Try-catch blocks in constexpr functions

Document #:	P1002R0
Date:	2018-04-01
Project:	Programming Language C++
Audience:	Evolution Working Group

Reply-to: Louis Dionne <ldionne.2@gmail.com>

1 Proposal

Try-catch blocks can't currently appear in `constexpr` functions:

```
constexpr int f(int x) {
  try { return x + 1; } // ERROR: can't appear in constexpr function
  catch (...) { return 0; }
}
```

This paper proposes allowing this usage, but without changing the fact that a `throw` statement can't appear in a constant expression. This way, compilation errors are still triggered by throwing in a `constexpr` function, and hence a `catch` block is simply never entered. In other words, try blocks are allowed in `constexpr` functions, but they behave like no-ops when the function is evaluated as a constant expression.

This proposal does not close the door to implementing error-handling in `constexpr` functions in the future if we so desire.

This proposal does not break any code, since `constexpr` functions that contain try-catch blocks are currently ill-formed.

2 Motivation

The underlying motivation is reflection and metaprogramming, just like [P0784R1]. Concretely, this limitation was encountered whilst surveying `std::vector` in libc++ with the purpose of making it `constexpr`-enabled. Indeed, `vector::insert` uses a try-catch block to provide the strong exception guarantee.

3 Proposed wording

This wording is based on the working draft [N4727]. Change in **[dcl.constexpr] 10.1.5/3**:

> The definition of a constexpr function shall satisfy the following requirements:

- it shall not be virtual (13.3);
- its return type shall be a literal type;
- each of its parameter types shall be a literal type;
- its *function-body* shall be `= delete`, `= default`, or a *compound-statement* that does not contain
  - an *asm-definition*,
  - a `goto` statement,
  - an identifier label (9.1), or
  - ~~a *try-block*, or~~
  - a definition of a variable of non-literal type or of static or thread storage duration or for which no initialization is performed.

Change in **[dcl.constexpr] 10.1.5/4**:

The definition of a constexpr constructor shall satisfy the following requirements:

- the class shall not have any virtual base classes;
- each of the parameter types shall be a literal type~~;~~.
- ~~its *function-body* shall not be a *function-try-block*.~~

4 References

[N4727] Richard Smith, *Working Draft, Standard for Programming Language C++* http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/n4727.pdf

[P0784R1] Multiple authors, *Standard containers and constexpr* http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0784r1.html