Subject: why no virtual destructor for RegExp?
Posted by navi on Sat, 19 Jan 2013 05:22:49 GMT
View Forum Message <> Reply to Message

I was thinking of extending the RegExp class of the pcre plugin and implementing a ReplaceMatch() for it. Sort of like a replica of preg_replace() of php or String.Replace() of javascript. However while digging through RegExp I realize that RegExp doesn't have a virtual destructor! So there are no safe ways of extending RegExp class other-then to use private inheritance, composition or free-functions instead. In other word, extending RegExp safely is not possible!?

Regards
navi

Subject: Re: why no virtual destructor for RegExp?
Posted by mirek on Sat, 19 Jan 2013 11:03:34 GMT
View Forum Message <> Reply to Message

You are right. I think the author considers RegExp to be "sealed". Your contract is with public interface only.

Mirek

Subject: Re: why no virtual destructor for RegExp?
Posted by navi on Sat, 19 Jan 2013 11:30:58 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 19 January 2013 12:03You are right. I think the author considers RegExp to be "sealed". Your contract is with public interface only.

Mirek

Thanks Mirek. I guess I better get started with writing my own RegExp class then.

Regards
Navi

Subject: Re: why no virtual destructor for RegExp?
Posted by dolik.rce on Sat, 19 Jan 2013 12:19:09 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 19 January 2013 12:03You are right. I think the author considers RegExp to be "sealed". Your contract is with public interface only.

Mirek
Hi guys,

What about updating the RegExp class? I personally would like it if it had ReplaceMatch(). As long as the addition doesn't break backward compatibility I don't see any reason not to add it. The class is definitely not 'sealed' for this, I remember adding the Study() method to it a while ago

Navi, would you mind contributing your code to U++?

Best regards,
Honza

Subject: Re: why no virtual destructor for RegExp?
Posted by navi on Sat, 19 Jan 2013 12:26:49 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 19 January 2013 13:19

Navi, would you mind contributing your code to U++?

Best regards,
Honza

Not at all. let me get cracking. lets see if my codes is up to the u++ standers or not. will post the code here asap.

Thanks & Regards
navi.

*off-topic any thoughts on this and this?

Subject: Re: why no virtual destructor for RegExp?
Posted by navi on Fri, 25 Jan 2013 13:58:33 GMT
View Forum Message <> Reply to Message

Hi All,

Just an update what I am up to so far. I manage to code a replica of preg_replace of php. due to

the difference in nature of variable type, I had to almost 4 overloaded function to match all the things "preg_replace" can do in a single function definition.

```
//=================================================================================
//        My Extension to RegExp
//=================================================================================

protected:
 bool ReplacePos(String& t, int p, int q, const String r);
 int  Replace0(String& t, Vector<String>& rv, const int& rv_count, int& offset);
 Vector<String> ResolveBackRef(Vector<String>& rv);
 Vector<String> Make_rv(const String& r);


public:

 int  ReplaceGlobal(String& t, Vector<String>& rv, const String pattern=Null, bool backref=false);
//DONE
 int  ReplaceGlobal(String& t, Vector<String>& rv, int offset, const String pattern=Null, bool
backref=false); //DONE

 int  ReplaceGlobal(String& t, const String r, const String pattern=Null, bool backref=false);
//DONE
 int  ReplaceGlobal(String& t, const String r, int offset, const String pattern=Null, bool
backref=false); //DONE

 int  ReplaceGlobal(String& t, Callback cbr, const String pattern=Null, bool backref=false); // NOT
YET
 int  ReplaceGlobal(String& t, Callback cbr, int offset, const String pattern=Null, bool
backref=false); //NOT YET
```

My plan is to implement 6 flavor of ReplaceGlobal() and 6 flavor of Replace(). all based on how replace text or callback function are given.

replace text can either be given as single String in Parentheses format just like a regular expression. ie (replace_a)(replace_b)(replace_c). Or can be given a vector containing three Strings. Also callback function can be passed in place of replace texts.

regexp pattern can also be given as argument. instead of setting it using SetPattern()

backref = is a on/off switch for resolving/expanding back reference given in replace text as /1, /2, /3 etc. where the number is the number of the match string cough by the regexp pattern.

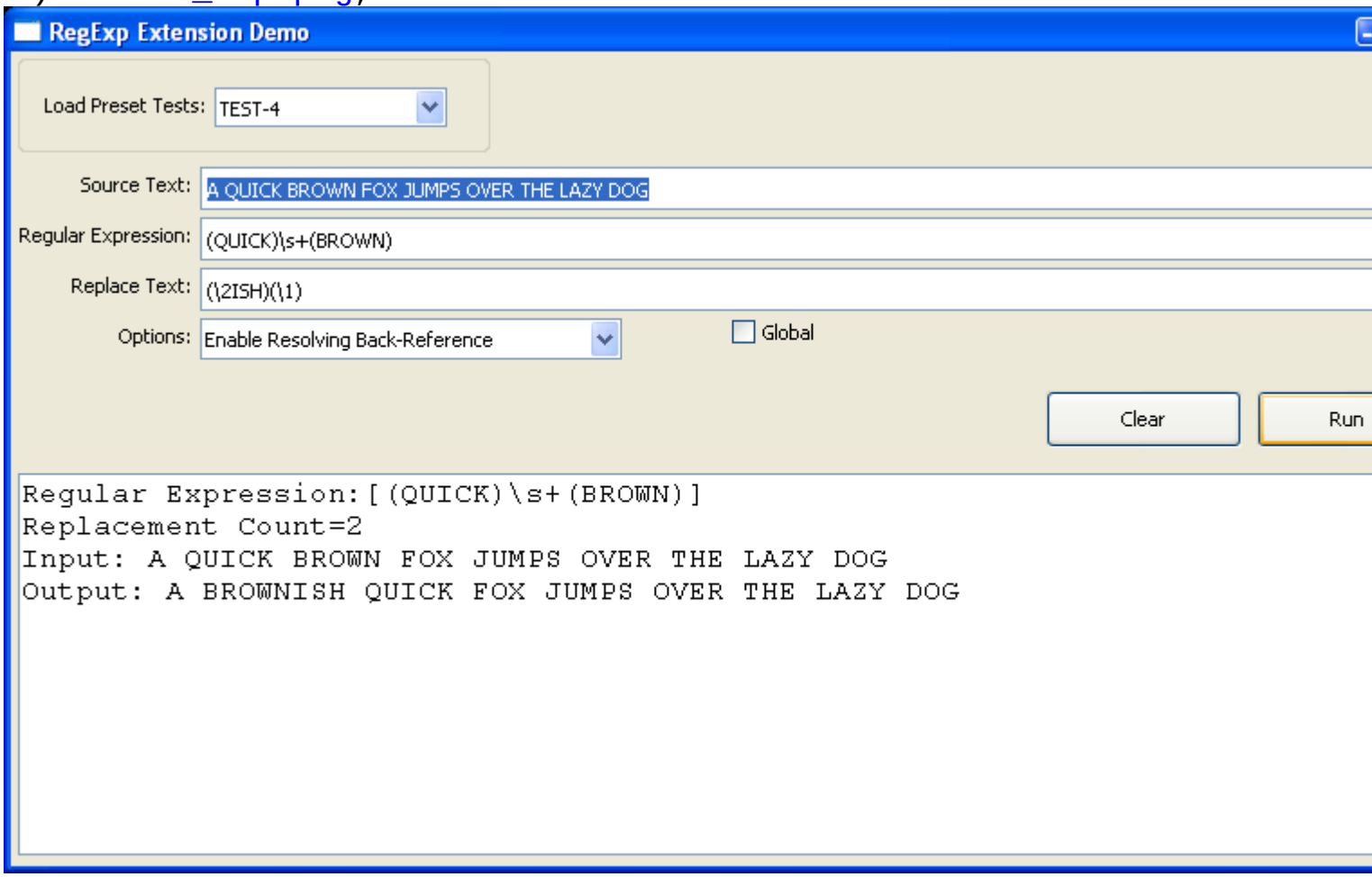
Thanks & Regards
Navi

Subject: Re: why no virtual destructor for RegExp?
Posted by navi on Fri, 25 Jan 2013 14:01:02 GMT
View Forum Message <> Reply to Message

Screen Cap.

File Attachments

1) screen_cap.png, downloaded 801 times



Subject: Re: why no virtual destructor for RegExp?
Posted by dolik.rce on Fri, 25 Jan 2013 15:04:28 GMT
View Forum Message <> Reply to Message

Hi Navi

IMHO the pattern parameter should not be there, it is kind of confusing and against the whole idea that RegExp object represents the regexp pattern. If you want to have a chance to specify the

replace pattern a separate global function might be better for this... It could use the RegExp object internaly, so it's not much work for you

Quote: int ReplaceGlobal(String& t, const String r, const String pattern=Null, bool backref=false); //DONE
 int ReplaceGlobal(String& t, const String r, int offset, const String pattern=Null, bool backref=false); //DONE

Why not simply int ReplaceGlobal(String& t, const String r, offset=0, bool backref=false); ... and similar for the vector and callback variant. It would make the interface simpler, without reducing its capabilities.

Note: The above is not any official requirement, just my ideas and opinions

Honza

Subject: Re: why no virtual destructor for RegExp?
Posted by dolik.rce on Fri, 25 Jan 2013 15:10:41 GMT
View Forum Message <> Reply to Message

PS: I forgot... About the returning, I think it is okay to modify the referenced string. If necessary it should be simple to add a another method with separate string for output, that would not modify the input.

Subject: Re: why no virtual destructor for RegExp?
Posted by navi on Fri, 25 Jan 2013 16:37:28 GMT
View Forum Message <> Reply to Message

Quote:IMHO the pattern parameter should not be there, it is kind of confusing and against the whole idea that RegExp object represents the regexp pattern. If you want to have a chance to specify the replace pattern a separate global function might be better for this... It could use the RegExp object internally, so it's not much work for you
You are right. I think its best to get rid of pattern from arguments. Actually I was trying to match preg_replace style. which is not particularly suited for regexp class.

Quote:Why not simply
int ReplaceGlobal(String& t, const String r, offset=0, bool backref=false);

well you see, the functions Match() and GlobalMatch() has no option for offset. in order for offset to work I had to device my own function using Execute(). however then I realize if written using GlobalMatch() even-though no option for offset, but it is much more solid and more compatible then to my own version with offset. now I wanted to have the best of both hence two different function and not option=0 as default value. I believe we can still mimic the default value effect if

we write the following:

```
int  ReplaceGlobal(String& t, const String r, bool backref=false);
int  ReplaceGlobal(String& t, const String r, offset, bool backref=false);
```

essentially if offset is not given then the function using GlobalMatch() will be called. if offset given, then the version using Execute() will be called instead. or perhaps drop the whole idea of starting offset, like the author of Match() and GlobalMatch()? I cant get my head around function GlobalMatch(). if I could understand it fully, then I can rewrite the exact thing using Execute() alone and we then could have one function for like you pointed out.

Quote:PS: I forgot... About the returning, I think it is okay to modify the referenced string. If necessary it should be simple to add a another method with separate string for output, that would not modify the input.
I agree, I have no problem doing either way. the only issue being if we wish to return the output text then, we will have 2 return values, 1: the replace count, 2: the output text. one of which have to go, or go in to function argument as reference variable.

Thanks for all your help & pointers.
Regards
Navi

Subject: Re: why no virtual destructor for RegExp?
Posted by dolik.rce on Fri, 25 Jan 2013 18:16:17 GMT
View Forum Message <> Reply to Message

Btw, there is a quite nice looking C++ wrapper around PCRE by Google. You might find some good inspiration in there

Honza

Subject: Re: why no virtual destructor for RegExp?
Posted by navi on Mon, 28 Jan 2013 06:11:31 GMT
View Forum Message <> Reply to Message

DONE.

Final Interface. Got rid of pattern and offset. 3 flavors of each. Also updated the demo, see updated screen cap above.

```
int  Replace(String& t, Vector<String>& rv, bool backref=false); //DONE
```

```
int  Replace(String& t, const String r, bool backref=false); //DONE
int  Replace(String& t, Callback1<Vector<String>&> cbr); //DONE

int  ReplaceGlobal(String& t, Vector<String>& rv, bool backref=false); //DONE
int  ReplaceGlobal(String& t, const String r, bool backref=false); //DONE
int  ReplaceGlobal(String& t, Callback1<Vector<String>&> cbr); //DONE
```

Callback works in same fashion as in U++ Menu and Bar. User writes a callback function with void return and "Vector<String>& v" argument. And uses the THISBACK macro to pass it to Replace() or ReplaceGlobal(). Inside the match_callback_fun() user gets a vector already populated with all the matches. any changes made to the vector will be reflected in the source string 't'.

```
void match_callback_fun(Vector<String>& v){
...
}

regx.Replace(t, THISBACK(match_callback_fun));
regx.ReplaceGlobal(t, THISBACK(match_callback_fun));
```

Regards
Navi

File Attachments

1) RegExp Extension 28-01-2013 #1700.rar, downloaded 388 times

---

Subject: Re: why no virtual destructor for RegExp?
Posted by mirek on Tue, 19 Feb 2013 16:15:50 GMT
View Forum Message <> Reply to Message

After minor fixes in the interface (const added where possible, const String -> const String&), I have applied your changes to plugin/pcre.

Thanks for contribution.

Mirek

Subject: Re: why no virtual destructor for RegExp?
Posted by navi on Tue, 19 Feb 2013 23:23:22 GMT
View Forum Message <> Reply to Message

Thank you very much for the fixes Mirek. Also many thanks for making it part of upp plugin/pcre.

Really appreciate that you guys have made such an elegant framework as UPP and made it freely available to all.

Thanks & Regards
navi
