decltype for the C++0x Standard Library

Authors: Douglas Gregor, Indiana University
Jaakko Järvi, Texas A&M University
Document number: N2194=07-0054
Date: 2007-03-08
Project: Programming Language C++, Library Working Group
Reply-to: Douglas Gregor <doug.gregor@gmail.com>

Introduction

This document describes specific changes to the C++0x working paper that make use of `decltype` [2] within the Standard Library. The changes are very minor, because the library facilities that benefit from `decltype` where incorporated after `decltype` had already been designed. In particular the `result_of` hook—which is the only aspect of the Standard Library that this document changes—was designed with forward-compatibility in mind [1]. `result_of` currently says that implementations are permitted to get the return type of a particular function call by any means possible, so long as they get the answer right; if they cannot do so, `result_of` specifies a protocol that the implementation should follow to extract the return type from library- and user-provided information. With `decltype`, every implementation can get the answer right, so we need only eliminate the weasel-wording `result_of` currently uses. We note that a C++0x `result_of` meets the requirements of a TR1 `result_of`.

Proposed Wording

Modify 20.5.4 "Function object return types" [func.ret] as follows:

```
namespace std {
  template <class FunctionCallTypes> // F(T1, T2, ..., TN)
  class result_of {
  public :
    // types
    typedef see below type;
  };
} // namespace std
```

1 Given an rvalue `fn` of type `Fn` and values `t1, t2, ..., tN` of types `T1, T2, ..., TN`, respectively, the `type` member is the result type of the expression `f(t1, t2, ...,tN)`. The values `ti` are lvalues when the corresponding type `Ti` is a reference type, and rvalues otherwise.

2 ~~The implementation may determine the type member via any means that produces the exact type of the expression f(t1, t2, ..., tN) for the given types. [Note: The intent is that implementations are permitted to use special compiler hooks —end note]~~

3 ~~If Fn is not a function object defined by the standard library, and if either the implementation cannot determine the type of the expression fn(t1, t2, ..., tN) or the expression is ill-formed, the implementation shall use the following process to determine the type member:~~

1. ~~If Fn is a function pointer or function reference type, type shall be the return type of the function type.~~
2. ~~If Fn is a member function pointer type, type shall be the return type of the member function type.~~
3. ~~If Fn is a possibly cv-qualified class type with a member type result_type, type shall be typename F::result_type.~~
4. ~~If Fn is a possibly cv-qualified class type with no member named result_type or if typename Fn::result_type is not a type:~~
   (a) ~~If N=0 (no arguments), type shall be void.~~
   (b) ~~If N>0, type shall be typename Fn::template result<Fn(T1, T2,..., TN)>::type.~~
5. ~~Otherwise, the program is ill-formed.~~

References

[1] Douglas Gregor. A uniform method for computing function object return types (revision 1). Technical Report N1454=03-0037, ISO/IEC JTC 1, Information Technology, Subcommittee SC 22, Programming Language C++, 2003. `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2003/n1454.html`.

[2] Jaakko Järvi, Bjarne Stroustrup, and Gabriel Dos Reis. Decltype (revision 6): Proposed wording. Technical Report N2115=06-0185, ISO/IEC JTC 1, Information Technology, Subcommittee SC 22, Programming Language C++, November 2006. `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2115.pdf`.