CSE101-Lec#24

- Some other functions from string handling library

Outline

- String Conversion Functions
- Character arithmetic

String Conversion Functions

- String Conversion functions
  - In `<stdlib.h>` (general utilities library)
- Convert strings of digits to **integer** and **floating-point values and vice versa**

Function prototype	Function description
`double atof( const char *nPtr );`	Converts the string nPtr to double.
`int atoi( const char *nPtr );`	Converts the string nPtr to int.
`long atol( const char *nPtr );`	Converts the string nPtr to long int.
char * itoa(int value, char *str,int base);	Converts the integer value to string

atof()

Function atof

- Function **atof converts its argument—a string that represents a floating-point** number—to a double value.
- The function returns the double value.
- If the converted value cannot be represented—for example, if the first character of the string is a letter—the behavior of function atof is undefined.

atof()-Program example

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    double d;
    d=atof("99.23");
    printf("%.2lf",d);
    return 0;
}
```

atoi()

Function atoi

- Function **atoi converts its argument—a string of digits that represents an integer—** to an int value.
- The function returns the int value.
- If the converted value cannot be represented, the behavior of function atoi is undefined.

atoi()-Program example

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char x[]="99";
    int i;
    i=atoi(x);
    printf("%d",i);
    return 0;
}
```

atol()

Function atol

- Function **atol converts its argument—a string of digits representing a long integer—** to a long value.
- The function returns the long value.
- If the converted value cannot be represented, the behavior of function atol is undefined.
- If int and long are both stored in 4 bytes, function atoi and function atol work identically.

atol()-Program example

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
	long int i;
	char x[]="10000000";
	i=atol(x);
	printf("%ld",i);
	return 0;
}
```

itoa()

- itoa () function in C language converts int data type to string data type. Syntax for this function is given below.
- char * itoa (int value, char * str, int base);
- Base values could be: 2, 8, 10,16(Depending upon the number system)

itoa()-Program example

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
        int a=123;
        char str[100];
        itoa(a,str,2);
        printf("\n Binary value:%s",str);
        itoa(a,str,10);
        printf("\n Decimal value:%s",str);
        itoa(a,str,16);
        printf("\n Hexadecimal value:%s",str);
        itoa(a,str,8);
        printf("\n Octal value:%s",str);
        return 0;
}
```

Converting from float to string

- There is no inbuilt function for this type of conversion, so we need to write the explicit code for the same,
- In the following ftoa1() is a user defined function which is converting floating type value to string
- Example:

```
#include<stdio.h>
void ftoa1(float f1,char s[])
{
sprintf(s,"%f",f1);// s is array, %f format specifier and f1 is float variable
}
int main()
{
char str[20];
float f=12.340000;
ftoa1(f,str);//it means convert float value f to string and store it in str// function call//
printf("\n%s",str);
return 0;
}
```

Character arithmetic

- To perform increment , decrement, addition subtraction operations on the characters.
- These operations work on the **ASCII** value of characters.
- Starting from ASCII value of ‘a’ = 97 to the ASCII value of ‘z’ = 122

Increment

- To display next char value

```
int main()
{
char x = 'a' + 1;
printf("%c", x); // Display Result = 'b‘
printf("%c", ++x); // Display Result = ‘c‘

}
```

Decrement

- To display previous char value

```
int main()
{
char x = ‘b' - 1;
printf("%c", x); // Display Result = ‘a‘
}
```

Addition

- Adding two ASCII values

```
int main()
{
  char x =  'a‛ + ‘c’;
  printf("%c", x); /* Display Result = - ( addition of
   ASCII of a and c is 196) */
}
```

Subtraction

- Subtracting two ASCII values

```
int main()
{
  char x =  'z' - 'a';
  printf("%c",x); /* Display Result = ↓ (difference
   between ASCII of z and a ) */
}
```