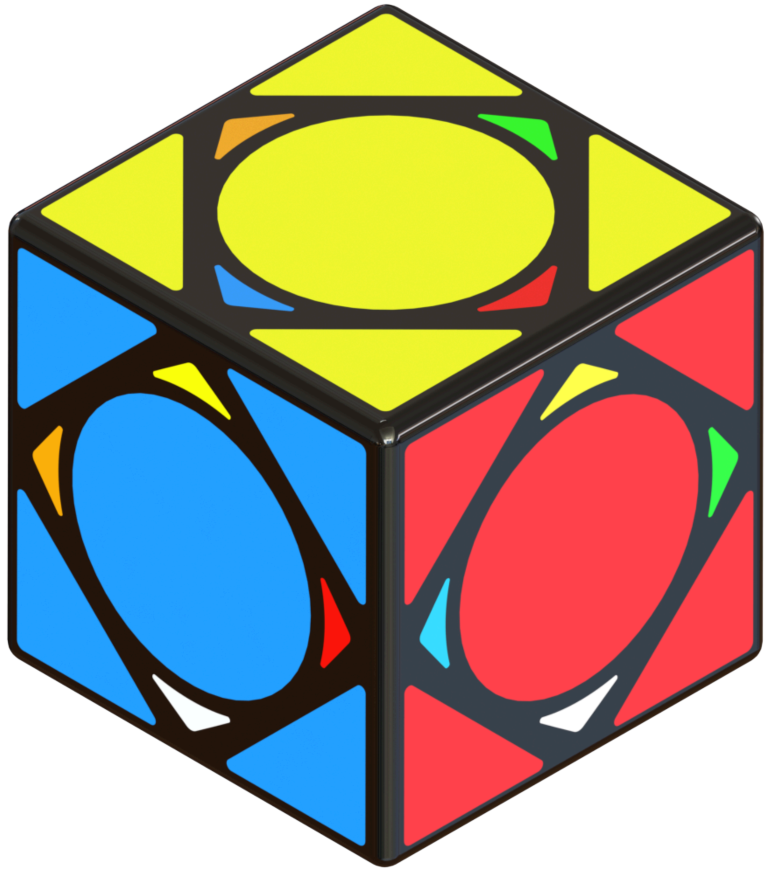
Brainplus: A user experience



Braden Ganetsky

C++Now 2024 Lightning Talk

Goals of this talk

- User experience
- Innovative API

Brainplus

- Parser combinators
- Expression templates in C++20 (empty types)
- Fully capable at compile-time
- Sounds like a famous language?

Epsilon

tok3n

```cpp
auto p = eps.of<char>;
```

```cpp
constexpr auto result = p.parse("ABC");
```

```cpp
static_assert(result.has_value());
static_assert(*result == "");
static_assert(result.remaining() == "ABC");
```

Brainplus

```cpp
auto p = plus;
```

```cpp
constexpr auto result = p.plarse("ABC");
```

```cpp
static_assert(result.has_value());
static_assert(*result == "");
static_assert(result.remaining() == "ABC");
```

Single "A"

tok3n

```
auto p = "A"_any_of;
```

```
constexpr auto result = p.parse("ABC");
```

```
static_assert(result.has_value());
static_assert(*result == "A");
static_assert(result.remaining() == "BC");
```

Brainplus

```
auto p =
  plus++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++;
```

```
constexpr auto result = p.plarse("ABC");
```

```
static_assert(result.has_value());
static_assert(*result == "A");
static_assert(result.remaining() == "BC");
```

Literal "ABC"

tok3n

```
auto p = "ABC"_all_of;
```

```
constexpr auto result = p.parse("ABC");
```

```
static_assert(result.has_value());
static_assert(*result == "ABC");
static_assert(result.remaining() == "");
```

Brainplus

```
auto p = ++(
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++plus+ ++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++plus+ ++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++plus
);
```

```
constexpr auto result = p.plarse("ABC");
```

```
static_assert(result.has_value());
static_assert(*result == "ABC");
static_assert(result.remaining() == "");
```

Version string

`tok3n`

```cpp
auto d = "0123456789"_any_of;
auto s = "."_all_of % ignore;
auto p = d >> s >> d >> s >> d;
```

```cpp
constexpr auto result = p.parse("1.2.3");
```

```cpp
static_assert(result.has_value());
static_assert(*result == std::tuple("1", "2", "3"));
static_assert(result.remaining() == "");
```

Version string

Brainplus

```
auto d = plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;
auto s = ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++plus+ +plus;
auto p = ++(d+s+d+s+d);
```

Version string

Brainplus

```
auto d = ...;
auto s = ...;
auto p = ++(d+s+d+s+d);
```

```
constexpr auto result = p.parse("1.2.3");
```

```
static_assert(result.has_value());
static_assert(*result == std::tuple("1", "2", "3"));
static_assert(result.remaining() == "");
```

Modifiers in `tok3n`

```
p % ignore   // or `ignore(p)`
p % join     // or `join(p)`
p % complete // or `complete(p)`, etc
```

```
p % constant<value>
```

```
p % fn<foo>
p % apply<bar>
```

```
p % into<S>
p % apply_into<T>
p % defaulted<U>
```

How many ways can we use `+`?

- Unary `+a`
- Binary `a + b`
- Prefix increment `++a`
- Postfix increment `a++`
- Anything else? (without `+=`)
- `a.operator++()` (stay with me)

How can we inject template parameters?

- `a.operator++<Value>()`
- `a.operator++<Type>()`
- **Overload set, therefore both can exist**
- **Necessary because expression templates**

Modifiers in `Brainplus`

```
p % constant<value>

p % fn<foo>
p % apply<bar>

p % into<S>
p % apply_into<T>
p % defaulted<U>
```

```
p + plus.operator++<value>()

p + ++plus.operator++<foo>()
p + plus.operator++<bar>()++

p + plus.operator++<S>()
p + ++plus.operator++<T>()
p + plus.operator++<U>()++
```

Version string v2

```
struct Version { int major; int minor; int patch; };
```

```
constexpr auto f = [](auto span) { return std::atoi(span); };
```

```
auto d = (...)+ ++plus.operator++<f>();
auto s = ...;
auto p = ++(d+s+d+s+d)+ ++plus.operator++<Version>();
```

```
constexpr auto result = p.parse("1.2.3");
```

```
static_assert(result.has_value());
static_assert(*result == Version{1,2,3});
static_assert(result.remaining() == "");
```

Recursion in `tok3n`

```
struct P : Custom<P>
{
  using result_type = int;
  static consteval auto get_parser();
};
```

Recursion in Brainplus

```
struct P : Plustom<P>
{


};
```

Recursion in Brainplus

```
struct P : Plustom<P>
{
  using plusult_type = int;

};
```

Recursion in Brainplus

```cpp
struct P : Plustom<P>
{
  using plusult_type = int;
  static consteval auto get_plarser();
};
```

Version string v3

```cpp
struct Version { int major; int minor; int patch; };
constexpr auto f = [](auto span) { return std::atoi(span); };
```

```cpp
auto p =
  ++((plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ ++plus.operator++<f>())+(++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++++++plus+ +plus)+(plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ++plus.operator++<f>())+(++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus+ +plus)+(plus++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++plus+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++plus++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +++++++++++++++++++++++++++++++++++++++ ++plus.operator++<f>()))+ ++plus.operator++<Version>();
```

```cpp
constexpr auto result = p.parse("1.2.3");
```

```cpp
static_assert(*result == Version{1,2,3});
```

Next up

- `tok3n` to Brainplus converter
- Propose to Boost

Thank you!

Braden Ganetsky

braden@ganets.ky
GitHub @k3DW