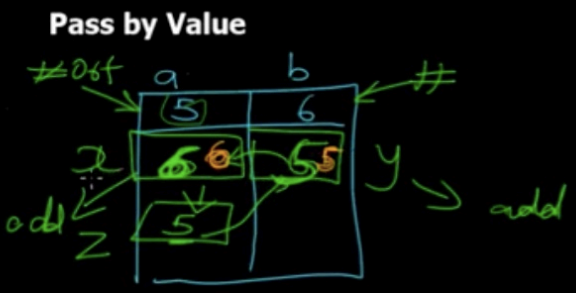
Pass by Value

- The call/pass by value method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function.
- In this case, changes made to the parameter inside the function have no effect on the argument.
- By default, C++ uses call by value to pass arguments. In general, this means that code within a function cannot alter the arguments used to call the function.

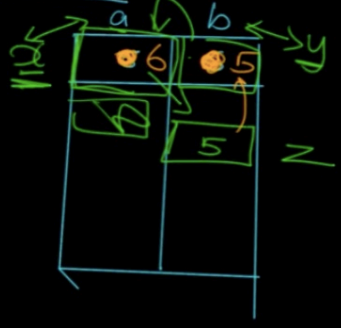



```
#include <iostream>
using namespace std;

void passByValue(int x, int y){
 int z = x;
 x=y;
 y=z;
 }

int main()
{
int a =5, b=6;
cout<<"Before swapping "<<endl<<"a: "<<a<<endl<<"b: "<<b<<endl<<endl;
passByValue(a,b);
cout<<"After swapping "<<endl<<"a: "<<a<<endl<<"b: "<<b<<endl<<endl;
   return 0;
}
```

Pass by Reference

- The call/pass by reference method of passing arguments to a function copies the reference of an argument into the formal parameter.
- Inside the function, the reference is used to access the actual argument used in the call. This means that changes made to the parameter affect the passed argument.
- To pass the value by reference, argument reference is passed to the functions just like any other value.

Pass by Reference




```
#include <iostream>
using namespace std;

void passByReference(int &x, int &y){
 int z = x;
 x=y;
 y=z;
 }

int main()
{
int a =5, b=6;
cout<<"Before swapping "<<endl<<"a: "<<a<<endl<<"b: "<<b<<endl<<endl;
passByReference(a,b);
cout<<"After swapping "<<endl<<"a: "<<a<<endl<<"b: "<<b<<endl<<endl;
   return 0;
}
```

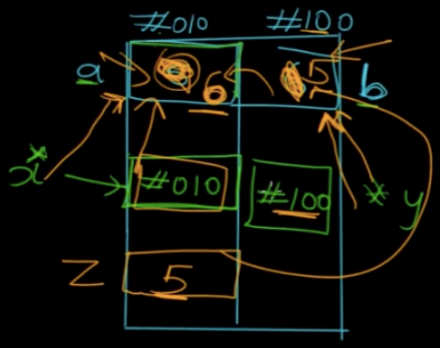
a = 6
b = 5

//fn call

Pass by Pointer or Call by Address

- The call by pointer method of passing arguments to a function copies the address of an argument into the formal parameter.
- Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the passed argument.
- To pass the value by pointer, argument pointers are passed to the functions just like any other value.

Pass by Address




```
#include <iostream>
using namespace std;

void passByAddress(int *x, int *y){
 int z = *x;
 *x=*y;
 *y=z;
 }

int main()
{
int a =5, b=6;
cout<<"Before swapping "<<endl<<"a: "<<a<<endl<<"b: "<<b<<endl<<endl;
passByAddress(&a,&b);
cout<<"After swapping "<<endl<<"a: "<<a<<endl<<"b: "<<b<<endl<<endl;
   return 0;
}
```

Pass by Value	**Pass by Reference**	**Pass by Address**
Passes actual value of argument into parameters of function	Passes reference of an argument into parameters of function	Passes address of an argument into parameters of function
Ex: a=5	Ex: &a	Ex: *a
Changes made to the parameter inside the function have no effect on the argument	Changes made to the parameter inside the function affect the argument.	Changes made to the parameter inside the function affect the argument.
In C++, CbyValue is default. Mean: code inside function cannot alter arguments used to call function	Address is used	Pointer is required
Ex: passbyvalue(a,b)	Ex: passbyref(a,b);	Passbyaddr(&a,&b);

`void passbyvalue(int x, int y)`
`{`
`  int z=x;`
`  x=y;`
`  y=z;`
`}`
	`void passbyref(int &x, int &y)`
	`{`
	`  int z=x;`
	`  x=y;`
	`  y=z;`
	`}`
		`void passbyaddr(int  *x, int *y)`
		`{`
		`  int z= *x;`
		`  *x=*y;`
		`  *y=z;`
		`}`

CODE 1:

```
#include<iostream>
using namespace std;

void passbyvalue(int x, int y)
{
        int z=x;
        x=y;
        y=z;
}

void passbyref(int &x, int &y)
{
        int z=x;
        x=y;
        y=z;
}

void passbyaddr(int  *x, int *y)
{
        int z= *x;
        *x=*y;
        *y=z;
}

int main()
{

        int a=5, b=6;
        cout<<"Before Swapping"<<endl<<"a is "<<a<<endl<<"b
is"<<b<<endl;

//        passbyvalue(a,b);
  //      passbyref(a,b);

        passbyaddr(&a,&b);
        cout<<"After Swapping"<<endl<<"a is "<<a<<endl<<"b
is"<<b<<endl;


return 0;
}
```

Code: 2

```
#include<iostream>
using namespace std;

void passbyval(int val)
{
        val = 10;
        cout<<"passed number is"<<val<<endl;
}

void passbyref(int &ref)
{
        ref = 20;
        cout<<"passed number is"<<ref<<endl;
}

void passbyaddr(int *ptr)
{
        *ptr = 30;
        cout<<"passed number is"<<*ptr<<endl;
}

int main()
{

        int x=5;
        cout<<"x is"<<x<<endl;
        passbyval(x);
        cout<<"final val is "<<x<<endl;

        cout<<"x is"<<x<<endl;
        passbyref(x);
        cout<<"final val(ref) is"<<x<<endl;

        int *xptr = &x;
        cout<<"x is"<<x<<endl;
        cout<<"xptr is "<<*xptr<<endl;
        passbyaddr(xptr);
        cout<<"final val(ptr) is"<<*xptr<<endl;

        return 0;
}
```