Functions

Functions

(1) // with return type with argument.

```
int func1 (int x, int y) {
        int sum = x+y;
        return sum;
}

int main() {
    int a, b;
    cin>> a>> b;
    int c = fun1(a,b);
    cout << c;
}
```

(2) without return type with argument.

```
void fun2 (int x, int y)
{
    int sum = x+y;
    cout << sum;
}

int main() {
    int a, b;
    cin>> a>> b;
    fun(a,b);
}
```

(3) With return type without argument

```
int fun3()
{
    int x, y;
    cin >> x >> y;
    int sum = x+y;
    return sum;
}

int main()
{
    int c = fun3();
    cout << c;
}
```

(4) without return type, , without argument.


```
void fun4()
{
    int x, y;
    cin>> x>> y;
    int sum = x+y;
    cout << sum;
}

int main()
{
    fun4();
}
```

Points

→ when we are writing void as function's return type, we cannot store it in a variable and then call it.
and we cannot return it also.

call by value :

call by value is a method in which it passes the copies of actual parameters to the formal parameters inside the function.

So, as you know it passes the copies, so even if there is any change in the values inside the function, it will not reflect the change in the actual values.

```
#include <bits/stdc++.h>
using namespace std;

int fun(int x, int y)          // 2 3.
{
    x=10, y=12;                // 10 12.
    return x+y;                // 22.
}
int main()
{
    int a, b;                  // 2, 3.
    cin>>a>>b;
    int c = fun(a, b);         // fun(2, 3).

    cout<<a<<" "<<b<<" "<<c;
}
```

Call By Reference

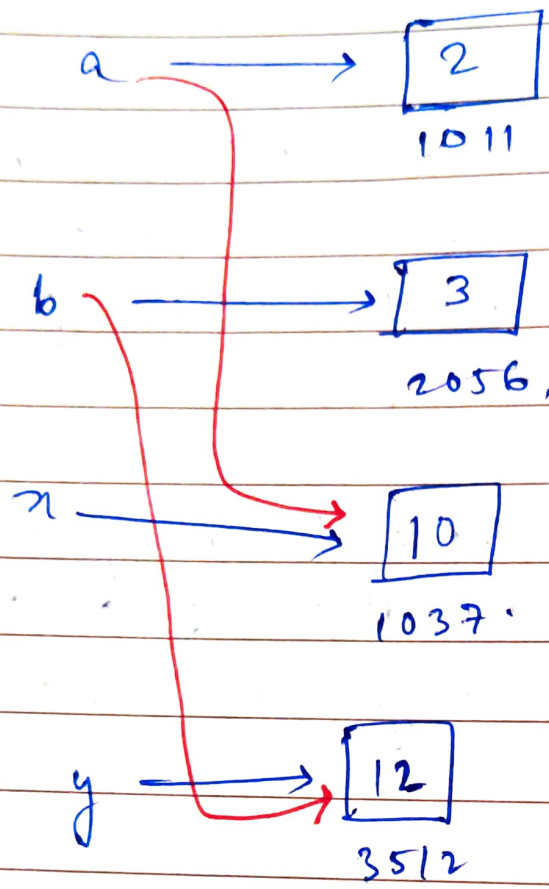
Call by reference is a method in which it passes the reference or address of the actual parameter to the function's formal parameters,

which means if there is any change in the values inside the function, it reflects that change in the actual values.

```
#include <bits/stdc++.h>
using namespace std;

int fun(int &x, int &y)   // address of a and b (2 and 3) will be passed, means 1011 and 2056
{
    // x will contain a's address.
    // y will contain b's address.
    x=10, y=12;   // The address of x and y will be changed here and so does the address of a and b.
    return x+y
}   //22

int main()
{
    int a, b;
    cin>> a>> b;
    int c = fun(a,b);   // fun(2, 3)
    cout<<a<<" "<<b<<" "<<c;   //10 12 22
}
```



(int x, int y) ✗

(int &x, &y) ✓