Source File: `~/1337/40/lab40.(C|CPP|cpp|c++|cc|cxx|cp)`
Input: Under control of `main` function
Output: Under control of `main` function
Value: 2

The purpose of this assignment is to write a version of the selection sort that uses pointers. A version of the selection sort that uses indexing can be found in the textbook. The description and prototype of each of the functions can be found in the `main` function shown in Figure 1. A sample execution sequence is shown in Figure 2. To use the `Makefile` as distributed in class, add a target of `lab40` to `targets2srcfiles`.

```
#include <iostream>
#include <cstdlib>

using namespace std;

// selectionSortUsingIndexing:  Using indexing, this function sorts
// in ascending order the n-element integer array using a selection sort.
void selectionSortUsingIndexing(int array[], int n);

// selectionSortUsingPointers:  Using pointers, this function sorts
// in ascending order the n-element integer array using a selection sort.
void selectionSortUsingPointers(int *array, int n);

// printArrayUsingPointers:  prints the n-element integer array to
// output stream os; the function uses pointers
void printArrayUsingPointers(const int *array, int n, ostream& os);

// swap: The swap function swaps a and b in memory.
void swap(int &a, int &b);

int main()
{
  int i, array[100], *copyOfArray, *ptr, *ptr2;

  // initialize array via values read from the keyboard
  i = 0;
  ptr = array;
  while (i < 100 && cin >> *ptr++)
    i++;

  // dynamically allocate an array large enough to hold a copy of array
  try
  {
    copyOfArray = new int[i];
  }
  catch (bad_alloc error)
  {
    cerr << "Couldn't allocate memory. Exiting." << endl;
    cerr << "Exception occurred: " << error.what() << endl;
    exit(EXIT_FAILURE);
  }
```

Figure 1. `/usr/local/1337/src/lab40main.C` (Part 1 of 3)

```
42
43    // make the copy of array element-by-element to copyOfArray
44    for (ptr = array, ptr2 = copyOfArray; ptr < array + i; ++ptr, ++ptr2)
45      *ptr2 = *ptr;
46
47    if (i == 0)
48      cout << "No data" << endl;
49    else
50    {
51      cout << "Before call to selectionSortUsingIndexing:" << endl;
52      printArrayUsingPointers(array, i, cout);
53      selectionSortUsingIndexing(array, i);
54      cout << endl << "After call to selectionSortUsingIndexing:" << endl;
55      printArrayUsingPointers(array, i, cout);
56
57      cout << endl << "Before call to selectionSortUsingPointers:" << endl;
58      printArrayUsingPointers(copyOfArray, i, cout);
59      selectionSortUsingPointers(copyOfArray, i);
60      cout << endl << "After call to selectionSortUsingPointers:" << endl;
61      printArrayUsingPointers(copyOfArray, i, cout);
62    }
63
64    delete [] copyOfArray;
65
66    return EXIT_SUCCESS;
67  }
68
69  //*********************************************************************
70  // Gaddis, Tony, "Starting Out with C++:  From Control Structures      *
71  // through Objects," Ninth ed., Addison Wesley, 2018, pp. 482 - 486.   *
72  //                                                                     *
73  // Definition of function selectionSort.                               *
74  // This function performs an ascending order selection sort on         *
75  // array.  size is the number of elements in the array.                *
76  //*********************************************************************
77
78  void selectionSortUsingIndexing(int array[], int size)
79  {
80    int minIndex, minValue;
81
82    for (int start = 0; start < (size - 1); start++)
83    {
84      minIndex = start;
85      minValue = array[start];
86      for (int index = start + 1; index < size; index++)
87      {
88        if (array[index] < minValue)
89        {
90          minValue = array[index];
```

Figure 1. `/usr/local/1337/src/lab40main.C` (Part 2 of 3)

```
91          minIndex = index;
92        }
93      }
94      swap(array[minIndex], array[start]);
95    }
96  }
97
98  void swap(int &a, int &b)
99  {
100   int temp = a;
101   a = b;
102   b = temp;
103 }
104
105 void printArrayUsingPointers(const int *array, int n, ostream& os)
106 {
107   const int *ptr;
108   const int *const endPtr = array + n;
109
110   if (n > 0)
111   {
112     os << "array" << endl << '{' << endl;
113     for (ptr = array; ptr < endPtr; ++ptr)
114       os << "  [" << ptr - array << "] = " << *ptr << endl;
115     os << '}' << endl;
116   }
117 }
```

Figure 1. `/usr/local/1337/src/lab40main.C` (Part 3 of 3)

```
newuser@csunix ~> cd 1337
newuser@csunix ~/1337> mkdir 40
newuser@csunix ~/1337> cd 40
newuser@csunix ~/1337/40> cp /usr/local/1337/data/40/* .
newuser@csunix ~/1337/40> cp /usr/local/1337/src/lab40main.C .
newuser@csunix ~/1337/40> cp /usr/local/1337/src/Makefile .
newuser@csunix ~/1337/40> touch lab40.cpp
newuser@csunix ~/1337/40> # Edit Makefile and lab40.cpp
newuser@csunix ~/1337/40> make lab40
g++ -g -Wall -std=c++11 -c lab40main.C -I/usr/local/1337/include -I.
g++ -g -Wall -std=c++11 -c lab40.cpp -I/usr/local/1337/include -I.
g++ -o lab40 lab40main.o lab40.o -L/usr/local/1337/lib -lm -lbits
newuser@csunix ~/1337/40> cat 01.dat
2305 1361 1362 1331 1341 1351
```

```
newuser@csunix ~/1337/40> cat 01.dat | ./lab40
Before call to selectionSortUsingIndexing:
array
{
  [0] = 2305
  [1] = 1361
  [2] = 1362
  [3] = 1331
  [4] = 1341
  [5] = 1351
}

After call to selectionSortUsingIndexing:
array
{
  [0] = 1331
  [1] = 1341
  [2] = 1351
  [3] = 1361
  [4] = 1362
  [5] = 2305
}
```

```

Before call to selectionSortUsingPointers:
array
{
  [0] = 2305
  [1] = 1361
  [2] = 1362
  [3] = 1331
  [4] = 1341
  [5] = 1351
}

After call to selectionSortUsingPointers:
array
{
  [0] = 1331
  [1] = 1341
  [2] = 1351
  [3] = 1361
  [4] = 1362
  [5] = 2305
}
```

```
newuser@csunix ~/1337/40> cat 01.dat | ./lab40 > my.out
newuser@csunix ~/1337/40> diff 01.out my.out
newuser@csunix ~/1337/40> cat 02.dat | ./lab40 > my.out
newuser@csunix ~/1337/40> diff 02.out my.out
newuser@csunix ~/1337/40>
```

Figure 2. Commands to Compile, Link, & Run Lab 40