fast way to copy one vector into another

I prefer two ways:

```
void copyVecFast(const vec<int>& original)
{
  vector<int> newVec;
  newVec.reserve(original.size());
  copy(original.begin(),original.end(),back_inserter(newVec));
}

void copyVecFast(vec<int>& original)
{
```

```
  vector<int> newVec;
  newVec.swap(original);
}
```

How do you do it?

c++ algorithm stl

edited Jul 20 '16 at 2:23

gsamaras
22.9k 17 42 87

asked Mar 13 '09 at 21:23

Sasha

8 Second one has misleading name - as it is not a copy (although it is fast). – Anonymous Mar 13 '09 at 21:29

6 Answers

Your second example does not work if you send the argument by reference. Did you mean

```
void copyVecFast(vec<int> original) // no reference
{

  vector<int> new_;
  new_.swap(original);
}
```

That would work, but an easier way is

```
vector<int> new_(original);
```

answered Mar 13 '09 at 21:30

rlbond
33.9k 35 139 196

Good, it works. But it's not working for an array of vectors: for e.g: vector<int> A[n]; – ABcDexter Dec 28 '14 at 7:51

They aren't the same though, are they? *One is a copy, the other is a swap*. Hence the function names.

My favourite is:

```
a = b;
```

Where `a` and `b` are vectors.

edited Jul 20 '16 at 2:31

gsamaras
22.9k 17 42 87

answered Mar 13 '09 at 21:25

Daniel Earwicker
84k 28 164 242

2 In fact the approach is passing by value, the compiler calls the copy constructor, and then swapping that newly created element. That is why rlbond suggests calling the copy constructor directly to achieve the same effect. – David Rodríguez - dribeas Mar 14 '09 at 0:17

1 However, you can't call rlbon without a function that passes the original as val. Otherwise, the original one will be empties. The second solution made sure that you will always call by value and hence you will not lose the date in the original vector. (Assuming swap deals with pointers) – Eyad Ebrahim Mar 31 '13 at 13:57

Won't that move the elements of b to a (leaving b with size == 0)? – Jonathan. Oct 23 '14 at 16:00

1 @Jonathan. Assuming you're talking about `a = b` then no. Assignment means: make `a` equal `b` without changing `b`. By contrast, `std::swap(a, b)` would exchange their contents (so `b`'s `size` would now be whatever `a`'s had been before). You are perhaps thinking of a move operation (as occurs in C++11, but not in an ordinary assignment like this). Such a move would leave `b` in an, ahem, "interesting" state - see stackoverflow.com/questions/17730689/… – Daniel Earwicker Oct 24 '14 at 11:39

1 @Jonathan. Note the double ampersand `&&`. That version will only be used for an rvalue reference. It won't match any non-const value (such as `b` in my example above). You can turn `b` into one by saying `a = std::move(b);` See en.cppreference.com/w/cpp/language/value_category for even greater levels of complexity. – Daniel Earwicker Oct 24 '14 at 15:29

This is another valid way to make a copy of a vector, just use its constructor:

```
std::vector<int> newvector(oldvector);
```

This is even simpler than using `std::copy` to walk the entire vector from start to finish to `std::back_insert` them into the new vector.

That being said, your `.swap()` one is not a copy, instead it swaps the two vectors. You would modify the original to not contain anything anymore! Which is not a copy.

edited Mar 20 at 20:03

phoenix
545 5 11

answered Mar 13 '09 at 21:28

X-Istence
11k 3 36 62

---

you should not use swap to copy vectors, it would change the "original" vector.

pass the original as a parameter to the new instead.

answered Mar 13 '09 at 21:27

Raz
1,674 2 16 22

You can try this:

```
new_vector.assign(old_vector.begin(),old_vector.end());
```

edited Aug 31 '16 at 23:44

Guillaume Racicot
7,802 3 17 56

answered Aug 31 '16 at 19:27

Fake_Death
41 1

In case the vector ALREADY existed and you wanted to just copy, you could do this:

```
newVec.resize(oldVec.size());
memcpy(&newVec.at(0), &oldVec.at(0), oldVec.size());
```

answered Sep 21 '15 at 18:48

sgowd
171 3 12

Please don't memcpy. Also this won't work since memcpy takes the size in bytes. Also if the other vector already exists you can just do `newVec = oldVec` which is the same as one of the other answers. – FDinoff Sep 29 '15 at 18:39

Yes, you are right. I didn't see that. @FDinoff, although below one works, why do you suggest not using memcpy? It seems to be much faster than newVec = oldVec. memcpy(&newVec.at(0), &oldVec.at(0), oldVec.size() * sizeof(int)); – sgowd Sep 29 '15 at 23:16

In the general case, Copying an object without calling its copy constructor could lead to subtle bugs. In this case I would have thought they would have had the same performance. If it didn't I would say vector wasn't performance optimized, since it should already be doing this. Did you actually write a benchmark? – FDinoff Oct 3 '15 at 3:23

I wasn't criticizing you.. My team lead also suggested the same and I was trying to understand. – sgowd Oct 8 '15 at 23:54

(I didn't think you were criticizing me.) Is there still something you don't understand? – FDinoff Oct 8 '15 at 23:57