Subject: Transfer Semantics - Initializer list and copy constructors
Posted by captainc on Tue, 10 Jul 2007 13:33:26 GMT
View Forum Message <> Reply to Message

I've been trying to get used to Upp NTL library and the pick semantics, an idea that I think is great, but still have a few questions.

Mainly, do initializer lists use deep copy/copy constructors? Also, the do Vector and Array Add() functions use copy constructors while the AddPick() functions do not?

I read the NTL tutorial and the Transfer Semantics section, but I think you should improve on this section with other examples of using it and not using it so that it is easier to grasp at first. I would love to see examples for appending Arrays or Vectors and an expanded explanation on the use of the Pick() and Copy() functions. This is all probably because I am still learning C++ as a language, and want to see it easy to learn and know the differences between standard implementations and NTL; especially for the cases where we use libraries that utilize standard library containers and we have to use them along with NTL containers.

Subject: Re: Transfer Semantics - Initializer list and copy constructors
Posted by mirek on Tue, 10 Jul 2007 20:11:01 GMT
View Forum Message <> Reply to Message

The simple rule: Do what you need. If you get crash because of broken pick semantics (the ASSERT in Vector with items = -1), use "deep copy variant" (<<= instead of =, constructor with additional int parameter (e.g. 1).

Subject: Re: Transfer Semantics - Initializer list and copy constructors
Posted by forlano on Sat, 29 Sep 2007 22:27:31 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 10 July 2007 22:11The simple rule: Do what you need. If you get crash because of broken pick semantics (the ASSERT in Vector with items = -1), use "deep copy variant" (<<= instead of =, constructor with additional int parameter (e.g. 1).

Hello,

for first time I needed to copy the content of a vector in another vector WITHOUT to destroy the original vector. This is my class:

```
class Person : Moveable<Person> {
 int  currentProposer;
 int  cursor;
  public:
 Index<int> preference;
```

```
 int  RankProposer(int k);
       int  GetNext();     int  GetCurrentRank();    int  GetCurrentProposer();
       int  GetProposedTo();
 int  FindPosition(int k);
 void AcceptProposal(int k);
 String ToString() const { String s; for (int i=0; i<preference.GetCount();i++)
s<<AsString(preference[i]) + ' '; return s; }
 Person() {currentProposer = 0; cursor = -1; }
};
```

Then I declared two vectors:

Vector<Person> M, R;

After some operations made with M I would like to copy M in R. I used
R<<=M;
But the program crashed. Which is the correct syntax for this task?
Thanks,
Luigi

Subject: Re: Transfer Semantics - Initializer list and copy constructors
Posted by mirek on Sun, 30 Sep 2007 06:32:11 GMT
View Forum Message <> Reply to Message

forlano wrote on Sat, 29 September 2007 18:27luzr wrote on Tue, 10 July 2007 22:11The simple rule: Do what you need. If you get crash because of broken pick semantics (the ASSERT in Vector with items = -1), use "deep copy variant" (<<= instead of =, constructor with additional int parameter (e.g. 1).

Hello,

for first time I needed to copy the content of a vector in another vector WITHOUT to destroy the original vector. This is my class:

```
class Person : Moveable<Person> {
 int  currentProposer;
 int  cursor;
  public:
 Index<int> preference;
 int  RankProposer(int k);
       int  GetNext();     int  GetCurrentRank();    int  GetCurrentProposer();
       int  GetProposedTo();
 int  FindPosition(int k);
 void AcceptProposal(int k);
 String ToString() const { String s; for (int i=0; i<preference.GetCount();i++)
s<<AsString(preference[i]) + ' '; return s; }
 Person() {currentProposer = 0; cursor = -1; }
```

};

Then I declared two vectors:

Vector<Person> M, R;

After some operations made with M I would like to copy M in R. I used
R<<=M;
But the program crashed. Which is the correct syntax for this task?
Thanks,
Luigi

Unfortunately, this case is a little bit more complicated, you also have to provide deep copy operations for Person (which, frankly, is a tedious process, but there is no other way).

At minimum provide Person(const Person&, int) constructor that creates deep copy, and use DeepCopyOption template as base class to generate the rest.

Search

http://www.ultimatepp.org/srcdoc$Core$pick_$en-us.html

for "DeepCopyOption".

Mirek
