CppIndia

Move semantic – 0 to 1

LinkedIn: https://www.linkedin.com/company/cppindia
Youtube: https://www.youtube.com/channel/UCwB-WjSJI2D97YZcACFxJDw
Facebook: https://www.facebook.com/CppIndiaUG
Discord Community: https://discord.gg/D6a2jNh

Dheeraj Jha
www.jhadheeraj.com
LinkedIn: https://www.linkedin.com/in/jhadheeraj/

Move semantic – 0 to 1

CppIndia

lvalue - rvalue

- Lvalue it appears at left.
- Rvalue can only be on right side.

l = r

- Lvalue will have a name.
- We can get the address of lvalue

- Rvalue does not have a name.
- We can not get the address of rvalue

Let's jump into code.

```
string s{};
s + s = s;
```

rvalue

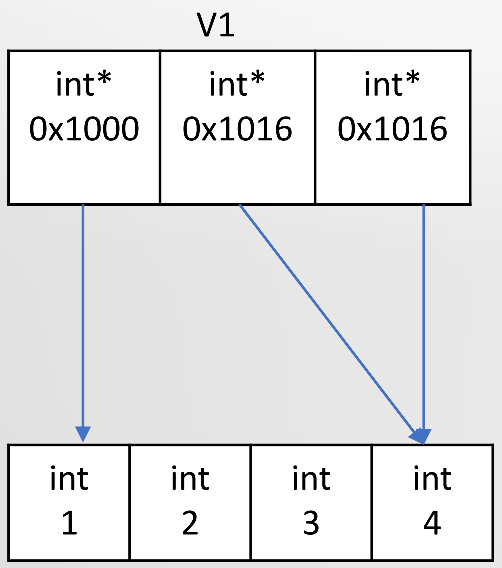
lvalue

If the expression has an identifiable memory address, it's lvalue otherwise, rvalue

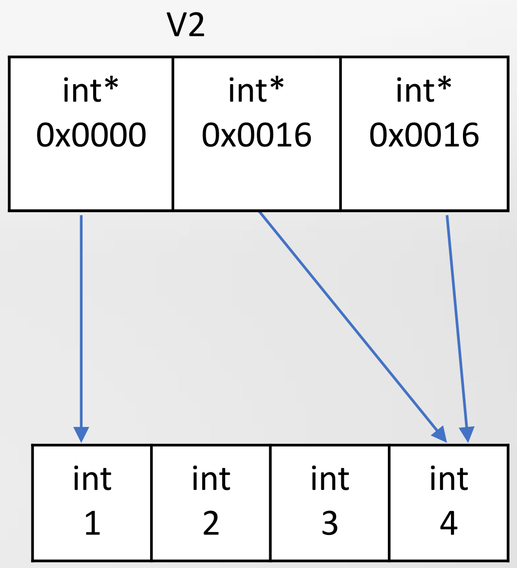
Move semantic – 0 to 1

std::vector<int> V1{1,2,3,4};

std::vector<int> v2{};

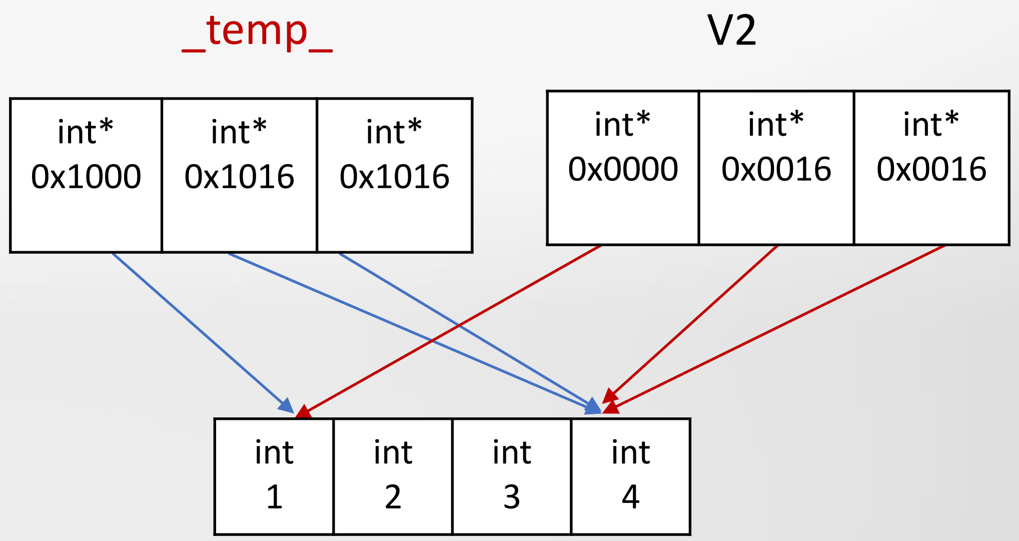


V2 = V1;



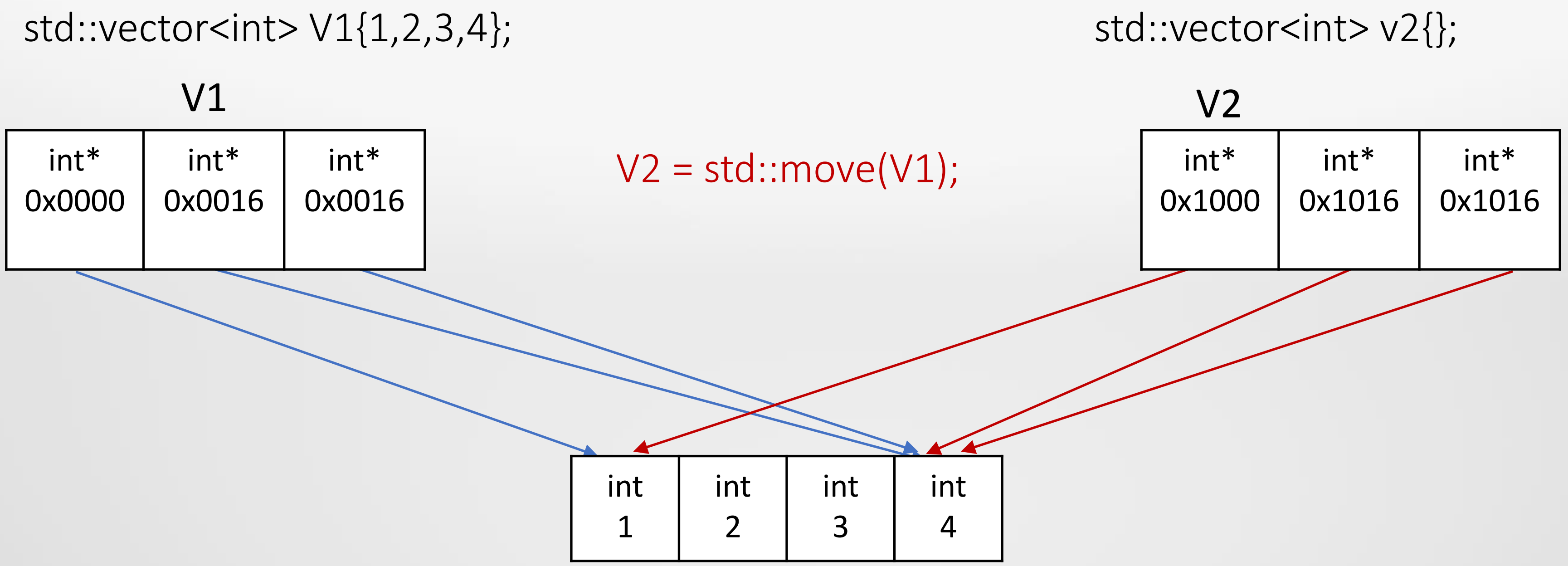
Move semantic – 0 to 1

```
std::vector<int> createVector(){
            return std::vector<int>{1,2,3,4};
}
std::vector<int> V2{};
V2 = createVector();
```



Move semantic – 0 to 1

std::move()



Move semantic – 0 to 1

CppIndia

std::move()

```
template <typename T>
typename remove_reference<T>::type&& move(T&& arg) {
  return static_cast<typename remove_reference<T>::type&&>(arg);
}
```

Move semantic – 0 to 1

Move Constructor and Move Assignment operator

```
//Move constructor
test(test&& obj) = default;

//Move assignment operator
test& operator=(test&& obj) = default;
```

Now in Modern C++, there are 6 special functions a class has.

- Constructor
- Destructor
- Copy-constructor
- Assignment operator
- Move-constructor
- Move Assignment operator

Move semantic – 0 to 1

Move Constructor

What to expect:

- Transfer the content
- moved-from should be the valid but undefined state.

Let’s jump into code.

2 Steps:

- Member-wise move
- Reset

Move semantic – 0 to 1

Move Assignment operator

What to expect:

- Clean up all resources
- Transfer the content
- Leave move from object in a valid but undefined state.

Let’s jump into code.

3 steps:

- cleanup
- Member-wise move
- Reset

Move semantic – 0 to 1

C.ctor: Constructors, assignments, and destructors

These are default operations:

- a default constructor: X()
- a copy constructor: X(const X&)
- a copy assignment: operator=(const X&)
- a move constructor: X(X&&)
- a move assignment: operator=(X&&)
- a destructor: ~X()

Move semantic – 0 to 1

C.21: If you define or =delete any copy, move, or destructor function, define or =delete them all

The semantics of copy, move, and destruction are closely related, so if one needs to be declared, the odds are that others need consideration too.

Declaring any copy/move/destructor function, even as =default or =delete, will suppress the implicit declaration of a move constructor and move assignment operator.

Declaring a move constructor or move assignment operator, even as =default or =delete, will cause an implicitly generated copy constructor or implicitly generated copy assignment operator to be defined as deleted.

So as soon as any of these are declared, the others should all be declared to avoid unwanted effects like turning all potential moves into more expensive copies, or making a class move-only.

This is known as “the rule of five.”

Move semantic – 0 to 1

Implicitly-declared move constructor

If no user-defined move constructors are provided for a class type.

- there are no user-declared copy constructors.
- there are no user-declared copy assignment operators.
- there are no user-declared move assignment operators.
- there is no user-declared destructor.
- the implicitly-declared move constructor is not defined as *deleted*

Then the compiler will declare a move constructor as a non-explicit inline public member of its class.

CppIndia

C.66: Make move operations noexcept

These are default operations:

- a default constructor: X()
- a copy constructor: X(const X&)
- a copy assignment: operator=(const X&)
- a move constructor: X(X&&)
- a move assignment: operator=(X&&)
- a destructor: ~X()

Move semantic – 0 to 1

CppIndia

Any Questions?

Move semantic – 0 to 1

CppIndia

Reference

- https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#S-ctor
- https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#c21-if-you-define-or-delete-any-copy-move-or-destructor-function-define-or-delete-them-all
- https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#c66-make-move-operations-noexcept
- https://en.cppreference.com/w/cpp/language/move_constructor
- https://en.cppreference.com/w/cpp/language/move_assignment
- https://www.youtube.com/watch?v=St0MNEU5b0o&t=2s

CppIndia

LinkedIn: https://www.linkedin.com/company/cppindia
Youtube: https://www.youtube.com/channel/UCwB-WjSJI2D97YZcACFxJDw
Facebook: https://www.facebook.com/CppIndiaUG
Discord Community: https://discord.gg/D6a2jNh