bubbleSort vector template class

IBM Bluemix
Move your data. Not your cloud spend.
Cut your cloud server bandwidth bill.*
Get Started
* Up to 10TB/mo
IBM

I am working on an assignment where I need to sort vectors using a template so I can pass different data types to my class and sort them not using std::sort. so I am very stuck so far, I don't really get how to use the template to accept my input from main(). PS. this is for Homework, that is why i am not using arrays and the sort function.

Here is my code so far.

```
using namespace std;

template <class T>
class SortableVector
{
```

```
private:
 T a =0;
public:
  SortableVector();                // constructor
 ~SortableVector();               // destructor

void bubble_sort(vector<T>, a)


{
    for (int i = a.size(); i > 0;i--)
    {
      for (int j = 0, k = 1; k < i;j++, k++)
      {
        if (a[j] > a[k])
        {
          int swap = a[j];
          a[j] = a[k];
          a[k] = swap;
        }
      }
    }
}
};
```

And my main is looking like this:

```
int main()
{


  int alen, val;
  vector<int> a;
  cout << "Enter the number of elements : ";
  cin >> alen;
  for(int i = 0; i < alen; i++)
  {
   cin >> val;
    a.push_back(val);
  }
  SortableVector::bubble_sort(a);
  cout << "List of sorted elements: " << endl;
  for(int i = 0; i < alen; i++)
  {
    cout << a[i] << " ";
  }
      cout << endl;
}
```

Any help will be welcome :)

Ok... So I made some changes thanks to Namfuak

now I have a totally different problem

command line output;

Hw8_3.cpp:(.text+0x11): undefined reference to `SortableVector<int>::SortableVector()'` `Hw8_3.cpp:(.text+0x138): undefined reference to` SortableVector::~SortableVector()' Hw8_3.cpp:(.text+0x170): undefined reference to `SortableVector::~SortableVector()' collect2: error: ld returned 1 exit status

I am really not getting this one. This is the code I have so far;

```
template <class T>
class SortableVector
{
private:
 vector<T> vec;
 public:
  SortableVector();                  // constructor
 ~SortableVector();                 // destructor
void push_back(T push) {vec.push_back(push);}
T bubble_sort(vector<T> a);
};

template <class T>
T SortableVector<T>::bubble_sort(vector<T> a)
{


for (int i = a.size(); i > 0;i--)
{
  for (int j = 0, k = 1; k < i;j++, k++)
  {
    if (a[j] > a[k])
    {
      T swap = vec[j];
      vec[j] = vec[k];
      vec[k] = swap;
   }
 }
   }return 0;
}
```

And my main() ;

```
{
 SortableVector<int> L;

  int alen, val;
  vector<int> a;
  cout << "Enter the number of elements : ";
  cin >> alen;
  for(int i = 0; i < alen; i++)
  {
    cin >> val;
    L.push_back(val);
  }
 L.SortableVector<int>::bubble_sort(a);
  cout << "List of sorted elements: " << endl;
  for(int i = 0; i < alen; i++)
  {
    cout << a[i] << " ";
  }

}
```

Any other Idea? I am really lost here...

c++ vector bubble-sort

edited May 15 '14 at 3:52

asked May 15 '14 at 2:31

Mrodri13
10 2

4 Answers

If you are using a template, your `swap` variable needs to be of that template type. IE:

```
T swap = a[j];
```

EDIT: Looking through, I don't think you are using the correct design. Your `SortableVector` should probably have a `vector<T>` as a member, IE:

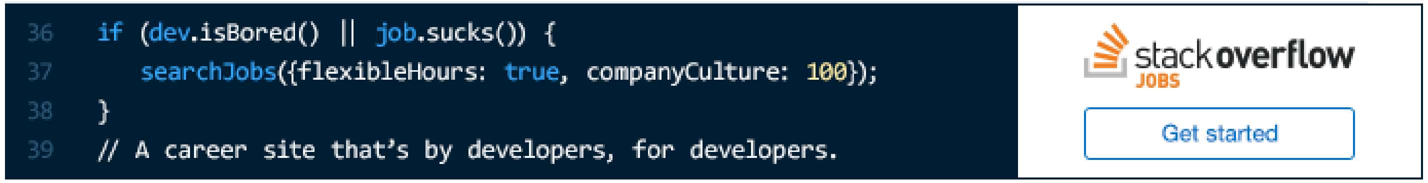
```
template <class T>
class SortableVector
{
private:
     std::vector<T> vec;
public:
    SortableVector();
    ~SortableVector();
    void push_back(T push) {vec.push_back(push);}
    void bubble_sort() //You could also return a sorted version of vec
}
```

Otherwise, there is no reason for a `SortableVector` class, you can just take the function and put it in the global space as a templated function.

answered May 15 '14 at 2:37
IllusiveBrian
2,393 2 6 17

Perhaps the idea was to inherit from the vector class? Which is not a good idea, but that's possibly where some confusion lies – Henry May 15 '14 at 2:46

I see, Thank you very much, This is my first time working with vectors so I am not really sure of where I am going. my direction are; Write a class template SortableVector. The class should have a member function that sorts the vector elements in ascending order... and we are not supposed to use std::sort. so I thought about implementing bubble sort... Now I see that my logic might be wrong. – Mrodri13 May 15 '14 at 2:53



```
if (dev.isBored() || job.sucks()) {
   searchJobs({flexibleHours: true, companyCulture: 100});
}
// A career site that's by developers, for developers.
```

You need to provide definitions for the constructor and destructor.

If you don't declare them, the compiler will automatically create them for you. The constructor will call the default constructor on each element (and the base class if there is one) and the destructor will call the destructor of each class element.

However, since you have declared them, you also need to provide definitions.

answered May 15 '14 at 3:59

Sam S
61 3

The error is pretty clear:

```
Hw8_3.cpp:(.text+0x11): undefined reference to
SortableVector<int>::SortableVector()'
Hw8_3.cpp:(.text+0x138): undefined reference toSortableVector::~SortableVector()'
Hw8_3.cpp:(.text+0x170): undefined reference to `SortableVector::~SortableVector()'
collect2: error: ld returned 1 exit status
```

You're not defining the constructor and the destructor...

```
SortableVector() { }
~SortableVector() { }
```

Or if you want:

```
template<class T>
SortableVector::SortableVector()
{
...
}

template<class T>
SortableVector::~SortableVector()
{
...
}
```

Also, this line `L.SortableVector<int>::bubble_sort(a);` why not just `L.buble_sort(a);`
Also, here:

```
template <class T>
T SortableVector<T>::bubble_sort(vector<T> a)
```

Did you know you're passing a copy of `a` instead of a reference? This means you any modifications to that vector will not be live in `vector<int> a;` in your main function or whatever function you're calling it from:

```
template <class T>
T SortableVector<T>::bubble_sort(vector<T>& a)
```

Notice the `&`

This:

```
  vec[j] = vec[k];
  vec[k] = swap;
```

Did you mean `a` ?

This class member is useless:

```
vector<T> vec;
```

I'd suggest providing functions for appending to that member instead and remove the `vector<T>& a` parameter from the `bubble_sort` function in the class, then instead of making another vector in your main or w/e, use push on your `SortableVector` class instance.

answered May 15 '14 at 4:05

user1551592

```
#include <vector>
#include <algorithm> //This library give us some handy function swap
#include <iostream>
#include <random>

// This is a function to create random number between 1 and 10
static int random_int(){
    static std::random_device rd;
    static std::mt19937 prng{ rd() };
    static std::uniform_int_distribution<> d10{1, 10};
    return d10(prng);
}


template<class T>
```

```
class SortableVector{
    //By default element define in classes are private so no need to specify
    std::vector<T> v_;
public:
    //You dont need to specify constructor
    //Also, you dont need to specify destructor.
    //This is because the default constructor and destructor are ok.
    //The default constructor use item to item copy. In this case it's gonna copy
the vector val_ wich is ok.
    //For the destructor, we don't need to do anything

    //Usually you use const T& to avoid copy
    void push_back(const T& val){
        v_.push_back(val);
    }

    //Here i don't really know what u want to do.
    //Do you want to sort your inner vector and return it ?
    //Or do you want to make a copy and return it ?

    //Let's make a static method so it helps us covering both cases.
    //This method is static so it can be used outside of an instance.
    //It's like a function that is define inside the scope of this class.
    static void bubble_sort(std::vector<T>& v){
        //Using the correct type is always better.
        //In this case it's the type that define the size of this particular vector
is :
        size_t size = v.size();//Or with c++11 : auto size = v.size()
        bool change;
        do{
            change = false;
            for(int i = 0; i < size - 1; ++i){
                if(v[i-1] > v[i]){
                    std::swap(v[i-1],v[i]);
                    change = true;
                }
            }
        }
        while(change);
    }

    //This is the method version using the static one.
    std::vector<T>& bubble_sort(){
        bubble_sort(v_);
        return v_;
    }
};
int main() {
    SortableVector<int> ss;
    for(int k = 0; k < 10; ++k){
        ss.push_back(random_int());
    }
    for(auto& elem : ss.bubble_sort()){//C++ 11 foreach
        std::cout << elem << std::endl;
    }
}
```

answered Feb 23 '16 at 10:50

XOR
1,347 1 8 8