Subject: Issues using 2015.2 version
Posted by Giorgio on Thu, 21 Jan 2016 10:31:33 GMT
View Forum Message <> Reply to Message

Hi there,
I have just a brand new pc with windows 10. I used the 2015.1 ultimate++ version till yesterday. So, new OS, new upp version and new compiler... I am having trouble compiling my old projects and having all this new stuff I do not know where the problem sits. In this specific case, I am trying to compile a test code for a barcode generator library. The library was downloaded here in the forum. I am trying to compile the test application that comes with the library, and get the following error:

Barcode\Code128.cpp(701): error C2280: 'Upp::Vector<Upp::byte> &Upp::Vector<Upp::byte>::operator =(const Upp::Vector<Upp::byte> &)': trying to reference a deleted function

The same application compiles fine on my previous platform: windows 7, upp 2015.1 and MSC 12.

Any suggestion?
Thanks,
Gio

Subject: Re: Issues using 2015.2 version
Posted by dolik.rce on Thu, 21 Jan 2016 12:17:57 GMT
View Forum Message <> Reply to Message

Hi Giorgio,

Wild shoot in the dark: Have you used C++11 before the upgrade? If I remember correctly, deleted assignment operator can happen only with C++11, so perhaps the package is not quite C++11 ready.

Best regards,
Honza

Subject: Re: Issues using 2015.2 version
Posted by Giorgio on Thu, 21 Jan 2016 12:48:24 GMT
View Forum Message <> Reply to Message

Hi Dolik,
actually I do not really know :(, programming is not my first duty and that package was found in the forum...
Regards,

Gio

Subject: Re: Issues using 2015.2 version
Posted by Lance on Thu, 21 Jan 2016 17:34:28 GMT
View Forum Message <> Reply to Message

Not sure if you are directly using the BarcodeTest program, anyway, the following line generates the problem you referred to

```
GUI_APP_MAIN
{
 auto a=MyApp();                    <============ This line
 a.Sizeable().MaximizeBox();
 a.Maximize();
 a.Run();

}
```

while MyApp has this definition

```
struct MyApp : WithBarcodeTestLayout<TopWindow>
{
 MyApp(){
  CtrlLayout(*this, "Barcode Test");
  input<<=THISBACK(Updated);
  print.SetImage(CtrlImg::print())<<=THISBACK(Print);
  top<<=2;
  left<<=2;
 }

 virtual void Paint(Draw& w);

 void GenBarcode();
 void Print();

 void Updated()
 {
  Refresh();
  Code128 c(String().Cat()<<~input);
  EAN ean(AsString(~input));
  PDF417 pdf(AsString(~input));
  richview.SetQTF(String("[ ").Cat()
```

```
   <<c.DisplayText("Hi, U++ user!")
    .Color(Red()).BarRatio(36)
   <<"&&" // new line
   <<ean.Type(EAN::EAN8UPCAEAN13)
    .DisplayText("Best Seller")
   <<"&&"
   <<pdf.FixedAspectRatio(3,2)
  );
 }

 typedef MyApp CLASSNAME;

};
```

There are not even any data members (except ones inherits from its ancestors) in the MyApp struct definition, so it could not be because of anything from my barcode library.

I am also puzzled on that line. Logically local variable a should be default constructed once, instead of a default construct of a temporary and then an assignment, so it should be equivalent to

```
    MyApp a;
```

Apparently I was wrong. Anyway, change the line to above fixed the problem. I was wondering why should I use the more cumbersome form at first place. So if you are using the BarcodeTest program directly, change the GUI_MAIN to the following form will fix your problem:

```
GUI_APP_MAIN
{
 MyApp a;
 a.Sizeable().MaximizeBox();
 a.Maximize();
 a.Run();

}
```

HTH

Subject: Re: Issues using 2015.2 version
Posted by Mindtraveller on Thu, 21 Jan 2016 19:16:11 GMT
View Forum Message <> Reply to Message

Hi, Giorgio.
Good news is that I know exactly why you have such error.
It's simple: your new compiler is C++11 compiler. And thus U++ is compiled in C++11 mode. It means, among other things, that you must now specify whether you use Pick or Clone operation.
For example, this code is valid for U++ in 'old' mode:
Vector<int> a,b; a = b;
But it fails to compile in U++11 mode because you should use pick/clone and call it explicitly:
Vector<int> a,b; a = pick(b);
Of course, you may use clone() only if your object supports it.
Please refer to the updated Help documentation for details.

Subject: Re: Issues using 2015.2 version
Posted by Giorgio on Fri, 22 Jan 2016 14:08:20 GMT
View Forum Message <> Reply to Message

Hi there,
first of all I want say a big thank you to everybody. Dorik noticed who the problem was a c++ 11 related issue, Lance who developed the library and find the problem and MindTraveller who solved the puzzle. I really appreciate your support. You guys made the open source great.

That said, and going back to my problem, the problem is solved :). The issue was at line 701 of the "code128.cpp" source file in the BarCode package. There is here an assignment not valid for c++ 11. This is how it is:

data=enc.GetEncoded();

Adding pick or clone solves the issues:

data= clone(enc.GetEncoded());

As I said before I am not a really expert c++ developer (my duties are just 15-20% related to programming and less than 5% related to c++ programming), so I do not know if is better using pick or clone. In my case both work. Any suggestion about what is better are welcome. Finally, maybe Lance can update its library (for future users).
Regards,
Giorgio

Subject: Re: Issues using 2015.2 version
Posted by mr_ped on Sat, 23 Jan 2016 11:08:40 GMT
View Forum Message <> Reply to Message

I think this ideal case for pick(), as you pick up content of return value (leaving it empty), which is discarded anyway right away.

In programming you always should be sort of aware how you use the computer memory.
In C++ you have great control over it, actually just by reading source you can tell quite exactly, how the memory is used.

Vector is built of two parts, the Vector container data like size, capacity and pointer to the data itself; this part is of constant memory size, and is cheap to allocate/copy/destroy, so it's often instantiated on the actual scope stack.
And then there're the data itself, which are allocated on the heap memory, as there may be lot of them, and the pointer to the heap is stored inside the container inner structure.

pick() is sort of "copy constructor", which will copy the size/capacity/pointer from one Vector to the other Vector container, and it will disconnect and invalidate the old container's data pointer. Effectively "moving" the Vector without copying millions of data, only the small constant size Vector container inner structure is copied.

clone() is full copy constructor, which will allocate another (second) block of heap memory for the actual data inside the new Vector instance, and then will copy the actual data from one block of heap memory to the new one.
The old instance of Vector remains fully operational. This is performance expensive operation, and you don't want to do it, unless you actually need a second copy of data.

So when you have a cheap way to move the Vector data around with "pick", you can instantiate Vector inside function, fill it up with data, return the temporary function instantiated Vector (to be released on the very next line after return from function), and pick the data payload to caller's Vector instance, saving them from release, but not copying them one by one.

If you are still confused, which of those two used, use always "pick" in debug mode. In case you mess it up, and access the picked container after, it will crash with some assertion, that you tried to access data from picked Vector, then you can decide if you need a copy, or you accidentally access old Vector instead of the new one.

In case of function returning Vector there's no reason for "clone()", as the source instance is destroyed right away in the function call clean up.

Subject: Re: Issues using 2015.2 version
Posted by Oblivion on Sat, 23 Jan 2016 21:59:27 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Sat, 23 January 2016 13:08I think this ideal case for pick(), as you pick up content of return value (leaving it empty), which is discarded anyway right away.

In programming you always should be sort of aware how you use the computer memory.
In C++ you have great control over it, actually just by reading source you can tell quite exactly, how the memory is used.

Vector is built of two parts, the Vector container data like size, capacity and pointer to the data

itself; this part is of constant memory size, and is cheap to allocate/copy/destroy, so it's often instantiated on the actual scope stack.
And then there're the data itself, which are allocated on the heap memory, as there may be lot of them, and the pointer to the heap is stored inside the container inner structure.

pick() is sort of "copy constructor", which will copy the size/capacity/pointer from one Vector to the other Vector container, and it will disconnect and invalidate the old container's data pointer. Effectively "moving" the Vector without copying millions of data, only the small constant size Vector container inner structure is copied.

clone() is full copy constructor, which will allocate another (second) block of heap memory for the actual data inside the new Vector instance, and then will copy the actual data from one block of heap memory to the new one.
The old instance of Vector remains fully operational. This is performance expensive operation, and you don't want to do it, unless you actually need a second copy of data.

So when you have a cheap way to move the Vector data around with "pick", you can instantiate Vector inside function, fill it up with data, return the temporary function instantiated Vector (to be released on the very next line after return from function), and pick the data payload to caller's Vector instance, saving them from release, but not copying them one by one.

If you are still confused, which of those two used, use always "pick" in debug mode. In case you mess it up, and access the picked container after, it will crash with some assertion, that you tried to access data from picked Vector, then you can decide if you need a copy, or you accidentally access old Vector instead of the new one.

In case of function returning Vector there's no reason for "clone()", as the source instance is destroyed right away in the function call clean up.

+1

This clear and simple explanation of the pick() and clone() taking vector as an example, should be included in the U++ docs, really. :)

Regards,

Oblivion

Subject: Re: Issues using 2015.2 version
Posted by Giorgio on Mon, 25 Jan 2016 08:48:14 GMT
View Forum Message <> Reply to Message

Really clean and neat explanation, in my local copy of the library I used clone() but indeed a copy of the vector is not necessary, pick() will fit.

Subject: Re: Issues using 2015.2 version
Posted by forlano on Tue, 26 Jan 2016 22:32:14 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 21 January 2016 20:16
It's simple: your new compiler is C++11 compiler. And thus U++ is compiled in C++11 mode. It means, among other things, that you must now specify whether you use Pick or Clone operation.
For example, this code is valid for U++ in 'old' mode:
Vector<int> a,b; a = b;
But it fails to compile in U++11 mode because you should use pick/clone and call it explicitly:
Vector<int> a,b; a = pick(b);
Of course, you may use clone() only if your object supports it.
Please refer to the updated Help documentation for details.

Hello,

Very interesting. I am getting a lot of similar errors (C2280) even where old assignment between Vector<> is not apparently involved. For example when Vector<int> is passed in a function.
Moreover in my case the pick/clone trick does not work.
Anyway I would like to come back to the old U++ good way and get my code compiled correctly.
Does anybody know how to set VC2015 in order to avoid the C2280 error?

Thanks,
Luigi

Subject: Re: Issues using 2015.2 version
Posted by mirek on Wed, 27 Jan 2016 06:27:32 GMT
View Forum Message <> Reply to Message

Hi,

I would rather recommend fixing the code. Could you please show me the example of code that is causing the problem?

Note that very often, old behavior masked bugs.

Mirek

Subject: Re: Issues using 2015.2 version
Posted by forlano on Wed, 27 Jan 2016 20:10:00 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 27 January 2016 07:27Hi,

I would rather recommend fixing the code. Could you please show me the example of code that is

causing the problem?

Here it is one:

```
class ReadExcelDlg : public WithReadExcel<TopWindow> {
 typedef ReadExcelDlg CLASSNAME;
 OfficeSheet sheet;
 void MakeList();
 void SaveDetailRecord(int count, int idteam, Vector<String> p);
 int  SaveMasterRecord(Vector<String> t);
public:
 bool updated;
 void SetTournamentDir( String dir);
 void ReadSheetDetail();
 ReadExcelDlg();
};
// code snippet

  Vector<String> t;
              //...
  int idteam = SaveMasterRecord( t); <--- error C2280
```

another

```
class Pair : Moveable<Pair> {
public:
 int idw, idb;
 String ToString() const { return AsString(idw) + ' ' + AsString(idb); }
 Pair(int idw, int idb) : idw(idw), idb(idb) {}
 Pair() {}
};

class SortPairing {
 Vector<Pair> pairs;
 int comppair(int j1, int j2);
 public:
 Vector<Pair> SortPairs(Vector<Pair> pairing);
 SortPairing() {}
 ~SortPairing(){;}
};

  //code snippet
       SortPairing S, pairs;
       //...
```

```
Vector<Pair> sortedPairing;
    sortedPairing = S.SortPairs(pairs);  <--- error C2280
```

There are several of this kind that appear when Vector<something> is passed in a function.
With non C11 everything worked without complain.

Thanks!
Luigi

Subject: Re: Issues using 2015.2 version
Posted by mr_ped on Wed, 27 Jan 2016 21:42:21 GMT
View Forum Message <> Reply to Message

Do you really want to pass full copy?

I think for example for "Save" a const reference is much better, i.e.:
int SaveMasterRecord(const Vector<String> & t);

About the sort... you probably want to keep "pairs" intact?
Then you can again use const reference as input for the function.

But maybe you should check for the built-in Sort, like this:

```
Vector<int> pairs{1, 4, 2, 3};
Vector<int> sortedPairs = clone(pairs);
Sort(sortedPairs, [](const int & a, const int & b){return a < b;} );
// lambda expression used for simple "a is less than b" predicate here
// to give you idea, how you can code your own comparator for "Pair" type
//(the "less than" is default predicate of Upp::Sort(T & container); )
```

So if you would define operator < in Pair class, you would be able to write:

```
Vector<Pair> sortedPairs = clone(pairs);
Sort(sortedPairs);
```

Anyway, passing full copy of Vector to function does make little sense to me, only when you actually store the parameter somewhere... but in most cases you want either just to read the data (const Vector & v), or you want to
manipulate them, but the owner of the Vector is still the caller, so you want to operate on his instance: (Vector & v);

Subject: Re: Issues using 2015.2 version

Posted by forlano on Thu, 28 Jan 2016 14:42:06 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Wed, 27 January 2016 22:42Do you really want to pass full copy?

Hi,

thanks for the answer. That code was a bit old. It never gave problem and I forgot it. You are right of course.
After your observation I cleaned it and used Vector<...>& everywhere.

Now it compiles but fail to link 8o with error:

regarding Browser.lib. Does anybody know what is it?

Thanks again,
Luigi

File Attachments
1) 2016-01-28_153308.png, downloaded 677 times

Subject: Re: Issues using 2015.2 version
Posted by forlano on Sat, 30 Jan 2016 13:03:15 GMT
View Forum Message <> Reply to Message

forlano wrote on Thu, 28 January 2016 15:42mr_ped wrote on Wed, 27 January 2016 22:42Do you really want to pass full copy?

Hi,

thanks for the answer. That code was a bit old. It never gave problem and I forgot it. You are right of course.
After your observation I cleaned it and used Vector<...>& everywhere.

Now it compiles but fail to link 8o with error:

Solved.
I needed to remove the package Ide/Browser from my project. I do not know how they entered in my package being foreigner.

thank you,
Luigi