The format library

- Topics:
  - printing and formatting facilities from c++23
  - How mainstream compilers support them
  - The fmt library to the rescue

The format library

```cpp
//C++ 23 adds std::print and std::println, but they only work well on Visual C++.
//We'll stick to fmt going forward.

auto value = std::format("Hello, {}!", "world");
std::cout << value << "\n";

std::print("Hello, {}!", "world\n");
std::print("Hello, {}!", "world\n");
std::println("Unformatted table : ");
std::println("{} {} {}", "Daniel", "Gray", "25");
std::println("{} {} {}", "Stanley", "Woods", "33");
std::println("{} {} {}", "Jordan", "Parker", "45");
std::println("{} {} {}", "Joe", "Ball", "21");
std::println("{} {} {}", "Josh", "Carr", "27");
std::println("{} {} {}", "Izaiah", "Robinson", "29");

std::println("-----");

std::println("Formatted table : ");
std::println("{:<10} {:<10} {:<5}", "Lastname", "Firstname", "Age");
std::println("{:<10} {:<10} {:<5}", "Daniel", "Gray", "25");
std::println("{:<10} {:<10} {:<5}", "Stanley", "Woods", "33");
std::println("{:<10} {:<10} {:<5}", "Jordan", "Parker", "45");
std::println("{:<10} {:<10} {:<5}", "Joe", "Ball", "21");
std::println("{:<10} {:<10} {:<5}", "Josh", "Carr", "27");
std::println("{:<10} {:<10} {:<5}", "Izaiah", "Robinson", "29");
```

```
Hello, world!
Hello, world
!Hello, world
!Unformatted table :
Daniel Gray 25
Stanley Woods 33
Jordan Parker 45
Joe Ball 21
Josh Carr 27
Izaiah Robinson 29
```

The format library

```cpp
auto value = fmt::format("Hello, {}!", "world");
std::cout << value << std::endl;
fmt::print("Hello, {}!", "world\n");
fmt::println("Unformatted table : ");
fmt::println("{} {} {}", "Daniel", "Gray", "25");
fmt::println("{} {} {}", "Stanley", "Woods", "33");
fmt::println("{} {} {}", "Jordan", "Parker", "45");
fmt::println("{} {} {}", "Joe", "Ball", "21");
fmt::println("{} {} {}", "Josh", "Carr", "27");
fmt::println("{} {} {}", "Izaiah", "Robinson", "29");

fmt::println("-----");

fmt::println("Formatted table : ");
fmt::println("{:<10} {:<10} {:<5}", "Lastname", "Firstname", "Age");
fmt::println("{:<10} {:<10} {:<5}", "Daniel", "Gray", "25");
fmt::println("{:<10} {:<10} {:<5}", "Stanley", "Woods", "33");
fmt::println("{:<10} {:<10} {:<5}", "Jordan", "Parker", "45");
fmt::println("{:<10} {:<10} {:<5}", "Joe", "Ball", "21");
fmt::println("{:<10} {:<10} {:<5}", "Josh", "Carr", "27");
fmt::println("{:<10} {:<10} {:<5}", "Izaiah", "Robinson", "29");
```

```
-----
Formatted table :
Lastname   Firstname  Age
Daniel     Gray       25
Stanley    Woods      33
Jordan     Parker     45
Joe        Ball       21
Josh       Carr       27
Izaiah     Robinson   29
```

The format library

```cpp
// dynamic width
int col_width{ 10 };
fmt::println("Formatted table with dynamic width: ");
fmt::println("{:<{}} {:<{}} {:<{}}", "Lastname", col_width, "Firstname", col_width, "Age", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Daniel", col_width, "Gray", col_width, "25", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Stanley", col_width, "Woods", col_width, "33", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Jordan", col_width, "Parker", col_width, "45", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Joe", col_width, "Ball", col_width, "21", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Josh", col_width, "Carr", col_width, "27", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Izaiah", col_width, "Robinson", col_width, "29", col_width / 2);

fmt::println("-----");

// right justified
fmt::println("Right justified table:  ");
col_width = 20;
fmt::println("{:>{}} {:>{}} {:>{}}", "Lastname", col_width, "Firstname", col_width, "Age", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Daniel", col_width, "Gray", col_width, "25", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Stanley", col_width, "Woods", col_width, "33", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Jordan", col_width, "Parker", col_width, "45", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Joe", col_width, "Ball", col_width, "21", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Josh", col_width, "Carr", col_width, "27", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Izaiah", col_width, "Robinson", col_width, "29", col_width / 2);
```

```
-----
Formatted table :
Lastname    Firstname   Age
Daniel      Gray        25
Stanley     Woods       33
Jordan      Parker      45
Joe         Ball        21
Josh        Carr        27
Izaiah      Robinson    29
```

The format library

```
// right justified
fmt::println("Right justified table:  ");
col_width = 20;
fmt::println("{:>{}} {:>{}} {:>{}}", "Lastname", col_width, "Firstname", col_width, "Age", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Daniel", col_width, "Gray", col_width, "25", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Stanley", col_width, "Woods", col_width, "33", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Jordan", col_width, "Parker", col_width, "45", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Joe", col_width, "Ball", col_width, "21", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Josh", col_width, "Carr", col_width, "27", col_width / 2);
fmt::println("{:>{}} {:>{}} {:>{}}", "Izaiah", col_width, "Robinson", col_width, "29", col_width / 2);

// left justified
fmt::println("Left justified table :  ");
col_width = 20;
fmt::println("{:<{}} {:<{}} {:<{}}", "Lastname", col_width, "Firstname", col_width, "Age", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Daniel", col_width, "Gray", col_width, "25", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Stanley", col_width, "Woods", col_width, "33", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Jordan", col_width, "Parker", col_width, "45", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Joe", col_width, "Ball", col_width, "21", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Josh", col_width, "Carr", col_width, "27", col_width / 2);
fmt::println("{:<{}} {:<{}} {:<{}}", "Izaiah", col_width, "Robinson", col_width, "29", col_width / 2);
```

```
Right justified table:
            Lastname            Firstname          Age
              Daniel                 Gray           25
             Stanley                Woods           33
              Jordan               Parker           45
                 Joe                 Ball           21
                Josh                 Carr           27
              Izaiah             Robinson           29
-----
Left justified table :
Lastname            Firstname           Age
Daniel              Gray                25
Stanley             Woods               33
Jordan              Parker              45
Joe                 Ball                21
Josh                Carr                27
Izaiah              Robinson            29
```

The format library

```
// setfill
fmt::println("Table with fill characters :  ");
col_width = 20;
fmt::println("{:*<{}} {:*<{}} {:*<{}}", "Lastname", col_width, "Firstname", col_width, "Age", col_width / 2);
fmt::println("{:*<{}} {:*<{}} {:*<{}}", "Daniel", col_width, "Gray", col_width, "25", col_width / 2);
fmt::println("{:*<{}} {:*<{}} {:*<{}}", "Stanley", col_width, "Woods", col_width, "33", col_width / 2);
fmt::println("{:*<{}} {:*<{}} {:*<{}}", "Jordan", col_width, "Parker", col_width, "45", col_width / 2);
fmt::println("{:*<{}} {:*<{}} {:*<{}}", "Joe", col_width, "Ball", col_width, "21", col_width / 2);
fmt::println("{:*<{}} {:*<{}} {:*<{}}", "Josh", col_width, "Carr", col_width, "27", col_width / 2);
fmt::println("{:*<{}} {:*<{}} {:*<{}}", "Izaiah", col_width, "Robinson", col_width, "29", col_width / 2);
```

```
Table with fill characters :
Lastname********** Firstname********** Age*******
Daniel************ Gray************** 25********
Stanley*********** Woods************* 33********
Jordan************ Parker************ 45********
Joe*************** Ball*************** 21********
Josh************** Carr************** 27********
Izaiah************ Robinson*********** 29********
```

The format library

```cpp
// control bool output format : 1/0 or true/false
bool condition{ true };
bool other_condition{ false };

fmt::println("condition: {}", condition);// true
fmt::println("other_condition: {}", other_condition);// false

// This is a hack - ref: https://github.com/fmtlib/fmt/issues/170
fmt::println("condition: {:d}", condition);
fmt::println("other_condition: {:d}", other_condition);
```

```
condition: true
other_condition: false
condition: 1
other_condition: 0
```

The format library

```cpp
//show or hide the +  sign for positive numbers
fmt::println("show or hide the +  sign for positive numbers: ");
int pos_num{ 34 };
int neg_num{ -45 };

fmt::println("pos_num: {}", pos_num);
fmt::println("neg_num: {}", neg_num);

fmt::println("pos_num: {:+}", pos_num);
fmt::println("neg_num: {:-}", neg_num);
```

```
show or hide the +  sign for positive numbers:
pos_num: 34
neg_num: -45
pos_num: +34
neg_num: -45
```

The format library

```cpp
// different number systems : std::dec, std::hex, std::oct
  fmt::println("different number systems: dec, hex, oct");
  int pos_int{ 717171 };
  int neg_int{ -5 };
  double double_var{ 498.32 };

  fmt::println("default base format : ");
  fmt::println("pos_int : {}", pos_int);
  fmt::println("neg_int : {}", neg_int);
  fmt::println("double_var : {}", double_var);

  fmt::println("pos_int in different bases : ");
  fmt::println("pos_int (dec) : {}", pos_int);
  fmt::println("pos_int (hex) : {:x}", pos_int);
  fmt::println("pos_int (oct) : {:o}", pos_int);
  fmt::println("pos_int (binary) : {:b}", pos_int);
  fmt::println("pos_int (decimal) : {:d}", pos_int);
```

```
different number systems: dec, hex, oct
default base format :
pos_int : 717171
neg_int : -5
double_var : 498.32
pos_int in different bases :
pos_int (dec) : 717171
pos_int (hex) : af173
pos_int (oct) : 2570563
pos_int (binary) : 10101111000101110011
pos_int (decimal) : 717171
```

The format library

```rust
// uppercase and nouppercase
fmt::println("uppercase and nouppercase: ");
pos_int = 717171;
fmt::println("pos_int (nouppercase - default): ");
fmt::println("pos_int (dec): {}", pos_int);
fmt::println("pos_int (hex): {:x}", pos_int);
fmt::println("pos_int (oct): {:o}", pos_int);

// For integers, the # toogles the alternative format flag. This shows the base prefix like 0b, and 0x.
fmt::println("pos_int (uppercase): ");
fmt::println("{:#X}", pos_int);
fmt::println("{:#B}", pos_int);
```

```
uppercase and nouppercase:
pos_int (nouppercase - default):
pos_int (dec): 717171
pos_int (hex): af173
pos_int (oct): 2570563
pos_int (uppercase):
0XAF173
0B10101111000101110011
```

The format library

```cpp
// fixed and scientific : for floating point values
// control the precision.
fmt::println("fixed and scientific: for floating point values: ");
double a{ 3.14159265358979323846264338327950 };
double b{ 2006.0 };
double c{ 1.34e-10 };

fmt::println("double values (default : use scientific where necessary) : ");
fmt::println("a : {}", a);
fmt::println("b : {}", b);
fmt::println("c : {}", c);

fmt::println("double values (precision) : ");
fmt::println("a: {:.6}", a);
fmt::println("b: {:.6}", b);
fmt::println("c: {:.6}", c);

fmt::println("double values (fixed) : ");
fmt::println("a: {:.6f}", a);
fmt::println("b: {:.6f}", b);
fmt::println("c: {:.6f}", c);

fmt::println("double values (scientific) : ");
fmt::println("a: {:.6e}", a);
fmt::println("b: {:.6e}", b);
fmt::println("c: {:.6e}", c);
```

```
fixed and scientific: for floating point values:
double values (default : use scientific where necessary) :
a : 3.141592653589793
b : 2006
c : 1.34e-10
double values (precision) :
a: 3.14159
b: 2006
c: 1.34e-10
double values (fixed) :
a: 3.141593
b: 2006.000000
c: 0.000000
double values (scientific) :
a: 3.141593e+00
b: 2.006000e+03
c: 1.340000e-10
```

The format library

```
// argument indexes: Allow us to order arguments. This is good for example for applications that use localization.
fmt::println("argument indexes: ");
fmt::println("It's {:.2f} degrees outside and it's {}", 34.5, "sunny");
fmt::println("It's {1} today. And the temperature is {0:.2f} degrees outside", 34.5, "sunny");
```

```
argument indexes:
It's 34.50 degrees outside and it's sunny
It's sunny today. And the temperature is 34.50 degrees outside
```