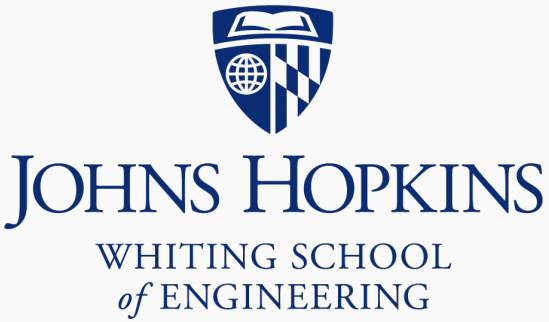
C++: pair and tuple

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

C++: pair and tuple

We want a function that takes integers a & b and returns both a/b (quotient) and a%b (remainder)

- `divmod(10, 5)` returns `2, 0`
- `divmod(10, 3)` returns `3, 1`

Functions only return *one* thing

C++: pair

One solution: pass-by-pointer arguments:

```
void divmod(int a, int b, int *quo, int *rem) {
    *quo = a / b;
    *rem = a % b;
}
```

Another: define `struct` for `divmod`'s return type:

```
struct quo_rem {
    int quotient;
    int remainder;
};

quo_rem divmod(int a, int b) { ... }
```

C++: pair

STL solution: return pair<int, int>, a pair with items of types int & int:

```
pair<int, int> divmod(int a, int b) {
    return make_pair(a/b, a%b);
}

int main() {
    pair<int, int> qr_10_5 = divmod(10, 5);
    pair<int, int> qr_10_3 = divmod(10, 3);
    cout << "10/5 quotient=" << qr_10_5.first
         << ", remainder=" << qr_10_5.second << endl;
    cout << "10/3 quotient=" << qr_10_3.first
         << ", remainder=" << qr_10_3.second << endl;
    return 0;
}
```

C++: pair

We've used `pair` already; dereferenced `map` iterator is a pair:

```
for(map<int, string>::iterator it = jhed_to_name.begin();
    it != jhed_to_name.end();
    ++it)
{
    // it->first has type int
    // it->second has type string
    cout << "  " << it->first << ": " << it->second << endl;
}
```

C++: `pair`

Relational operators for `pair` work as expected:

- Compares first field first...
- ...if there's a tie, compares second field

`make_pair(2, 3) < make_pair(3, 2)` is true

C++: tuple

`tuple` is like `pair` but with as many fields as you like

```
#include <tuple>
using std::tuple; using std::make_tuple;

tuple<int, int, float> divmod(int a, int b) {
    return make_tuple(a/b, a%b, (float)a/b);
}
```

`std::get<N>(tup)` gets the Nth field of `tuple` called `tup`:

```
using std::get;
tuple<int, int, float> tup = divmod(10, 3);
cout << "10/3 quotient="      << get<0>(tup)
     << ", remainder="        << get<1>(tup)
     << ", decimal quotient=" << get<2>(tup) << endl;
```

C++: tuple

```
$ g++ -c quo_rem_tuple.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o quo_rem_tuple quo_rem_tuple.o
$ ./quo_rem_tuple
10/3 quotient=3, remainder=1, decimal quotient=3.33333
```