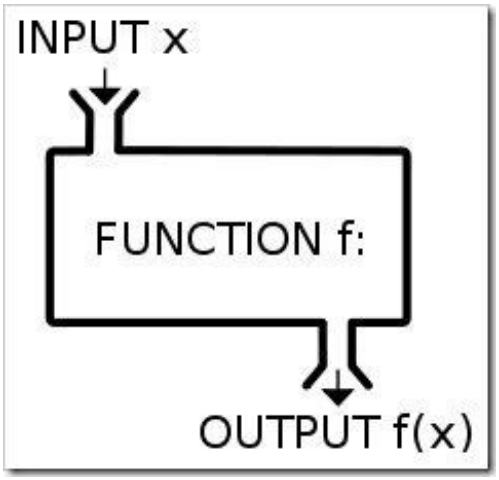
Assignment#4

Functions in C



In this section we will explore some of the **built in functions** in C. Examples you've seen so far:

- printing to a screen, `printf()`
- collecting input from the user, `fscanf()`
- calculating the square root, `sqrt()`

There are several more **built-in** function in C you should become familiar with. Many tasks in C can be completed without out having to program at all. Becoming familiar with C's **built in** functions can save you hours of time.

Future lessons you *may* be asked to create a computer program that can replicate these tasks, but normally, **you are free to use these built-in functions as a short cut.**

Each built-in function in C is contained in a "**library**". You must call the appropriate **library** at the beginning of you C program to use a function.

Example:

the following code calls the **function** `sqrt()` using the **library** `math.h`

```
#include <stdio.h>
#include <math.h>      <-- Library

int main()

{
 float num, root;
 printf("Enter a number: ");
 scanf("%f", &num);
 root = sqrt(num);     <-- Function
 printf("Square root of %.2f = %.2f", num, root);
 return 0;
}
```

Below is a brief list of some common **libraries** and **functions** you should become familiar with. Please note that each function uses specific **parameters** (needed input). To find out how each function works, you can google the required functions and see examples of their implementation.

`math.h` Library (Math Stuff)

Function	Description
floor ( )	This function returns the nearest integer which is less than or equal to the argument passed to this function.
round ( )	This function returns the nearest integer value of the float/double/long double argument passed to this function. If decimal value is from ".1 to .5", it returns integer value less than the argument. If decimal value is from ".6 to .9", it returns the integer value greater than the argument.
ceil ( )	This function returns nearest integer value which is greater than or equal to the argument passed to this function.
sin ( )	This function is used to calculate sine value.
cos ( )	This function is used to calculate cosine.
exp ( )	This function is used to calculate the exponential "e" to the x^{th} power. e^x
tan ( )	This function is used to calculate tangent.
sqrt ( )	This function is used to find square root of the argument passed to this function.
pow ( )	This is used to find the power of the given number.
trunc ( )	This function truncates the decimal value from floating point value and returns integer value.

`stdlib.h` **Library** (Standard common stuff)

rand()	This function returns the random integer numbers
abs()	This function returns the absolute value of an integer . The absolute value of a number is always positive. Only integer values are supported in C.

Exercise 4.1 (you try then check – answers on next page)

a) Use **Floor** and **Ceiling** functions to find the floor and ceiling of a float variable

Sample Input: 34.67

Output : floor 34 Ceiling 35

b) Use the **power** and **square root** functions to write a program that find the hypotenuse of the right angle triangle give the adjacent and opposite sides. Use floats.

Sample input: 3, 4

Sample output: 5

c) Use the `rand()` function to create a program that generates **pairs** of random numbers from 1 to 6 (like rolling two separate dice). Then have the program continue to roll until one pair is two 6's. The program should print out the random pairs as they are generated, stop when it gets two 6's, and announce to the user how many rolls it took.

Bonus: prompt the user to start the rolling.

Super Bonus: have a half second delay between each roll.

Sample Solutions EXERCISE 4.1

a)

```
int main()
{
   float x;
   printf("\nInput Number: ");
   scanf("%f",&x);
   printf("Floor: %.0f\n", floor(x));
   printf("Ceiling: %.0f", ceil(x));
   return 0;
}
```

b)

```
int main()
{
   float x,y,tot,tot1;
   printf("\nInput Number: ");
   scanf("%f",&x);
   printf("\nInput Second Number: ");
   scanf("%f",&y);
   tot=pow(x,2)+pow(y,2);
   printf("%.2f",sqrt(tot));

   return 0;
}
```

c)

```
#include <stdio.h>
#include <time.h>

int main()
{
    int d1, d2, count;
    srand(time(NULL));

    printf("Press ENTER to begin:");
    while( getchar() != '\n' );

    do
    {
        d1 = rand()%6 + 1;
        d2 = rand()%6 + 1;

        printf("\nRoll %d: (%d, %d)", count, d1, d2);
        count++;

    } while((d1!=6) || (d2!=6));
    printf("\nTwo 6's were rolled in %d rolls.", count);
    return 0;
```

Alterative solution for c)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int
main ()
{
  srand (time (NULL));
  int sum, x, y;
  char d;
  sum = 0;
  int check;
  printf ("\nPress Enter To Start Program\n");
  scanf("%c",&d);
  check = 0;
  while (check == 0)
    {
      sum = sum + 1;
      y = rand () % 6 + 1;
      x = rand () % 6 + 1;
      printf ("%d,", x);
      printf ("%d\n", y);
      if (x == 6 && y == 6)
           {
           printf ("\n\nDice 1 & 2 are #6 & It Took %d Tries", sum);
           check = 1;

           }
    }
  return 0;
}
```

Library for manipulating "**strings**" (word stuff)
<string.h>

Strings are groups of **characters** that can be stored as **one** entity….basically a string is a **word**. You can also think of strings as an **ARRAY of characters**. Strings are very useful for manipulating and sorting data that is based on **words**. Some of the functions **listed on the next page** are **not available on our online compiler**. But it is helpful to know they exist.

How to use strings:

Ways to **declare** strings:

```
char c[] = "abcd";
char c[50] = "abcd";
char c[] = {'a', 'b', 'c', 'd', '\0'}
char c[5] = {'a', 'b', 'c', 'd', '\0'};
```

```
#include <stdio.h>
int main ()
{
   char c[6] = "Hello";
   printf("Greeting message: %s\n", greeting );
   return 0;
}
```

This will output:

```
Greeting message: Hello
```

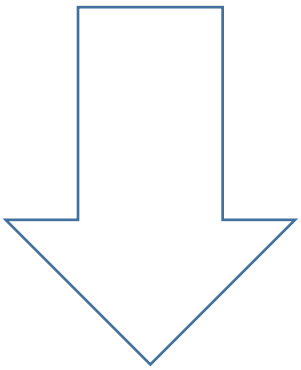
Note that the specifier for a string is `%s`

Library for manipulating **"strings"** (word stuff)
`<string.h>`

These are function that can manipulate strings

strlen ( )	Gives the length of str1
strlwr ( )	Converts string to lowercase
strupr ( )	Converts string to uppercase
strrev ( )	Reverses the given string
strcmp ( )	Compares two strings character by character to see if they are equal
strcmpi ( )	Same as strcmp() function. But, "A" and "a" are treated as same.
strchr ( )	Returns pointer to first occurrence of char in str1
strrchr ( )	last occurrence of given character in a string is found
strstr ( )	Returns pointer to first occurrence of str2 in str1
strrstr ( )	Returns pointer to last occurrence of str2 in str1
strdup ( )	Duplicates the string

Try the following exercises to help you familiarize yourself with managing strings:



Exercise 4.2 (copy and paste)

The **`strcmp()`** compares two strings character by character. If the first character of two strings are equal, next character of two strings are compared. This continues until the corresponding characters of two strings are different or a null character or the end of the string is reached. **The output is the difference in ASCII of the character values that are different**

Return Values for `strcmp()`

Return Value	Remarks
0	if both strings are identical (equal)
negative	if the ASCII value of first unmatched character is less than second.
positive integer	if the ASCII value of first unmatched character is greater than second.

```
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[] = "abcd", str2[] = "abed", str3[] = "abcd";
    int result;

    result = strcmp(str1, str2);
    printf("strcmp(str1, str2) = %d\n", result);

    result = strcmp(str1, str3);
    printf("strcmp(str1, str3) = %d\n", result);
    return 0;
}
```

Copy and paste the code above into OnlineGDB. Notice how strings are created (declared) and see if the **Return Values** that get printed make sense.

Exercise 4.3 (you try then check)

a) Use string functions to create the programs that *outputs* the length of a string.

Sample input: Enter a work: Bicycle
Sample input: This word is 7 letter long

b) Use the strcmp() function above to wirte a program that is able to determine the alphabetical order of 3 words. Each work must start with a different letter.

Sample Input:
Enter three words(each starting with a different letter):
Cookie
Ball
Tree

Sample Output:
These words in alphabetical order are:
Cookie
Ball
Tree