Subject: Please help! New transfer semantic issue!
Posted by sergeynikitin on Mon, 21 Mar 2016 11:36:49 GMT
View Forum Message <> Reply to Message

Maybe problem is trivial, but I stopped on it.

There is class Line

```
class Line {
    ....
       Line( Point p1, Point p2 ) // constructor
}
```

and class Figure

```
class Line {
  public:
      Array<Line> lines;
}
```

and if I use compiler option -std=c++11

I get error message while adding new Line to Lines

```
lines.Add(Line(p1,p2));
```

Without option -std=c++11 all OK.

What right way to do this with -std=c++11 option?

Subject: Re: Please help! New transfer semantic issue!
Posted by dolik.rce on Mon, 21 Mar 2016 12:52:55 GMT
View Forum Message <> Reply to Message

sergeynikitin wrote on Mon, 21 March 2016 12:36if I use compiler option -std=c++11

I get error message while adding new Line to Lines

```
lines.Add(Line(p1,p2));
```

Hi Sergey,

What does the error message say? It is really hard to guess without seeing the entire class nor the error...

Best regards,
Honza

Subject: Re: Please help! New transfer semantic issue!
Posted by Novo on Mon, 21 Mar 2016 12:55:17 GMT
View Forum Message <> Reply to Message

AFAIK, in case of C++11 you should explicitly call pick() here.
Mirek is working on refactoring of Upp, so this requirement will be gone soon.

Subject: Re: Please help! New transfer semantic issue!
Posted by sergeynikitin on Mon, 21 Mar 2016 13:02:31 GMT
View Forum Message <> Reply to Message

Error message is

required from 'T& Upp::Array<T>::Add(const T&) [with T = Line]'
error: use of deleted function 'Line::Line(const Line&)'

Subject: Re: Please help! New transfer semantic issue!
Posted by sergeynikitin on Mon, 21 Mar 2016 13:07:21 GMT
View Forum Message <> Reply to Message

Code of adding

```
void AddLine(Point p1, Point p2){
 lines.Add(Line(p1,p2));
 moving_line = lines.GetCount()-1;
}
```

Subject: Re: Please help! New transfer semantic issue!
Posted by forlano on Mon, 21 Mar 2016 13:47:21 GMT
View Forum Message <> Reply to Message

sergeynikitin wrote on Mon, 21 March 2016 14:07Code of adding

void AddLine(Point p1, Point p2){

```
 lines.Add(Line(p1,p2));
 moving_line = lines.GetCount()-1;
}
```

Hi Sergey,

I guess the error disappear if you use

void AddLine(Point& p1, Point& p2)

Luigi

Subject: Re: Please help! New transfer semantic issue!
Posted by sergeynikitin on Mon, 21 Mar 2016 15:22:38 GMT
View Forum Message <> Reply to Message

Thanks for opinion.
But Next function has no parameters:

```
void AddMovingLine(){
 lines.Add(Line(checked_elem_point+Point(0,0),checked_elem_point+Point(0,0)));
 moving_line = lines.GetCount()-1;
}
```

But have same problem.

There are problems while creating array item.

Subject: Re: Please help! New transfer semantic issue!
Posted by Lance on Mon, 21 Mar 2016 21:48:12 GMT
View Forum Message <> Reply to Message

Quote:
Error message is

required from 'T& Upp::Array<T>::Add(const T&) [with T = Line]'
error: use of deleted function 'Line::Line(const Line&)'

it appears that you have explicitly deleted copy ctor for Line class.

change the line to

lines.AddPick(Line(p1,p2));

should fix your problem.

Subject: Re: Please help! New transfer semantic issue!
Posted by mirek on Tue, 22 Mar 2016 05:18:58 GMT
View Forum Message <> Reply to Message

This depends on Line definition. Could you post complete Line?

Mirek

Subject: Re: Please help! New transfer semantic issue!
Posted by sergeynikitin on Tue, 22 Mar 2016 06:18:26 GMT
View Forum Message <> Reply to Message

Yes.

```
class Line {
  typedef Line CLASSNAME;
 public:
  void Paint ( Draw& w, bool moving );
  Line();
  Line( Point _p1, Point _p2 );
  String ToString() const;
 public:
  Point p1, p2;
  ElementPin* elp1;
  ElementPin* elp2;
  Array<Segment> segs;
  Array<ElementPin> lines;
};
```

Subject: Re: Please help! New transfer semantic issue!
Posted by Lance on Tue, 22 Mar 2016 20:47:49 GMT
View Forum Message <> Reply to Message

Contained members Array segs and Array<ElementPin> lines are the culprits.

Mirek would be able to give you the best suggestion. However explicitly define a copy constructor

and a move constructor will fix the problem.

Something like

```
// public:

    Line(const Line& ln):p1(ln.p1),p2(ln.p2), elp1(???),elp2(???),
                  segs(ln.segs,1), // make a deep copy
                  lines(ln.lines,1)  // ditto
                 {}

    Line(Line&& ln): p1(ln.p1),p2(ln.p2),elp1(???), elp2(???),
                  segs(pick(ln.segs)),
                  lines(pick(ln.lines))
                 {}
```

Subject: Re: Please help! New transfer semantic issue!
Posted by mirek on Fri, 25 Mar 2016 17:54:00 GMT
View Forum Message <> Reply to Message

I you want to maintain pick/clone distinction, you should implement "clone" constructor/operator:

```
    Line(const Line& ln, int):p1(ln.p1),p2(ln.p2), elp1(???),elp2(???),
                  segs(clone(ln.segs)), // make a deep copy
                  lines(clone(ln.lines))  // ditto
                 {}
// and operator=....
```

If you do not care, you can also use WithDeepCopy

```
...
WithDeepCopy<Array<Segment> > segs;
...
```

Things might get simpler with C++11...

Mirek

Subject: Re: Please help! New transfer semantic issue!
Posted by sergeynikitin on Wed, 06 Apr 2016 08:48:08 GMT
View Forum Message <> Reply to Message

Thanks. It's works with or without C++11 option!
Real crossplatforming (even through C++ version :))!