

```

1  /*****
2  * AUTHOR          : Nick Reardon
3  * Extra Credit #1 : Extendable Array
4  * CLASS           : CS1D
5  * SECTION         : MW - 2:30p
6  * DUE DATE        : 02 / 12 / 20
7  *****/
8  #ifndef _EXCEPT_H_
9  #define _EXCEPT_H_
10
11 #include <string>
12 #include <exception>
13 #include <sstream>
14
15 /**
16 * @class   Except Except.h Except.h
17 *
18 * @brief   Generic Exception class with basic output setup
19 *
20 * @author  Nick Reardon
21 * @date    12/09/2020
22 */
23 class Except : virtual public std::runtime_error {
24
25 protected:
26
27     int error_number;          ///< Error number
28     int error_offset;          ///< Error offset
29
30 public:
31
32     /**
33     * @fn    explicit Except::Except(const std::string& msg, int err_num, int
34             err_off)
35             : std::runtime_error(msg)
36     * @brief Constructor
37     *
38     * @param msg      The error message.
39     * @param err_num  Error number.
40     * @param err_off  Error offset.
41     */
42     explicit Except(const std::string& msg, int err_num, int err_off) :
43         std::runtime_error(msg)
44     {
45         error_number = err_num;
46         error_offset = err_off;
47     }
48
49     /** Destructor.
50     * Virtual to allow for subclassing.

```

```
52     */
53     virtual ~Except() throw () {}
54
55     /** Returns error number.
56      * @return #error_number
57      */
58     virtual int getErrorNumber() const throw() {
59         return error_number;
60     }
61
62     /**Returns error offset.
63      * @return #error_offset
64      */
65     virtual int getErrorOffset() const throw() {
66         return error_offset;
67     }
68
69     virtual void outputError(std::ostream& output) const throw()
70     {
71         output << "Exception - Error number " << error_number
72             << ":" << std::string(error_offset, ' ') << what() << '\n';
73     }
74
75 };
76
77
78 #endif // !_EXCEPT_H_
```