Subject: Improved heurestics in C++ parser...
Posted by mirek on Thu, 11 Sep 2008 20:46:27 GMT
View Forum Message <> Reply to Message

This is not the "final step" in C++ parser, but it seems to greatly improve its performance by implementing new error recovery heurestics.

In other words, C++ parser is now able to skip things that it does not undestand much better than before...

Mirek

Subject: Re: Improved heurestics in C++ parser...
Posted by Mindtraveller on Fri, 12 Sep 2008 07:36:49 GMT
View Forum Message <> Reply to Message

This must be really tricky algorithm.
As I understand finally TheIDE will have lightspeed & accurate parsing. And it`s great!

Subject: Re: Improved heurestics in C++ parser...
Posted by mirek on Fri, 12 Sep 2008 08:09:29 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Fri, 12 September 2008 03:36This must be really tricky algorithm.
As I understand finally TheIDE will have lightspeed & accurate parsing. And it`s great!

Actually, not yet. We still need macros - and that is EXTREMELY tricky (just think that you need to detect all macros changes, remember which files are using which macros and reparse them accordingly

Anyway, current heurestics is hopefully able to parse correctly most of U++ sources. Which in turn is quite needed for T++ efforts...

Mirek

Subject: Re: Improved heurestics in C++ parser...
Posted by cbpporter on Sat, 13 Sep 2008 15:38:03 GMT
View Forum Message <> Reply to Message

Just a little thought: when inserting separators into a package, or even opening and closing those separators, Assist++ is checking my packages. It goes real fast, but is quit redundant for separators.

Subject: Re: Improved heurestics in C++ parser...
Posted by cbpporter on Sat, 13 Sep 2008 16:08:36 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 12 September 2008 11:09
Anyway, current heurestics is hopefully able to parse correctly most of U++ sources.
After testing on a lot of code, I can confirm that too. The only thing I could find that kills auto complete up until now was something like this:
Vector<int*>* grade[11] = { NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL };
(And yes, that is not a made up example, I do have something similar in my code)

Subject: Re: Improved heurestics in C++ parser...
Posted by mirek on Sat, 13 Sep 2008 17:12:16 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sat, 13 September 2008 12:08luzr wrote on Fri, 12 September 2008 11:09
Anyway, current heurestics is hopefully able to parse correctly most of U++ sources.
After testing on a lot of code, I can confirm that too. The only thing I could find that kills auto complete up until now was something like this:
Vector<int*>* grade[11] = { NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL };
(And yes, that is not a made up example, I do have something similar in my code)

Does it kill it for 'grade' only, or anything that follows is affected as well?

Mirek

Subject: Re: Improved heurestics in C++ parser...
Posted by cbpporter on Sat, 13 Sep 2008 17:23:30 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 13 September 2008 20:12
Does it kill it for 'grade' only, or anything that follows is affected as well?

Anything that follows, but only in current function. The problem seems to be related to initializer list, not the type. More precisely, it doesn't handle the ',' in the initializer list. Initializing with one item works.

Subject: Re: Improved heurestics in C++ parser...
Posted by mirek on Sat, 13 Sep 2008 17:56:28 GMT

Ah, yeah, the new heurestics is basically about detecting the next file-level item in error recovery and putting parser into file-level too. It is not able to solve what happens inside body.

I have noticed the initializer list problem before too (just before implementing heurestics), I guess it is worth fixing (and easy to fix too in fact...).

Mirek

Subject: Re: Improved heurestics in C++ parser...
Posted by mirek on Sun, 14 Sep 2008 16:48:44 GMT

luzr wrote on Sat, 13 September 2008 13:56
I have noticed the initializer list problem before too

Hopefuly, initializer list should be now fixed...

Please check with your testcase.

Mirek
