Subject: Clang vs. GCC
Posted by dolik.rce on Sat, 06 Feb 2010 22:10:46 GMT
View Forum Message <> Reply to Message

Hello,

I'm not sure if this is the right place to post this, but I couldn't find any category where it would fit. I installed fresh svn version of Clang (C++ frontend for LLVM) inspired by a news that it finally can build itself. I was playing with it a bit and of course I tried to compile some U++ code.

I'm not going to talk about the compilation problems, but about interesting error that popped out. In topt.h there is this template:template <class T>

```
inline void DestroyArray(T *t, const T *lim) {
 while(t < lim) {
  t->T::~T();
  t++;
 }
}
```

Clang complained about using this template with T=unsigned int, since "type 'unsigned int' cannot be used prior to '::' because it has no members". After this I was curious about how GCC solves this problem. Well, the answer is simple: GCC ignores it.

This template is being called with T equal to types like int, const char*, void* without any problems. When I stepped through this part of code in debugger, the destructor line was simply skipped, the loop run through the given range of pointers doing nothing usefull.

My knowledge of C++ is quite limited, so I have few questions: How is that possible? Is it a GCC, Clang or U++ bug? Or is it just me, missing some deep knowledge about inlined functions, templates or some other dark corner of C++?

I hope someone can enlighten me a bit... I really like Clangs verbosity an it would be great if U++ supported it once (But rewriting half of the Core is just too high price).

Best regards,
Honza