

```
1: // $Id: iterintvec.cpp,v 1.41 2016-04-14 16:09:31-07 - - $
2:
3: //
4: // iterintvec - implementation of an int vector with iterator.
5: //
6:
7: #include <algorithm>
8: #include <iostream>
9: #include <stdexcept>
10:
11: using namespace std;
12:
13: //////////////////////////////////////
14: // iterintvec.h
15: //////////////////////////////////////
16:
17: class iterintvec {
18:     public:
19:         using value_type = int;
20:         using reference = int&;
21:         using const_reference = const int&;
22:         using pointer = int*;
23:         using const_pointer = const int*;
24:         using difference_type = ptrdiff_t;
25:         using size_type = size_t;
26:         class iterator;
27:     private:
28:         size_type size_;
29:         value_type *data_;
30:     public:
31:         iterintvec(); // default ctor
32:         iterintvec (const iterintvec&); // copy ctor
33:         iterintvec (iterintvec&&); // move ctor
34:         iterintvec& operator= (const iterintvec&); // copy operator=
35:         iterintvec& operator= (iterintvec&&); // move operator=
36:         ~iterintvec(); // dtor
37:         explicit iterintvec (size_type size);
38:         size_type size() const;
39:         reference at (size_type index);
40:         const_reference at (size_type index) const;
41:         iterator begin();
42:         iterator end();
43: };
44:
```

```
45:
46: //////////////////////////////////////
47: // iterintvec.h
48: //////////////////////////////////////
49:
50: class iterintvec::iterator {
51:     private:
52:         pointer curr;
53:         friend class iterintvec;
54:         iterator (pointer init): curr(init) {
55:             };
56:     public:
57:         iterator(): curr(nullptr) {};
58:         reference operator*() {
59:             return *curr;
60:         }
61:         const_reference operator*() const {
62:             return *curr;
63:         }
64:         iterator& operator++() {
65:             ++curr;
66:             return *this;
67:         }
68:         iterator operator++ (value_type) {
69:             iterator tmp {*this};
70:             ++curr;
71:             return tmp;
72:         }
73:         bool operator== (const iterator& that) {
74:             return curr == that.curr;
75:         }
76:         bool operator!= (const iterator& that) {
77:             return not (*this == that);
78:         }
79:         operator bool() {
80:             return curr != nullptr;
81:         }
82: };
83:
```

```
84:
85: ///////////////////////////////////////////////////////////////////
86: // iterintvec.cpp
87: ///////////////////////////////////////////////////////////////////
88:
89: // Default ctor.
90: iterintvec::iterintvec(): size_(0), data_(nullptr) {
91: }
92:
93: // Copy constructor.
94: iterintvec::iterintvec (const iterintvec& that):
95:     size_(that.size_), data_ (new value_type[that.size_]) {
96:     std::copy (&that.data_[0], &that.data_[that.size_], &data_[0]);
97: }
98:
99: // Move constructor.
100: iterintvec::iterintvec (iterintvec&& that):
101:     size_(that.size_), data_ (that.data_) {
102:     that.size_ = 0;
103:     that.data_ = nullptr;
104: }
105:
106: // Copy operator=
107: iterintvec& iterintvec::operator= (const iterintvec& that){
108:     if (this != &that) {
109:         if (data_ != nullptr) delete[] data_;
110:         size_ = that.size_;
111:         data_ = new value_type[that.size_];
112:         std::copy (&that.data_[0], &that.data_[that.size_], &data_[0]);
113:     }
114:     return *this;
115: }
116:
117: // Move operator=
118: iterintvec& iterintvec::operator= (iterintvec&& that){
119:     if (this != &that) {
120:         if (data_ != nullptr) delete[] data_;
121:         size_ = that.size_;
122:         data_ = that.data_;
123:         that.size_ = 0;
124:         that.data_ = nullptr;
125:     }
126:     return *this;
127: }
128:
```

```
129:
130: //////////////////////////////////////
131: // iterintvec.cpp
132: //////////////////////////////////////
133:
134: // Destructor.
135: iterintvec::~iterintvec() {
136:     if (data_ != nullptr) delete[] data_;
137: }
138:
139: // Fixed-size allocator.
140: iterintvec::iterintvec (size_type size):
141:     size_(size), data_ (new value_type[size_]) {
142:     std::fill (&data_[0], &data_[size_], 0);
143: }
144:
145: iterintvec::size_type iterintvec::size() const {
146:     return size_;
147: }
148:
149: iterintvec::reference
150: iterintvec::at (iterintvec::size_type index) {
151:     if (index >= size_) throw out_of_range ("iterintvec::at");
152:     return data_[index];
153: }
154:
155: iterintvec::const_reference
156: iterintvec::at (iterintvec::size_type index) const {
157:     if (index >= size_) throw out_of_range ("iterintvec::at");
158:     return data_[index];
159: }
160:
161: iterintvec::iterator iterintvec::begin() {
162:     return iterator (&data_[0]);
163: }
164:
165: iterintvec::iterator iterintvec::end() {
166:     return iterator (&data_[size_]);
167: }
168:
```

```
169:
170: //////////////////////////////////////
171: // main.cpp
172: //////////////////////////////////////
173:
174: int main() {
175:     iterintvec v1(10);
176:     v1.at(3) = 99;
177:     int x = v1.at(3);
178:     cout << x << endl;
179:     try {
180:         v1.at(999);
181:     } catch (out_of_range error) {
182:         cerr << error.what() << endl;
183:     }
184:     iterintvec v2 = v1;
185:     v2.at(3) = 1234;
186:     cout << v1.at(3) << " " << v2.at(3) << endl;
187:     v2 = v1;
188:     cout << v1.at(3) << " " << v2.at(3) << endl;
189:     for (iterintvec::iterator i = v1.begin(); i != v1.end(); ++i) {
190:         cout << " " << *i;
191:     }
192:     cout << endl;
193:     for (const auto& n: v1) {
194:         cout << " " << n;
195:     }
196:     cout << endl;
197:     for_each (v1.begin(), v1.end(), [](int n){cout << " " << n;});
198:     cout << endl;
199:     return 0;
200: }
201:
202: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
203: //TEST// grind iterintvec >iterintvec.out 2>&1
204: //TEST// mkpspdf iterintvec.ps iterintvec.cpp* iterintvec.out*
205:
```

[illegible]

```
1: ==15192== Memcheck, a memory error detector
2: ==15192== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
3: ==15192== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright
info
4: ==15192== Command: iterintvec
5: ==15192==
6: 99
7: iterintvec::at
8: 99 1234
9: 99 99
10: 0 0 0 99 0 0 0 0 0 0
11: 0 0 0 99 0 0 0 0 0 0
12: 0 0 0 99 0 0 0 0 0 0
13: ==15192==
14: ==15192== HEAP SUMMARY:
15: ==15192==      in use at exit: 0 bytes in 0 blocks
16: ==15192==    total heap usage: 6 allocs, 6 frees, 319 bytes allocated
17: ==15192==
18: ==15192== All heap blocks were freed -- no leaks are possible
19: ==15192==
20: ==15192== For counts of detected and suppressed errors, rerun with: -v
21: ==15192== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```