Should I use a pointer or a reference to remotely assign a variable? [duplicate]

Asked 16 years, 3 months ago Modified 1 year, 2 months ago Viewed 130k times

277

This question already has answers here:
How should I pass objects to functions? (8 answers)
When should I use pointers instead of references in API-design? (17 answers)
Closed last year.

What would be better practice when giving a function the original variable to work with:

```
unsigned long x = 4;

void func1(unsigned long& val) {
     val = 5;
}
func1(x);
```

or:

```
void func2(unsigned long* val) {
     *val = 5;
}
func2(&x);
```

IOW: Is there any reason to pick one over another?

c++ variables pointers reference

Share
Improve this question
Follow

edited Sep 28, 2023 at 0:19
Jan Schultke
38.3k ● 8 ● 87 ● 168

asked Sep 22, 2008 at 10:38
Jack Reza
2,789 ● 3 ● 16 ● 4

1 References are of course valuable , but i come from C , where pointers are everywhere. One has to be proficient with pointers first to understand the value of references. – Jay D May 8, 2012 at 19:43

How does this fit in with a goal such as referential transparency from functional programming? What if you always want functions to return new objects and never internally mutate the state, especially not of variables passed to the function. Is there a way this concept is still used with pointers and references in a language like C++. (Note, I'm assuming someone already has the goal of referential transparency. I'm not interested in talking about whether or not it is a good goal to have.) – ely Sep 30, 2013 at 17:31

Prefer references. User pointers when you don't have a choice. – Ferruccio Jul 5, 2014 at 13:14

12 Answers

Sorted by: Highest score (default)

306

My rule of thumb is:

Use pointers if you want to do pointer arithmetic with them (e.g. incrementing the pointer address to step through an array) or if you ever have to pass a NULL-pointer.

Use references otherwise.

Share

Improve this answer

Follow

edited Jul 31, 2014 at 0:11
aaronsnoswell
6,221 ● 5 ● 48 ● 72

answered Sep 22, 2008 at 10:40
Nils Pipenbrinck
86.2k ● 33 ● 155 ● 223

11 Excelent point regarding a pointer being NULL. If you have a pointer parameter then you must either check explicitly that it is not NULL, or search all usages of the function to be sure that it is never NULL. This effort is not required for references. – Richard Corden Sep 22, 2008 at 11:10

28 Explain what you mean by arithmetic. A new user may not understand that you want to adjust what the pointer is pointing at. – Loki Astari Sep 22, 2008 at 16:30

9 Martin, By arithmetic I mean that you pass a pointer to a structure but know that it's not a simple structure but an array of it. In this case you could either index it using [] or do arithmetic by using ++/-- on the pointer. That's the difference in a nutshell. – Nils Pipenbrinck Nov 25, 2008 at 20:35

2 Martin, You can only do this with pointers directly. Not with references. Sure you can take a pointer to a reference and do the same thing in practice, but if you do so you end with very dirty code.. – Nils Pipenbrinck Nov 25, 2008 at 20:38

1 What about polymorphism (e.g. `Base* b = new Derived()`)? This seems like a case that can't be handled without pointers. – Chris Redford Mar 7, 2013 at 19:25

75

I really think you will benefit from establishing the following function calling coding guidelines:

1. As in all other places, always be `const`-correct.

   - Note: This means, among other things, that only out-values (see item 3) and values passed by value (see item 4) can lack the `const` specifier.
2. Only pass a value by pointer if the value 0/NULL is a valid input in the current context.
   - Rationale 1: As **a caller**, you see that whatever you pass in *must be* in a usable state.
   - Rationale 2: As **called**, you know that whatever comes in *is* in a usable state. Hence, no NULL-check or error handling needs to be done for that value.
   - Rationale 3: Rationales 1 and 2 will be *compiler enforced*. Always catch errors at compile time if you can.
3. If a function argument is an out-value, then pass it by reference.
   - Rationale: We don't want to break item 2...
4. Choose "pass by value" over "pass by const reference" only if the value is a POD (Plain old Datastructure) or small enough (memory-wise) or in other ways cheap enough (time-wise) to copy.
   - Rationale: Avoid unnecessary copies.
   - Note: *small enough* and *cheap enough* are not absolute measurables.

Share
Improve this answer
Follow

edited May 23, 2017 at 12:10
Community Bot
1 ● 1

answered Sep 22, 2008 at 11:37
Johann Gerell
25.5k ● 11 ● 76 ● 125

It lacks the guideline when:... "when to use const &"... The guideline 2 should be written "for [in] values, only pass by pointer if NULL is valid. Otherwise, use const reference (or for "small" objects, copy), or reference if it is an [out] value. I'm monitoring this post to potentially add a +1. – paercebal Sep 22, 2008 at 11:54

Item 1 covers the case you describe. – Johann Gerell Sep 22, 2008 at 13:14

It's a bit hard to pass an out-parameter by reference if it's not default-constructible. That's quite common in my code - the whole reason to have a function create that out-object is because it's non-trivial. – MSalters Sep 22, 2008 at 15:02

@MSalters: If you're going to allocate the memory inside the function (which I think is what you mean), then why not just return a pointer to the allocated memory? – Kleist Feb 16, 2011 at 9:27

@Kleist: On behalf of @MSalters, there are many possible reasons. One is that you might already have allocated memory to fill, like a pre-sized `std::vector<>`. – Johann Gerell Feb 16, 2011 at 10:28

This ultimately ends up being subjective. The discussion thus far is useful, but I don't think there is a correct or decisive answer to this. A lot will depend on style guidelines

24

and your needs at the time.

While there are some different capabilities (whether or not something can be NULL) with a pointer, the largest practical difference for an output parameter is purely syntax. Google's C++ Style Guide (https://google.github.io/styleguide/cppguide.html#Reference_Arguments), for example, mandates only pointers for output parameters, and allows only references that are const. The reasoning is one of readability: something with value syntax should not have pointer semantic meaning. I'm not suggesting that this is necessarily right or wrong, but I think the point here is that it's a matter of style, not of correctness.

Share

Improve this answer

Follow

edited Sep 18, 2016 at 11:19
Roland
23 ● 5

answered Sep 22, 2008 at 13:11
Aaron N. Tubbs
4,031 ● 1 ● 18 ● 6

What does it mean that references have value syntax but pointer semantic meaning?
– Eric Andrew Lewis Aug 16, 2015 at 13:43

It looks like you are passing a copy since the "pass by reference" part is only apparent from the funciton definition (value syntax), but you're not copying the value you pass, you essentially pass a pointer under the hood, which allows the function to modify your value.
– phant0m May 19, 2016 at 10:35

One should not forget that the Google C++ style guide is much detested. – Deduplicator Dec 20, 2018 at 0:54

16

Pointers

- A pointer is a variable that holds a memory address.
- A pointer declaration consists of a base type, an *, and the variable name.
- A pointer can point to any number of variables in lifetime
- A pointer that does not currently point to a valid memory location is given the value null (Which is zero)

```
BaseType* ptrBaseType;
BaseType objBaseType;
ptrBaseType = &objBaseType;
```

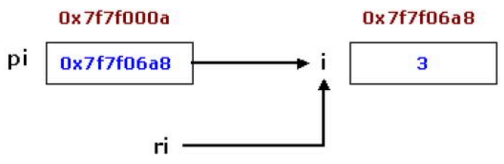
- The & is a unary operator that returns the memory address of its operand.
- Dereferencing operator (*) is used to access the value stored in the variable which pointer points to.

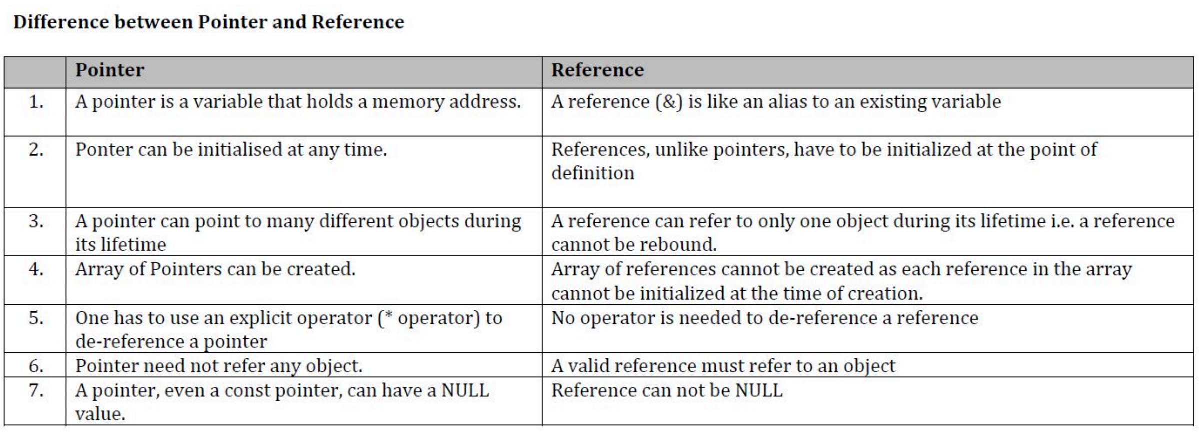
```
    int nVar = 7;
    int* ptrVar = &nVar;
    int nVar2 = *ptrVar;
```

Reference

- A reference (&) is like an alias to an existing variable.
- A reference (&) is like a constant pointer that is automatically dereferenced.
- It is usually used for function argument lists and function return values.
- A reference must be initialized when it is created.
- Once a reference is initialized to an object, it cannot be changed to refer to another object.
- You cannot have NULL references.
- A const reference can refer to a const int. It is done with a temporary variable with value of the const

  ```
  int i = 3;    //integer declaration
  int * pi = &i;    //pi points to the integer i
  int& ri = i;    //ri is refers to integer i – creation of reference and
  initialization
  ```





Difference between Pointer and Reference

	Pointer	Reference
1.	A pointer is a variable that holds a memory address.	A reference (&) is like an alias to an existing variable
2.	Ponter can be initialised at any time.	References, unlike pointers, have to be initialized at the point of definition
3.	A pointer can point to many different objects during its lifetime	A reference can refer to only one object during its lifetime i.e. a reference cannot be rebound.
4.	Array of Pointers can be created.	Array of references cannot be created as each reference in the array cannot be initialized at the time of creation.
5.	One has to use an explicit operator (* operator) to de-reference a pointer	No operator is needed to de-reference a reference
6.	Pointer need not refer any object.	A valid reference must refer to an object
7.	A pointer, even a const pointer, can have a NULL value.	Reference can not be NULL

Share Improve this answer Follow

answered Mar 13, 2018 at 12:43

Saurabh Raoot
1,411 ● 3 ● 28 ● 33

1 I liked your comparision table. I noticed a little typo on the second line: "*Pointer* can be initialised at any time". – Ikem Krueger Mar 4, 2022 at 0:10

7

You should pass a pointer if you are going to modify the value of the variable. Even though technically passing a reference or a pointer are the same, passing a pointer in your use case is more readable as it "advertises" the fact that the value will be changed by the function.

Share Improve this answer Follow

answered Sep 22, 2008 at 15:28

Max Caceres
1,986 ● 3 ● 18 ● 18

2 If you follow Johann Gerell guidelines, a non-const reference also advertises a changeable variable, so the pointer does not have that advantage here. – Alexander Kondratskiy Jul 19, 2011 at 13:39

5 @AlexanderKondratskiy: you're missing the point... you can't see instantly *at the call site* whether the called function accepts a paramter as a `const` or non- `const` reference, but you can see if the parameter's passed ala `&x` vs. `x`, and use that convension to encode whether the parameter's liable to be modified. (That said, there are times when you'll want to pass a `const` pointer, so the convension's just a hint. Arguable suspecting something might be modified when it won't be is less dangerous than thinking it won't be when it will be....) – Tony Delroy Jun 17, 2015 at 7:29

6

Use a reference when you can, use a pointer when you have to. From [C++ FAQ: "When should I use references, and when should I use pointers?"]

Share

Improve this answer

Follow

edited Feb 23, 2015 at 7:37

seaotternerd
6,409 ● 2 ● 48 ● 59

answered Mar 1, 2011 at 18:59

RezaPlusPlus
545 ● 1 ● 8 ● 18

5

If you have a parameter where you may need to indicate the absence of a value, it's common practice to make the parameter a pointer value and pass in NULL.

A better solution in most cases (from a safety perspective) is to use boost::optional. This allows you to pass in optional values by reference and also as a return value.

```
// Sample method using optional as input parameter
void PrintOptional(const boost::optional<std::string>& optional_str)
{
    if (optional_str)
    {
       cout << *optional_str << std::endl;
    }
    else
    {
        cout << "(no string)" << std::endl;
    }
}

// Sample method using optional as return value
boost::optional<int> ReturnOptional(bool return_nothing)
{
    if (return_nothing)
    {
        return boost::optional<int>();
    }

    return boost::optional<int>(42);
}
```

Share Improve this answer Follow

answered Oct 17, 2008 at 21:53

Kiley Hykawy
921 ● 5 ● 5

3

A reference is an implicit pointer. Basically you can change the value the reference points to but you can't change the reference to point to something else. So my 2 cents is that if you only want to change the value of a parameter pass it as a reference but if you need to change the parameter to point to a different object pass it using a pointer.

Share Improve this answer Follow

answered Sep 22, 2008 at 15:23

3

Consider C#'s out keyword. The compiler requires the caller of a method to apply the out keyword to any out args, even though it knows already if they are. This is intended to enhance readability. Although with modern IDEs I'm inclined to think that this is a job for syntax (or semantic) highlighting.

Share
Improve this answer
Follow

edited Jun 25, 2011 at 9:29

answered Oct 17, 2008 at 22:22

Daniel Earwicker
117k ● 38 ● 208 ● 286

typo: semantic, not symantic; +1 I agree about possibility of highlighting instead of writing out (C#), or & (in case of C, no references) – peenut Jun 25, 2011 at 9:23

2

Pass by const reference unless there is a reason you wish to change/keep the contents you are passing in.

This will be the most efficient method in most cases.

Make sure you use const on each parameter you do not wish to change, as this not only protects you from doing something stupid in the function, it gives a good indication to other users what the function does to the passed in values. This includes making a pointer const when you only want to change whats pointed to...

Share Improve this answer Follow

answered Sep 22, 2008 at 11:43
NotJarvis
1,247 ● 11 ● 18

2

Pointers:

- Can be assigned `nullptr` (or `NULL`).
- At the call site, you must use `&` if your type is not a pointer itself, making explicitly you are modifying your object.
- Pointers can be rebound.

References:

- Cannot be null.
- Once bound, cannot change.
- Callers don't need to explicitly use `&`. This is considered sometimes bad because you must go to the implementation of the function to see if your parameter is modified.

Share
Improve this answer
Follow

edited Dec 20, 2018 at 0:45
Pang
10.1k ● 146 ● 85 ● 124

answered Oct 29, 2013 at 13:09
Germán Diago
7,664 ● 1 ● 38 ● 66

A small point to those who do not know: nullptr or NULL is simply a 0. stackoverflow.com/questions/462165/... – Serguei Fedorov Jan 9, 2014 at 5:41

4 nullptr is not the same as 0. Try int a=nullptr; stackoverflow.com/questions/1282295/what-exactly-is-nullptr – Johan Lundberg Apr 27, 2015 at 18:58

0

A reference is similar to a pointer, except that you don’t need to use a prefix `*` to access the value referred to by the reference. Also, a reference cannot be made to refer to a different object after its initialization.

References are particularly useful for specifying function arguments.

for more information see "A Tour of C++" by "Bjarne Stroustrup" (2014) Pages 11-12

Share Improve this answer Follow

answered Apr 26, 2019 at 14:35

amirfg
281 ● 2 ● 7 ● 23