```
1  /*-----------------------------------------------------------------
2  Copyright (c) 2014 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  -- All heap blocks were freed -- no leaks are possible
9
10 -----------------------------------------------------------------*/
11
12 /*-----------------------------------------------------------------
13 This file test dstack object
14 -----------------------------------------------------------------*/
15
16 /*-----------------------------------------------------------------
17 All includes here
18 -----------------------------------------------------------------*/
19 #include "dstack.h"
20 #include "../complex/complex.h"
21
22 /*-----------------------------------------------------------
23 static definition - only once at the start
24 Change to false, if you don't need verbose
25 -----------------------------------------------------------*/
26 template <typename T>
27 bool dstack<T>::_display = false;
28
29 /*-----------------------------------------------------------
30 local to this file. Change verbose = true for debugging
31 -----------------------------------------------------------*/
32 static bool verbose = true;
33
34 /*-----------------------------------------------------------------
35 Print an integer - value given
36 -----------------------------------------------------------------*/
37 static void print(int& x) {
38   cout << x << " ";
39 }
40
41 /*-----------------------------------------------------------------
42 Print an integer - address given
43 -----------------------------------------------------------------*/
44 static void print(int*& x) {
45   print(*(x));
46 }
47
48 /*-----------------------------------------------------------------
49 Print a complex - value given
50 -----------------------------------------------------------------*/
51 static void print(complex& x) {
52   cout << x << " ";
53 }
54
55 /*-----------------------------------------------------------------
56 Print a complex  - address given
57 -----------------------------------------------------------------*/
58 static void print(complex*& x) {
59   print(*(x));
60 }
61
62 /*-----------------------------------------------------------------
63 add by 2000 - value given
64 -----------------------------------------------------------------*/
65 static void add2000(int& x) {
66   x = x + 2000;
```

```
67 }
68
69 /*--------------------------------------------------------------
70 add by 2000 - address given
71 --------------------------------------------------------------*/
72 static void add2000(int*& x) {
73   add2000(*(x));
74 }
75
76 /*--------------------------------------------------------------
77 add by 2000 - value given
78 --------------------------------------------------------------*/
79 static void add2000(complex& c) {
80   int x, y;
81   c.getxy(x, y);
82   x = x + 2000;
83   y = y + 2000;
84   c.setxy(x, y);
85 }
86
87 /*--------------------------------------------------------------
88 add by 2000 - address given
89 --------------------------------------------------------------*/
90 static void add2000(complex*& x) {
91   add2000(*(x));
92 }
93
94 /*--------------------------------------------------------------
95 delete integer - address given
96 --------------------------------------------------------------*/
97 static void delete_obj(int*&  x) {
98   delete(x);
99 }
100
101 /*--------------------------------------------------------------
102 delete complex - address given
103 --------------------------------------------------------------*/
104 static void delete_obj(complex*&  x) {
105   delete(x);
106 }
107
108
109 /*--------------------------------------------------------------
110 array of integers
111 --------------------------------------------------------------*/
112 static void test_stack_of_integers(){
113   dstack<int>::set_display(verbose);
114   dstack<int> s(3);
115   cout << "Number of element in the stack is: " << s.num_elements() << endl;
116   for (int i = 0; i < 8; i++) {
117     s.push(1000 + i);
118   }
119   s.for_each_element_of_stack_from_top_to_bottom(print);
120   cout << endl;
121   s.for_each_element_of_stack_from_top_to_bottom(add2000);
122   s.for_each_element_of_stack_from_top_to_bottom(print);
123   cout << endl;
124   for (int i = 0; i < 7; i++){
125     int& x = s.top();
126     cout << "Top element = " << x << endl;
127     s.pop();
128   }
129   int& x = s.top();
130   cout << "top element = " << x << endl;
131   for (int i = 0; i < 8; i++) {
132     s.push(-(8000 + i));
```

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

```
199   }
200   s.for_each_element_of_stack_from_top_to_bottom(print);
201   cout << endl;
202   complex& y = s.top();
203   y = 9999;
204   s.for_each_element_of_stack_from_top_to_bottom(print);
205   cout << endl;
206   complex z = s.top();
207   z = 8888;
208   s.for_each_element_of_stack_from_top_to_bottom(print);
209   cout << endl;
210 }
211
212 /*--------------------------------------------------------------------
213 array of user defined pointer type
214 --------------------------------------------------------------------*/
215 static void test_stack_of_ptr_udt(){
216   dstack<complex*>::set_display(verbose);
217   dstack<complex*> s(3);
218   cout << "Number of element in the stack is: " << s.num_elements() << endl;
219   for (int i = 0; i < 8; i++) {
220     s.push(new(complex)(complex(1000 + i, -(1000 + i))));
221   }
222   s.for_each_element_of_stack_from_top_to_bottom(print);
223   cout << endl;
224   s.for_each_element_of_stack_from_top_to_bottom(add2000);
225   s.for_each_element_of_stack_from_top_to_bottom(print);
226   cout << endl;
227   for (int i = 0; i < 7; i++){
228     complex*& x = s.top();
229     cout << "top element = " << *(x) << endl;
230     delete(x);
231     s.pop();
232   }
233   cout << "top element = " << s.top() << endl;
234   for (int i = 0; i < 8; i++) {
235     s.push(new(complex)((-(8000 + i), -(-(8000 + i)))));
236   }
237   s.for_each_element_of_stack_from_top_to_bottom(print);
238   cout << endl;
239   s.for_each_element_of_stack_from_top_to_bottom(delete_obj);
240 }
241
242 /*--------------------------------------------------------------------
243 main
244 --------------------------------------------------------------------*/
245 int main() {
246   test_stack_of_integers();
247   cout << "_____________________________" << endl;
248   test_stack_of_ptr_integers();
249   cout << "_____________________________" << endl;
250   test_stack_of_udt();
251   cout << "_____________________________" << endl;
252   test_stack_of_ptr_udt();
253   cout << "_____________________________" << endl;
254   return 0;
255 }
256
257
258 //EOF
259
260
```