Lecture - 24

Programming in C

functions (Part - 02)

function
↳ Theory < Declare, Define } function

→ Built in : Pre-defined
↳ Header files <stdio.h>

→ User defined : Defined by the Programmer

Syntax to define a function

Rules are same for identifiers.

```
return type  functionName ( _ ) {
    Body of the function
    return value;
}
```

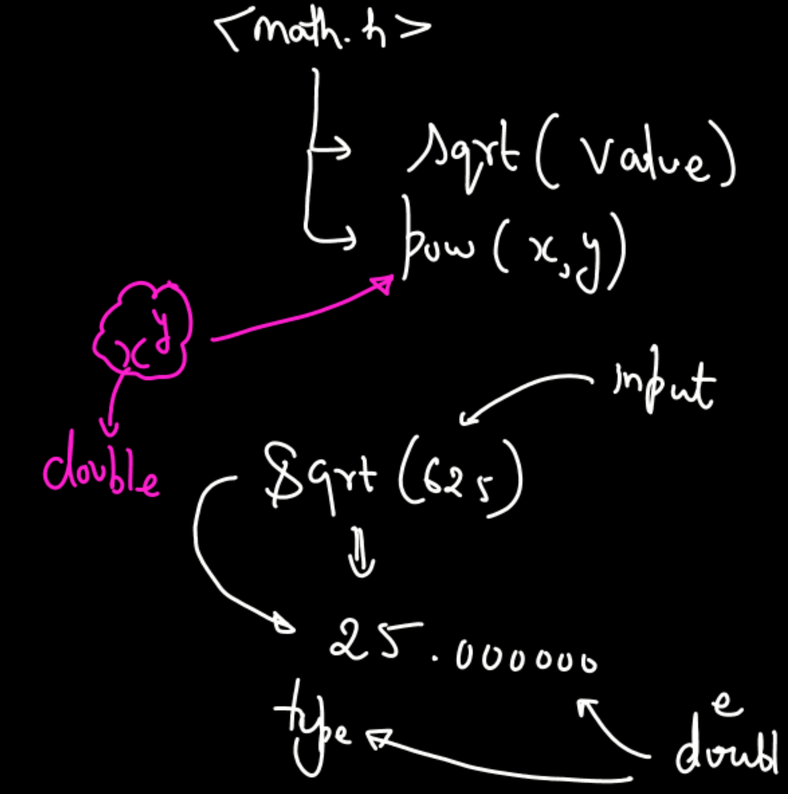
Input of the function (Parameter)

default (int)

Takes place if the function [written at the last]

we have to mention the type and variable name to assign the input value

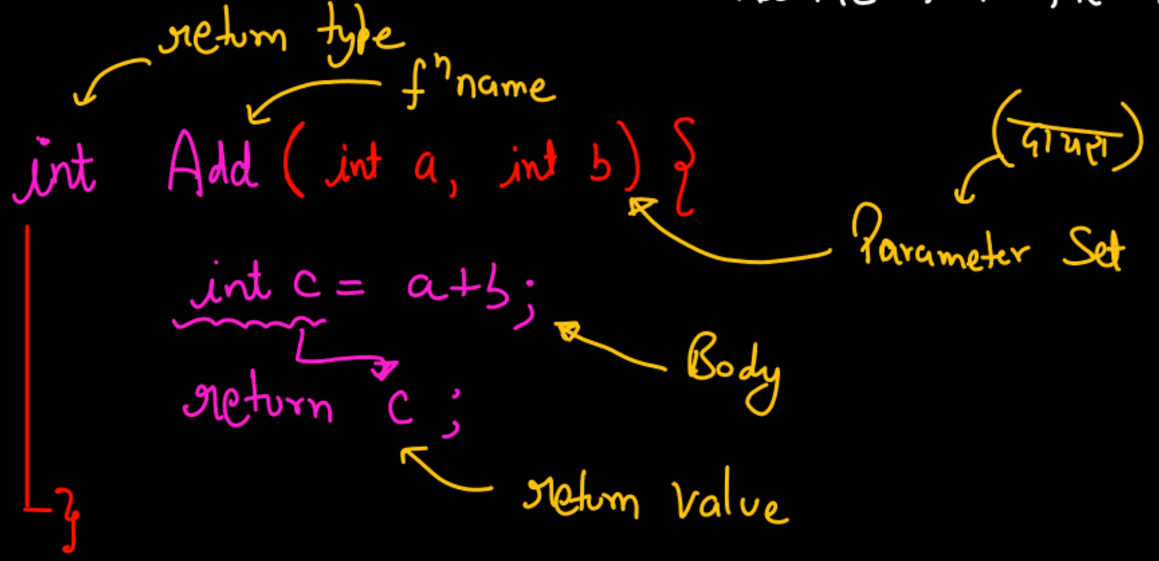
1. Return Type
2. function Name
3. Parameter
4. Return value



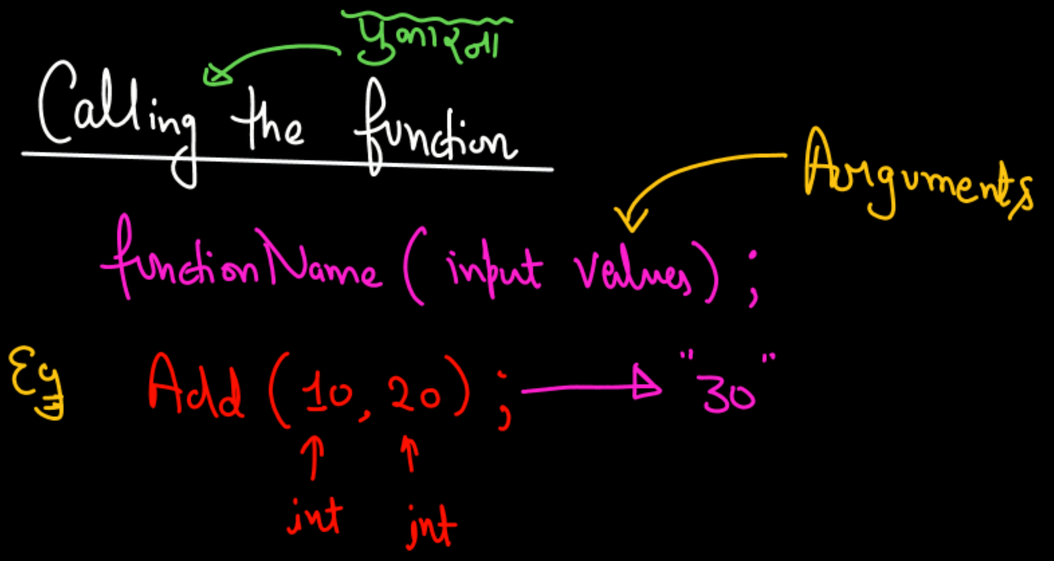
Define a function that takes two numbers and return their sum.

↓
Assume that the numbers are integer

int + int = int



Calling the function



Definition & Declaration of the function :

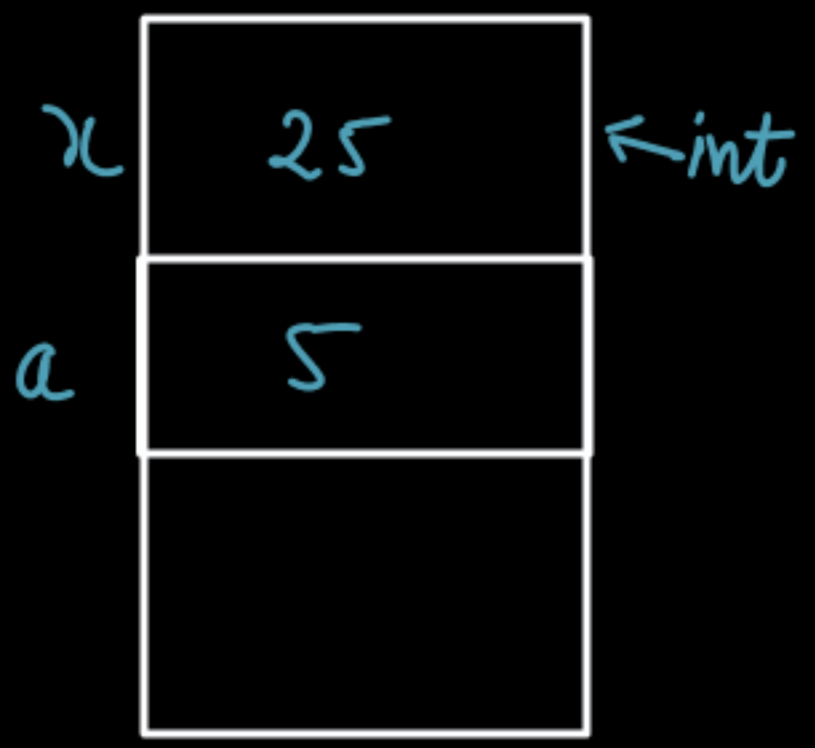
```c
# include <stdio.h>
int square ( int x) ;
int main ( ) {
    int x = square( 5);
    printf( "%d", x) ;
    return 0;
}

int square (int a) {
    return (a * a);
}
```

or → int square (int) ✔

int square (int x) ; ← declare → This function exists / defined

square(5) → 25



int square (int a) { return (a * a); } → define

(a * a) → 5x5 = 25

Load inside the memory

f(x) ⇒ Define

→ After main() ↳ Declare first

→ Before main() ↳ No need to declare

int y = square(100)

y ⇒ 10000

int z = square(25) → 625

z ⇒ 625

Calling the fn

Welcome | function.c

function.c > mul(int, int)

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
//declare
// int mul(int a, int b);
// or
int mul(int,int);

int main(){
    int x = mul(10,20);
    printf("%d", x);
    return 0;
}
//define
int mul(int a , int b){
    int c = a*b;
    return c;
}
```

declaration

a b

10 20

200

fn definition

```
200
PS C:\Use
i Chand\
```

function Prototype : प्रारूप

→ It Provides the details of the function. (name, parameters, argument, return value, Body, limit, Scope etc.)

void → no need to write return

```
int main ( ) {
    f(10);          ← Call
    return 0;
}
```

f(10) → Actual Parameter ⇓ Actual values are passed while calling the function.

```
void f ( int a) {        ← define
    printf("The Square of a is %d", a*a);
}
```

(void replaces int)

int a → formal Parameter → The parameter which set during function definition ⇓ declare

* It is not necessary that function will return a value

formal Parameter ⇐ during Compile time → Source Code . C ⟶ Compiled Program . exe
.obj
└ GCC ↗

Actual Parameters → during Runtime → Compiled Program-exe ⟶ Memory ∵ Load ⇒ Processing

formal Parameter : Void function x (int x, int y) {

}

=

Variables declared during the function definition

Arguments (values)
⇓
formal

Actual Parameter : function x (500, 6000)

Actual values ← Arguments (values)
⇓
Actual argument

int function x(int a, b) {

error

(int a, int b)