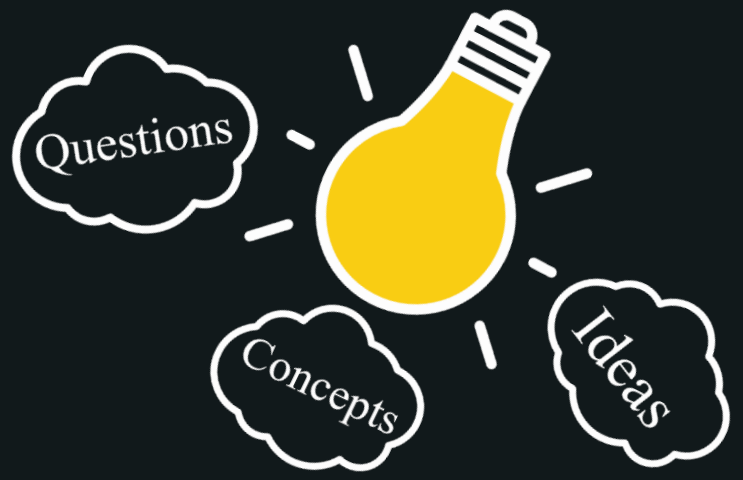


Functions

C++ Programming

VEDINESH

What is a Function ?

A function is a block of code, are used to perform certain task

Input

Put repea

MILK

Output

User Defined

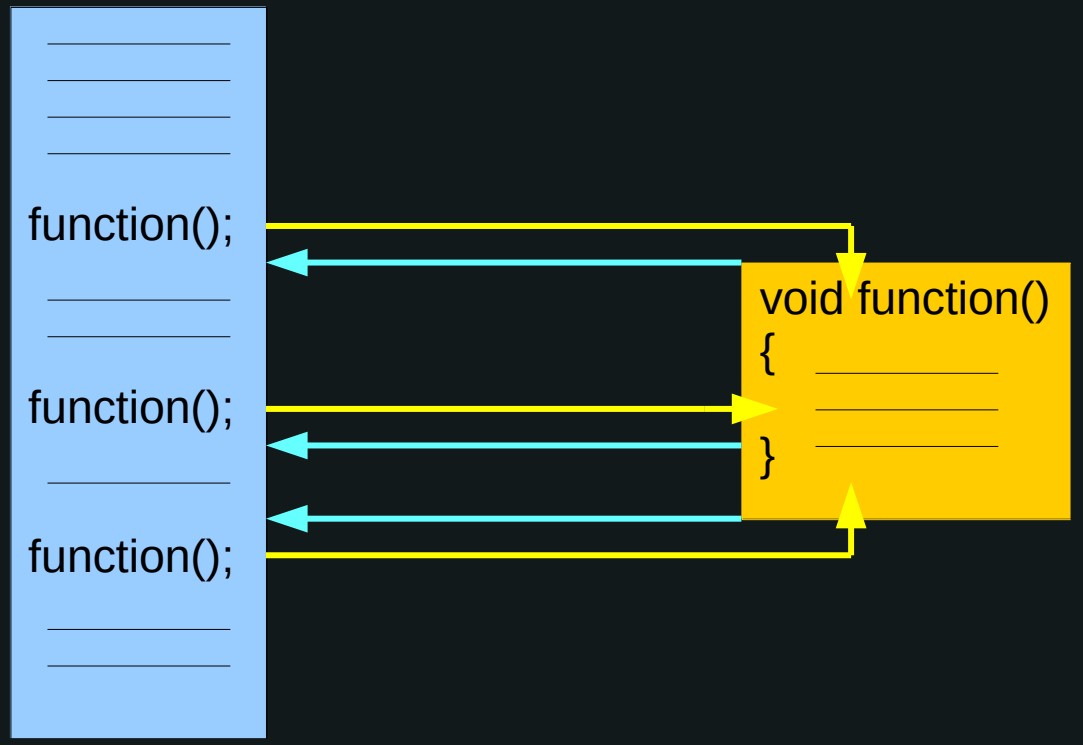
Pre Defined

Why Function ?

Functions help us in reducing code redundancy (reuse same functionality)

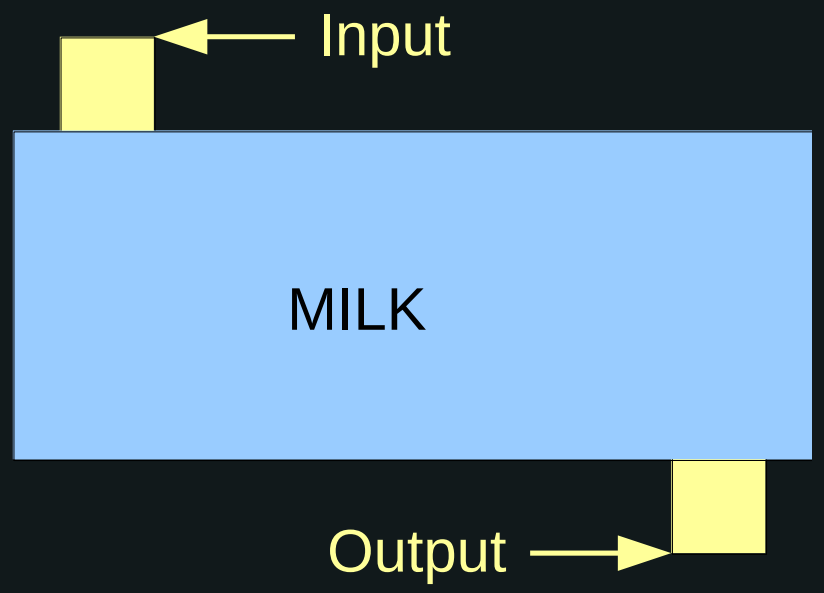
Functions make code modular (Breaking bigger problem in smaller chunks)

Functions provide abstraction (hide complications)



Simple Function

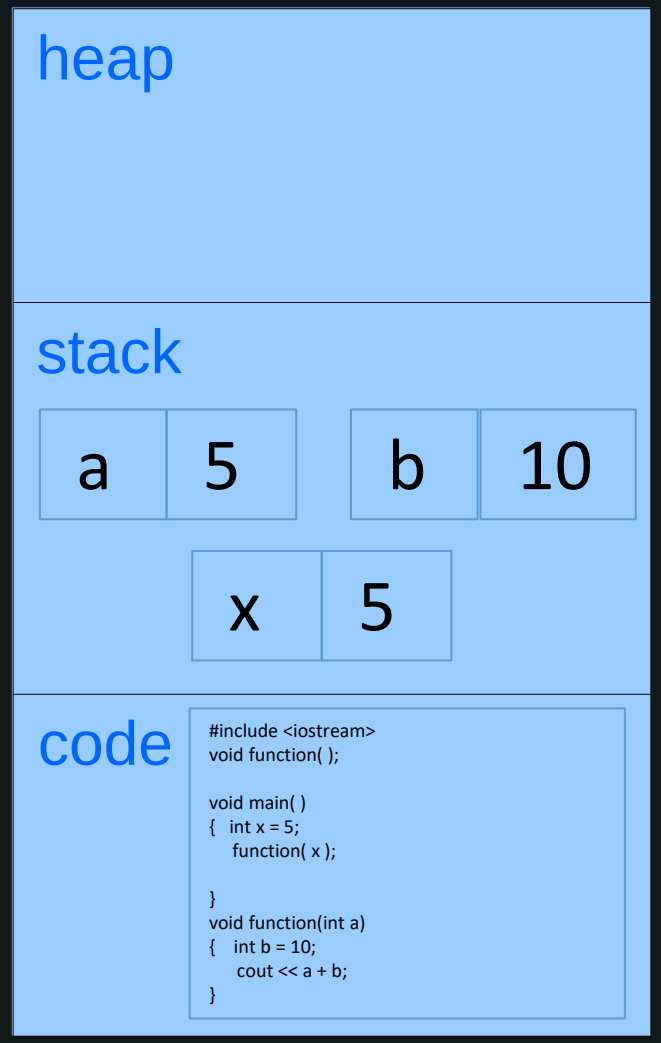
```
#include <iostream>

void fun( void );                    // function declaration
-------------------------------------------------------------
void main( )
{
    fun( );                          //  function calling
}
-------------------------------------------------------------
void fun()                           // function body
{
     cout << “Hello World”;
}
```



Function in Memory

```
#include <iostream>
void fun( int );
----------------------------------------
void main( )
{   int x = 5;
    fun( x );              // actual
}
----------------------------------------
void fun(int a)            // formal
{    int b = 10;
     cout << a + b;
}
```

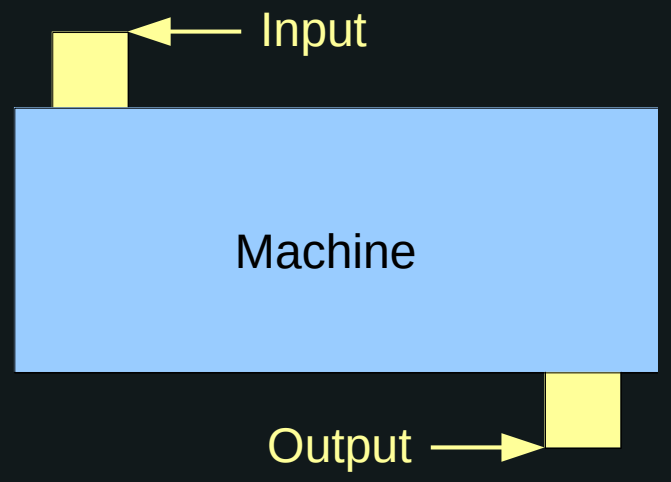
Main Memory



Function I / O

VEDINESH

```
Output   function_name ( input1 , input 2 ,... )
```




```
void function_name ( void )
```

```
int function_name ( void )
```

```
void  function_name ( int a , int b )
```

```
int function_name ( int a , int b )
```

Function Types

> Function with no arguments and no return value

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

```
void function (void)
{
   cout << "Hey"
}
```

Function Types

> Function with arguments and no return value.

```
void main ()
{
  ------------------
  ------------------
  function( 5 );
  ------------------
  ------------------
}
```

```
void function ( int x )
{
   cout << x ;
}
```

Function Types

> Function with no arguments and a return value.

```
void main ()
{
  int x;
  ------------------
  x = function( void );
  ------------------
  ------------------
}
```

```
int function ( void )
{
   return 0;
}
```

Function Types

> Function with arguments and a return value.

```
void main ()
{
  int x;
  -------------------
  x = function( 5 );
  -------------------
  -------------------
}
```

```
int function ( int x )
{
   return (x*x);
}
```

Function Types

> Pass By Address

```
void main ()
{
  int x = 10;

  function( &x );
  -------------------
  -------------------
}
```

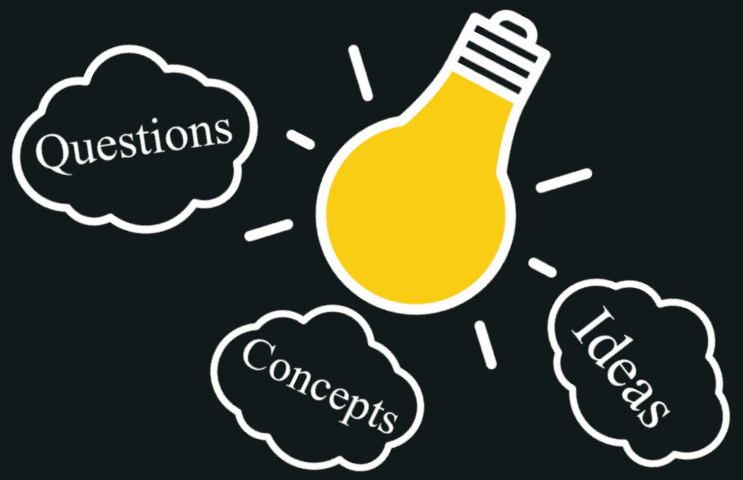
```
int function ( int *y )
{
   *y = 5;
}
```

10

x (1000)

5 ← 1000

x (1000) y



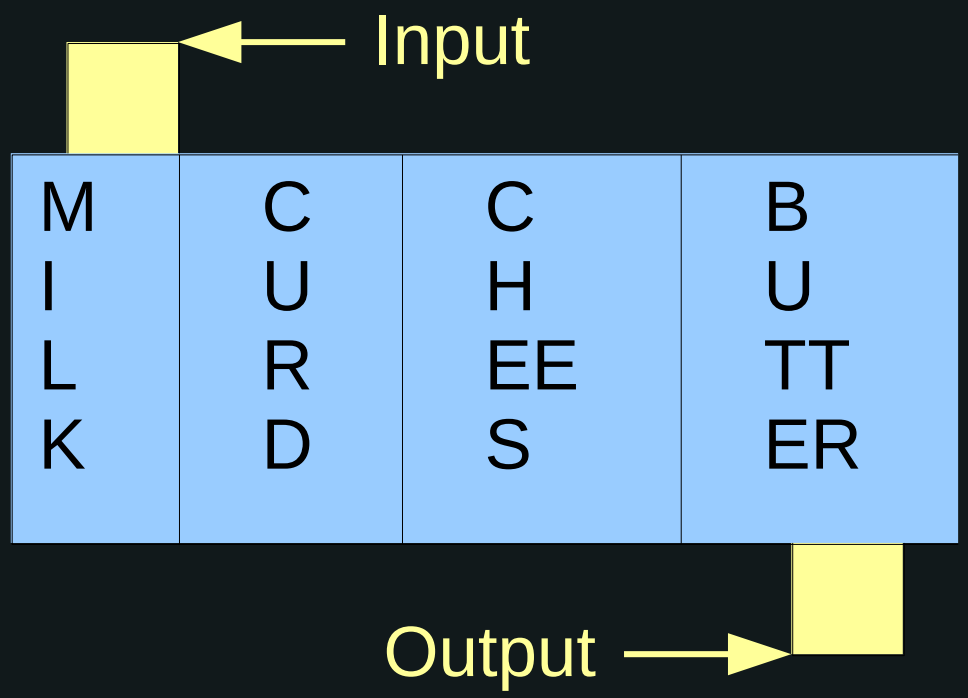
Functions Overloading

C++ Programming

VEDINESH

Function Overloading

An overloaded function perform, different activities depending upon type of input.



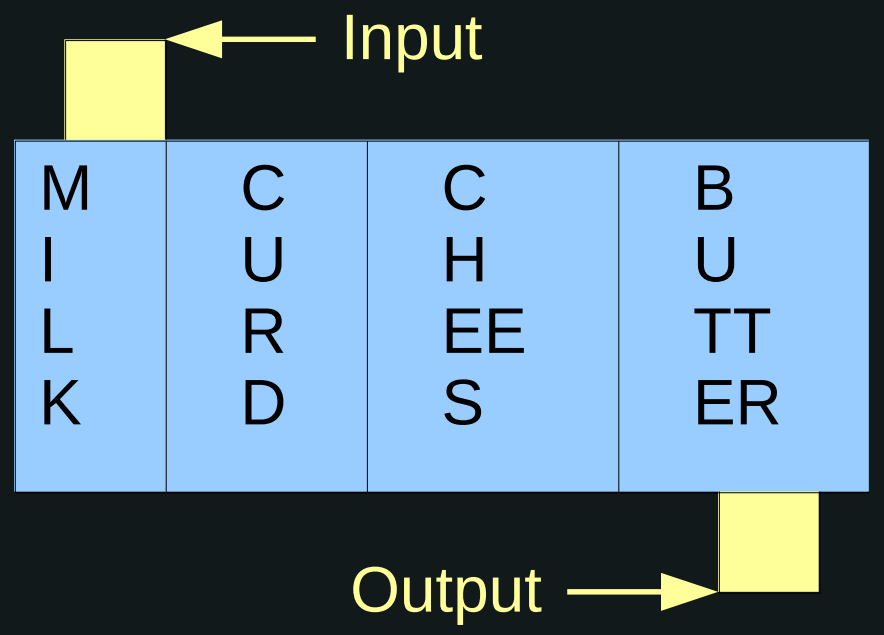
Multitasking

Benefits of Function Overloading

> Reduce code length.

> Program becomes easy to understand.

> Easy maintainability of the code



Function Overloading - Way 1

VEDINESH

Different type
of
arguments

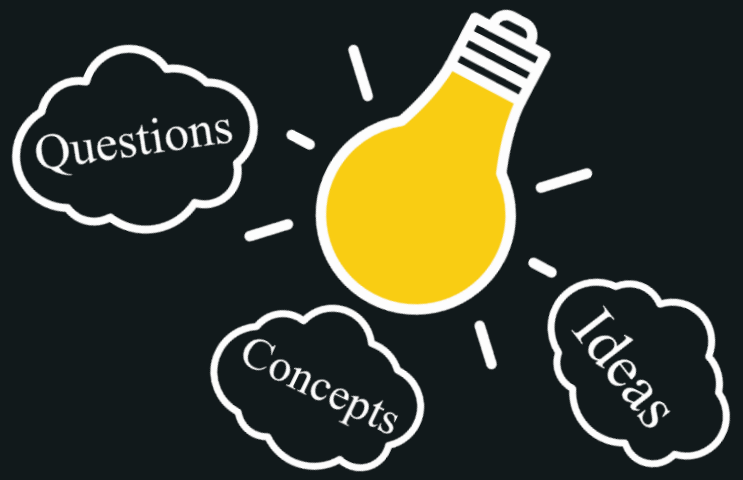
```
#include<iostream>
void main()
{
  add( 3, 7 );
  add( 4.5, 6.1 );
}
----------------------------------------
void add (int a, int b)
{ cout << a+b; }
----------------------------------------
void add (float a, float b)
{ cout << a+b; }
```

Function Overloading - Way 2

VEDINESH

Different number
of
arguments

```
#include<iostream>
void main()
{
  add( 3, 7 );
  add( 4, 6, 1 );
}
----------------------------------------
void add (int a, int b)
{ cout << a+b; }
----------------------------------------
void add (int a, int b, int c)
{ cout << a+b+c; }
```



Functions Issues

C++ Programming

VEDINESH

Function Issues

> Extra time is required:-

i) Jump to the function

ii) pushing arguments into stack

iii) removing them from stack (on completion)

iv) Jump back to main program.

```
void main ()
{
  int x;
  -------------------
  x = function( 5 );
  -------------------
  -------------------
}
```

```
int function ( int x )
{
   return (x*x);
}
```

Main Memory

heap

stack

x	5

code

```
#include <iostream>
void function( );

void main( )
{  int x = 5;
   function( x );

}
void function(int a)
{   int b = 10;
    cout << a + b;
}
```

Inline Function

```
void main ()
{
  int x = 10;
  -----------------
  function( x );
  -------------------
  -------------------
}

inline void function (int a)
{
      cout<<a;
}
```

Compilation

```
void main ()
{
  int x = 10;
  ------------------
  {
      cout<<a;
  }
  --------------------
  --------------------
}
```