Subject: The right container
Posted by koldo on Sat, 16 Feb 2019 16:54:18 GMT
View Forum Message <> Reply to Message

I wanted to know the right U++ container for this.
Data type could be int or double.

Container<Data type> dat;

dat.SetCount(300);
dat[23] = 12;
dat.FindAdd(34);
Thank you.

Subject: Re: The right container
Posted by Novo on Sat, 16 Feb 2019 19:53:06 GMT
View Forum Message <> Reply to Message

VectorMap<int, int>

Subject: Re: The right container
Posted by koldo on Sun, 17 Feb 2019 09:35:10 GMT
View Forum Message <> Reply to Message

Thank you Novo. But I want to save one data, not two.

Subject: Re: The right container
Posted by Novo on Sun, 17 Feb 2019 10:00:05 GMT
View Forum Message <> Reply to Message

koldo wrote on Sun, 17 February 2019 04:35Thank you Novo. But I want to save one data, not two.
In this case
dat[23] = 12;
won't work.

An one-value alternative is Index<int>, but you will be unable to write the code line above.
If your range of indices is small and sparse, you can just use regular Vector.
Another approach is to store data in a Vector and sort it, after that you will be able to use algorithms on sorted ranges.

Hope this helps.

Subject: Re: The right container
Posted by koldo on Sun, 17 Feb 2019 17:03:47 GMT
View Forum Message <> Reply to Message

Thank you Novo. Is what I have done.
This array is filled from text files with different formats. For some formats the Vector is best and for others Index is best.
I have used Vector, adding a FindAdd() function.

```
template <class Range, class V>
void FindAdd(Range& r, const V& value, int from = 0)
{
 for(int i = from; i < r.GetCount(); i++)
  if(r[i] == value)
   return;
 r.Add(value);
}
```

Subject: Re: The right container
Posted by mirek on Sun, 24 Feb 2019 10:28:00 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 17 February 2019 11:00koldo wrote on Sun, 17 February 2019 04:35Thank you Novo. But I want to save one data, not two.
In this case
dat[23] = 12;
won't work.

An one-value alternative is Index<int>, but you will be unable to write the code line above.
If your range of indices is small and sparse, you can just use regular Vector.
Another approach is to store data in a Vector and sort it, after that you will be able to use algorithms on sorted ranges.

Hope this helps.

dat.Set(23, 12);

Mirek

Subject: Re: The right container
Posted by mirek on Sun, 24 Feb 2019 10:30:07 GMT
View Forum Message <> Reply to Message

koldo wrote on Sat, 16 February 2019 17:54I wanted to know the right U++ container for this.
Data type could be int or double.

Container<Data type> dat;

dat.SetCount(300);
dat[23] = 12;
dat.FindAdd(34);
Thank you.

I would be helpful if you have explained what is supposed to happen at each point. Esp. what you expect from SetCount and FindAdd...

(But in general, I would say use Index. That really IS Vector with search...)

Mirek

Subject: Re: The right container
Posted by koldo on Sun, 24 Feb 2019 17:29:43 GMT
View Forum Message <> Reply to Message

Hi Mirek

It is used in a program that reads data from different file formats. In one of them the size is known in the beginning and data is filled randomly when reading the file. In other format data is got sequentially and you do not know total size until the end. It is possible to scan the file first to get the dimension, although it is not so efficient and a little more complex.

However do not worry, files are not huge and Vector with FindAdd works adequately.
