
C++Builder® 13
embarcadero®

Qui suis-je ?

Laurent Navarro

ln@altidev.com

https://www.altidev.com

MVP C++ Builder & Delphi - Toulouse

MAJ du compilateur C++ Moderne

- Il y a maintenant 4 compilateurs C++ Windows.
  - Compilateur classique « Borland » 32 bits
  - Compilateur Clang 5.0 32 bits
  - Compilateur Clang 5.0 64 bits
  - Compilateur Moderne Clang 20 64 bits
- Le compilateur moderne est passé de Clang 15 à Clang 20
- Le support de C++20 et C++23 est presque complet. Cf. doc statut compilateur & LLVM lib C++

Principales nouveautés C++ 20

- Opérateur <=> et évolutions comparaisons
- Librairie format
- Librairie range
- SPAN
- Source_location
- Librairie chrono : Gestion calendrier
- Jthread

Exemple C++20

```cpp
#include <format>
#include <ranges>
#include <chrono>
#include <source_location>
#include <thread>
using namespace std;
using namespace std::chrono;
struct Point2D {
  int x;
  int y;
  Point2D(int x_, int y_) :x(x_), y(y_) {};
  std::strong_ordering operator <=> (const Point2D& OpDroite) const
  {
    if (auto r = x <=> OpDroite.x; r != 0)
      return r;
    return y <=> OpDroite.y;
  }
  bool operator == (const Point2D& OpDroite) const
  {
    return (x == OpDroite.x) && (y == OpDroite.y);
  }
};

template <>
struct std::formatter<Point2D> {
  constexpr auto parse(std::format_parse_context& ctx) {
    return ctx.begin();
  }
  auto format(const Point2D& p, std::format_context& ctx) const {
    return std::format_to(ctx.out(), "({};{})", p.x, p.y);
  }
};
void FctBasic(int no)
{
  cout << "Debut thread " << no << endl;
  for (int i = 0; i < 3; i++) {
    this_thread::sleep_for(200ms);
    cout << "Thread " << no << " iteration " << i << endl;
  }
  cout << "Fin thread " << no << endl;
}
```

Exemple C++20

```
int _tmain(int argc, _TCHAR* argv[])
{
  // Format
  cout << "Hello " << format("the answer is {}.", 42) << endl;
  // <=> Operator
  Point2D A{ 1,1 };
  Point2D B{ 1,2 };

  cout << format("A < B = {}<{} = {}", A, B, (A < B)) << endl;
  cout << format("A > B = {}>{} = {}", A, B, (A > B)) << endl;

  cout << "Ranges " << endl;
  vector<int> myVec{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
  auto vue = myVec | views::reverse | views::take(5);
  for (auto v : vue) {
    std::cout << v << " ";
  }
  cout << endl;
  cout << "Chrono " << endl;
  auto date1{ 29d / month(03) / 2022y };
  cout << "date1 = " << date1;
  auto sysnow = system_clock::now();
  cout << format("\nd/m/Y H:M:S ={:%d/%m/%Y %H:%M:%S}\n", sysnow);

  auto location = source_location::current();
  cout << format("source_location\n{} ({}:{}) \"{}\" \n"
    , location.file_name(), location.line(), location.column()
    , location.function_name());
  cout << "JThread " << endl;
  {
    jthread t1(FctBasic, 1);
    this_thread::sleep_for(100ms);
    jthread t2(FctBasic, 2);
  }
  cout << "Fin JThread " << endl;

  getchar();
}
```

Exemple C++ 20

```
Hello the answer is 42.
A < B = (1;1)<(1;2) = true
A > B = (1;1)>(1;2) = false
Ranges
9 8 7 6 5
Chrono
date1 = 2022-03-29
d/m/Y H:M:S =15/09/2025 06:31:08.032153
source_location
DemoCpp20_1_Main.cpp (70:19) "int wmain(int, _TCHAR **)"
JThread
Debut thread 1
Debut thread 2
Thread 1 iteration 0
Thread 2 iteration 0
Thread 1 iteration 1
Thread 2 iteration 1
Thread 1 iteration 2
Fin thread 1
Thread 2 iteration 2
Fin thread 2
Fin JThread
```

Principales nouveautés C++ 23
implémentées dans clang 20

- Opérateur [] multidimensionnel
- Mdspan (Span multidimensionnel)
- Evolution Format : support des conteneurs
- println
- Gestion des erreurs par Expected
- Monadic optional & expected

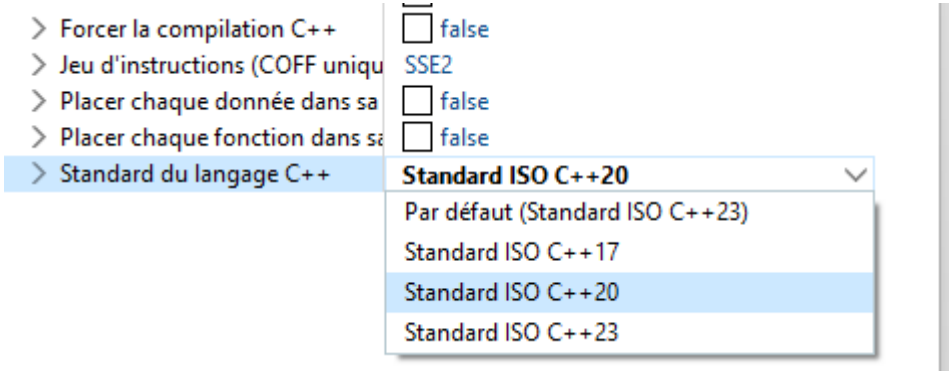
Exemple C++23

```cpp
#include <iostream>   #include <tchar.h>
#include <array>      #include <mdspan>
#include <format>     #include <expected>
#include <charconv>
using namespace std;
expected<int, string> parse_int(const string& s) {
  int value{};
  auto [ptr, ec] = from_chars(s.data(), s.data() + s.size(), value);
  if (ec == errc{}) {
    return value;
  }
  else {
    return unexpected("Error: unable to convert  \"" + s + "\" to int");
  }
}

int _tmain(int argc, _TCHAR* argv[])
{
  cout << "multidimensional operator[] and mdspan" << endl;

  array A{ 1, 2, 3,4, 5, 6 };
  mdspan Span{ A.data(), 2, 3 };
  for (size_t i = 0; i < Span.extent(0); ++i)
    for (size_t j = 0; j < Span.extent(1); ++j)
      cout << format("Span[{},{}]={}\n", i, j, Span[i, j]);
  println("println of array A={}", A);

  cout << "std::expected & monadic" << endl;
  for (auto test : { "42"s, "abc"s }) {
    auto result = parse_int(test);
    if (result)
      cout << "Pass: value = " << *result << "\n";
    else
      cout << "Fail: " << result.error() << "\n";
    cout << "value_or(0): " << result
      .and_then([](int x)->expected<int, string> {return x * 5; })
      .value_or(0) << "\n";
  }
  getchar();
}
```

```
multidimensional operator[] and mdspan
Span[0,0]=1
Span[0,1]=2
Span[0,2]=3
Span[1,0]=4
Span[1,1]=5
Span[1,2]=6
println of array A=[1, 2, 3, 4, 5, 6]
std::expected & monadic
Pass: value = 42
value_or(0): 210
Fail: Error: unable to convert  "abc" to int
value_or(0): 0
```

Autres nouveautés

- Dans les options du projet > Compilateur C++ à présent nous avons un sélecteur de la version du standard C++ à utiliser.



- Visual Assist est à présent disponible sur l'IDE 64 bits et son intégration à été améliorée.
- Meilleur support de ASAN (Address Sanitizer) sous Windows 11

Présentation précédente

- Ma présentation de C++ Builder 12.3
  https://www.youtube.com/watch?v=80vZrro_Btc&t=582s
  sur la chaine YouTube de Barnsten France
- Elle aborde :
  - Résumé des nouveautés de C++ Builder 12 à 12.3
  - Clang ASAN
  - Le time travel debugging
  - Dr Memory