Paper 102: Programming & Problem solving through C

Lecture-10:Unit-II
Functions

INTRODUCTION

1. A number of statements group into a single logical unit is referred to as a **function**
2. Large programs raises level of complexity, so we must design and develop a program in pieces, or modules
3. In C these modules are called *functions*
4. A C program is a collection of functions
5. The function main() in a C program is executed first.
6. The other functions are executed when the function main calls them (or transfer the control to them)

Function Declaration/Prototype

Functions should be declared before it is being used.

Syntax:

Return- data- type **function-name(***data type arg1, datatype arg2,..***);**

Example:

int **addnumbers(***int a, int b***);**

int **subnumbers(***int,int***);**

A function can return zero or one value

Function Definition

Function definition means specifying the task of a function by writing it as a block of statements.

Syntax:

Return- data- type **function-name(***data type arg1, datatype arg2,..***)**

```
{
        statement 1;
        statement 2;
        …
        return(exp);
}
```

1. *Return- data- type* specifies what data type will be returned by the function (**int**, **double**, etc., if any; if no value is returned, use **void**)
2. ***Function-name*** meets the rules for variable names; should be descriptive
3. *argument list* contains list of data types (and optional variable names) the function expects to receive; the calling function will "pass" these values as *arguments*, the called function will receive them into its *parameters*; if the function accepts no arguments, use **void**

Function Definition: Example

```
 int addnumbers(int a, int b)
{
        int sum;
        sum=a + b;
        return(sum);
}
```

```
 int addnumbers(int a, int b)
{
        return(a+b);
}
```

Function calls

A function can be called in two ways

1.If it does not return any value then it can be called by simply giving the following statement

Function-name(list of arguments);

Sum(2,5);

2.If it return a value then it can be used in any expression

Variable-name=function-name(list of arguments);

sum=sum + i/fact(i);

It can also be included as an argument in another function like

printf("The sum of two numbers is =%d",addnumbers(2,3));

addfunc.c

```
#include<stdio.h>
 int addnumbers(int,int);
 void main()
{
    int a,b;
    int sum;
    printf("enter the values of two numbers:");
    scanf("%d%d",&a,&b);
    sum=addnumbers(a,b);
    printf("The sum of %d and %d is = %d",a,b,sum);
}
int addnumbers(int a,int b)
{
    int addition;
    addition=a + b;
    return(addition);
}
```

Function without arguments and without return value

```
#include<stdio.h>
void addnumbers(void);
 void main()
{
    printf(“ Program for adding two numbers using
    functions \n”);
    addnumbers();
    printf(“\n we are back in main function”);
}
void addnumbers()
{
    int a,b,sum;
    printf(“enter the values of two numbers:”);
    scanf(“%d%d”,&a,&b);
    sum=a + b;
    printf(“The sum of %d and %d is = %d”,a,b,sum);
}
```

- One function can also calls another function i.e., function calls can be nested.
- A function can also calls itself. These types of functions are called recursion.

```
void main()
{
        calculate();
}
void calculate()
{
        add();
        mul();
}
void add()
{
}
void mul()
{
}
```

```
int fact(int a)
{
        f=a*fact(a-1);
}
```

Library functions

- Each standard library has a corresponding *header file*
- Header files contain function prototypes and other definitions
- Can also create library of your own functions
- There's a list of available functions in many Libraries
- stdio.h, math.h, string.h, conio.h etc…

clrscr() in conio.h

Clears text mode window

Declaration: void clrscr(void);

Remarks:

clrscr clears the current text window and places the cursor in the upper left-hand corner (at position 1,1).

Return Value: None

Math.h

abs	acos	acosl	asin	asinl	
atan	atanl	atan2	atan2l	atof	_atold
cabs	cabsl	ceil	ceill	**cos**	cosl
cosh	coshl	exp	expl	fabs	fabsl
floor	floorl	fmod	fmodl	frexp	frexpl
hypot	hypotl	labs	ldexp	ldexpl	
log	logl	log10	log101	matherr	_matherrl
modf	modfl	poly	polyl	**pow**	powl
pow10	pow10l	**sin**	sinl	sinh	sinhl
sqrt	sqrtl	**tan**	tanl	tanh	tanhl

stdio.h

clearerr	fclose	fcloseall	fdopen	feof	ferror
fflush	fgetc	fgetchar	fgetpos	fgets	fileno
flushall	fopen	fprintf	fputc	fputchar	fputs
fread	freopen	fscanf	fseek	fsetpos	ftell
fwrite	**getc**	**getchar**	gets	getw	perror
printf	**putc**	**putchar**	puts	putw	remove
rename	rewind	rmtmp	**scanf**	setbuf	setvbuf
sprintf	sscanf	strerror	_strerror	tempnam	tmpfile
tmpnam	ungetc	unlink	vfprintf	vfscanf	vprintf
vscanf	vsprintf	vsscanf			

Class Assignment

1. Write a program to simulate a basic calculator to perform the four basic operations (+,-,/,*). Use functions for each of the operations. The interface should be menu driven.