Assert

When a programmer writes an *assert* statement he's saying:

"This condition must be true or we have an error!" For example:

```
#include <assert.h>
#define MAX_INTS 100
int main() {
    int ints[MAX_INTS];
    // i should be in bounds, but is it really?
    i = foo(<something complicated>);
    // safety assertions
    assert(i>=0);
    assert(i<MAX_INTS);
    ints[i] = 0;
    return 0;
}</pre>
```

However if we'll add the line "#define NDEBUG" just before "#include <assert.h>, the two asserts will be translated into nothing by the preprocessor and the program will run without the overhead of the bounds checking.

Using assert

assert is a simple yet powerful tool for debugging. A few seconds putting in *assert* statements can save you hours of debugging. In the final release you get rid of the overhead of the checking by simply defining NDEBUG. A common error in using *asserts* is to put expressions in *assert* tests which need to be evaluated for the proper functioning of the program:

```
// bad, foo() will not be called the
// if compiler removes the assert() ⇔
// if NDEBUG is defined
assert(foo() == 0);
// better
int errCode = foo();
assert(errCode == 0);
// much better (possible only if you have the
// source code of foo())
```

Notice: *assert* is not meant as a tool to check user input. *assert* should also not be used as a way to check error code from functions such as *malloc* which may fail because of shortage in resources (it's not a bug). *assert* should only be used for catching bugs.