**Reference in C++**

**Introduction**

A reference variable is an alias, another name for an already present variable. As soon as a reference is initialized with a variable, either the variable name or the reference name may consult the variable.

**Creating References in C++**

Reference can be created by certainly using an & operator. When we create a variable, it occupies some memory location. We will access the original variable using either name of the variable or reference. for example,

int a=10;

Now, we create a reference variable for the above variable.  
 int &reference=a;

The above declaration means that ‘ref’ is a reference variable of ‘a’, i.e., we can use the ‘ref’ variable instead of the ‘a’ variable.

**Types of References**

C++ provides two types of references:

1. References to non-const values

2. References as aliases

**References to Non-Const Values**

It can be declared by using & an operator with the reference type variable.

#include <bits/stdc++.h>

using namespace std;

int main()

{

int a=10;

int &value=a;

cout << value << endl;

return 0;

}

Output

10

**References As Aliases**

It is another name of the variable which is being referenced.

For example,

int x=10; // 'x' is a variable.

int &y=x; // 'y' reference to x.

int &z=x; // 'z' reference to x.

**Properties of References**

**Initialization**

It will be initialized at the time of the declaration.

#include <bits/stdc++.h>

using namespace std;

int main()

{

int a=20;

int &b=a;

cout << "value of a is " <<b<< endl;

return 0;

}

Output

value of a is 20

**Reassignment**

It can’t be a reassigned method that is the reference variable cannot be modified.

#include <bits/stdc++.h>

using namespace std;

int main()

{

   int x=11;

   int z=67;

   int &y=x;

   cout<<"Address Before Reassingemnt "<<&y<<endl;

   y=z;

   cout<<"Address After Reassingemnt "<<&y<<endl;

   return 0;

}

Output

Address Before Reassingemnt 0x7ffe25f37748

Address After Reassingemnt 0x7ffe25f37748

Here we changed reference but still the address of y is same.

**Function Parameter**

References also can be passed as a function parameter. It does not create a copy of the argument and behaves as an alias for a parameter. It complements the performance as it does now, not making a copy of the statement.

#include <bits/stdc++.h>

using namespace std;

void swap(int &x, int &y) // function definition

{

   int temp; // variable declaration

   temp=x;

   x=y;

   y=temp;

}

int main()

{

   int a=9; // variable initialization

   int b=10; // variable initialization

   swap(a, b); // function calling

   cout << "value of a is :" <<a<< endl;

   cout << "value of b is :" <<b<< endl;

   return 0;

}

Output

value of a is : 10

value of b is : 9

**References as Shortcuts**

With the help of references, we can effortlessly access the nested data.

#include <bits/stdc++.h>

using namespace std;

struct profile

{

   int id;

};

struct employee

{

   profile p;

};

int main()

{

   employee e;

   int &ref=e.p.id;

   ref=34;

   cout << e.p.id << endl;

}

Output

34

**References VS Pointers**

References are often confused with pointers; however, three essential differences between references and pointers are −

* You can not have NULL references. You always have to assume that a reference is connected to a useful storage piece.
* When a reference is initialized to an object, it can’t be changed to refer to another object. Pointers may be pointed to another object at any time.
* A reference has to be initialized when it’s far created. Pointers can be initialized at any time.

**Frequently Asked Questions**

**1. What is the difference between pointer and reference?**

When a reference is created, it can’t reference some other object. This can be done with pointers. References cannot be null, while pointers can be. References can’t be uninitiated, and it is impossible to refer directly to a reference item after it is defined.

**2. What means pass by reference?**

The callee function gets hard and fast of references which might be aliased to variables. If a change is made to the reference variable, the original value may also be changed. The pointers handle all the references. Multiple values modification may be done by passing more than one variable.

**3. What are References in C++?**

A restrained type of pointer in C++ is referred to as a reference. A reference may be assigned best once and can’t be null.