The quote about implicitly-declared copy constructor is unclear on logic

Asked 4 years, 6 months ago Modified 4 years, 6 months ago Viewed 99 times

1

> The implicitly-declared copy constructor for a class X will have the form
>
> > X::X(const X&)
>
> if each potentially constructed subobject of a class type M (or array thereof) has a copy constructor whose first parameter is of type `const M&` or `const volatile M&`
>
> Otherwise, the implicitly-declared copy constructor will have the form
>
> > X::X(X&)

I think about this sentence, the logic can be transformed as that:

> It's my box if each **red** card in this box has a shape of circle, otherwise it's jack's.

So, I have searched this box and I found there's no any red card in this box at all. Therefore,Is this box is mine or jack's? I think it's an ambigous proposition. It's a typically logic issue.

```
class A{
  int b;
};
int main(){
   A a;  //A::A(A&) or A::A(A const&) ?
}
```

I.E., the questions also can be transformed to:

```
//For this condition, A is either 0 or 1, Now I set A = 2
 if(all(A) == 0){
   then A belong to you
 }else {
   then A belong to mine
 }
when there's no A in the set.  A belong to mine?
```

I think modify the sentence like this:

> if **exists** any potentially constructed subobject of a class type M (or array thereof) has a copy constructor whose first parameter **is not** of type const M& or const volatile M&, the implicitly-declared copy constructor for a class X will have the form:
>
> > X::X(X&)
>
> otherwise, the implicitly-declared copy constructor will have the form:
>
> > X::X(const X&)

That would be more clear.

```
if(exsit(A)!=1){
  then A will belong to you
}else{
  A will belong to mine
}
```

So, for `A`, as long as exist `A` that value of the A is not 1, it will be the first branch, otherwise it will be the second branch, It does not exist a value of A that make there's no branch can match it or ambigours.

c++ c++17 language-lawyer

Share

Improve this question

Follow

edited Jun 20, 2020 at 9:12

Community Bot
1 ● 1

asked Jun 12, 2020 at 4:45

xmh0511
7,349 ● 1 ● 11 ● 41

@cigien The question is, for `class A{ int b;}`, the implicitly-defined copy constructor would be `A::A(A&)` ? – xmh0511 Jun 12, 2020 at 4:48

I find "*if each .. has*" more to the point than "*if any .. has not*" here, but it's just a matter of style and preference. – dxiv Jun 12, 2020 at 4:50

2 All I am saying is that your "*if(exist non-P) B else A*" is entirely equivalent to the original "*if(all P) A else B*" and I'll just leave it at that. – dxiv Jun 12, 2020 at 5:08

3 Then `all P` is vacuously true. – dxiv Jun 12, 2020 at 5:28

1 It's yours. For Jack to claim the box, he would have to produce a red card that's not a circle. But he has no such card to show. – dxiv Jun 12, 2020 at 5:37

1 Answer

Sorted by: Highest score (default)

3

The two formulations are entirely equivalent. The standard statement is of the form `if(all P) then A else B`, while the proposed re-statement is simply `if(exist non-P) then B else A` which is logically the same.

The question raised boils down to how this works when the set `all P` is empty i.e. class `X` has no "*potentially constructed subobject of a class type (or array thereof)*". In that case `if(all P)` is vacuously true, so case `A` applies. In other words, the implicitly-declared copy constructor for a class `X` with no potentially constructed subobjects of a class type (or array thereof) has the form `X::X(const X&)`.

As a side note, the notion of "*vacuous truth*" may sound a bit like logic sophistry, but it is often used in different disguises. For a C++ example:

```
bool IsAllPositive(vector<int> const &v)
{
    for(auto n : v)
        if(n <= 0) return false;
    return true;
}
```

This returns `true` if the vector is empty, and indeed `IsAllPositive` is vacuously true in that case.

Share

Improve this answer

Follow

edited Jun 12, 2020 at 16:41

answered Jun 12, 2020 at 6:36

dxiv
17.6k ● 2 ● 32 ● 53

Suggestion: modify `potentially constructed subobject` to `potentially constructed subobject of class type M which...`, because `int a` is also a `potentially constructed subobject`. – xmh0511 Jun 12, 2020 at 7:05

1 Exactly, `the notion of "vacuous truth" may sound a bit like logic sophistry` +1, such as "all the cell phone in the room are bigger than earth" even true if there's no cell phone in the room at all. It's so weird. – xmh0511 Jun 12, 2020 at 7:11

@jackX I edited the missing part in ("subobject *of a class type (or array thereof)*"). That does not include the "*which*" clause since, in the context, that's immaterial once no qualifying subobjects exist. As for `int`, fundamental types are certainly not "*of a class type*". – dxiv Jun 12, 2020 at 16:46

Similarity issue: If moscow is the captial of American, then you couldn't have went to moscow but haven't wen to American, otherwise..., For this proposition, if I have found moscow is in Russia, then I can say the `if` is false, hence the `otherwise` branch works, Right? – xmh0511 Jun 14, 2020 at 1:43

`If moscow is the captial of American, then ...` You can replace `...` with *anything* and the logical implication will be true. This is not only "*similar*" but actually the *same* as "*vacuous truth*". That said, I think you are splitting hairs that no longer have much anything to do with C++ at this point. – dxiv Jun 14, 2020 at 2:21