```
 1: // $Id: intvec.cpp,v 1.24 2014-04-14 13:16:36-07 - - $
 2:
 3: //
 4: // intvec - implementation of an int vector, similar to
 5: // Stroustrup, ch.17 example.
 6: //
 7:
 8: #include <iostream>
 9: #include <stdexcept>
10:
11: using namespace std;
12:
13: ////////////////////////////////////////////////////////////////
14: // intvec.h
15: ////////////////////////////////////////////////////////////////
16:
17: class intvec {
18:    private:
19:       size_t _size;
20:       int *_data;
21:       void copy_data (int *data);
22:       void range_check (size_t index) const;
23:    public:
24:       intvec ();                           // default ctor
25:       intvec (const intvec&);              // copy ctor
26:       intvec (intvec&&);                   // move ctor
27:       intvec& operator= (const intvec&); // copy operator=
28:       intvec& operator= (intvec&&);        // move operator=
29:       ~intvec();                           // dtor
30:       // Other members.
31:       explicit intvec (size_t size);
32:       size_t size() const;
33:       int get (size_t index) const;
34:       void put (size_t index, int value);
35: };
36:
```

```
37:
38: //////////////////////////////////////////////////////////////////
39: // intvec.cpp
40: //////////////////////////////////////////////////////////////////
41:
42: // Private.
43: void intvec::copy_data (int *data) {
44:    for (size_t index = 0; index < _size; ++index) {
45:       _data[index] = data[index];
46:    }
47: }
48:
49: // Private.
50: void intvec::range_check (size_t index) const {
51:    if (index >= _size) throw out_of_range ("intvec::range_check");
52: }
53:
54: // Default ctor.
55: intvec::intvec(): _size(0), _data(nullptr) {
56: }
57:
58: // Copy constructor.
59: intvec::intvec (const intvec& that):
60:             _size(that._size), _data (new int[that._size]) {
61:    copy_data (that._data);
62: }
63:
64: // Move constructor.
65: intvec::intvec (intvec&& that):
66:             _size(that._size), _data (that._data) {
67:    that._size = 0;
68:    that._data = nullptr;
69: }
70:
71: // Copy operator=
72: intvec& intvec::operator= (const intvec& that){
73:    if (this != &that) {
74:       if (_data != nullptr) delete[] _data;
75:       _size = that._size;
76:       _data = new int[that._size];
77:       copy_data (that._data);
78:    }
79:    return *this;
80: }
81:
82: // Move operator=
83: intvec& intvec::operator= (intvec&& that){
84:    if (this != &that) {
85:       if (_data != nullptr) delete[] _data;
86:       _size = that._size;
87:       _data = that._data;
88:       that._size = 0;
89:       that._data = nullptr;
90:    }
91:    return *this;
92: }
93:
```

```
 94:
 95: // Destructor.
 96: intvec::~intvec() {
 97:    if (_data != nullptr) delete[] _data;
 98: }
 99:
100: // Fixed-size allocator.
101: intvec::intvec (size_t size): _size(size), _data (new int[_size]) {
102:    for (size_t index = 0; index < _size; ++index) {
103:       _data[index] = 0;
104:    }
105: }
106:
107: size_t intvec::size() const {
108:    return _size;
109: }
110:
111: int intvec::get (size_t index) const {
112:    range_check (index);
113:    return _data[index];
114: }
115:
116: void intvec::put (size_t index, int value) {
117:    range_check (index);
118:    _data[index] = value;
119: }
120:
```

```
121:
122: //////////////////////////////////////////////////////////////////////
123: // main.cpp
124: //////////////////////////////////////////////////////////////////////
125:
126: int main () {
127:    intvec v1(10);
128:    v1.put (3, 99);
129:    int x = v1.get (3);
130:    cout << x << endl;
131:    try {
132:       v1.get (999);
133:    }catch (out_of_range error) {
134:       cerr << error.what() << endl;
135:    }
136:    intvec v2 = v1;
137:    v2.put (3, 1234);
138:    cout << v1.get (3) << " " << v2.get (3) << endl;
139:    v2 = v1;
140:    cout << v1.get (3) << " " << v2.get (3) << endl;
141:    return 0;
142: }
143:
144: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
145: //TEST// grind intvec >intvec.out 2>&1
146: //TEST// mkpspdf intvec.ps intvec.cpp* intvec.out*
147:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting intvec.cpp
2: intvec.cpp: $Id: intvec.cpp,v 1.24 2014-04-14 13:16:36-07 - - $
3: g++ -g -O0 -Wall -Wextra -std=gnu++11 intvec.cpp -o intvec -lm
4: rm -f intvec.o
5: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished intvec.cpp
```

```
    1: ==15134== Memcheck, a memory error detector
    2: ==15134== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al
.
    3: ==15134== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright i
nfo
    4: ==15134== Command: intvec
    5: ==15134==
    6: 99
    7: intvec::range_check
    8: 99 1234
    9: 99 99
   10: ==15134==
   11: ==15134== HEAP SUMMARY:
   12: ==15134==     in use at exit: 0 bytes in 0 blocks
   13: ==15134==   total heap usage: 5 allocs, 5 frees, 308 bytes allocated
   14: ==15134==
   15: ==15134== All heap blocks were freed -- no leaks are possible
   16: ==15134==
   17: ==15134== For counts of detected and suppressed errors, rerun with: -v
   18: ==15134== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```