displayString(str);     // Passing string to a function.
        return 0;
}
void displayString(char str[])
{
        printf("String Output: ");
        puts(str);
}
```

Example: Strings and Pointers

```
#include <stdio.h>

int main(void) {
  char name[] = "Harry Potter";
  char *namePtr;

  namePtr = name;
  printf("%c", *namePtr);       // Output: H
  printf("%c", *(namePtr+1));   // Output: a
  printf("%c", *(namePtr+7));   // Output: o

  printf ("\n");

  printf("%c", *name);          // Output: H
  printf("%c", *(name+1));   // Output: a
  printf("%c", *(name+7));   // Output: o
  return 0;
}
```

C supports a large number of string handling functions in the standard library "string.h".

strcpy(dest, src): Copies string src into string dest. **strcat(dest, src):** Concatenates string src onto the end of string dest. **strlen(s)**: Returns the length of string s.	**strcmp(dest, src):** Returns 0 if dest and src are the same; less than 0 if dest<src; greater than 0 if dest>src. **strchr(s1, ch):** Returns a pointer to the first occurrence of character ch in string s1, , or NULL if the character is not found. **strstr(dest, src):** Returns a pointer to the first occurrence of string s2 in string s1.

Example: strlen(str), strcat(str1,str2) & strcpy(str1,str2) function in C

```
#include<stdio.h>
#include<string.h>
int main()
{
        char str1[10],str2[10],str3[20];
        int len;
        printf("Enter String 1:");
        gets(str1);
        printf("Enter String 2:");
        gets(str2);
        len=strlen(str1);
        printf("The length of the string 1 is: %d\n", len);
        strcat(str1,str2);
        printf("%s\n",str1);
        strcpy(str3,str1);
        printf("%s",str3);
        return 0;
}
```

Problems:

1. Write a program to compare two strings without using C library function.

```
Enter first strings :abc
Enter Second strings :abc
Strings are equal
```

2. Check whether an input string is palindrome or not. A string is a palindrome if it remains the same after you reverse it.
   For example, “racecar”, “level”, “12321”, “madam” etc.

   Sample:
   Enter a string: racecar

It's a palindrome