Array Class Documentation

1.0

Generated by Doxygen 1.16.1

Chapter 1

Test List

Class Array

Comprehensive test of Array class features

Member Array::Array (int arraySize=10)

Test with positive, negative, and zero sizes

Member Array::Array (const Array &init)

Test copying arrays of various sizes

Member Array::getArrayCount ()

Verify count increases with construction and decreases with destruction

Member Array::getSize () const

Verify size matches what was specified during construction

Member Array::operator!= (const Array &right) const

Test with equal and unequal arrays

Member Array::operator<< (ostream &output, const Array &a)

Test output formatting with various array sizes

Member Array::operator= (const Array &right)

Test assignment between arrays of different sizes

Member Array::operator== (const Array &right) const

Test with arrays of different sizes and contents

Test with equal and unequal arrays

Member Array::operator>> (istream &input, Array &a)

Test with valid integer input

Member Array::operator[] (int subscript)

Test valid and invalid indices

Member main ()

Comprehensive test of Array class features

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

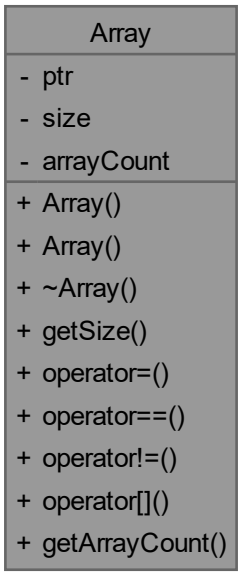
Class Documentation

4.1 Array Class Reference

Dynamic integer array class with bounds checking and I/O operators Array class: like an int array (retains all functionality) but also includes additional features: – allows input and output of the whole array – allows for comparison of 2 arrays, element by element – allows for assignment of 2 arrays – size is part of the class (so no longer needs to be passed) – includes range checking, program terminates for out-of-bound subscripts.

`#include <array.h>`

Collaboration diagram for Array:



Public Member Functions

- Array (int arraySize=10)

  Default constructor for Array class.
- Array (const Array &init)

  Copy constructor for Array class.
- ~Array ()

  Destructor for Array class.
- int getSize () const

  Get the size of the array.
- const Array & operator= (const Array &right)

  Assignment operator for Array class.
- bool operator== (const Array &right) const

  Equality comparison operator.
- bool operator!= (const Array &right) const

  Inequality comparison operator.
- int & operator[ ] (int subscript)

  Subscript operator for array access.

Static Public Member Functions

- static int getArrayCount ()

  Get the number of Array objects instantiated.

Private Attributes

- int * ptr

  pointer to first element of array
- int size

  size of the array

Static Private Attributes

- static int arrayCount = 0

Friends

- istream & operator>> (istream &input, Array &a)

  Overloaded input operator for class Array.
- ostream & operator<< (ostream &output, const Array &a)

  Overloaded output operator for class Array.

4.1.1 Detailed Description

Dynamic integer array class with bounds checking and I/O operators Array class: like an int array (retains all functionality) but also includes additional features: – allows input and output of the whole array – allows for comparison of 2 arrays, element by element – allows for assignment of 2 arrays – size is part of the class (so no longer needs to be passed) – includes range checking, program terminates for out-of-bound subscripts.

Driver program for the Array class This program tests the functionality of the Array class including:

Dynamic integer array class with bounds checking and I/O operators.

Assumptions: – size defaults to a fixed size of 10 if size is not specified – array elements are initialized to zero – user must enter valid integers when using $>>$ – in $<<$, integers are displayed 10 per line

Array class: like an int array (retains all functionality) but also includes additional features: – allows input and output of the whole array – allows for comparison of 2 arrays, element by element – allows for assignment of 2 arrays – size is part of the class (so no longer needs to be passed) – includes range checking, program terminates for out-of-bound subscripts

Implementation and assumptions: – size defaults to a fixed size of 10 if size is not specified – array elements are initialized to zero – user must enter valid integers when using $>>$ – in $<<$, integers are displayed 10 per line

- Construction and destruction
- Input and output operators
- Comparison operators (== and !=)
- Assignment operator
- Subscript operator with bounds checking
- Static member function for counting instances

Note

Ensure to provide valid integer input when prompted

Test Comprehensive test of Array class features

Definition at line 32 of file array.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Array() [1/2]

```
Array::Array (
            int arraySize = 10)
```

Default constructor for Array class.

Default constructor.

Parameters

in	*arraySize*	Initial size of the array (defaults to 10)

Precondition

None

Postcondition

ptr points to array of size arraySize, all elements zero, arrayCount is incremented

Returns

Newly constructed Array object

Note

Negative input values result in the default size of 10

Test Test with positive, negative, and zero sizes

Parameters

in	*arraySize*	Initial size of the array

Precondition

None

Postcondition

ptr points to an array of size arraySize and all elements of the array have been initialized to zero. arrayCount is incremented. Negative input values result in the default size of 10

Returns

Newly constructed Array object

Note

Uses assert to ensure memory allocation succeeds

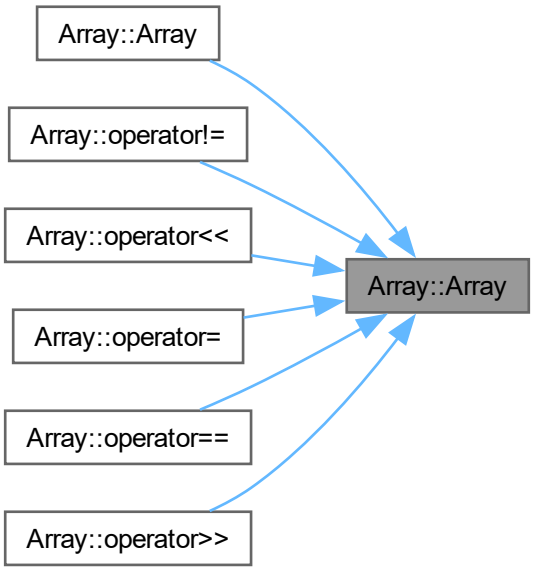
Definition at line 44 of file array.cpp.

```
00045 {
00046     ++arrayCount;
00047     size = (arraySize > 0 ? arraySize : 10);
00048     ptr = new int[size];
00049     assert(ptr != NULL);
00050
00051     for (int i = 0; i < size; i++)
00052         ptr[i] = 0;
00053 }
```

References arrayCount, ptr, and size.

Referenced by Array(), operator!=(), operator<<, operator=(), operator==(), and operator>>.

Here is the caller graph for this function:



4.1.2.2 Array() [2/2]

```
Array::Array (
            const Array & init)
```

Copy constructor for Array class.

Copy constructor.

Parameters

in	*init*	Array to copy from

Precondition

init.ptr points to an array of size at least init.size

Postcondition

init is copied into ∗this, arrayCount is incremented

Returns

Newly constructed Array object as a copy of init

Note

Creates deep copy of the array

Test Test copying arrays of various sizes

Parameters

in	*init*	Array to copy from

Precondition

init.ptr points to an array of size at least init.size

Postcondition

init is copied into ∗this, arrayCount is incremented

Returns

Newly constructed Array object as a copy of init

Note

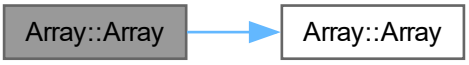
Creates deep copy of the array

Definition at line 65 of file array.cpp.

```
00066 {
00067     ++arrayCount;
00068     size = init.size;
00069     ptr = new int[size];
00070     assert(ptr != NULL);
00071
00072     for (int i = 0; i < size; i++)
00073         ptr[i] = init.ptr[i];
00074 }
```

References Array(), arrayCount, ptr, and size.

Here is the call graph for this function:



4.1.2.3 ~Array()

```
Array::~Array ()
```

Destructor for Array class.

Destructor.

Precondition

ptr points to memory on the heap

Postcondition

Array for ptr is deallocated, arrayCount is decremented

Returns

None

Note

Automatically called when Array object goes out of scope

Precondition

ptr points to memory on the heap

Postcondition

Array for ptr is deallocated, arrayCount is decremented

Returns

None

Note

Automatically called when object goes out of scope

Definition at line 84 of file array.cpp.

```
00085 {
00086     --arrayCount;
00087     delete[] ptr;
00088 }
```

References arrayCount, and ptr.

4.1.3 Member Function Documentation

4.1.3.1 getArrayCount()

```
int Array::getArrayCount ()  [static]
```

Get the number of Array objects instantiated.

Precondition

None

Postcondition

Returns the number of arrays

Returns

Count of Array objects currently instantiated

Note

Static member function

Test Verify count increases with construction and decreases with destruction

Precondition

None

Postcondition

Returns the number of arrays

Returns

Number of Array objects currently instantiated

Note

Static member function

Definition at line 187 of file array.cpp.

```
00187 { return arrayCount; }
```

References arrayCount.

Referenced by main().

Here is the caller graph for this function:

4.1.3.2 getSize()

```
int Array::getSize () const
```

Get the size of the array.

Precondition

None

Postcondition

Returns the size of the array

Returns

Current size of the array

Test Verify size matches what was specified during construction

Precondition

None

Postcondition

Returns the size of the array

Returns

Size of the array

Definition at line 97 of file array.cpp.

```
00097 { return size; }
```

References size.

Referenced by main().

Here is the caller graph for this function:

4.1.3.3 operator"!=()

```
bool Array::operator!= (
            const Array & right) const
```

Inequality comparison operator.

Inequality operator.

Parameters

in	*right*	Array to compare with

Precondition

ptr and right.ptr point to arrays with size at least size and right.size, respectively

Postcondition

false is returned if the arrays have the same size and elements, true otherwise

Returns

true if arrays are not equal, false otherwise

Test Test with equal and unequal arrays

Remarks

Uses operator== for implementation

Parameters

in	*right*	Array to compare with

Precondition

ptr and right.ptr point to arrays with size at least size and right.size, respectively

Postcondition

false is returned if the arrays have the same size and elements true is return otherwise

Returns

true if arrays are not equal, false otherwise

Remarks

Uses operator== for implementation

Definition at line 158 of file array.cpp.

```
00159 {
00160     return !(*this == right);
00161 }
```

References Array().

Here is the call graph for this function:

4.1.3.4 operator=()

```
const Array & Array::operator= (
            const Array & right)
```

Assignment operator for Array class.

Assignment operator.

Parameters

in	*right*	Array to assign from

Precondition

right.ptr points to an array of size at least right.size

Postcondition

∗this is assigned the same array as right

Returns

Reference to the assigned array

Note

Self-assignment is checked and handled properly

Test Test assignment between arrays of different sizes

Parameters

in	*right*	Array to assign from

Precondition

right.ptr points to an array of size at least right.size

Postcondition

∗this is assigned the same array as right

Returns

Reference to ∗this

Note

Handles self-assignment and creates deep copy

Definition at line 109 of file array.cpp.

```
00110 {
00111     if (&right != this)
00112     {
00113         delete[] ptr;
00114         size = right.size;
00115         ptr = new int[size];
00116         assert(ptr != NULL);
00117 
00118         for (int i = 0; i < size; i++)
00119             ptr[i] = right.ptr[i];
00120     }
00121 
00122     return *this;
00123 }
```

References Array(), ptr, and size.

Here is the call graph for this function:

4.1.3.5 operator==()

```
bool Array::operator== (
            const Array & right) const
```

Equality comparison operator.

Equality operator.

Parameters

in	*right*	Array to compare with

Precondition

ptr and right.ptr point to arrays with size at least size and right.size, respectively

Postcondition

true is returned if the arrays have the same size and elements, false otherwise

Returns

true if arrays are equal, false otherwise

Test Test with equal and unequal arrays

Parameters

in	*right*	Array to compare with

Precondition

ptr and right.ptr point to arrays with size at least size and right.size, respectively

Postcondition

true is returned if the arrays have the same size and elements false is return otherwise

Returns

true if arrays are equal, false otherwise

Test Test with arrays of different sizes and contents

Definition at line 136 of file array.cpp.

```
00137 {
00138     if (size != right.size)
00139         return false;
00140
00141     for (int i = 0; i < size; i++)
00142         if (ptr[i] != right.ptr[i])
00143             return false;
00144     return true;
00145 }
```

References Array(), ptr, and size.

Here is the call graph for this function:

4.1.3.6 operator[]()

```
int & Array::operator[] (
            int subscript)
```

Subscript operator for array access.

Subscript operator.

Parameters

in	*subscript*	Index to access

Precondition

0 <= subscript < size

Postcondition

Returns the array value at position "subscript"

Returns

Reference to array element at given index

Note

Terminates program if subscript is out of range

Test Test valid and invalid indices

Parameters

in	*subscript*	Index to access

Precondition

0 <= subscript < size

Postcondition

Returns the array value at position "subscript"

Returns

Reference to array element at given index

Note

Terminates program if subscript out of range

Definition at line 173 of file array.cpp.

```
00174 {
00175     assert(0 <= subscript && subscript < size);
00176     return ptr[subscript];
00177 }
```

References ptr, and size.

4.1.4 Friends And Related Symbol Documentation

4.1.4.1 operator<<

```
ostream & operator<< (
            ostream & output,
            const Array & a)  [friend]
```

Overloaded output operator for class Array.

Parameters

in,out	*output*	Output stream
in	*a*	Array to output

Returns

Reference to the output stream

Precondition

a.ptr must point to an array with size at least a.size

Postcondition

The first a.size elements of a.ptr are sent to output stream

Note

Outputs 10 integers per line with trailing newline

Test Test output formatting with various array sizes

Parameters

in,out	*output*	Output stream
in	*a*	Array to output

Precondition

a.ptr must point to an array with size at least a.size

Postcondition

The first a.size elements of a.ptr are sent to the output istream 10 per line with a trailing endl

Returns

Reference to the output stream

Note

Outputs 10 integers per line

Definition at line 220 of file array.cpp.

```
00221 {
00222     int i;
00223     for (i = 0; i < a.size; i++)
00224     {
00225         output « a.ptr[i] « ' ';
00226         if ((i + 1) % 10 == 0)
00227             output « endl;
00228     }
00229
00230     if (i % 10 != 0)
00231         output « endl;
00232     return output;
00233 }
```

References Array(), ptr, and size.

4.1.4.2 operator>>

```
istream & operator>> (
            istream & input,
            Array & a)  [friend]
```

Overloaded input operator for class Array.

Parameters

in,out	*input*	Input stream
in,out	*a*	Array to read into

Returns

Reference to the input stream

Precondition

a.ptr must point to an array with size at least a.size

Postcondition

The first a.size elements of a.ptr are filled with integers

Note

Reads exactly a.size integers from the stream

Test Test with valid integer input

Parameters

in,out	*input*	Input stream
in,out	*a*	Array to read into

Precondition

a.ptr must point to an array with size at least a.size

Postcondition

The first a.size elements of a.ptr are filled with integers read from the input istream

Returns

Reference to the input stream

Note

Reads exactly a.size integers from the stream

Definition at line 201 of file array.cpp.

```
00202 {
00203     for (int i = 0; i < a.size; i++)
00204         input » a.ptr[i];
00205     return input;
00206 }
```

References Array(), ptr, and size.

4.1.5 Member Data Documentation

4.1.5.1 arrayCount

```
int Array::arrayCount = 0  [static], [private]
```

4.1.6 of Arrays instantiated (static counter)

Definition at line 162 of file array.h.

Referenced by Array(), Array(), getArrayCount(), and ~Array().

4.1.6.1 ptr

```
int* Array::ptr  [private]
```

pointer to first element of array

Definition at line 160 of file array.h.

Referenced by Array(), Array(), operator<<, operator=(), operator==(), operator>>, operator[ ](), and ~Array().

4.1.6.2 size

```
int Array::size  [private]
```

size of the array

Definition at line 161 of file array.h.

Referenced by Array(), Array(), getSize(), operator<<, operator=(), operator==(), operator>>, and operator[ ]().

The documentation for this class was generated from the following files:

- array.h
- array.cpp

Chapter 5

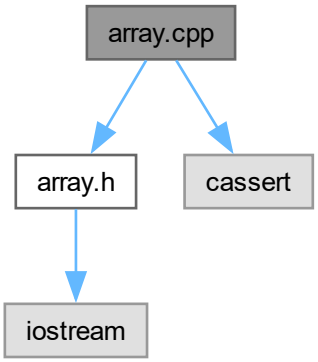
File Documentation

5.1 array.cpp File Reference

Implementation file for the Array class with bounds checking.

```
#include "array.h"
#include <cassert>
```

Include dependency graph for array.cpp:



Functions

- istream & operator>> (istream &input, Array &a)

  Overloaded input operator for class Array.
- ostream & operator<< (ostream &output, const Array &a)

  Overloaded output operator for class Array.

5.1.1 Detailed Description

Implementation file for the Array class with bounds checking.

Date

01/28/2026

Author

Chidera Izuora

Version

1.0

Definition in file array.cpp.

5.1.2 Function Documentation

5.1.2.1 operator<<()

```
ostream & operator<< (
            ostream & output,
            const Array & a)
```

Overloaded output operator for class Array.

Parameters

in,out	*output*	Output stream
in	*a*	Array to output

Precondition

a.ptr must point to an array with size at least a.size

Postcondition

The first a.size elements of a.ptr are sent to the output istream 10 per line with a trailing endl

Returns

Reference to the output stream

Note

Outputs 10 integers per line

Definition at line 220 of file array.cpp.

```
00221 {
00222     int i;
00223     for (i = 0; i < a.size; i++)
00224     {
00225         output « a.ptr[i] « ' ';
00226         if ((i + 1) % 10 == 0)
00227             output « endl;
00228     }
00229
00230     if (i % 10 != 0)
00231         output « endl;
00232     return output;
00233 }
```

5.1.2.2 operator>>()

```
istream & operator>> (
            istream & input,
            Array & a)
```

Overloaded input operator for class Array.

Parameters

in,out	*input*	Input stream
in,out	*a*	Array to read into

Precondition

a.ptr must point to an array with size at least a.size

Postcondition

The first a.size elements of a.ptr are filled with integers read from the input istream

Returns

Reference to the input stream

Note

Reads exactly a.size integers from the stream

Definition at line 201 of file array.cpp.

```
00202 {
00203     for (int i = 0; i < a.size; i++)
00204         input » a.ptr[i];
00205     return input;
00206 }
```

5.2 array.cpp

Go to the documentation of this file.

```
00001
00024
00025 #include "array.h"
00026 #include <cassert>
00027
00028 // Initialize static data member at file scope
00029 int Array::arrayCount = 0;
00030
00031 //---------------------------------------------------------------------------
00044 Array::Array(int arraySize)
00045 {
00046     ++arrayCount;
00047     size = (arraySize > 0 ? arraySize : 10);
00048     ptr = new int[size];
00049     assert(ptr != NULL);
00050
00051     for (int i = 0; i < size; i++)
00052         ptr[i] = 0;
```

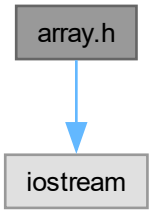
```
00053 }
00054
00055 //-----------------------------------------------------------------------
00065 Array::Array(const Array &init)
00066 {
00067     ++arrayCount;
00068     size = init.size;
00069     ptr = new int[size];
00070     assert(ptr != NULL);
00071
00072     for (int i = 0; i < size; i++)
00073         ptr[i] = init.ptr[i];
00074 }
00075
00076 //-----------------------------------------------------------------------
00084 Array::~Array()
00085 {
00086     --arrayCount;
00087     delete[] ptr;
00088 }
00089
00090 //---------------------------- getSize ----------------------------------
00097 int Array::getSize() const { return size; }
00098
00099 //-----------------------------------------------------------------------
00109 const Array &Array::operator=(const Array &right)
00110 {
00111     if (&right != this)
00112     {
00113         delete[] ptr;
00114         size = right.size;
00115         ptr = new int[size];
00116         assert(ptr != NULL);
00117
00118         for (int i = 0; i < size; i++)
00119             ptr[i] = right.ptr[i];
00120     }
00121
00122     return *this;
00123 }
00124
00125 //-----------------------------------------------------------------------
00136 bool Array::operator==(const Array &right) const
00137 {
00138     if (size != right.size)
00139         return false;
00140
00141     for (int i = 0; i < size; i++)
00142         if (ptr[i] != right.ptr[i])
00143             return false;
00144     return true;
00145 }
00146
00147 //-----------------------------------------------------------------------
00158 bool Array::operator!=(const Array &right) const
00159 {
00160     return !(*this == right);
00161 }
00162
00163 //-----------------------------------------------------------------------
00173 int &Array::operator[](int subscript)
00174 {
00175     assert(0 <= subscript && subscript < size);
00176     return ptr[subscript];
00177 }
00178
00179 //-----------------------------------------------------------------------
00187 int Array::getArrayCount() { return arrayCount; }
00188
00189 //-----------------------------------------------------------------------
00201 istream &operator»(istream &input, Array &a)
00202 {
00203     for (int i = 0; i < a.size; i++)
00204         input » a.ptr[i];
00205     return input;
00206 }
00207
00208 //-----------------------------------------------------------------------
00220 ostream &operator«(ostream &output, const Array &a)
00221 {
00222     int i;
00223     for (i = 0; i < a.size; i++)
00224     {
00225         output « a.ptr[i] « ' ';
00226         if ((i + 1) % 10 == 0)
00227             output « endl;
00228     }
```

```
00229
00230     if (i % 10 != 0)
00231         output « endl;
00232     return output;
00233 }
```

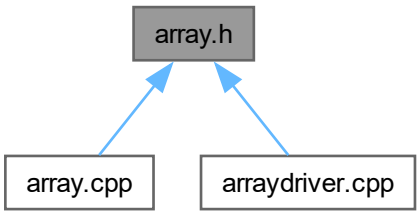
5.3 array.h File Reference

Header file for the Array class with bounds checking.

`#include <iostream>`
Include dependency graph for array.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Array

  Dynamic integer array class with bounds checking and I/O operators Array class: like an int array (retains all functionality) but also includes additional features: – allows input and output of the whole array – allows for comparison of 2 arrays, element by element – allows for assignment of 2 arrays – size is part of the class (so no longer needs to be passed) – includes range checking, program terminates for out-of-bound subscripts.

5.3.1 Detailed Description

Header file for the Array class with bounds checking.

Date

01/28/2026

Author

Chidera Izuora

Version

1.0

Definition in file array.h.

5.4 array.h

Go to the documentation of this file.

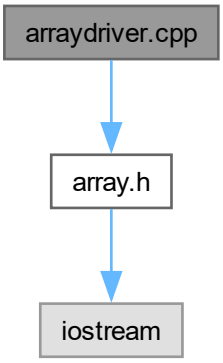
```
00001
00025
00026 #ifndef ARRAY_H
00027 #define ARRAY_H
00028
00029 #include <iostream>
00030 using namespace std;
00031
00032 class Array {
00043     friend istream& operator»(istream &input, Array &a);
00044
00055     friend ostream& operator«(ostream &output, const Array &a);
00056
00057 public:
00069     Array(int arraySize = 10);
00070
00080     Array(const Array &init);
00081
00089     ~Array();
00090
00098     int getSize() const;
00099
00110     const Array& operator=(const Array &right);
00111
00122     bool operator==(const Array &right) const;
00123
00135     bool operator!=(const Array &right) const;
00136
00147     int& operator[](int subscript);
00148
00157     static int getArrayCount();
00158
00159 private:
00160     int* ptr;
00161     int size;
00162     static int arrayCount;
00163 };
00164
00165 #endif
```

5.5 arraydriver.cpp File Reference

Driver program to test the Array class.

```
#include "array.h"
```

Include dependency graph for arraydriver.cpp:



Functions

- int main ()

  Main function to test Array class.

5.5.1 Detailed Description

Driver program to test the Array class.

Date

01/28/2026

Author

Chidera Izuora

Version

1.0

Definition in file arraydriver.cpp.

5.5.2 Function Documentation

5.5.2.1 main()

```
int main ()
```

Main function to test Array class.

Precondition

None

Postcondition

Tests all Array class functionality

Returns

0 on successful execution

Test Comprehensive test of Array class features

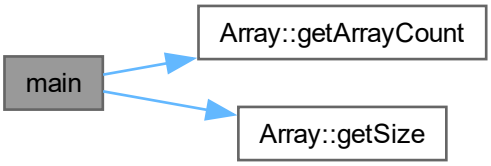
Definition at line 31 of file arraydriver.cpp.

```
00031           {
00032    // no objects yet
00033    cout « "# of arrays instantiated = "
00034         « Array::getArrayCount() « endl;
00035
00036    // create two arrays and print Array count
00037    Array integers1(7), integers2;
00038    cout « "# of arrays instantiated = "
00039         « Array::getArrayCount() « endl « endl;
00040
00041    // print integers1 size and contents
00042    cout « "Size of array integers1 is " « integers1.getSize() « endl
00043         « "Array after initialization:" « endl « integers1 « endl;
00044
00045    // print integers2 size and contents
00046    cout « "Size of array integers2 is " « integers2.getSize() « endl
00047         « "Array after initialization:" « endl « integers2 « endl;
00048
00049    // input and print integers1 and integers2
00050    cout « "Input 17 integers:" « endl;
00051    cin » integers1 » integers2;
00052    cout « "After input, the arrays contain:" « endl
00053         « "integers1: " « integers1
00054         « "integers2: " « integers2 « endl;
00055
00056    // use overloaded inequality (!=) operator
00057    cout « "Evaluating: integers1 != integers2" « endl;
00058    if (integers1 != integers2)
00059       cout « "They are not equal" « endl;
00060
00061    // create array integers3 using integers1 as an
00062    // initializer; print size and contents
00063    Array integers3(integers1);
00064
00065    cout « endl « "Size of array integers3 is " « integers3.getSize() « endl
00066         « "Array after initialization:" « endl « integers3 « endl;
00067
00068    // use overloaded assignment (=) operator
00069    cout « "Assigning integers2 to integers1:" « endl;
00070    integers1 = integers2;
00071    cout « "integers1: " « integers1
00072         « "integers2: " « integers2 « endl;
00073
00074    // use overloaded equality (==) operator
00075    cout « "Evaluating: integers1 == integers2" « endl;
00076    if (integers1 == integers2)
00077       cout « "They are equal" « endl « endl;
```

```
00078
00079    // use overloaded subscript operator to create rvalue
00080    cout « "integers1[5] is " « integers1[5] « endl;
00081
00082    // use overloaded subscript operator to create lvalue
00083    cout « "Assigning 1000 to integers1[5]" « endl;
00084    integers1[5] = 1000;
00085    cout « "integers1: " « integers1 « endl;
00086
00087    // attempt to use out of range subscript
00088    cout « endl « "Attempt to assign 1000 to integers1[15]" « endl;
00089    integers1[15] = 1000;                          // ERROR: out of range
00090
00091    return 0;
00092 }
```

References Array::getArrayCount(), and Array::getSize().

Here is the call graph for this function:



5.6 arraydriver.cpp

Go to the documentation of this file.

```
00001
00021
00022 #include "array.h"
00023
00031 int main() {
00032    // no objects yet
00033    cout « "# of arrays instantiated = "
00034         « Array::getArrayCount() « endl;
00035
00036    // create two arrays and print Array count
00037    Array integers1(7), integers2;
00038    cout « "# of arrays instantiated = "
00039         « Array::getArrayCount() « endl « endl;
00040
00041    // print integers1 size and contents
00042    cout « "Size of array integers1 is " « integers1.getSize() « endl
00043         « "Array after initialization:" « endl « integers1 « endl;
00044
00045    // print integers2 size and contents
00046    cout « "Size of array integers2 is " « integers2.getSize() « endl
00047         « "Array after initialization:" « endl « integers2 « endl;
00048
00049    // input and print integers1 and integers2
00050    cout « "Input 17 integers:" « endl;
00051    cin » integers1 » integers2;
00052    cout « "After input, the arrays contain:" « endl
00053         « "integers1: " « integers1
00054         « "integers2: " « integers2 « endl;
00055
00056    // use overloaded inequality (!=) operator
00057    cout « "Evaluating: integers1 != integers2" « endl;
00058    if (integers1 != integers2)
00059       cout « "They are not equal" « endl;
00060
00061    // create array integers3 using integers1 as an
```

```
00062    // initializer; print size and contents
00063    Array integers3(integers1);
00064
00065    cout « endl « "Size of array integers3 is " « integers3.getSize() « endl
00066         « "Array after initialization:" « endl « integers3 « endl;
00067
00068    // use overloaded assignment (=) operator
00069    cout « "Assigning integers2 to integers1:" « endl;
00070    integers1 = integers2;
00071    cout « "integers1: " « integers1
00072         « "integers2: " « integers2 « endl;
00073
00074    // use overloaded equality (==) operator
00075    cout « "Evaluating: integers1 == integers2" « endl;
00076    if (integers1 == integers2)
00077       cout « "They are equal" « endl « endl;
00078
00079    // use overloaded subscript operator to create rvalue
00080    cout « "integers1[5] is " « integers1[5] « endl;
00081
00082    // use overloaded subscript operator to create lvalue
00083    cout « "Assigning 1000 to integers1[5]" « endl;
00084    integers1[5] = 1000;
00085    cout « "integers1: " « integers1 « endl;
00086
00087    // attempt to use out of range subscript
00088    cout « endl « "Attempt to assign 1000 to integers1[15]" « endl;
00089    integers1[15] = 1000;                          // ERROR: out of range
00090
00091    return 0;
00092 }
```