References in C++

When a variable is declared as reference, it becomes an alternative name for an existing variable. A variable can be declared as 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 << endl ;

  // Value of x is now changed to 30
  x = 30;
  cout << "ref = " << ref << endl ;

  return 0;
}
```

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, in 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 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;
}
```

3. **In For Each Loops to modify all objects** : We can use references in for each loops to modify all elements

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

int main()
{
    vector<int> vect{ 10, 20, 30, 40 };

    // We can modify elements if we
    // use reference
    for (int &x : vect)
        x = x + 5;

    // Printing elements
    for (int x : vect)
       cout << x << " ";

    return 0;
}
```

4. **In For Each Loops to avoid copy of objects** : We can use references in for each loops to avoid 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 << endl;

    return 0;
}
```

References vs Pointers

Both references and pointers can be used to change local variables of one function inside another function. Both of them can also be used to save copying of big objects when passed as arguments to functions or returned from functions, to get efficiency gain.
Despite above similarities, there are following differences between references and pointers.

A pointer can be declared as void but a reference can never be void
For example

```
int a = 10;
void* aa = &a;. //it is valid
void &ar = a; // it is not valid
```

Thanks to Shweta Bansal for adding this point.

References are less powerful than pointers

1) Once a reference is created, it cannot be later made to reference another object; it cannot be reseated. This is often done with pointers.
2) References cannot be NULL. Pointers are often made NULL to indicate that they are not pointing to any valid thing.
3) A reference must be initialized when declared. There is no such restriction with pointers

Due to the above limitations, references in C++ cannot be used for implementing data structures like Linked List, Tree, etc. In Java, references don't have above restrictions, and can be used to implement all data structures. References being more powerful in Java, is the main reason Java doesn't need pointers.

References are safer and easier to use:
1) Safer: Since references must be initialized, wild references like wild pointers are unlikely to exist. It is still possible to have references that don't refer to a valid location (See questions 5 and 6 in the below exercise)
2) Easier to use: References don't need dereferencing operator to access the value. They can be used like normal variables. '&' operator is needed only at the time of declaration. Also, members of an object reference can be accessed with dot operator ('.'), unlike pointers where arrow operator (->) is needed to access members.

Together with the above reasons, there are few places like copy constructor argument where pointer cannot be used. Reference must be used pass the argument in copy constructor. Similarly references must be used for overloading some operators like ++.

Exercise:
Predict the output of following programs. If there are compilation errors, then fix them.

Question 1

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

int &fun()
{
    static int x = 10;
    return x;
}
int main()
{
    fun() = 30;
    cout << fun();
    return 0;
}
```

Question 2

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

int fun(int &x)
{
    return x;
}
int main()
{
    cout << fun(10);
    return 0;
}
```

Question 3

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

void swap(char * &str1, char * &str2)
{
  char *temp = str1;
  str1 = str2;
  str2 = temp;
}

int main()
{
  char *str1 = "GEEKS";
  char *str2 = "FOR GEEKS";
  swap(str1, str2);
  cout<<"str1 is "<<str1<<endl;
  cout<<"str2 is "<<str2<<endl;
  return 0;
}
```

Question 4

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

int main()
{
   int x = 10;
   int *ptr = &x;
   int &*ptr1 = ptr;
}
```

Question 5

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

int main()
{
   int *ptr = NULL;
   int &ref = *ptr;
   cout << ref;
}
```

Question 6

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

int &fun()
{
    int x = 10;
    return x;
}
int main()
{
    fun() = 30;
    cout << fun();
    return 0;
}
```

Related Articles :

- Pointers vs References in C++
- When do we pass arguments by reference or pointer?
- Can references refer to invalid location in C++?
- Passing by pointer Vs Passing by Reference in C++

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Pointers and References in C++

C++ | References | Question 1

C++ | References | Question 6

Pointers vs References in C++

C++ | References | Question 4

C++ | References | Question 6

C++ | References | Question 6

Default Assignment Operator and References

Can references refer to invalid location in C++?

Strings formed from given characters without any consecutive repeating characters

Check if N and M can be made equal by increasing N by A and decreasing M by B

Operator Overloading '<<' and '>>' operator in a linked list class

Highest power of 2 less than or equal to given Integer

How to find the Entry with largest Value in a C++ Map

Improved By : pkthapa, BabisSarantoglou

Article Tags : C C++ School Programming cpp-references

Practice Tags : C CPP

50

3.1

Based on **154** vote(s)

To-do Done

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Previous

Playing with Destructors in C++

Next

'this' pointer in C++

Most popular articles

- Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe, ...
- MAQ Software Interview Experience (Aug 2019)
- 7 Steps for Mastering the Intro Email to a Prospective Employer
- Best Books to Learn Python for Beginners and Experts in 2019
- 12 Reasons Why You Should Learn Python in 2019

Most visited in C++

- Storage Classes in C++ with Examples
- Difference between Private and Protected in C++ with Example

Difference between Inheritance and Polymorphism

Difference between Single and Multiple Inheritance in C++

Difference between Abstraction and Encapsulation in C++