

Functions

✓ A function is a set of statement grouped together and given a valid name

- Pre defined functions (built-in function)
- User-defined functions

Function definition

Interface(Prototype)



- Function name
- Function Parameter
- Function return type
- statements

Body



- Using a function involves 'calling', it
- Calling function consists of function name followed by arguments

`y=x(15,3)`

```
type name ( parameter1 , parameter2 , ... )  
{ statement }
```

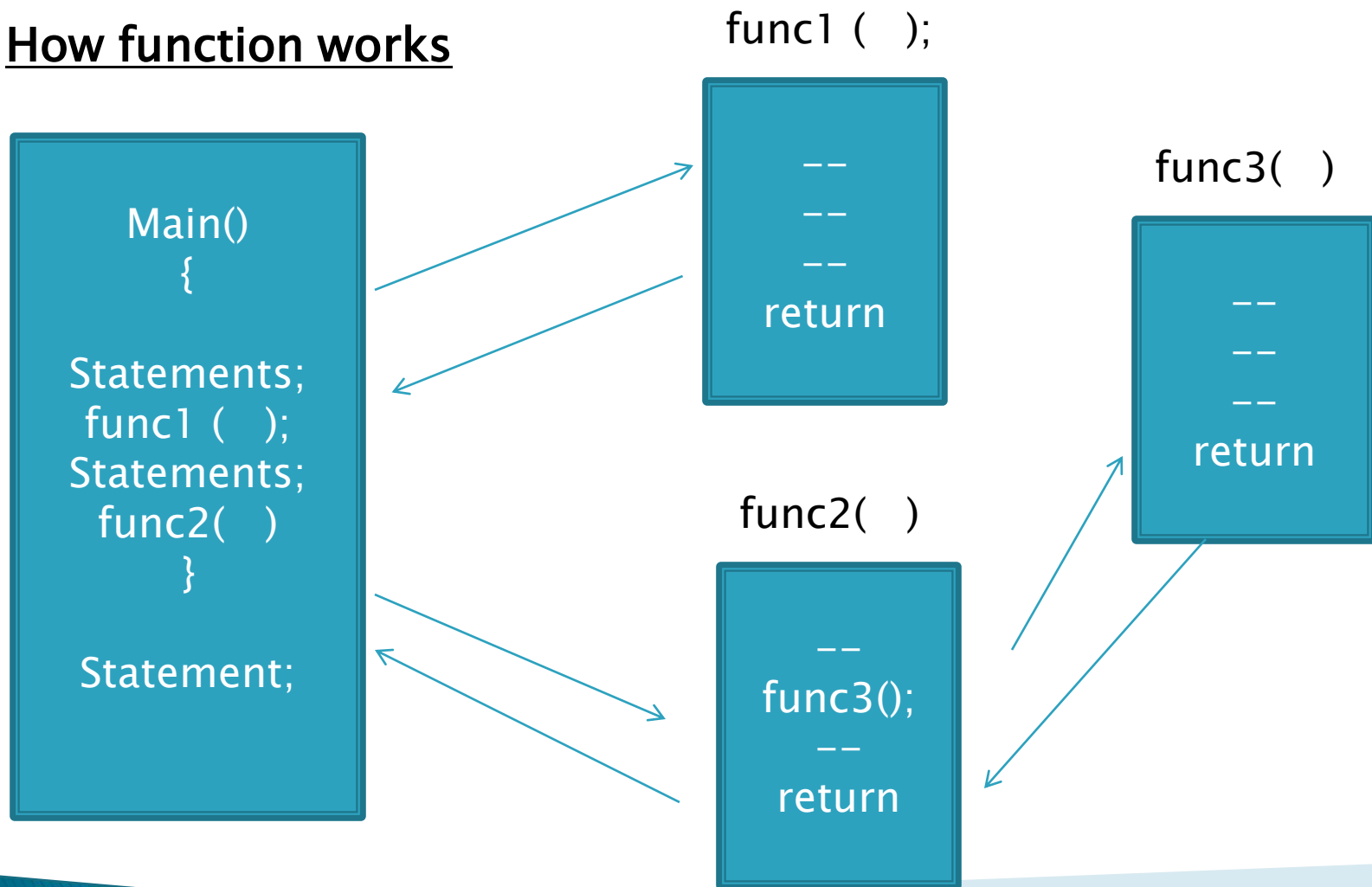
`int addition (int a,int b)`

Function name – This is simply a unique identifier

Function Parameter (signature) – This is set of zero or more typed identifiers used for passing values to and from the function

Function return type – This specifies the type of value the function returns

How function works



Definition of simple function

```
int add(int p, int q) //function interface
{
int r;           //local variable
r=p+q;          // result
return (r);      // return value of the function
}
```

How function declare:

Function Prototype & body
main Function

or

declaration of function
main function
function prototype & body

Function with return value

```
#include<iostream.h>
int add(int p,int q)
{
int r;
r=p+q;
return (r);
}
```

```
int main()
{
int z;
z=add(10,5);
cout<<"Addition of 2 numbers:"<<z;
return 0;
}
```

```
# include<iostream.h>
int sum(int a, int b);
```

```
int main()
```

```
{
int z,x,y;
cout<<"Enter x,y";
cin>>x>>y;
z=sum(x,y);
cout<<"The sum="<<z;
return 0;
}
```

```
int sum(int a, int b)
{
int r;
r=a+b;
return (r);
}
```


Scope of variables

```
#include<iostream.h>
```

```
int x;
```

```
char character;
```

```
unsigned int num;
```

} Global Variable

```
main()
```

```
{
```

```
short age;
```

```
float number;
```

```
cout<<"enter your age";
```

```
cin>>age;
```

```
}
```

} Local Variable

type name(parameter1,parameter2,) statement

We want to make a function just to show message on the screen. We don't need it to return any value. In this case we use the void type specifier for the function. If we want to return no value? We can use void

```
#include<iostream.h>
void message();
int main()
{
    message();
    return 0;
}
void message()
{
    cout<<"Functions we can structure our programs in a modular way ";
}
```

Print 20 stars in 10th row ,30th column on the screen using user define function.

```
# include<iostream.h>
#include<conio.h>
void star(int a, int b,int c);
```

```
int main()
```

```
{
    star(10,30,50);
    return 0;
}
```

```
void star(int a, int b,int c)
{
    int i;
    for(i=b;i<=c;i++)
    {
        gotoxy(i,a);
        cout<<"*";
    }
}
```

Default values in Parameters

function declaration we can specify a default value for each parameter. This value will be used if the corresponding argument is left blank when calling to the function.

If a value for that parameter is not passed when the function is called, the default value is used, but if a value is specified this default value is ignored and the passed value is used instead. For example

```
#include <iostream,h>
int divide(int a, int b=2)
{
int r;
r=a/b;
return (r);
}
```

```
int main()
{
int x,y;
x=divide(12);
cout<<x<<endl;
y=divide(20,4);
cout<<y;
return 0;
}
```

Function Overloading

c++ overloading mean using some thing for more than one purpose
(>>= insertion operator =right shift operator)

Two function with the same name , but different parameter types or number .That means you can give the same name to more then one function if they have either a different number of parameters or different types in their parameters.


```
# include<iostream.h>
#include<conio.h>
```

```
int operate(int a, int b)
{
return(a*b);
}
float operate (float a,float b)
{
return (a/b);
}
```

```
int main()
{
int x=5, y=2;
float n=8.0 , m=3.0;
cout<<operate(x,y)<<endl;
cout<<operate(n,m)<<endl;
return 0;
}
```