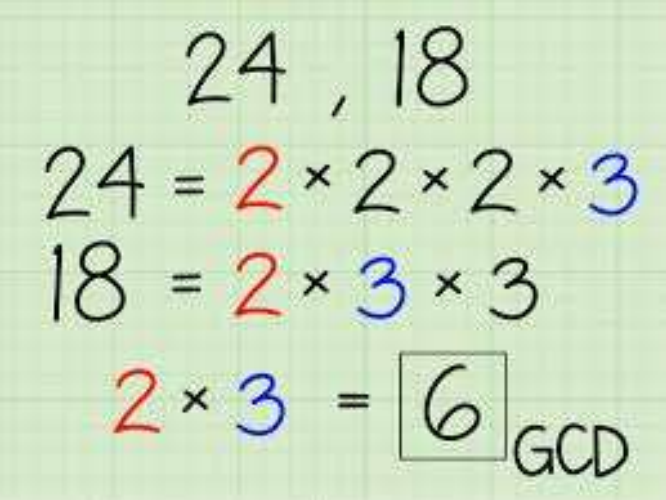
USER DEFINED FUNCTIONS

How to design and implement?

ALGORITHM



"Middle-school" procedure

Step 1 Find the prime factorisation of m

Step 2 Find the prime factorisation of n

Step 3 Find all the common prime factors

Step 4 Compute the product of all the common prime factors and return it as $\gcd(m,n)$

MAJOR TASK

MAJOR TASK

- We Need an algorithm for prime factorization to make middle-school procedure into an algorithm.

- As well as need an algorithm to find common prime factors.

MAIN FUNCTION

"Middle-school" procedure

Step 1 Find the prime factorisation of m

Step 2 Find the prime factorisation of n

Step 3 Find all the common prime factors

Step 4 Compute the product of all the common prime factors and return it as $\gcd(m,n)$

SUB FUNCTION 1

Prime Factorisation

Input: Integer $x \geq 2$,

Output: List F of prime factors of x

```
P ← Sieve(x)
while n > 1 do
   while n mod P[i] = 0 do
      F ← F + P[i]
      x ← x / P[i]
      i ← i + 1
```

Sieve of Eratosthenes

Input: Integer $x \geq 2$

Output: List of primes less than or equal to x

```
for p ← 2 to x do  A[p] ← p
for p ← 2 to x do
   if A[p] ≠ 0
         j ← p·p
      while j ≤ x  do
            A[j] ← 0
            j ← j + p
```

SUB FUNCTION 2

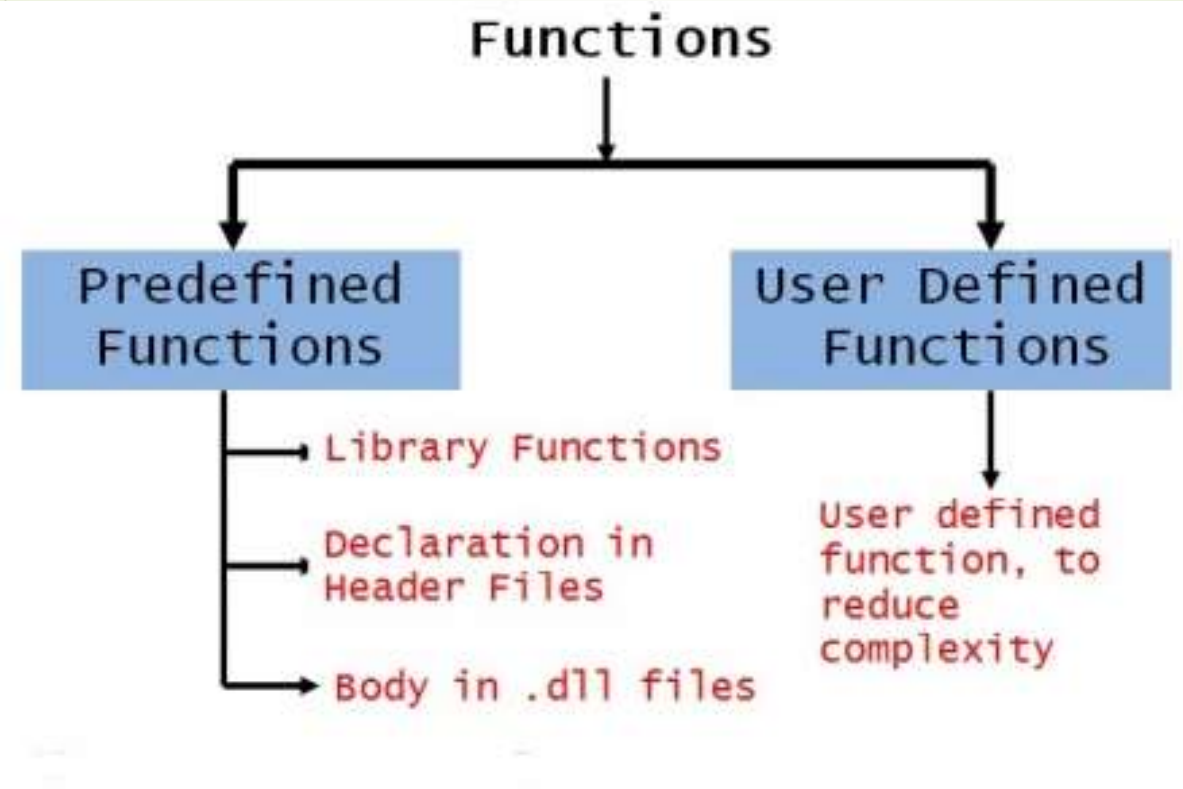
FUNCTIONS-PURPOSE

- Functions break large computing tasks into smaller ones.
- Avoids repetition of Codes.
- Every C program has at least one function.

```
main( )
{
    printf ( "\nI am in main" ) ;
    argentina( )
    {
        printf ( "\nI am in argentina" ) ;
    }
}
```

A function is a set of statements that take inputs, do some specific computation and produces output.

TYPES OF FUNCTIONS



Elements of the functions

Function prototype / function declaration

Function Call

Function definition

Actual Arguments

Formal Arguments

Return Value

Function name

Elements of Functions

1.Function prototype / function declaration

Syntax :

return_type function_name(arguments);

Eg: int add(int, int);

or

int add(int num1, int num2);

Elements of Functions

2. Function definition

Syntax :

```
return_type function_name(formal arguments)
{
    ----- function body ----
}
```

Eg:

```
int add(int x, int y)
{
    ------
    return c;
}
```

Elements of Functions

3. Function call

Syntax:

function_name(Actual Arguments);

Eg : add(a,b);

or

int result = add(a, b);

Elements of Functions

4. Actual arguments

The arguments used in the function call

5. Formal arguments

The arguments used in the function definition

6. Return Value

Return statement used to return the values back to the calling function.

```
/* Function to add two numbers */
void add(int,int);                    /* function declaration */
#include<stdio.h>
int main()
{
    int a,b,d;
    printf("Enter a,b:");
    scanf("%d%d",&a,&b);
    d= add(a,b);          /* function call   , a, b – actual arguments */
    printf("%d",d);
}

int add(int x,int y)        /* function definition x, y – formal arguments */
{
    int c;
    c=x+y;
    return c;
}
```

Four types of function prototypes

1. Function with no argument and no return value
2. Function with no argument and with return value
3. Function with argument and no return value
4. Function with argument and with return value

Function Prototypes

1. Function with no argument and no return value

- In this prototype, no data transfer takes place between the calling function and the called function. (ie from function call to function definition).
- The function is only executed, does not return any value to the calling program.

Example

```
void add();
main()
{
        ------
        ------
        add();
        ------
}
```

```
add()
{
        ------
        ------
        ------
        ------

}
```

```
//Program for no argument and no return value
#include<stdio.h>
void add();                    // function declaration
main()
{
    add();                     // function call
}

void add()                     // function definition
{
    int a, b, c;
    a=10,b=20;
    c=a+b;
    printf("sum is %d",c);
}
```

Output:- sum is 30

Function Prototypes

2. Function with no argument and with return value

- In this prototype, the calling program cannot pass any arguments to the called program.
- But the called/invoked program may send some value return to the calling program.

Example

```
int add();
main ()
{
        ------
        ------
        sum=add( );
        ------
}
```

```
int add( )
{
        ------
        ------
        ------
return(c);
}
```

```
//Program for no argument and with return value
#include<stdio.h>
int add();                    // function declaration
int main()
{
    int sum;
    sum=add();                          // function call
    printf("sum is %d",sum);
}
int add()                     // function definition
{
    int a, b, c;
    a=10,b=20;
    c=a+b;
    return c ;
}
```

Output:- sum is 30

Function Prototypes

3. Function with argument and no return value

- In this prototype, data is transferred from calling function to called function.
- i.e called function receives some data from the calling function and does not return any values to the calling function.

Example

```
void add(int,int);
main()
{
        ------
        ------
        add(a,b );
        ------
}
```

```
add( x,y)
{
        ------
        ------
        ------
        ------
}
```

```
//Program for argument and no return value
#include<stdio.h>
void add(int,int);          // function declaration
int main()
{
    int a=10,b=20;
    add(a,b);               // function call
 }

void add(int a, int b)      // function definition
{
    int c;
    c=a+b;
    printf("sum is %d",c);
}
```

Output:- sum is 30

Function Prototypes

4. Function with argument and with return value

- In this prototype, the data is transferred between the calling function and called function.
- i.e the called function receives some data from the calling function and return a value to the calling function.

Example

```
int add(int,int);
main()
{
        ------
        ------
        sum=add(a,b );
        ------
}
```

```
int add( x,y)
{
        ------
        ------
        ------
        return (c);
}
```

```
//Program for argument and with return value
#include<stdio.h>
int add(int,int);          // function declaration
int  main()
{
    int a=10,b=20,c;
    c = add(a, b);                    // function call
    printf("sum is %d",c);
}

int add(int a,int b)          // function definition
{
    int c;
    c=a+b;
    return c ;
}
```

Output:- sum is 30

Sample Programs

1. Write a program to find the GCD of a given numbers

Test Data :

- Input :

  98 56

- Expected Output :

  GCD : 14

Solution

```
#include <stdio.h>
 int findgcd(int a, int b)
{
    int r;
    while( b!=0 ){
        r = a % b;
        a = b;
        b = r;
    }
    return a;
}
```

```
int main()
{
    int a = 98, b = 56;
    int gcd;
    gcd = findgcd(a,b);
    printf("%d",gcd);
    return 0;
}
```

Sample Programs

2. Write a program in C to get the factors of a given number.

Test Data :

- Enter a number :

  10

- Expected Output :

  1 2 5 10

Solution

```
#include <stdio.h>
void FindFactors(int num){
    int i;
    printf("Factors of %d are: ", num);
    for (i = 1; i <= num; ++i) {
        if (num % i == 0) {
            printf("%d ", i);
        }
    }
}
```

```
int main() {
    int num;
    printf("Enter a number : ");
    scanf("%d", &num);
    FindFactors(num);
    return 0;
}
```

Sample Programs

3. Write a program in C to convert decimal number to binary number using the function

Test Data :

- Enter decimal number :

  65

- Expected Output :

  The Binary value is : 1000001

Solution

```
long toBin(int);
int main()
{
        long bno;
        int dno;
        printf(" Enter any decimal number : ");
        scanf("%d",&dno);
        bno = toBin(dno);
        printf("\n The Binary value is :
                                %ld\n\n",bno);
        return 0;
}
```

```
long toBin(int dno)
{
        long bno=0,remainder,f=1;
        while(dno != 0)
        {
                remainder = dno % 2;
                bno = bno + remainder * f;
                f = f * 10;
                dno = dno / 2;
        }
        return bno;
}
```

QUIZ

What is the output of the following program?

```
#include <stdio.h>
void f()
{
        int i = 0;
        ++i;
        printf("%d ", i);
}
int main()
{
        f();
        f();
        f();
        return 0;
}
```

A. 1 1 1
B. 0 0 0
C. 3 2 1
D. 1 2 3

Answer : A

QUIZ

What will be the output of the C program?

```
#include <stdio.h>
int main()
{
        int num = _a_123(4);
        printf("%d\n", --num);
        return 0;
}
int _a_123(int num)
{
        return (num++);
}
```

A. 3
B. Compilation error
C. 4
D. 5

Answer : A

QUIZ

What is the output of this C code?

```
#include <stdio.h>
int main()
{
        void rec();
        printf("1 ");
        rec();
        return 0;
}
void rec()
{
        printf("2 ");
}
```

A. 1 2
B. Compile time error
C. 1 2 1 2
D. Depends on the compiler

Answer : A

QUIZ

What is the output of the program?

```
#include <stdio.h>
int f(int i)
{
      if (i % 2 == 0)
            return 0;
      else
            return 1;
}
int main()
{
      int i = 3;
      i = f(i);
      i = f(i);
      printf("%d", i);
      return 0;
}
```

A. 3
B. 1
C. 0
D. 2

Answer : B

QUIZ

What is the output of the following C program?

```
#include <stdio.h>
void foo(), f();
int main()
{
	f();
	return 0;
}
void foo()
{
	printf("2 ");
}
void f()
{
	printf("1 ");
	foo();
}
```

A. Compiler error as foo() is not declared in main
B. 1 2
C. 2 1
D. Compile time error due to declaration of functions inside main

Answer : B

QUIZ

What is the output of following program?

```
#include <stdio.h>
void func(int x)
{
        x = 40;
}

int main()
{
        int y = 30;
        func(y);
        printf("%d", y);
        return 0;
}
```

A. 40
B. 30
C. Compilation error
D. Runtime error

Answer : B

QUIZ

What is the error in the following program?

```
#include <stdio.h>
int f(int a)
{
      a > 2 ? return(1): return(2);
}
int main()
{
      int b;
      b = f(1);
      printf("%d\n", b);
      return 0;
}
```

A. Error: Return statement cannot be used with conditional operators
B. Error: Prototype declaration
C. Error: Two return statements cannot be used in any function
D. No error

Answer : B

QUIZ

What will happen after compiling and running following code?

```
#include <stdio.h>
int main()
{
        printf("%p", main);
        return 0;
}
```

A. Error
B. Will make an infinite loop.
C. Some address will be printed.
D. None of the mentioned.

Answer : C