References in C++

References are like constant pointers that are automatically dereferenced. It is a new name given to an existing storage. So when you are accessing the reference, you are actually accessing that storage.

```
int main()
{
    int y=10;
    int &r = y;  // r is a reference to int y
    cout << r;
}
```

There is no need to use the * to dereference a reference variable.

Difference between Reference and Pointer

References	**Pointers**
Reference must be initialized when it is created.	Pointers can be initialized any time.
Once initialized, we cannot reinitialize a reference.	Pointers can be reinitialized any number of time.
You can never have a NULL reference.	Pointers can be NULL.
Reference is automatically dereferenced.	* is used to dereference a pointer.

References in Funtions

References are generally used for function argument lists and function return values, just like pointers.

CASE 1:

```
int* first (int* x)

{

    (*x++);

    return x;   // SAFE, x is outside this scope

}
```

first() takes a pointer as argument and returns a pointer, it will work fine. The returning pointer points to variable declared outside first(), hence it will be valid even after the first() ends.

```
CASE 2
int& second (int& x)

{

    x++;

    return x;   // SAFE, x is outside this scope

}
```

Similarly, **second()** will also work fine. The returning reference is connected to valid storage, that is `int a` in this case.

CASE 3

```
int& third ()

{

    int q;

    return q;   // ERROR, scope of q ends here

}
```

But in case of **third()**, we declare a variable `q` inside the function and try to return a reference connected to it. But as soon as function third() ends, the local variable q is destroyed, hence nothing is returned.

CASE 4:

```
int& fourth ()

{

    static int x;
    return x;   // SAFE, x is static, hence lives till the end.
}
```

To remodify above problem, we make x as **static** in function **fourth()**, giving it a lifetime till main() ends, hence now a reference connected to x will be valid when returned.

```
int main()
{
    int a=0;
    first(&a); // UGLY and explicit
    second(a);// CLEAN and hidden
}
```

Const Reference in C++

Const reference is used in function arguments to prevent the function from changing the argument.

```
void g(const int& x)
{
    x++;
}   // ERROR

int main()
{
    int i=10;
    g(i);
}
```

We cannot change the argument in the function because it is passed as const reference.