FUNCTIONS

Saturday, 23. July 2022 23:59

What is functions ?

- Function is a block of code of C Program.
- Function is a block of code, which has some name for identification.
- A C Program can have any number of functions.

Syntax of function

```
function name ( )
{
    // code
}
```
→ function definition

- Function name must be unique in a program.

FUNCTION DEFINITION VS FUNCTION CALL

```
f1 ( )
{
    printf ("Ashishpuru");
    printf ("I am function");
}
```
⇒ Function definition

```
{
    ==
    f1 ( );
    ==
}
```
⇒ Function call

- Function is a way to implement modularization.
- Modularization is a spliting up of a bigger task into several smaller sub-tasks to reduce the complexity of a problem.
- We can compile a C file without having main ( ) function but cannot run.

TYPES OF FUNCTION

(i) Predefined functions
i.e printf ( ), scanf ( ) etc.

(ii) User defined functions
i.e main ( ) { == } , addition ( ) { == }

NOTE:- keywords are not function
switch ( ), while ( ), sizeof ( ) etc.

FLOW OF PROGRAM

```
main ( )
{
    a ( );
    b ( );
    a ( );
}
a ( )
{
    printf ("Hello");
}
b ( )
{
    printf ("Bye");
    a ( );
}
```

So here first a ( ) will create and take memory and we will see Hello on screen, after execution memory will release then b( ) will execute and then we will see that Bye is printed on screen after Hello like HelloBye so after printf (" ") ther is one more line of code and that code is calling a ( ) function so again Hello will be printed like this HelloByeHello. after calling of b( ) function a( ) will execute and we will see HelloByeHelloHello

BENIFITS OF FUNCTIONS

1. Easy to read
2. Reduce complexity
3. Easy to modify
4. Easy to debug
5. Code reusability
6. Avoids rewriting
7. Better memory utilization

WAYS TO DEFINE FUNCTION

1. Takes nothing, Returns Nothing – TNRN
2. Takes something, Returns Nothing – TSRN
3. Takes nothing, Returns something – TNRS
4. Takes something, Returns something – TSRS

→ Takes Nothing, Returns Nothing

```
#include <stdio.h>
void add ( );            --> Function declaration (must)
int main ( )
{
    add ( );             --> Function call
    return 0;
}
void add ( )
{
    int a,b,c;
    printf ("Enter two numbers");
    scanf ("%d %d", &a, &b);
    c = a+b;
    printf ("Sum is %d", c);
}
```
(void add ( ) { ... } → Function definition)

NOTE: void add ( ) ↳ There is nothing that's why it's take nothing
↓
Here it's written void means empty ∴ it's return nothing

* Function can only made by 3 steps by now.
(①) → Functionname ( ) → ① ② ③

TAKE SOMETHING RETURN NOTHING

```
#include <stdio.h>
void add (int, int);
int main ( )
{
    int a,b;
    printf ("Enter two numbers");
    scanf ("%d %d", &a, &b);
    add (a, b);          --> actual argument
    return 0;
}
void add (int x, int y)  --> Formal argument
{
    int c = x+y;
    printf ("Sum is %d", c);
}
```

TAKE NOTHING RETURN SOMETHING

```
#include <stdio.h>
int add ( );
int main ( )
{
    int a;
    a = add ( );
    printf ("Sum is %d", a);
    return 0;
}
int add ( )
{
    int b,c,d;
    printf ("Enter two numbers");
    scanf ("%d %d", &b, &c);
    d = b+c;
    return d;
}
```

TAKE SOMETHING RETURN SOMETHING

```
#include <stdio.h>
int add (int, int);
int main ( )
{
    int a,b,c;
    printf ("Enter two numbers");
    scanf ("%d %d", &a, &b);
    c = add (a, b);      --> Function call by passing values
    printf ("Sum is %d", c);
    return 0;
}
int add (int x, int y);
{
    int d;
    d = x+y;
    return d;
}
```

SUMMARIZE FUNCTION BY ANSWERING THESE

1. Function call V/S Function Definition V/S Function Declaration
2. Function saves memory → By this way, we can run program of 1 GB in the System having 100 mb RAM.
3. main ( ) is a user defined function
4. Header file V/S Library file
5. when to write void
6. when to write return
7. Return type of main ( )
8. Call by value
9. Function prototype / declaration
10. Can we call main ( ) function ?