How to order set<pair<unsigned int, double> > by the second value of pair?

As the title says, I've built a `set` of `pair` of values that I need ordered by the `double value` (second):

```
set<pair<unsigned int, double> > s
```

c++ set order

asked Mar 24 '12 at 17:28

Luis
379 ● 2 ● 10 ● 24

Have you tried `std::sort`? – Jon Mar 24 '12 at 17:32

1 Duplicate of stackoverflow.com/questions/2958226/… ? – Guy Sirton Mar 24 '12 at 17:42

4 Answers

You should use the comparator:

```
struct Cmp
{
    bool operator ()(const pair<unsigned int, double> &a, const pair<unsigned int, double>
&b)
    {
        return a.second < b.second;
    }
};
```

and then you can define your set like:

```
set <pair<unsigned int, double>, Cmp> your_set;
```

answered Mar 24 '12 at 17:43

maxteneff
116 ● 5

`set`, in fact, has three parameters, the last two of which are optional. But in your case you want to use the second parameter, which tells how the elements will be sorted. By default (ignoring the third parameter), a `set` is:

```
set<typename Key, typename Compare = less<Key>>
```

You just need to write an appropriate class for the second parameter, such as:

```
template<typename Pair>
class CompareSecond
{
  bool operator()(const Pair& firstPair, const Pair& secondPair)
  {
    return firstPair.second < secondPair.second;
  }
}
```

And then you construct your `set` as:

```
typedef pair<unsigned int, double> MyPair;
set<MyPair, CompareSecond<MyPair>> s;
```

answered Mar 24 '12 at 17:47

Gorpik
7,696 ● 1 ● 17 ● 41

You could use a map instead of set. With a key doulbe and value unsigned int. std::map< double, unsigned int > s;

answered Mar 24 '12 at 17:33

AlexTheo
2,239 ● 1 ● 6 ● 24

In C++11 you will probably want to use a lambda. Unfortunately it is not possible to get the type of lambda in an unevaluated context. So we need to cheat a little to get argument deduction:

```
#include <set>

template<typename T, typename Cmp>
std::set< T, Cmp > make_set(Cmp cmp) {
  return std::set<T, Cmp>(cmp);
}

int main()
{
  auto s = make_set<std::pair<int, int> >(
    [](const std::pair<int, int>& p1, const std::pair<int, int>& p2)
    { return p1.second < p2.second; } );

  return 0;
}
```

`make_set` is a nice functor to have around but it wont work if you need to specify the allocator for the `set`.

answered Mar 24 '12 at 18:02

pmr
34.4k ● 4 ● 51 ● 104