```
 1: // $Id: iterintvec.cpp,v 1.21 2014-05-30 16:45:52-07 - - $
 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: };
67:
```

```
 68:
 69: //////////////////////////////////////////////////////////////////
 70: // iterintvec.cpp
 71: //////////////////////////////////////////////////////////////////
 72:
 73: // Private.
 74: void iterintvec::copy_data (int *data) {
 75:    for (size_t index = 0; index < _size; ++index) {
 76:       _data[index] = data[index];
 77:    }
 78: }
 79:
 80: // Private.
 81: void iterintvec::range_check (size_t index) const {
 82:    if (index >= _size) throw out_of_range ("iterintvec::range_check");
 83: }
 84:
 85: // Default ctor.
 86: iterintvec::iterintvec(): _size(0), _data(nullptr){}
 87:
 88: // Copy constructor.
 89: iterintvec::iterintvec (const iterintvec& that):
 90:             _size(that._size), _data (new int[that._size]) {
 91:    copy_data (that._data);
 92: }
 93:
 94: // Move constructor.
 95: iterintvec::iterintvec (iterintvec&& that):
 96:             _size(that._size), _data (that._data) {
 97:    that._size = 0;
 98:    that._data = nullptr;
 99: }
100:
101: // Copy operator=
102: iterintvec& iterintvec::operator= (const iterintvec& that){
103:    if (this != &that) {
104:       if (_data != nullptr) delete[] _data;
105:       _size = that._size;
106:       _data = new int[that._size];
107:       copy_data (that._data);
108:    }
109:    return *this;
110: }
111:
112: // Move operator=
113: iterintvec& iterintvec::operator= (iterintvec&& that){
114:    if (this != &that) {
115:       if (_data != nullptr) delete[] _data;
116:       _size = that._size;
117:       _data = that._data;
118:       that._size = 0;
119:       that._data = nullptr;
120:    }
121:    return *this;
122: }
123:
```

```
124:
125: // Destructor.
126: iterintvec::~iterintvec() {
127:    if (_data != nullptr) delete[] _data;
128: }
129:
130: // Fixed-size allocator.
131: iterintvec::iterintvec (size_t size):
132:                 _size(size), _data (new int[_size]) {
133:    for (size_t index = 0; index < _size; ++index) {
134:       _data[index] = 0;
135:    }
136: }
137:
138: size_t iterintvec::size() const {
139:    return _size;
140: }
141:
142: int iterintvec::get (size_t index) const {
143:    range_check (index);
144:    return _data[index];
145: }
146:
147: void iterintvec::put (size_t index, int value) {
148:    range_check (index);
149:    _data[index] = value;
150: }
151:
152: iterintvec::iterator iterintvec::begin() {
153:    return iterator (&_data[0]);
154: }
155:
156: iterintvec::iterator iterintvec::end() {
157:    return iterator (&_data[_size]);
158: }
159:
```

```
160:
161: //////////////////////////////////////////////////////////////////////
162: // main.cpp
163: //////////////////////////////////////////////////////////////////////
164:
165: int main () {
166:    iterintvec v1(10);
167:    v1.put (3, 99);
168:    int x = v1.get (3);
169:    cout << x << endl;
170:    try {
171:       v1.get (999);
172:    }catch (out_of_range error) {
173:       cerr << error.what() << endl;
174:    }
175:    iterintvec v2 = v1;
176:    v2.put (3, 1234);
177:    cout << v1.get (3) << " " << v2.get (3) << endl;
178:    v2 = v1;
179:    cout << v1.get (3) << " " << v2.get (3) << endl;
180:    for (iterintvec::iterator i = v1.begin();
181:         i != v1.end(); ++i) {
182:       cout << *i << endl;
183:    }
184:    return 0;
185: }
186:
187: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
188: //TEST// grind iterintvec >iterintvec.out 2>&1
189: //TEST// mkpspdf iterintvec.ps iterintvec.cpp* iterintvec.out*
190:
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting iterintvec.cpp
    2: iterintvec.cpp:
    3:       $Id: iterintvec.cpp,v 1.21 2014-05-30 16:45:52-07 - - $
    4: g++ -g -O0 -Wall -Wextra -std=gnu++11 iterintvec.cpp -o iterintvec -lglu
t -lGLU -lGL -lX11 -lm -lrt
    5: rm -f iterintvec.o
    6: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished iterintvec.cpp
```

```
    1: ==21719== Memcheck, a memory error detector
    2: ==21719== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al
.
    3: ==21719== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright i
nfo
    4: ==21719== Command: iterintvec
    5: ==21719==
    6: 99
    7: iterintvec::range_check
    8: 99 1234
    9: 99 99
   10: 0
   11: 0
   12: 0
   13: 99
   14: 0
   15: 0
   16: 0
   17: 0
   18: 0
   19: 0
   20: ==21719==
   21: ==21719== HEAP SUMMARY:
   22: ==21719==     in use at exit: 0 bytes in 0 blocks
   23: ==21719==   total heap usage: 6 allocs, 6 frees, 321 bytes allocated
   24: ==21719==
   25: ==21719== All heap blocks were freed -- no leaks are possible
   26: ==21719==
   27: ==21719== For counts of detected and suppressed errors, rerun with: -v
   28: ==21719== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```