```cpp
/**
 * @file -  ArrayStackMain.cpp
 *
 * Uses ArrayStack<T> that has implemented StackADT<T> using an array as the internal data structure for the stack elements.
 *
 * @author - Humayun Kabir, Instructor, CSCI 161, VIU
 * @version - 0.0.1
 * @date - April 25, 2021
 *
 */


#include <iostream>
#include <string>
#include "StackADT.h"
#include "ArrayStack.cpp"

using namespace std;

ArrayStack<string> getColorStack() {
    int count = 7;
    string colors[] = {"Red", "Orange", "Yellow", "Green", "Cyan", "Blue", "Violet"};
    ArrayStack<string> arraystack(count);
    for (int i = 0; i<count; i++) {
        arraystack.push(colors[i]);
    }
    return arraystack;
}

int main(){


  cout << "************** Testing ArrayStack ***************" << endl;

  /*
   * ArrayStack default constructor
   */
  ArrayStack<double> arraystackdouble1;

  cout << "ArrayStack<double> default constructor..." << endl;
  cout << "ArrayStack<double> capacity: " << arraystackdouble1.getCapacity() << endl;
  cout << "ArrayStack<double> isEmpty: " << arraystackdouble1.isEmpty() << endl;
  cout << "ArrayStack<double> size: " << arraystackdouble1.getSize() << endl;

  try {
  cout << "ArrayStack<double> pushing 10.01"<<endl;
  arraystackdouble1.push(10.01);
  cout << "ArrayStack<double> peek(): " << arraystackdouble1.peek() << endl;

  cout << "ArrayStack<double> pushing 20.02"<<endl;
  arraystackdouble1.push(20.02);
  cout << "ArrayStack<double> peek(): " << arraystackdouble1.peek() << endl;

  cout << "ArrayStack<double> pushing 30.03"<<endl;
  arraystackdouble1.push(30.03);
  cout << "ArrayStack<double> peek(): " << arraystackdouble1.peek() << endl;

  cout << "ArrayStack<double> pushing 40.04"<<endl;
  arraystackdouble1.push(40.04);
  cout << "ArrayStack<double> peek(): " << arraystackdouble1.peek() << endl;

  cout << "ArrayStack<double> pushing 50.05"<<endl;
  arraystackdouble1.push(50.05);
  cout << "ArrayStack<double> peek(): " << arraystackdouble1.peek() << endl;

  }
  catch (const char* excp) {
      cout<<excp<<endl;
```

```
69      }
70      cout <<"ArrayStack<double> after pushing 10.01, 20.02, 30.03, 40.04, and 50.05" << endl
        ;
71      cout << "ArrayStack<double> capacity: " << arraystackdouble1.getCapacity() << endl;
72      cout << "ArrayStack<double> isEmpty: " << arraystackdouble1.isEmpty() << endl;
73      cout << "ArrayStack<double> size: " << arraystackdouble1.getSize() << endl;
74
75      try {
76        cout << "ArrayStack<double> pop(): " << arraystackdouble1.pop() << endl;
77        cout << "ArrayStack<double> pop(): " << arraystackdouble1.pop() << endl;
78        cout << "ArrayStack<double> pop(): " << arraystackdouble1.pop() << endl;
79        cout << "ArrayStack<double> pop(): " << arraystackdouble1.pop() << endl;
80        cout << "ArrayStack<double> pop(): " << arraystackdouble1.pop() << endl;
81        cout << "ArrayStack<double> pop(): " << arraystackdouble1.pop() << endl;
82      }
83      catch (const char* excp) {
84          cout << excp << endl;
85      }
86
87
88      /*
89       * ArrayStack regular constructor
90       */
91
92      cout << "ArrayStack<double> regular constructor..." << endl;
93
94      int capacity = 5;
95      ArrayStack<double> arraystackdouble2(capacity);
96
97      cout << "ArrayStack<double> capacity: " << arraystackdouble2.getCapacity() << endl;
98      cout << "ArrayStack<double> isEmpty: " << arraystackdouble2.isEmpty() << endl;
99      cout << "ArrayStack<double> size: " << arraystackdouble2.getSize() << endl;
100
101
102     try {
103
104     cout << "ArrayStack<double> pushing 1.1"<<endl;
105     arraystackdouble2.push(1.1);
106     cout << "ArrayStack<double> peek(): " << arraystackdouble2.peek() << endl;
107
108     cout << "ArrayStack<double> pushing 2.2"<<endl;
109     arraystackdouble2.push(2.2);
110     cout << "ArrayStack<double> peek(): " << arraystackdouble2.peek() << endl;
111
112     cout << "ArrayStack<double> pushing 3.3"<<endl;
113     arraystackdouble2.push(3.3);
114     cout << "ArrayStack<double> peek(): " << arraystackdouble2.peek() << endl;
115
116     cout << "ArrayStack<double> pushing 4.4"<<endl;
117     arraystackdouble2.push(4.4);
118     cout << "ArrayStack<double> peek(): " << arraystackdouble2.peek() << endl;
119
120     cout << "ArrayStack<double> pushing 5.5"<<endl;
121     arraystackdouble2.push(5.5);
122     cout << "ArrayStack<double> peek(): " << arraystackdouble2.peek() << endl;
123
124     }
125     catch (const char* excp) {
126         cout<<excp<<endl;
127     }
128
129     cout << "ArrayStack<double> after pushing 1.1, 2.2, 3.3, 4.4, and 5.5" << endl;
130     cout << "ArrayStack<double> capacity: " << arraystackdouble2.getCapacity() << endl;
131     cout << "ArrayStack<double> isEmpty: " << arraystackdouble2.isEmpty() << endl;
132     cout << "ArrayStack<double> size: " << arraystackdouble2.getSize() << endl;
133
134     cout << "Attempt to push 6.6 when a the ArrayStack<double> is full"<<endl;
135     try {
136       arraystackdouble2.push(6.6);
```

```
    }
    catch (const char* excp) {
        cout<<excp<<endl;
    }


    /*
     * ArrayStack copy constructor
     */
    cout << "ArrayStack<double> copy  constructor..." << endl;
    ArrayStack<double> arraystackdouble3(arraystackdouble2);

    cout << "ArrayStack<double> capacity: " << arraystackdouble3.getCapacity() << endl;
    cout << "ArrayStack<double> isEmpty: " << arraystackdouble3.isEmpty() << endl;
    cout << "ArrayStack<double> size: " << arraystackdouble3.getSize() << endl;

    while(arraystackdouble3.getSize() > 0) {
        try {
          cout << "ArrayStack<double> pop(): "<< arraystackdouble3.pop() << endl;
        }
        catch (const char* excp) {
            cout << excp << endl;
        }
    }

    cout << "ArrayStack<double> capacity: " << arraystackdouble3.getCapacity() << endl;
    cout << "ArrayStack<double> isEmpty: " << arraystackdouble3.isEmpty() << endl;
    cout << "ArrayStack<double> size: " << arraystackdouble3.getSize() << endl;



    /*
     * ArrayStack move constructor.
     */
    cout << "ArrayStack<string> move constructor......" <<endl;
    ArrayStack<string> arraystackstring1 = getColorStack();

    cout << "ArrayStack<string> capacity: " << arraystackstring1.getCapacity() << endl;
    cout << "ArrayStack<string> isEmpty: " << arraystackstring1.isEmpty() << endl;
    cout << "ArrayStack<string> size: " << arraystackstring1.getSize() << endl;


    while(arraystackstring1.getSize() > 0) {
        try {
          cout << "ArrayStack<string> pop(): "<< arraystackstring1.pop() << endl;
        }
        catch (const char* excp) {
            cout << excp << endl;
        }
    }


    cout << "ArrayStack<string> capacity: " << arraystackstring1.getCapacity() << endl;
    cout << "ArrayStack<string> isEmpty: " << arraystackstring1.isEmpty() << endl;
    cout << "ArrayStack<string> size: " << arraystackstring1.getSize() << endl;


    /*
     * ArrayStack copy assignment
     */
    cout << "ArrayStack<double> copy assignment..." << endl;
    ArrayStack<double> arraystackdouble4;

    cout << "ArrayStack<double> before copy assignment" << endl;
    cout << "ArrayStack<double> capacity: " << arraystackdouble4.getCapacity() << endl;
    cout << "ArrayStack<double> isEmpty: " << arraystackdouble4.isEmpty() << endl;
    cout << "ArrayStack<double> size: " << arraystackdouble4.getSize() << endl;

    arraystackdouble4 = arraystackdouble2;
```

```
206
207
208     cout << "ArrayStack<double> after copy assignment" << endl;
209     cout << "ArrayStack<double> capacity: " << arraystackdouble4.getCapacity() << endl;
210     cout << "ArrayStack<double> isEmpty: " << arraystackdouble4.isEmpty() << endl;
211     cout << "ArrayStack<double> size: " << arraystackdouble4.getSize() << endl;
212
213     while(arraystackdouble4.getSize() > 0) {
214         try {
215           cout << "ArrayStack<double> pop(): "<< arraystackdouble4.pop() << endl;
216         }
217         catch (const char* excp) {
218             cout << excp << endl;
219         }
220     }
221
222     cout << "ArrayStack<double> capacity: " << arraystackdouble4.getCapacity() << endl;
223     cout << "ArrayStack<double> isEmpty: " << arraystackdouble4.isEmpty() << endl;
224     cout << "ArrayStack<double> size: " << arraystackdouble4.getSize() << endl;
225
226
227
228      /*
229      * ArrayStack move assignment.
230      */
231     cout << "ArrayStack<string> move assignment......" <<endl;
232     ArrayStack<string> arraystackstring2;
233
234     cout << "ArrayStack<string> before move assignment" << endl;
235     cout << "ArrayStack<string> capacity: " << arraystackstring2.getCapacity() << endl;
236     cout << "ArrayStack<string> isEmpty: " << arraystackstring2.isEmpty() << endl;
237     cout << "ArrayStack<string> size: " << arraystackstring2.getSize() << endl;
238
239
240     arraystackstring2 = getColorStack();
241
242     cout << "ArrayStack<string> after move assignment" << endl;
243     cout << "ArrayStack<string> capacity: " << arraystackstring2.getCapacity() << endl;
244     cout << "ArrayStack<string> isEmpty: " << arraystackstring2.isEmpty() << endl;
245     cout << "ArrayStack<string> size: " << arraystackstring2.getSize() << endl;
246
247
248     while(arraystackstring2.getSize() > 0) {
249         try {
250           cout << "ArrayStack<string> pop(): "<< arraystackstring2.pop() << endl;
251         }
252         catch (const char* excp) {
253             cout << excp << endl;
254         }
255     }
256
257
258     cout << "ArrayStack<string> capacity: " << arraystackstring2.getCapacity() << endl;
259     cout << "ArrayStack<string> isEmpty: " << arraystackstring2.isEmpty() << endl;
260     cout << "ArrayStack<string> size: " << arraystackstring2.getSize() << endl;
261
262
263     cout << "ArrayStack<int>* ..........................................................."
     << endl;
264     ArrayStack<int>* arraystack = NULL;
265     capacity = 20;
266
267     arraystack = new ArrayStack<int>(capacity);
268
269     cout << "ArrayStack<int>* capacity: " << arraystack->getCapacity() << endl;
270     cout << "ArrayStack<int>* isEmpty: " << arraystack->isEmpty() << endl;
271     cout << "ArrayStack<int>* size: " << arraystack->getSize() << endl;
272
273     try{
```

```
274       arraystack->push(100);
275
276       cout << "ArrayStack<int>* isEmpty after pushing 100: " << arraystack->isEmpty() <<
          endl;
277       cout << "ArrayStack<int>* size after pushing 100: " << arraystack->getSize() << endl;
278       cout << "ArrayStack<int>* top after pushing 100: " << arraystack->peek() << endl;
279
280       arraystack->push(200);
281
282       cout << "ArrayStack<int>* isEmpty after pushing 200: " << arraystack->isEmpty() <<
          endl;
283       cout << "ArrayStack<int>* size after pushing 200: " << arraystack->getSize() << endl;
284       cout << "ArrayStack<int>* top after pushing 200: " << arraystack->peek() << endl;
285
286       arraystack->push(300);
287
288       cout << "ArrayStack<int>* isEmpty after pushing 300: " << arraystack->isEmpty() <<
          endl;
289       cout << "ArrayStack<int>* size after pushing 300: " << arraystack->getSize() << endl;
290       cout << "ArrayStack<int>* top after pushing 300: " << arraystack->peek() << endl;
291
292       int popped = arraystack->pop();
293
294       cout << "ArrayStack<int>* isEmpty after popping: " << popped << ": " << arraystack->
          isEmpty() << endl;
295       cout << "ArrayStack<int>* size after popping: " << popped << ": " << arraystack->
          getSize() << endl;
296       cout << "ArrayStack<int>* top after popping: " << popped << ": " << arraystack->peek
          () << endl;
297
298       arraystack->push(300);
299
300       int peeked = arraystack->peek();
301
302       cout << "ArrayStack<int>* isEmpty after peeking: " << peeked << ": " << arraystack->
          isEmpty() << endl;
303       cout << "ArrayStack<int>* size after peeking: " << peeked << ": " << arraystack->
          getSize() << endl;
304       cout << "ArrayStack<int>* top after peeking: " << peeked << ": " << arraystack->peek
          () << endl;
305
306       arraystack->push(400);
307       arraystack->push(500);
308       arraystack->push(600);
309       arraystack->push(700);
310       arraystack->push(800);
311       arraystack->push(900);
312       arraystack->push(1000);
313
314       cout << "ArrayStack<int>* isEmpty after pushing 1000: " << arraystack->isEmpty() <<
          endl;
315       cout << "ArrayStack<int>* size after pushing 1000: " << arraystack->getSize() << endl
          ;
316       cout << "ArrayStack<int>* top after pushing 1000: " << arraystack->peek() << endl;
317
318       arraystack->push(1100);
319       cout << "ArrayStack<int>* isEmpty after pushing 1100: " << arraystack->isEmpty() <<
          endl;
320       cout << "ArrayStack<int>* size after pushing 1100: " << arraystack->getSize() << endl
          ;
321       cout << "ArrayStack<int>* top after pushing 1100: " << arraystack->peek() << endl;
322     }
323     catch (const char* excp){
324         cerr << excp << endl;
325     }
326     int stackSize = arraystack->getSize();
327     try{
328       for (int i=0; i<stackSize; i++) {
329         cout << "ArrayStack<int>* pop(): " << arraystack->pop() << endl;
```

```
330       }
331       cout << "ArrayStck<int>* pop() " << arraystack->pop() << endl;
332     }
333     catch (const char* excp){
334         cerr << excp << endl;
335     }
336 
337     cout << "StackADT<int>* ................................................." << endl;
338     StackADT<int>* stack = NULL;
339 
340     stack = new ArrayStack<int>(capacity);
341 
342     //cout << "StackADT<int>* capacity: " << stack->getCapacity() << endl;
343     cout << "StackADT<int>* isEmpty: " << stack->isEmpty() << endl;
344     cout << "StackADT<int>* size: " << stack->getSize() << endl;
345 
346     try{
347       stack->push(100);
348 
349       cout << "StackADT<int>* isEmpty after pushing 100: " << stack->isEmpty() << endl;
350       cout << "StackADT<int>* size after pushing 100: " << stack->getSize() << endl;
351       cout << "StackADT<int>* top after pushing 100: " << stack->peek() << endl;
352 
353       stack->push(200);
354 
355       cout << "StackADT<int>* isEmpty after pushing 200: " << stack->isEmpty() << endl;
356       cout << "StackADT<int>* size after pushing 200: " << stack->getSize() << endl;
357       cout << "StackADT<int>* top after pushing 200: " << stack->peek() << endl;
358 
359       stack->push(300);
360 
361       cout << "StackADT<int>* isEmpty after pushing 300: " << stack->isEmpty() << endl;
362       cout << "StackADT<int>* size after pushing 300: " << stack->getSize() << endl;
363       cout << "StackADT<int>* top after pushing 300: " << stack->peek() << endl;
364 
365       int popped = stack->pop();
366 
367       cout << "StackADT<int>* isEmpty after popping: " << popped << ": " << stack->isEmpty
          () << endl;
368       cout << "StackADT<int>* size after popping: " << popped << ": " << stack->getSize()
          << endl;
369       cout << "StackADT<int>* top after popping: " << popped << ": " << stack->peek() <<
          endl;
370 
371       stack->push(300);
372 
373       int peeked = stack->peek();
374 
375       cout << "StackADT<int>* isEmpty after peeking: " << peeked << ": " << stack->isEmpty
          () << endl;
376       cout << "StackADT<int>* size after peeking: " << peeked << ": " << stack->getSize()
          << endl;
377       cout << "StackADT<int>* top after peeking: " << peeked << ": " << stack->peek() <<
          endl;
378 
379       stack->push(400);
380       stack->push(500);
381       stack->push(600);
382       stack->push(700);
383       stack->push(800);
384       stack->push(900);
385       stack->push(1000);
386 
387       cout << "StackADT<int>* isEmpty after pushing 1000: " << stack->isEmpty() << endl;
388       cout << "StackADT<int>* size after pushing 1000: " << stack->getSize() << endl;
389       cout << "StackADT<int>* top after pushing 1000: " << stack->peek() << endl;
390 
391       stack->push(1100);
392       cout << "StackADT<int>* isEmpty after pushing 1100: " << stack->isEmpty() << endl;
```

```
393       cout << "StackADT<int>* size after pushing 1100: " << stack->getSize() << endl;
394       cout << "StackADT<int>* top after pushing 1100: " << stack->peek() << endl;
395     }
396     catch (const char* excp){
397         cerr << excp << endl;
398     }
399     stackSize = stack->getSize();
400     try{
401       for (int i=0; i<stackSize; i++) {
402         cout << "StackADT<int>* pop(): " << stack->pop() << endl;
403       }
404       cout << "StackADT<int>* pop(): " << stack->pop() << endl;
405     }
406     catch (const char* excp){
407         cerr << excp << endl;
408     }
409 
410     /*
411      * Releasing memory from dynimically coreated objects
412      */
413     if(arraystack != NULL) {
414       delete arraystack;
415     }
416 
417     if(stack != NULL) {
418         delete stack;
419     }
420 
421     cout << "************** End of Testing ArrayStack ***************" << endl;
422 
423 
424     return 0;
425   }
426 
```