C++26
エラー性動作（erroneous behavior）

高橋 晶 (Akira Takahashi)
faithandbrave@gmail.com
Preferred Networks, Inc.
2024/12/20 (金) C++ MIX #12

C++26 エラー性動作

- C++26から、環境によって発生する可能性のあるエラー動作として、「エラー性動作 (erroneous behavior)」が追加された
- これまであった未定義動作、未規定動作などと比較しながら見ていこう

未定義動作（undefined behavior; UB）

- 特定の操作に対して、予期せぬ動作をする可能性がある
- 範囲外アクセスやゼロ割など
- クラッシュする可能性もあるし、しない可能性もある
  - **クラッシュしないとしても何が起こるかはわからない**
- プログラマは未定義動作がないコードを書かないといけない
  - UBSan (Undefined Behavior Sanitizer) ツールで検出できる

未規定動作（unspecified behavior）

- C++規格では動作を規定せず、処理系で規定する
- 例外のエラーメッセージ、sizeof(long)、ラムダ式のオブジェクトサイズなど
- 処理系のよって異なる動作をするが、危険ではない（クラッシュはしない）

エラー性動作（erroneous behavior）

- 未定義動作を安全側に倒した動作
- クラッシュする可能性もあるし、しない可能性もある
  - **クラッシュしない場合の動作が規定される**
- C++26でエラー性動作に分類されるのは、
  「未初期化値の読み取り」のみ

未初期化値の読み取り

```
int f(int x) {
  // 処理が続行した場合…
  int y = x; // エラー性動作ではない
}

int x; // エラー性の値 (erroneous value) をもつ
f(x);  // エラー性動作 (エラー性の値を読み取った)
```

- エラー性動作が起こったあとは、エラー性の値とは見なされない
- ただしunsigned char (とstd::byte) ではエラー性動作にはならない

不定値の使用を明示

```
int f(int x) {}
int g(int x [[indeterminate]]) {
  int y = x; // 未定義動作
}

int x [[indeterminate]]; // 意図して不定値を使う
f(x); // 未定義動作
g(x); // OK
```

- 不定値として初期化すること、不定値を受け取ることを明示する属性もいっしょに入る
- 不定値の読み取りは、エラー性動作ではなく未定義動作になる

将来、エラー性動作に分類されるかもしれない操作

- 符号付き整数のオーバーフロー
- 型変換をした結果、表現可能な範囲を超えた
- ゼロ割

例として、ゼロ割はARM CPUではクラッシュせず値0が出力されて処理が続行する

まとめ

- エラー分類が今後変更されていくことで、
  クラッシュしない場合の動作が規定されていく

- それによってプログラムの安全性が高くなっていく

- 「なにが起こるかわからない (未定義動作)」から
  「クラッシュもしくは規定された処理続行 (エラー性動作)」へ