C++20からC++23までの変化

高橋 晶 (Akira Takahashi)
faithandbrave@gmail.com
Preferred Networks, Inc.
2023/12/06 (水) C++ MIX #8

はじめに

- 前回のC++ MIXが2019年末だったので、C++20の話があまりできていませんでした
- なので今回は、C++20のおさらいと、C++23のかんたんな紹介をします

これまでの主なアップデート内容をおさらい

バージョン	主なアップデート内容
C++98 (1998年)	ISOで標準化された最初のバージョン
C++03 (2003年)	C++98で未規定だった仕様の補完 (C++98と同じだと考えていい)
C++11 (2011年)	初期化子リスト、範囲for文、型推論、ラムダ式、スレッド、スマートポインタなど
C++14 (2014年)	2進数リテラル、数値リテラルの桁区切り文字など
C++17 (2017年)	構造化束縛、optionalクラス、インライン変数、ファイルシステムなど
C++20 (2020年)	比較演算子の自動定義、文字列フォーマット、Range、テンプレートパラメータの制約、数学定数など

C++20 1/4 比較演算子の自動定義

- operator==を定義すると、operator!=が自動定義される
- operator<=>を定義すると、operator<、operator<=、operator>、operator>=が自動定義される
- operator<=> の正式名称は「three way comparison operator (三方比較演算子)」。宇宙船演算子とも言う
- memcmp関数と同じように、等しいか、小さいか、大きいかを一度に判定できる

C++20 2/4 文字列フォーマット

- std::format関数が入った
- Python風の書式文字列を使い、書式指定の文字列を生成できる
  - printfのように型を書式では指定せず、引数番号 + 書式を指定する
- ostringstreamを使う必要が (だいたい) なくなった
- 書式文字列は、コンパイル時にチェックされる
  - constevalなコンストラクタによって文字列リテラルのコンパイル時チェックを実現している

```
cout << format("{} {} {}", 3, 1.23, "hello") << endl;
cout << format("{0} {0:#x} {1}", 15, "hello") << endl;
```

```
3 1.23 hello
15 0xf hello
```

C++20 3/4 Range

- コンテナやイテレータ範囲などのRangeに対する操作が定義された
- <ranges>での遅延評価のRangeアダプタに加えて、<algorithm>のRange版がstd::ranges名前空間に定義される
  - 動向的には、新規アルゴリズムはstd::ranges名前空間にのみ定義される模様
  - C++23でRangeアダプタがさらにたくさん入った。インデックス付きでループするenumerateとかとか

```
vector v = {1, 2, 3, 4, 5};
for (auto x : v | filter([](int x) { return x % 2 == 0 })
                | transform([](int x) { return x * 3; })) {
  cout << x << endl;
}
```

C++20 4/4 数学定数

- <numbers>ヘッダに、std::numbers::piなどの数学定数が定義される
  - C++11 コンパイル時の関数評価 (constexpr)
  - C++14 変数テンプレート
  - C++17 インライン変数 (ヘッダに変数定義しても実体をひとつにできる)
  - C++20 コンセプト
- piがdouble版、pi_vがテンプレート版

```
template <class T>
T degree_to_radian(T x)
{ return x * std::numbers::pi_v<T> / static_cast<T>(180.0); }
```

```
template <class T>
inline constexpr T pi_v = static_cast<T>(3.14159265358979323846L);
```

C++23

- std::print() / std::println()
- import std;
- std::expected
- コルーチンを便利に使うためのstd::generatorクラス

C++23 print / println

- std::format()ベースの出力関数として、std::print()とstd::println()が入った
  - std::println()は改行コード付き
- <print>ヘッダにデフォルトのオーバーロードと、FILE*のオーバーロードが定義される
- <ostream>ヘッダにはstd::ostream&のオーバーロードが定義される
  - ただし、std::cerrとかは<iostream>で定義されるので注意

```
println("{} {} {}", 3, 1.23, "hello");
println("{0} {0:#x} {1}", 15, "hello");
```

```
3 1.23 hello
15 0xf hello
```

C++23 import std;

- 標準モジュールとして、stdとstd.compatが入った
- stdは、C互換ライブラリを含む全部入りで、全部std名前空間に入る
- std.compatは、C互換ライブラリをグローバル名前空間にも入れる
  - どちらも全部入り
- ただし、マクロは含まれないのでassertとかが必要ならインクルードが追加で必要

```
import std;

int main() {
  std::println("Hello World");
}
```

C++23 expected

- 正常値かエラー値どちらかが入る型としてstd::expectedが入る
- expected<int, string>だったら正常値int、エラー値string
  - expected<string, string>もできる

```
expected<double, string> safe_divide(double i, double j) {
  if (j == 0) { return unexpected("divide by zero"); }
  else { return i / j; }
}

if (auto r = safe_divide(3.0, 2.0); r.has_value()) {
  r.value(); // 正常値を取り出してなにかする
} else {
  r.error(); // エラー値を取り出してなにかする
}
```

C++23 generator

- C++20のコルーチンをよりかんたんに使える機能としてstd::generator<T>クラスが入った
- 生成した値をRangeとして使える

```
// 偶数値の無限Rangeを作る関数
std::generator<int> evens() {
  int n = 0;
  while (true) {
    co_yield n;
    n += 2;
  }
}

// 先頭5個だけ使う
for (int i : evens() | std::views::take(5)) {}
```

まとめ

- C++20とC++23で、普段のプログラミングをより簡単にする機能がたくさん入りました
- C++のアップデートについていけなくなる方も増えてくると思うので、情報発信がんばりますっ

スポンサー募集中

- https://cpprefjp.github.io/

- C++日本語リファレンスサイトcpprefjpのスポンサー募集を開始しました
- C++の最新情報を持続的に発信していくためにサポートしていただける方を募集しております
- いただいたお金は、編集者に分配します