X-Macro

How to avoid repetition

Example: printing enums for debugging

```
enum States {
  Connected,
  Disconnected,
  Error
};
States s = Connected;
std::cout << s << “\n”; //prints ‘0’
```

Solution: write a conversion function

```
const char* str(enum States s) {
switch(s) {
  case Connected:
    return “Connected”;
    …
}

std::cout << str(s) << “\n”; //prints Connected
```

Problem:

Don’t forget to update the conversion function when you add a new state.

Problem:

Don’t forget to update the conversion function when you add a new state.

Don’t forget to release your locks.
Don’t forget to free your memory.
Don’t forget where you put your keys.

Problem:

Don’t forget to update the conversion function when you add a new state.

Don’t forget to release your locks. Thanks std::lock_guard
Don’t forget to free your memory. Thanks RAII
Don’t forget where you put your keys. Where *did* I put my keys?

Solution: X-macros – nothing to forget

1) Define your list of states,and wrap it with X(…):

Solution: X-macros – nothing to forget

1) Define your list of states,and wrap it with X(…):

```
#define X_STATES_LIST \
 X(Connected) \
 X(Disconnected) \
 X(Error)
```

Solution: X-macros – nothing to forget

2) Declare and define your enum by implementing the X macro:

Solution: X-macros – nothing to forget

2) Declare and define your enum by implementing the X macro:

```
#define X(name) name,
```

Solution: X-macros – nothing to forget

2) Declare and define your enum by implementing the X macro:

```
#define X(name) name,

enum States{
    X_STATES_LIST
};
#undef X
```

Solution: X-macros – nothing to forget

2) Declare and define your enum by implementing X:

```
#define X(name) name,

enum States{                      enum States{
    X_STATES_LIST       ──────►      X(Connected)
};                                   X(Disconnected)
#undef X                             X(Error)
                                  };
```

Solution: X-macros – nothing to forget

2) Declare and define your enum by implementing X:

```
#define X(name) name,

enum States{                    enum States{                 enum States{
    X_STATES_LIST      →           X(Connected)         →       Connected,
};                                 X(Disconnected)              Disconnected,
#undef X                           X(Error)                     Error,
                                };                           };
```

Solution: X-macros – nothing to forget

3) Declare and define your conversion function by implementing X:

```
const char* str(enum States s) {
   #define X(name) case (name): return #name;
   switch(s) {
     X_STATE_LIST
     default: return “Unknown";
   }
   #undef X
}
```

That’s it. Adding new states to X_STATE_LIST will update your code!

X-MACROS: multiple arguments

```
#define X_ERROR_LIST \
  X(nomem, “out of memory”, fatal)\
  X(invalid_user, “user login error”, warn)

#define X(name, description, category) case name: return #category “: ” description “(” #name “)”;
const char* long_error(error_t e) {
    switch (e) {
      X_ERROR_LIST
    }
}
//returns “fatal: out of memory (nomem)”, or “warn: user login error (invalid_user)”
```

Disclaimer: Macros, use them judiciously

https://github.com/matthewaveryusa/xmacro/