Add tuple protocol to complex

Abstract
This paper proposes amending complex with the tuple protocol, enabling structured binding and easy referential access.

Tony Table

<table>
<thead>
<tr>
<th>Before</th>
<th>Proposed</th>
</tr>
</thead>
</table>
| complex<

```cpp
complex<double> c{}; auto & [r, i] = reinterpret_cast<
```

```cpp
complex<double>({2})>(c));
```

```cpp
-template<

```cpp
constexpr auto swap_parts(complex<T> c) -> complex<T> {
    if constexpr {
        auto & [r, i] = reinterpret_cast<double>({2})(c); swap(r, i);
    }
    else {
        // required fallback as reinterpret_cast is ill-formed in constexpr contexts!
        const auto r{c.real()};
        const auto i{c.imag()};
        c.imag(r);
        c.real(i);
        return c;
    }
}

```

```cpp
complex<double> c{};
```

```cpp
// interaction with pattern matching proposal P1371R3
inspector<

```cpp
inspect<

```cpp
inspector<
```

```cpp
is [0, 0] => cout << "on origin";
```

```cpp
is [0, _] => cout << "on imaginary axis";
```

```cpp
is [_, 0] => cout << "on real axis";
```

```cpp
[r, i] is_ => cout << r << ", " << i;
```

```cpp
};

```

```cpp
complex<double> c{};
```

```cpp
// interaction with pattern matching proposal P2392R2
inspect<

```cpp
inspect<

```cpp
inspector<
```

```cpp
is [0, 0] => cout << "on origin";
```

```cpp
is [0, _] => cout << "on imaginary axis";
```

```cpp
is [_, 0] => cout << "on real axis";
```

```cpp
[r, i] is_ => cout << r << ", " << i;
```

```cpp
};

```

```cpp
complex<double> c{};
```

```cpp
// interaction with pattern matching proposal P2392R2
inspector<

```cpp
inspect<

```cpp
inspector<
```

```cpp
is [0, 0] => cout << "on origin";
```

```cpp
is [0, _] => cout << "on imaginary axis";
```

```cpp
is [_, 0] => cout << "on real axis";
```

```cpp
[r, i] is_ => cout << r << ", " << i;
```

```cpp
};

```

Revisions
R0: Initial version

---

1 RISC Software GmbH, Softwarepark 32a, 4232 Hagenberg, Austria, michael.hava@risc-software.at
2 RISC Software GmbH, Softwarepark 32a, 4232 Hagenberg, Austria, christoph.hofer@risc-software.at
Motivation
Mathematically the set of complex numbers $\mathbb{C}$ is isomorphic to $\mathbb{R}^2$ as a vector space with the isomorphism $\Phi: \mathbb{C} \to \mathbb{R}^2$ such that $\Phi(a+bi) = (a,b)$. Therefore, complex numbers can be identified with tuples and should possess the same characteristics, which is covered by the tuple protocol.

Complex numbers can equivalently be represented in cartesian coordinates $(a,b)$ as well as in polar coordinates $(r,\theta)$ using radius $r$ and angle $\theta$. However, alternative representations of complex numbers such as polar coordinates $(r,\theta)$ are prohibited by the requirement of matching C's _Complex floating-point_ feature.

As the respective getters do not expose referential access (changing them to do so would result in an ABI-break), the only way to get a reference to the real and imaginary parts of a complex is by performing a reinterpret_cast (mandated to be valid, see [complex numbers general]), which is not valid in a constexpr context. Supporting the tuple protocol enables structured binding and referential access to the components of a complex number in a constexpr compatible way.

Lastly, the current pattern matching proposals ([P1371R3] and [P2392R2]) allow inspection of _tuple-like_ objects, the proposed changes make complex _tuple-like_.

Design Space
The tuple protocol traits (tuple_size<T> and tuple_element<I,T>) are partially specialized for complex<U> and four free function overloads of get are provided.

Impact on the Standard
This proposal is a pure library extension.

Implementation Experience
The proposed design has been implemented at https://github.com/MFHava/STL/tree/P2819.

Proposed Wording
Wording is relative to [N4928]. Additions are presented like this, removals like this and drafting notes like this.

[version.syn]
In [version.syn], add:

```c
#define __cpp_lib_complex_tuple YYYYMML //also in <complex>
```

[DRAFTING NOTE: Adjust the placeholder value as needed to denote the proposal's date of adoption.]
Acknowledgements

Thanks to RISC Software GmbH for supporting this work.