Subject: Linear sorted array vs. tree
Posted by mirek on Mon, 15 Jan 2007 13:33:25 GMT
View Forum Message <> Reply to Message

For some time I was thinking about adding something like this container to U++ Core:

```
template <class T>
class Order {
 Vector<T> data;

public:
 int  Find(const T& x);
 int  FindAdd(const T& x);
 void Add(const T& x);
 const T& operator[](int i) const { return data[i]; }
 int  GetCount()                  { return data.GetCount(); }
};

template <class T>
void Order<T>::Add(const T& x)
{
 data.Insert(FindLowerBound(data, x), x);
}

template <class T>
int Order<T>::Find(const T& x)
{
 return FindBinary(data, x);
}

template <class T>
int Order<T>::FindAdd(const T& x)
{
 int i;
 i = FindUpperBound(data, x);
 if(i && data[i - 1] == x)
  return i;
 data.Insert(i, x);
 return i;
}
```

basically, sorted array used as associative container.

- advantage: less data used, so it could be useful for storing small maps.

- disadvantage - gets slow if there is too much data inserted (Insert slows it down).

Today I decided to put the disadvantage to the test and benchmark it against tree (std::set).

To my surprise, if we consider rather unlikely 2:1 find:insert ratio, this trivial implementation keeps the pace with std::set up to about 2000 elements! I am surprised.

Mirek

P.S.: Does not mean that Order will be part of U++ Core soon, it was just experiment...
