```
// $Id: iterintvec.cpp,v 1.41 2016-04-14 16:09:31-07 - - $

//
// iterintvec - implementation of an int vector with iterator.
//

#include <algorithm>
#include <iostream>
#include <stdexcept>

using namespace std;

/////////////////////////////////////////////////////////////////
// iterintvec.h
/////////////////////////////////////////////////////////////////

class iterintvec {
   public:
      using value_type = int;
      using reference = int&;
      using const_reference = const int&;
      using pointer = int*;
      using const_pointer = const int*;
      using difference_type = ptrdiff_t;
      using size_type = size_t;
      class iterator;
   private:
      size_type size_;
      value_type *data_;
   public:
      iterintvec();                                  // default ctor
      iterintvec (const iterintvec&);                // copy ctor
      iterintvec (iterintvec&&);                     // move ctor
      iterintvec& operator= (const iterintvec&);     // copy operator=
      iterintvec& operator= (iterintvec&&);          // move operator=
      ~iterintvec();                                 // dtor
      explicit iterintvec (size_type size);
      size_type size() const;
      reference at (size_type index);
      const_reference at (size_type index) const;
      iterator begin();
      iterator end();
};

```

```

////////////////////////////////////////////////////////////////////////
// iterintvec.h
////////////////////////////////////////////////////////////////////////

class iterintvec::iterator {
   private:
      pointer curr;
      friend class iterintvec;
      iterator (pointer init): curr(init) {
      };
   public:
      iterator(): curr(nullptr) {};
      reference operator*() {
         return *curr;
      }
      const_reference operator*() const {
         return *curr;
      }
      iterator& operator++() {
         ++curr;
         return *this;
      }
      iterator operator++ (value_type) {
         iterator tmp {*this};
         ++curr;
         return tmp;
      }
      bool operator== (const iterator& that) {
         return curr == that.curr;
      }
      bool operator!= (const iterator& that) {
         return not (*this == that);
      }
      operator bool() {
         return curr != nullptr;
      }
};

```

```

////////////////////////////////////////////////////////////////////
// iterintvec.cpp
////////////////////////////////////////////////////////////////////

// Default ctor.
iterintvec::iterintvec(): size_(0), data_(nullptr) {
}

// Copy constructor.
iterintvec::iterintvec (const iterintvec& that):
            size_(that.size_), data_ (new value_type[that.size_]) {
   std::copy (&that.data_[0], &that.data_[that.size_], &data_[0]);
}

// Move constructor.
iterintvec::iterintvec (iterintvec&& that):
            size_(that.size_), data_ (that.data_) {
   that.size_ = 0;
   that.data_ = nullptr;
}

// Copy operator=
iterintvec& iterintvec::operator= (const iterintvec& that){
   if (this != &that) {
      if (data_ != nullptr) delete[] data_;
      size_ = that.size_;
      data_ = new value_type[that.size_];
      std::copy (&that.data_[0], &that.data_[that.size_], &data_[0]);
   }
   return *this;
}

// Move operator=
iterintvec& iterintvec::operator= (iterintvec&& that){
   if (this != &that) {
      if (data_ != nullptr) delete[] data_;
      size_ = that.size_;
      data_ = that.data_;
      that.size_ = 0;
      that.data_ = nullptr;
   }
   return *this;
}

```

```

////////////////////////////////////////////////////////////////
// iterintvec.cpp
////////////////////////////////////////////////////////////////

// Destructor.
iterintvec::~iterintvec() {
   if (data_ != nullptr) delete[] data_;
}

// Fixed-size allocator.
iterintvec::iterintvec (size_type size):
               size_(size), data_ (new value_type[size_]) {
   std::fill (&data_[0], &data_[size_], 0);
}

iterintvec::size_type iterintvec::size() const {
   return size_;
}

iterintvec::reference
iterintvec::at (iterintvec::size_type index) {
   if (index >= size_) throw out_of_range ("iterintvec::at");
   return data_[index];
}

iterintvec::const_reference
iterintvec::at (iterintvec::size_type index) const {
   if (index >= size_) throw out_of_range ("iterintvec::at");
   return data_[index];
}

iterintvec::iterator iterintvec::begin() {
   return iterator (&data_[0]);
}

iterintvec::iterator iterintvec::end() {
   return iterator (&data_[size_]);
}

```

```

////////////////////////////////////////////////////////////////////
// main.cpp
////////////////////////////////////////////////////////////////////

int main() {
   iterintvec v1(10);
   v1.at(3) = 99;
   int x = v1.at(3);
   cout << x << endl;
   try {
      v1.at(999);
   }catch (out_of_range error) {
      cerr << error.what() << endl;
   }
   iterintvec v2 = v1;
   v2.at(3) = 1234;
   cout << v1.at(3) << " " << v2.at(3) << endl;
   v2 = v1;
   cout << v1.at(3) << " " << v2.at(3) << endl;
   for (iterintvec::iterator i = v1.begin(); i != v1.end(); ++i) {
      cout << " " << *i;
   }
   cout << endl;
   for (const auto& n: v1) {
      cout << " " << n;
   }
   cout << endl;
   for_each (v1.begin(), v1.end(), [](int n){cout << " " << n;});
   cout << endl;
   return 0;
}

//TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
//TEST// grind iterintvec >iterintvec.out 2>&1
//TEST// mkpspdf iterintvec.ps iterintvec.cpp* iterintvec.out*

```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting iterintvec.cpp
iterintvec.cpp:
     $Id: iterintvec.cpp,v 1.41 2016-04-14 16:09:31-07 - - $
g++ -g -O0 -Wall -Wextra -rdynamic -std=gnu++14 iterintvec.cpp
        -o iterintvec -lglut -lGLU -lGL -lX11 -lrt -lm
rm -f iterintvec.o
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished iterintvec.cpp
```

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