```cpp
/**************************************************************************
 * AUTHOR           : Nick Reardon
 * Assignment #7    : Hashing Algorithms
 * CLASS            : CS1D
 * SECTION          : MW - 2:30p
 * DUE DATE         : 03 / 04 / 20
 **************************************************************************/
#ifndef _EXCEPT_H_
#define _EXCEPT_H_

#include <string>
#include <exception>
#include <sstream>

/**
 * @class   Except Except.h Except.h
 *
 * @brief   Generic Exception class with basic output setup
 *
 * @author  Nick Reardon
 * @date    12/09/2020
 */
class Except : virtual public std::runtime_error {

protected:

    int error_number;               ///< Error number
    int error_offset;               ///< Error offset

public:

    /**
     * @fn  explicit Except::Except(const std::string& msg, int err_num, int
       err_off)
     *      : std::runtime_error(msg)
     * @brief   Constructor
     *
     * @param   msg     The error message.
     * @param   err_num Error number.
     * @param   err_off Error offset.
     */
    explicit
        Except(const std::string& msg, int err_num, int err_off) :
        std::runtime_error(msg)
    {
        error_number = err_num;
        error_offset = err_off;

    }

    /** Destructor.
     *  Virtual to allow for subclassing.
     */
    virtual ~Except() throw () {}

    /** Returns error number.
     *  @return #error_number
     */
    virtual int getErrorNumber() const throw() {
        return error_number;
    }

    /**Returns error offset.
     * @return #error_offset
     */
    virtual int getErrorOffset() const throw() {
        return error_offset;
    }

    virtual void outputError(std::ostream& output) const throw()
    {
        output << "Exception - Error number " << error_number
            << ":" << std::string(error_offset, ' ') << what() << '\n';
    }

};


#endif // !_EXCEPT_H_
```