```
1  /*-----------------------------------------------------------------
2  Copyright (c) 2015 Author: Jagadeesh Vasudevamurthy
3  file: ../complex/complex.cpp dstacktest.cpp
4
5  On linux:
6  g++ ../complex/complex.cpp dstacktest.cpp
7  valgrind a.out
8  valgrind --leak-check=full -v a.out
9  -- All heap blocks were freed -- no leaks are possible
10
11 -----------------------------------------------------------------*/
12
13 /*-----------------------------------------------------------------
14 This file test dstack object
15 -----------------------------------------------------------------*/
16
17 /*-----------------------------------------------------------------
18 All includes here
19 -----------------------------------------------------------------*/
20 #include "dstack.h"
21 #include "../complex/complex.h"
22
23 /*-------------------------------------------------------
24 local to this file. Change verbose = true for debugging
25 -------------------------------------------------------*/
26 static bool verbose = true;
27
28 /*-----------------------------------------------------------------
29 Print an integer - value given
30 -----------------------------------------------------------------*/
31 static void print(int& x) {
32   cout << x << " ";
33 }
34
35 /*-----------------------------------------------------------------
36 Print an integer - address given
37 -----------------------------------------------------------------*/
38 static void print(int*& x) {
39   print(*(x));
40 }
41
42 /*-----------------------------------------------------------------
43 Print a complex - value given
44 -----------------------------------------------------------------*/
45 static void print(complex& x) {
46   cout << x << " ";
47 }
48
49 /*-----------------------------------------------------------------
50 Print a complex  - address given
51 -----------------------------------------------------------------*/
52 static void print(complex*& x) {
53   print(*(x));
54 }
55
56 /*-----------------------------------------------------------------
57 add by 2000 - value given
58 -----------------------------------------------------------------*/
59 static void add2000(int& x) {
60   x = x + 2000;
61 }
62
63 /*-----------------------------------------------------------------
64 add by 2000 - address given
65 -----------------------------------------------------------------*/
66 static void add2000(int*& x) {
```

```
  add2000(*(x));
}

/*--------------------------------------------------------------
add by 2000 - value given
--------------------------------------------------------------*/
static void add2000(complex& c) {
  int x, y;
  c.getxy(x, y);
  x = x + 2000;
  y = y + 2000;
  c.setxy(x, y);
}

/*--------------------------------------------------------------
add by 2000 - address given
--------------------------------------------------------------*/
static void add2000(complex*& x) {
  add2000(*(x));
}

/*--------------------------------------------------------------
delete integer - address given
--------------------------------------------------------------*/
static void delete_obj(int*&  x) {
  delete(x);
}

/*--------------------------------------------------------------
delete complex - address given
--------------------------------------------------------------*/
static void delete_obj(complex*&  x) {
  delete(x);
}


/*--------------------------------------------------------------
array of integers
--------------------------------------------------------------*/
static void test_stack_of_integers(){
  {
    //THIS will not compile
    //int x = 8;
    //dstack<int> s(8,verbose);
  }
  dstack<int> s(3,verbose);
  cout << "Number of element in the stack is: " << s.num_elements() << endl;
  for (int i = 0; i < 8; i++) {
    s.push(1000 + i);
  }
  s.for_each_element_of_stack_from_top_to_bottom(print);
  cout << endl;
  s.for_each_element_of_stack_from_top_to_bottom(add2000);
  s.for_each_element_of_stack_from_top_to_bottom(print);
  cout << endl;
  for (int i = 0; i < 7; i++){
    int& x = s.top();
    cout << "Top element = " << x << endl;
    s.pop();
  }
  int& x = s.top();
  cout << "top element = " << x << endl;
  for (int i = 0; i < 8; i++) {
    s.push(-(8000 + i));
  }
  s.for_each_element_of_stack_from_top_to_bottom(print);
```

```
133   cout << endl;
134   int& y = s.top();
135   y = 9999;
136   s.for_each_element_of_stack_from_top_to_bottom(print);
137   cout << endl;
138   int z = s.top();
139   z = 8888;
140   s.for_each_element_of_stack_from_top_to_bottom(print);
141   cout << endl;
142 }
143
144 /*-------------------------------------------------------------
145 array of integer pointers
146 -------------------------------------------------------------*/
147 static void test_stack_of_ptr_integers(){
148   dstack<int*> s(3,verbose);
149   cout << "Number of element in the stack is: " << s.num_elements() << endl;
150   for (int i = 0; i < 8; i++) {
151     s.push(new(int)(1000 + i));
152   }
153   s.for_each_element_of_stack_from_top_to_bottom(print);
154   cout << endl;
155   s.for_each_element_of_stack_from_top_to_bottom(add2000);
156   s.for_each_element_of_stack_from_top_to_bottom(print);
157   cout << endl;
158   for (int i = 0; i < 7; i++){
159     int*& x = s.top();
160     cout << "top element = " << *x << endl;
161     delete(x);
162     s.pop();
163   }
164   int x = *(s.top());
165   cout << "Top element = " << x << endl;
166   for (int i = 0; i < 8; i++) {
167     s.push(new(int)(-(8000 + i)));
168   }
169   s.for_each_element_of_stack_from_top_to_bottom(print);
170   cout << endl;
171   s.for_each_element_of_stack_from_top_to_bottom(delete_obj);
172 }
173
174 /*-------------------------------------------------------------
175 array of user defined type
176 -------------------------------------------------------------*/
177 static void test_stack_of_udt(){
178   dstack<complex> s(3, verbose);
179   cout << "Number of element in the stack is: " << s.num_elements() << endl;
180   for (int i = 0; i < 8; i++) {
181     s.push(complex(1000 + i, -(1000 + i)));
182   }
183   s.for_each_element_of_stack_from_top_to_bottom(print);
184   cout << endl;
185   s.for_each_element_of_stack_from_top_to_bottom(add2000);
186   s.for_each_element_of_stack_from_top_to_bottom(print);
187   cout << endl;
188   for (int i = 0; i < 7; i++){
189     cout << "Top element = " << s.top() << endl;
190     s.pop();
191   }
192   cout << "Top element = " << s.top() << endl;
193   for (int i = 0; i < 8; i++) {
194     s.push(complex(-(8000 + i), -(-(8000 + i))));
195   }
196   s.for_each_element_of_stack_from_top_to_bottom(print);
197   cout << endl;
198   complex& y = s.top();
```

```
199   y = 9999;
200   s.for_each_element_of_stack_from_top_to_bottom(print);
201   cout << endl;
202   complex z = s.top();
203   z = 8888;
204   s.for_each_element_of_stack_from_top_to_bottom(print);
205   cout << endl;
206 }
207
208 /*-------------------------------------------------------------------
209 array of user defined pointer type
210 -------------------------------------------------------------------*/
211 static void test_stack_of_ptr_udt(){
212   dstack<complex*> s(3, verbose);
213   cout << "Number of element in the stack is: " << s.num_elements() << endl;
214   for (int i = 0; i < 8; i++) {
215     s.push(new(complex)(complex(1000 + i, -(1000 + i))));
216   }
217   s.for_each_element_of_stack_from_top_to_bottom(print);
218   cout << endl;
219   s.for_each_element_of_stack_from_top_to_bottom(add2000);
220   s.for_each_element_of_stack_from_top_to_bottom(print);
221   cout << endl;
222   for (int i = 0; i < 7; i++){
223     complex*& x = s.top();
224     cout << "top element = " << *(x) << endl;
225     delete(x);
226     s.pop();
227   }
228   cout << "top element = " << s.top() << endl;
229   for (int i = 0; i < 8; i++) {
230     s.push(new(complex)((-(8000 + i), -(-(8000 + i)))));
231   }
232   s.for_each_element_of_stack_from_top_to_bottom(print);
233   cout << endl;
234   s.for_each_element_of_stack_from_top_to_bottom(delete_obj);
235 }
236
237 /*-------------------------------------------------------------------
238 main
239 -------------------------------------------------------------------*/
240 int main() {
241   complex::set_display(verbose);
242   test_stack_of_integers();
243   cout << "_____________________________" << endl;
244   test_stack_of_ptr_integers();
245   cout << "_____________________________" << endl;
246   test_stack_of_udt();
247   cout << "_____________________________" << endl;
248   test_stack_of_ptr_udt();
249   cout << "_____________________________" << endl;
250   return 0;
251 }
252
253
254 //EOF
255
256
```