C++ Macros

Marco Gallotta

28 February, 2009

1 Macros

- Define macro `MACRO` using `#define MACRO` on the top of your code.
- Can also `#define VERBOSE 2` and then use `VERBOSE` like a constant.
- Code between `#IFDEF MACRO` and `#ENDIF` is only included *if* `MACRO` is defined.
- `#IFNDEF MACRO` is similar, but checks if `MACRO` is **not** defined.
- `#define MIN(a, b) (a < b ? a : b)` defines a macro "function". All occurrences of `MIN(1, 2)` or the likes are replaced with `(1 < 2 ? 1 : 2)`.
- All above operations are performed at compile time, before the code is actually compiled.

Define the macro `NDEBUG` **before** `#include <cassert>` to disable assertions when submitting.

```
#define NDEBUG
#include <cassert>
int main() {
  // foo
  assert(N > 0); // we expect N > 0, so we assert that this is indeed true
#IFNDEF NDEBUG
  cerr << "debugging foo" << endl;
#ENDIF
}
```

Useful macro to print the file, line number, variable name and its value:

```
#include <cstdio>
#define PRINT(a, fmt) printf("%s:%u: %s=" fmt "\n", __FILE__, __LINE__, #a, a)
int main() {
  int num = 1;
  PRINT(num, "%d");
}
```