Let’s solve it

27

Functions (developer uses C lang)

→ It is set of instructions, having name and purpose. Optionally it may take input and (optionally) may generate output.

As a developer, you may define your own one or more functions in your program, they are called U D F. (User is developer.)

Elements of function

void or any other datatype

data type of Return value (return type)

function_name

is an example of identifier* ~~rules~~ applied

parameters arguments

{ body of the function }

function definition within fn definition Not Allowed.

Calling function

```
     function1(     )
     {
        ————  ——  ~
     }
void function2(     )
     {

     }
```

[illegible]

Single file program using U.D.F.

Function Declaration/Prototype

1
```
void displayline();
```

```
int main()
{
    displayline();
    displayline();
    displayline();
    return(0);
}
```

2 function call

```
void displayline()
{ int i;
  for(i=1;i<=50;i++)
  {
    printf("*");
  }
}
```

3 Function Definition

Result

Importance of User Defined Function

- + Reusability of code
- + Modularity

You can break down the bigger tasks into smaller modules.

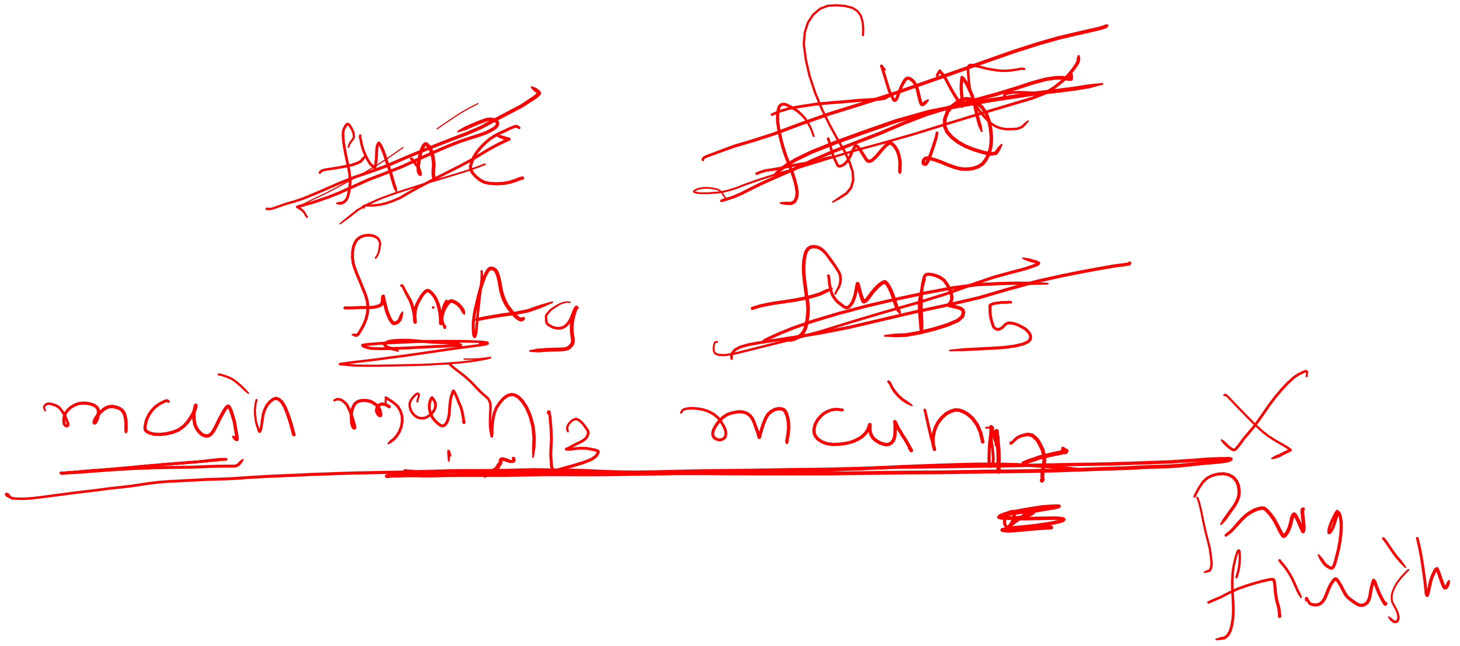
Also, allows ~~team~~ development by team of developers.

module is synonym of function

Top-Down approach (software products or Bottom-Up Approach for library building

Function call stack (LIFO or FILO)

```
int main()
{
   funA();      12
   funB();      17
}

funA()
{
   funC();
}

funB(   )
{
   funD();
   funE();      5
}

funC()
{
}

funD()
{
}

funE()
{
}
```



caller
```
main()
{
     A();
}
```
callee
```
A( )
{
     B();
}
```
caller
```
B( )
{
     C();
}
```

```
C( )
{
  D();
}
return
```

```
D( )
{
  return;
}
```

return

main → A → B → C → D

main A B C

Last In First Out

Data Stack

There will be data stack for variables created locally within one another functions.

Automatic Variables (local variables)