Valentine Timoshik

constexpr in action

constexpr - specifies that the value of a variable or function can appear in constant expressions (cppreference)

constant expression - defines an expression that can be evaluated at compile time (cppreference)

Limitations:

- Literal type
- Immediately initialized
- Initialization is constexpr

- Return type is literal
- Parameters are literal
- Must not contain goto, try-block
- Exactly one return statement (except for constructors)

Example:

```
constexpr int factorial(int n)
{
    return n <= 1 ? 1 : (n * factorial(n - 1));
}
```

So what?

Higher Order Functions:

lift - by Björn Fahller (github)

```
struct Employee { std::string name; unsigned number; };

const std::string& select_name(const Employee& e) { return e.name; }
unsigned select_number(const Employee& e) { return e.number; }

std::vector<Employee> staff; // sort employees by name
std::sort(staff.begin(), staff.end(), lift::compose(std::less<>{}, select_name);

// retire employee number 5
auto i = std::find_if(staff.begin(), staff.end(), lift::compose(lift::equal(5), select_number));
if (i != staff.end())
  staff.erase(i);
```

Higher Order Functions:

boostorg/hof - by Boost org(github)

```
BOOST_HOF_STATIC_LAMBDA_FUNCTION(print) = first_of(
[](const auto& x) -> decltype(std::cout << x, void())
 {
   std::cout << x << std::endl;
 },
 [](const auto& range)
 {
   for(const auto& x:range)
     std::cout << x << std::endl;
 });

std::vector<int> v = { 1, 2, 3, 4 };
print(v);
```

Parsers:

ctre - by Hana Dusíková (github)

```
struct date { std::string_view year; std::string_view month; std::string_view day; };

std::optional<date> extract_date(std::string_view s) noexcept
{
using namespace ctre::literals;
if (auto [whole, year, month, day] = ctre::match<"(\\d{4})/(\\d{1,2})/(\\d{1,2})">(s); whole)
{
   return date{year, month, day};
 }
 else
 {
   return std::nullopt;
 }
}
```

Parsers:

see-phit - by rep-movsd (github)

```
constexpr auto parser = R"*( <span > <p color="red" height='10' >
                             {{name}} is a {{profession}} in {{city}}</p > </span> )*"_html;
spt::tree spt_tree(parser);

spt::template_dict dct;
dct["name"] = "Mary";
dct["profession"] = "doctor";
dct["city"] = "London";
```

```
<HTML> <span> <p COLOR='red' HEIGHT='10'> Mary is a doctor in London </p> </span> </HTML>
<HTML> <span> <p COLOR='red' HEIGHT='10'> John is a janitor in New York </p> </span> </HTML>
```

Misc:

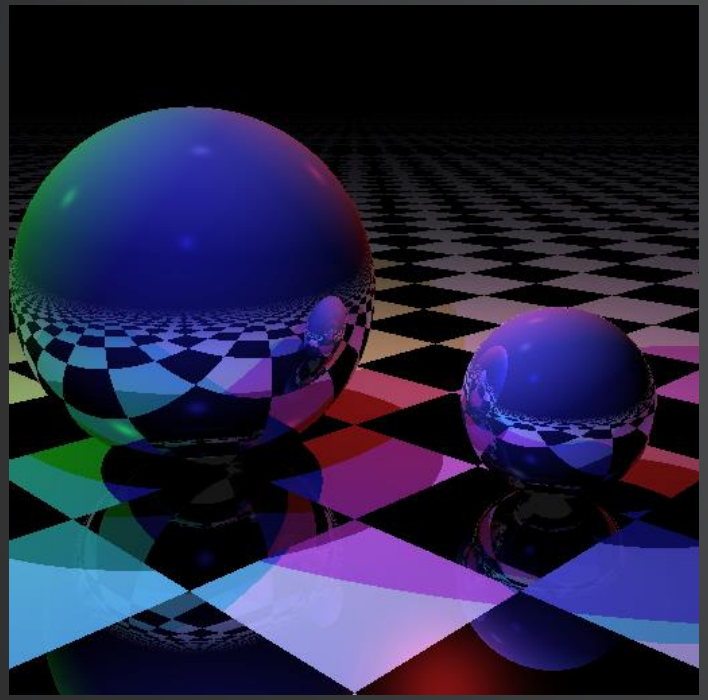
constexpr SQL - by Michael Kitzan (github)

```
using query = sql::query< "SELECT title AS book, name AS author, year, pages “
                          "FROM books NATURAL JOIN (SELECT * FROM authored
                          WHERE name = \"Harlan Ellison\") " "WHERE year = 1967 OR year >= 1972
                          AND genre = \"science fiction\"", books, authored >;
int main()
 {
   authored a{ sql::load<authored>("tests/data/authored.tsv", '\t') };
   books b{ sql::load<books>("tests/data/books.tsv", '\t') };

   for (query q{ b, a }; auto const& [book, author, year, pages] : q)
    {
      std::cout << book << '\t' << author << '\t' << year << '\t' << pages << '\n’;
    }
   return 0;
 }
```

Misc:

Compile-time Ray Tracer - by Tristan Brindle (github)



```
constexpr auto image = [] {
                ray_tracer r{};
                static_canvas<IMAGE_WIDTH, IMAGE_HEIGHT> c{};
                r.render(static_scene{}, c, c.width, c.height);
                return c; }();

stbi_write_png("render-ct.png", image.width,
                        image.height, 4, image.get_pixels().data(),
                        image.width * image.bpp);
```

Misc:

constexpr-8cc - by Keiichi Watanabe (github)

Useful links:

Link – a YouTube video about porting Doom to C++

Link – Talk by Antony Polukhin about constexpr

Link – “Constexpr all the things!” CppCon talk

Link - “Don’t constexpr all the things” by David Sankel

Thank you!

Slides - https://github.com/f1nal3/constexpr_in_action