References in C++

When a variable is declared as a reference, it becomes an alternative name for an existing variable. A variable can be declared as a reference by putting ‘&’ in the declaration.

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

int main()
{
    int x = 10;

    // ref is a reference to x.
    int& ref = x;

    // Value of x is now changed to 20
    ref = 20;
    cout << "x = " << x << '\n';

    // Value of x is now changed to 30
    x = 30;
    cout << "ref = " << ref << '\n';
```

Output

```
x = 20
ref = 30
```

Applications :

1. **Modify the passed parameters in a function**: If a function receives a reference to a variable, it can modify the value of the variable. For example, the following program variables are swapped using references.

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

```
void swap(int& first, int& second)
{
    int temp = first;
    first = second;
    second = temp;
}

int main()
{
    int a = 2, b = 3;
    swap(a, b);
    cout << a << " " << b;
    return 0;
}
```

Output

```
3 2
```

2. **Avoiding a copy of large structures**: Imagine a function that has to receive a large object. If we pass it without reference, a new copy of it is created which causes wastage of CPU time and memory. We can use references to avoid this.

```
struct Student {
    string name;
    string address;
    int rollNo;
}

// If we remove & in below function, a new
// copy of the student object is created.
// We use const to avoid accidental updates
// in the function as the purpose of the function
// is to print s only.
void print(const Student &s)
{
    cout << s.name << " " << s.address << " " << s.rollNo
```

```
        << '\n';
}
```

3. **For Each Loop to avoid the copy of objects**: We can use references in each loop to avoid a copy of individual objects when objects are large.

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

int main()
{
    vector<string> vect{ "geeksforgeeks practice",
                         "geeksforgeeks write",
                         "geeksforgeeks ide" };

    // We avoid copy of the whole string
    // object by using reference.
    for (const auto& x : vect) {
        cout << x << '\n';
    }

    return 0;
}
```