Macros

Eric McCreath

Constants

Macros provide a simple way of creating constants in your code.

```
#define MAXLEN 200
#define HEIGHT 5.98
```

They really are just text replacement rules.

```
#define MYSIZE 100 + 5

x = 5 * MYSIZE;
// would turn into
x = 5 * 100 + 5; // this may not be what we want!!

// If you have expressions in #define it is good to add brackets
#define MYSIZE (100 + 5)
//so
x = 5 * MYSIZE;
// would turn into
x = 5 * (100 + 5);
```

The macro will end at the end of the line, however, you can use backslash (\) to continue it onto the next line.

Parameters

We can also use parameters with #define

```
#define MAX(A,B) ((A)<(B)?(A):(B)) // note, no space between the X and (
```

These macros can be used to shorten our code without the use of a function call.

When creating macros it is good to overdo the brackets.

```
#define TWOTIMES(x) (2.0 * (x))
```

What problems would we run into if you left off the brackets in the above expression?

Australian National University

Using Macros

- They are also often used, in combination with #ifdef, #else, #ifndef, and #endif, to enable code to work on different platforms.
- Use them with care as they can make code harder to read and search.
- Macros are part of the pre-processor of c.