```
// $Id: abox.cpp,v 1.12 2014-04-10 14:12:07-07 - - $

// Class abox showing default ctor, copy ctor, operator=, dtor.

#include <cstdlib>
#include <iostream>
#include <string>
#include <typeinfo>
#include <vector>

using namespace std;

#define LINE "[" << __LINE__ << "] "
#define IDENT(name) \
        cout << LINE << reinterpret_cast<const void*> (this) \
             << "->" << name << ": value=" << value << endl;
#define SHOW(STMT) \
        cout << LINE << "SHOW(" << #STMT << "):" << endl; \
        STMT; \
        cout << endl;

////////////////////////////////////////////////////////////////////
// abox.h
////////////////////////////////////////////////////////////////////

class abox {
   private:
      int value;
   public:
      abox();                            // default ctor
      abox (const abox&);                // copy ctor
      abox& operator= (const abox&);     // copy operator=
      ~abox();                           // dtor
      abox (abox&&);                     // C++11 move ctor
      abox& operator= (abox&&);          // C++11 move operator=
      abox (int);                        // implicit int->abox conv
      operator int() const;              // implicit abox->int conv
      abox& operator++();                // prefix ++
      abox operator++ (int);             // postfix ++
};
```

```

////////////////////////////////////////////////////////////////////
// abox.cpp
////////////////////////////////////////////////////////////////////

abox::abox(): value(0) { // default ctor
   IDENT ("abox::abox()");
}

abox::abox (const abox& init): value(init.value) { // copy ctor
   IDENT ("abox::abox(const abox&)");
}

abox& abox::operator= (const abox& that) { // copy operator=
   if (this != &that) {
      this->value = that.value;
   }
   IDENT ("abox::operator= (const abox&)");
   return *this;
}

abox::~abox() { // dtor
   IDENT ("abox::~abox()");
}

abox::abox (abox&& init): value(init.value) { // C++11 move ctor
   IDENT ("abox::abox(abox&&)");
}

abox& abox::operator= (abox&& that) { // C++11 move operator=
   if (this != &that) {
      this->value = that.value;
   }
   IDENT ("abox::operator= (abox&&)");
   return *this;
}

abox::abox (int init): value(init) { // implicit int->abox conv
   IDENT ("abox::abox(int)");
}

abox::operator int() const { // implicit abox->int conv
   IDENT ("abox::operator int()");
   return value;
}

abox& abox::operator++() { // prefix ++
   ++value;
   return *this;
}

abox abox::operator++ (int) { // postfix ++
   abox result = value;
   value++;
   return result;
}

```

```

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

void ref_fn (const abox& that) {
   SHOW (cout << "ref_fn, that=" << that << endl);
}

void value_fn (const abox that) {
   SHOW (cout << "value_fn, that=" << that << endl);
}

int main() {
   SHOW (abox a);
   SHOW (abox b = a);
   SHOW (abox c (a));
   SHOW (abox d = 6);
   SHOW (b = 3);
   SHOW (ref_fn (6));
   SHOW (ref_fn (a));
   SHOW (value_fn (a));
   SHOW (abox ii = 255);
   SHOW (int i = ii);
   SHOW (cout << i << endl);
   cout << '\f' << endl;
   SHOW (abox *p = new abox (6));
   SHOW (delete p);
   SHOW (abox *bb = new abox[3]);
   SHOW (delete[] bb);
   SHOW (vector<abox> vb);
   SHOW (vb.push_back (6));
   SHOW (vb.push_back (8));
   SHOW (cout << (a = d++) << endl;);
   SHOW (cout << (a = ++d) << endl;);
   return EXIT_SUCCESS;
}

/*
//TEST// valgrind --leak-check=full --show-reachable=yes \
//TEST//       --log-file=abox.lis.grind abox >abox.lis 2>&1
//TEST// mkpspdf abox.ps abox.cpp* abox.lis*
*/

```

```
* @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting abox.cpp
* abox.cpp: $Id: abox.cpp,v 1.12 2014-04-10 14:12:07-07 - - $
* g++ -g -O0 -Wall -Wextra -std=gnu++11 abox.cpp -o abox -lm
* rm -f abox.o
* @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished abox.cpp
```

```
[112] SHOW(abox a):
[47] 0x7fefffb10->abox::abox(): value=0

[113] SHOW(abox b = a):
[51] 0x7fefffb00->abox::abox(const abox&): value=0

[114] SHOW(abox c (a)):
[51] 0x7fefffaf0->abox::abox(const abox&): value=0

[115] SHOW(abox d = 6):
[79] 0x7fefffae0->abox::abox(int): value=6

[116] SHOW(b = 3):
[79] 0x7fefffb20->abox::abox(int): value=3
[74] 0x7fefffb00->abox::operator= (abox&&): value=3
[63] 0x7fefffb20->abox::~abox(): value=3

[117] SHOW(ref_fn (6)):
[79] 0x7fefffb30->abox::abox(int): value=6
[104] SHOW(cout << "ref_fn, that=" << that << endl):
[83] 0x7fefffb30->abox::operator int(): value=6
ref_fn, that=6

[63] 0x7fefffb30->abox::~abox(): value=6

[118] SHOW(ref_fn (a)):
[104] SHOW(cout << "ref_fn, that=" << that << endl):
[83] 0x7fefffb10->abox::operator int(): value=0
ref_fn, that=0


[119] SHOW(value_fn (a)):
[51] 0x7fefffb40->abox::abox(const abox&): value=0
[108] SHOW(cout << "value_fn, that=" << that << endl):
[83] 0x7fefffb40->abox::operator int(): value=0
value_fn, that=0

[63] 0x7fefffb40->abox::~abox(): value=0

[120] SHOW(abox ii = 255):
[79] 0x7fefffad0->abox::abox(int): value=255

[121] SHOW(int i = ii):
[83] 0x7fefffad0->abox::operator int(): value=255

[122] SHOW(cout << i << endl):
255

```

```

[124] SHOW(abox *p = new abox (6)):
[79] 0x4c2b040->abox::abox(int): value=6

[125] SHOW(delete p):
[63] 0x4c2b040->abox::~abox(): value=6

[126] SHOW(abox *bb = new abox[3]):
[47] 0x4c2b098->abox::abox(): value=0
[47] 0x4c2b09c->abox::abox(): value=0
[47] 0x4c2b0a0->abox::abox(): value=0

[127] SHOW(delete[] bb):
[63] 0x4c2b0a0->abox::~abox(): value=0
[63] 0x4c2b09c->abox::~abox(): value=0
[63] 0x4c2b098->abox::~abox(): value=0

[128] SHOW(vector<abox> vb):

[129] SHOW(vb.push_back (6)):
[79] 0x7fefffb50->abox::abox(int): value=6
[67] 0x4c2b0f0->abox::abox(abox&&): value=6
[63] 0x7fefffb50->abox::~abox(): value=6

[130] SHOW(vb.push_back (8)):
[79] 0x7fefffb60->abox::abox(int): value=8
[67] 0x4c2b144->abox::abox(abox&&): value=8
[51] 0x4c2b140->abox::abox(const abox&): value=6
[63] 0x4c2b0f0->abox::~abox(): value=6
[63] 0x7fefffb60->abox::~abox(): value=8

[131] SHOW(cout << (a = d++) << endl;):
[79] 0x7fefffb70->abox::abox(int): value=6
[74] 0x7fefffb10->abox::operator= (abox&&): value=6
[83] 0x7fefffb10->abox::operator int(): value=6
6
[63] 0x7fefffb70->abox::~abox(): value=6

[132] SHOW(cout << (a = ++d) << endl;):
[58] 0x7fefffb10->abox::operator= (const abox&): value=8
[83] 0x7fefffb10->abox::operator int(): value=8
8

[63] 0x4c2b140->abox::~abox(): value=6
[63] 0x4c2b144->abox::~abox(): value=8
[63] 0x7fefffad0->abox::~abox(): value=255
[63] 0x7fefffae0->abox::~abox(): value=8
[63] 0x7fefffaf0->abox::~abox(): value=0
[63] 0x7fefffb00->abox::~abox(): value=3
[63] 0x7fefffb10->abox::~abox(): value=8
```

```
==24188== Memcheck, a memory error detector
==24188== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al
.
==24188== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright i
nfo
==24188== Command: abox
==24188== Parent PID: 24187
==24188==
==24188==
==24188== HEAP SUMMARY:
==24188==     in use at exit: 0 bytes in 0 blocks
==24188==   total heap usage: 4 allocs, 4 frees, 36 bytes allocated
==24188==
==24188== All heap blocks were freed -- no leaks are possible
==24188==
==24188== For counts of detected and suppressed errors, rerun with: -v
==24188== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```