# Pointers vs References in C++

Difficulty Level : Easy • Last Updated : 11 Mar, 2023



Prerequisite: Pointers, References

C and C++ support pointers, which is different from most other programming languages such as Java, Python, Ruby, Perl and PHP as they only support references. But interestingly, C++, along with pointers, also supports references.

On the surface, both references and pointers are very similar as both are used to have one variable provide access to another. With both providing lots of the same capabilities, it's often unclear what is different between these mechanisms. In this article, I will try to illustrate the differences between pointers and references.

Pointers: A pointer is a variable that holds the memory address of another variable. A pointer needs to be dereferenced with the * operator to access the memory location it points to.



References: A reference variable is an alias, that is, another name for an already existing variable. A reference, like a pointer, is also implemented by storing the address of an object.
A reference can be thought of as a constant pointer (not to be confused with a pointer to a

constant value!) with automatic indirection, i.e., the compiler will apply the * operator for you.

```
int i = 3;

// A pointer to variable i or "stores the address of i"
int *ptr = &i;

// A reference (or alias) for i.
int &ref = i;
```

**Differences**:

1. **Initialization:** A pointer can be initialized in this way:

```
int a = 10;
int *p = &a;
// OR
int *p;
p = &a;
```

We can declare and initialize pointer at same step or in multiple line.

2. While in references,

```
int a = 10;
int &p = a; // It is correct
// but
int &p;
p = a; // It is incorrect as we should declare and initialize references at
single step
```

**NOTE:** This difference may vary from compiler to compiler. The above difference is with respect to Turbo IDE.

3. **Reassignment:** A pointer can be re-assigned. This property is useful for the implementation of data structures like a linked list, a tree, etc. See the following example:

```
int a = 5;
int b = 6;
int *p;
p = &a;
p = &b;
```

4. On the other hand, a reference cannot be re-assigned, and must be assigned at initialization.

```
int a = 5;
int b = 6;
int &p = a;
int &p = b; // This will throw an error of "multiple declaration is not
allowed"

// However it is valid statement,
int &q = p;
```

5. **Memory Address:** A pointer has its own memory address and size on the stack, whereas a reference shares the same memory address with the original variable but also takes up some space on the stack.

```
int &p = a;
cout << &p << endl << &a;
```

6. **NULL value:** A pointer can be assigned NULL directly, whereas a reference cannot be. The constraints associated with references (no NULL, no reassignment) ensure that the underlying operations do not run into an exception situation.

7. **Indirection:** You can have a pointer to pointer (known as a double pointer) offering extra levels of indirection, whereas references only offer one level of indirection. For example,

```
In Pointers,
int a = 10;
int *p;
int **q; // It is valid.
p = &a;
q = &p;

// Whereas in references,
int &p = a;
int &&q = p; // It is reference to reference, so it is an error
```

8. **Arithmetic operations:** Various arithmetic operations can be performed on pointers, whereas there is no such thing called Reference Arithmetic (however, you can perform pointer arithmetic on the address of an object pointed to by a reference, as in &obj + 5).

**Tabular form of difference between References and Pointers in C++**

| | References | Pointers |
|---|---|---|
| **Reassignment** | The variable cannot be reassigned in Reference. | The variable can be reassigned in Pointers. |
| **Memory Address** | It shares the same address as the original variable. | Pointers have their own memory address. |
| **Work** | It is referring to another variable. | It is storing the address of the variable. |
| **Null Value** | It does not have null value. | It can have value assigned as null. |
| **Arguments** | This variable is referenced by the method pass by value. | The pointer does it work by the method known as pass by reference. |

**When to use What**

The performances are exactly the same as references are implemented internally as pointers. But still, you can keep some points in your mind to decide when to use what:

- Use references:
  - In function parameters and return types.
- Use pointers:
  - If pointer arithmetic or passing a NULL pointer is needed. For example, for arrays (Note that accessing an array is implemented using pointer arithmetic).
  - To implement data structures like a linked list, a tree, etc. and their algorithms. This is so because, in order to point to different cells, we have to use the concept of pointers.

Quoted in C++ FAQ Lite: Use references when you can, and pointers when you have to. References are usually preferred over pointers whenever you don't need "reseating". This usually means that references are most useful in a class's public interface. References typically appear on the skin of an object, and pointers on the inside.

The exception to the above is where a function's parameter or return value needs a "sentinel" reference – a reference that does not refer to an object. This is usually best done by returning/taking a pointer, and giving the "nullptr" value this special significance (references must always alias objects, not a dereferenced null pointer).

**Related Article:**

When do we pass arguments as Reference or Pointers?

This article is contributed by **Rishav Raj**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

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

217

## Related Articles

1. lvalues references and rvalues references in C++ with Examples
2. Pointers and References in C++
3. Difference between constant pointer, pointers to constant, and constant pointers to constants
4. Can References Refer to Invalid Location in C++?
5. Default Assignment Operator and References in C++
6. C++ | References | Question 1
7. C++ | References | Question 6
8. C++ | References | Question 6
9. C++ | References | Question 4
10. C++ | References | Question 6

Previous Next

**Article Contributed By :**

GeeksforGeeks