```
 1: // $Id: iterintvec.cpp,v 1.30 2015-04-14 13:25:24-07 - - $
 2:
 3: //
 4: // iterintvec - implementation of an int vector with iterator.
 5: //
 6:
 7: #include <iostream>
 8: #include <stdexcept>
 9:
10: using namespace std;
11:
12: //////////////////////////////////////////////////////////////////
13: // iterintvec.h
14: //////////////////////////////////////////////////////////////////
15:
16: class iterintvec {
17:    private:
18:       size_t _size;
19:       int *_data;
20:       void copy_data (int *data);
21:       void range_check (size_t index) const;
22:    public:
23:       using value_type = int;
24:       using reference = int&;
25:       using const_reference = const int&;
26:       using pointer = int*;
27:       using const_pointer = const int*;
28:       using difference_type = ptrdiff_t;
29:       using size_type = size_t;
30:       class iterator;
31:       friend class iterintvec::iterator;
32:       iterintvec ();                                  // default ctor
33:       iterintvec (const iterintvec&);                 // copy ctor
34:       iterintvec (iterintvec&&);                      // move ctor
35:       iterintvec& operator= (const iterintvec&); // copy operator=
36:       iterintvec& operator= (iterintvec&&);      // move operator=
37:       ~iterintvec();                                  // dtor
38:       explicit iterintvec (size_t size);
39:       size_t size() const;
40:       int get (size_t index) const;
41:       void put (size_t index, int value);
42:       iterator begin();
43:       iterator end();
44: };
45:
```

```
46:
47: class iterintvec::iterator {
48:    private:
49:       pointer curr;
50:       friend class iterintvec;
51:       iterator (pointer init): curr (init) {};
52:    public:
53:       iterator(): curr (nullptr) {};
54:       reference operator* () { return *curr; }
55:       const_reference operator* () const { return *curr; }
56:       iterator& operator++ () { ++curr; return *this; }
57:       iterator operator++ (int) {
58:          iterator tmp {*this}; ++curr; return tmp;
59:       }
60:       bool operator== (const iterator& that) {
61:          return curr == that.curr;
62:       }
63:       bool operator!= (const iterator& that) {
64:          return not (*this == that);
65:       }
66:       operator bool() { return curr != nullptr; }
67: };
68:
```

```
 69:
 70: //////////////////////////////////////////////////////////////////
 71: // iterintvec.cpp
 72: //////////////////////////////////////////////////////////////////
 73:
 74: // Private.
 75: void iterintvec::copy_data (int *data) {
 76:    for (size_t index = 0; index < _size; ++index) {
 77:       _data[index] = data[index];
 78:    }
 79: }
 80:
 81: // Private.
 82: void iterintvec::range_check (size_t index) const {
 83:    if (index >= _size) throw out_of_range ("iterintvec::range_check");
 84: }
 85:
 86: // Default ctor.
 87: iterintvec::iterintvec(): _size(0), _data(nullptr){}
 88:
 89: // Copy constructor.
 90: iterintvec::iterintvec (const iterintvec& that):
 91:             _size(that._size), _data (new int[that._size]) {
 92:    copy_data (that._data);
 93: }
 94:
 95: // Move constructor.
 96: iterintvec::iterintvec (iterintvec&& that):
 97:             _size(that._size), _data (that._data) {
 98:    that._size = 0;
 99:    that._data = nullptr;
100: }
101:
102: // Copy operator=
103: iterintvec& iterintvec::operator= (const iterintvec& that){
104:    if (this != &that) {
105:       if (_data != nullptr) delete[] _data;
106:       _size = that._size;
107:       _data = new int[that._size];
108:       copy_data (that._data);
109:    }
110:    return *this;
111: }
112:
113: // Move operator=
114: iterintvec& iterintvec::operator= (iterintvec&& that){
115:    if (this != &that) {
116:       if (_data != nullptr) delete[] _data;
117:       _size = that._size;
118:       _data = that._data;
119:       that._size = 0;
120:       that._data = nullptr;
121:    }
122:    return *this;
123: }
124:
```

```
125:
126: // Destructor.
127: iterintvec::~iterintvec() {
128:    if (_data != nullptr) delete[] _data;
129: }
130:
131: // Fixed-size allocator.
132: iterintvec::iterintvec (size_t size):
133:                 _size(size), _data (new int[_size]) {
134:    for (size_t index = 0; index < _size; ++index) {
135:       _data[index] = 0;
136:    }
137: }
138:
139: size_t iterintvec::size() const {
140:    return _size;
141: }
142:
143: int iterintvec::get (size_t index) const {
144:    range_check (index);
145:    return _data[index];
146: }
147:
148: void iterintvec::put (size_t index, int value) {
149:    range_check (index);
150:    _data[index] = value;
151: }
152:
153: iterintvec::iterator iterintvec::begin() {
154:    return iterator (&_data[0]);
155: }
156:
157: iterintvec::iterator iterintvec::end() {
158:    return iterator (&_data[_size]);
159: }
160:
```

```
161:
162: ////////////////////////////////////////////////////////////////////
163: // main.cpp
164: ////////////////////////////////////////////////////////////////////
165:
166: int main () {
167:    iterintvec v1(10);
168:    v1.put (3, 99);
169:    int x = v1.get (3);
170:    cout << x << endl;
171:    try {
172:       v1.get (999);
173:    }catch (out_of_range error) {
174:       cerr << error.what() << endl;
175:    }
176:    iterintvec v2 = v1;
177:    v2.put (3, 1234);
178:    cout << v1.get (3) << " " << v2.get (3) << endl;
179:    v2 = v1;
180:    cout << v1.get (3) << " " << v2.get (3) << endl;
181:    for (iterintvec::iterator i = v1.begin(); i != v1.end(); ++i) {
182:       cout << " " << *i;
183:    }
184:    cout << endl;
185:    for (const auto& i: v1) cout << " " << i << endl;
186:    return 0;
187: }
188:
189: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
190: //TEST// grind iterintvec >iterintvec.out 2>&1
191: //TEST// mkpspdf iterintvec.ps iterintvec.cpp* iterintvec.out*
192:
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting iterintvec.cpp
    2: iterintvec.cpp:
    3:       $Id: iterintvec.cpp,v 1.30 2015-04-14 13:25:24-07 - - $
    4: g++ -g -O0 -Wall -Wextra -rdynamic -std=gnu++11 iterintvec.cpp -o iterin
tvec -lglut -lGLU -lGL -lX11 -lrt -lm
    5: rm -f iterintvec.o
    6: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished iterintvec.cpp
```

```
    1: ==9512== Memcheck, a memory error detector
    2: ==9512== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
    3: ==9512== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright in
fo
    4: ==9512== Command: iterintvec
    5: ==9512==
    6: 99
    7: iterintvec::range_check
    8: 99 1234
    9: 99 99
   10:  0 0 0 99 0 0 0 0 0 0
   11:  0
   12:  0
   13:  0
   14:  99
   15:  0
   16:  0
   17:  0
   18:  0
   19:  0
   20:  0
   21: ==9512==
   22: ==9512== HEAP SUMMARY:
   23: ==9512==     in use at exit: 0 bytes in 0 blocks
   24: ==9512==   total heap usage: 6 allocs, 6 frees, 321 bytes allocated
   25: ==9512==
   26: ==9512== All heap blocks were freed -- no leaks are possible
   27: ==9512==
   28: ==9512== For counts of detected and suppressed errors, rerun with: -v
   29: ==9512== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```