Student Generated Example

Doxygen Project

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

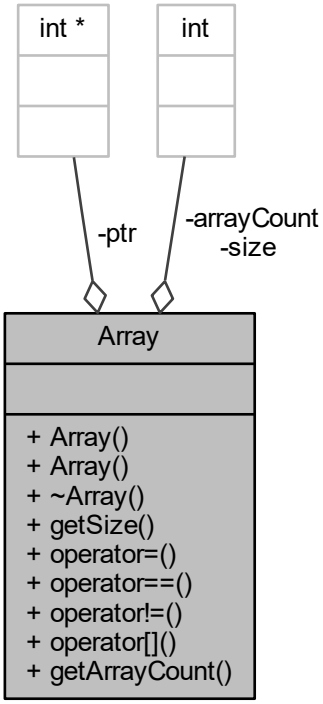
Chapter 3

Class Documentation

3.1 Array Class Reference

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

Collaboration diagram for Array:



Public Member Functions

- Array (int=10)
  Default constructor.
- Array (const Array &)
  Copy constructor.
- ~Array ()
  Destructor.
- int getSize () const
  getSize Get the size of the array
- const Array & operator= (const Array &)
  operator=
- bool operator== (const Array &) const
  operator== Determine if two arrays are equal
- bool operator!= (const Array &) const
  operator!= Determine if two arrays are not equal
- int & operator[ ] (int)
  operator[] Overloaded subscript operator, terminates if subscript out of range error

Static Public Member Functions

- static int getArrayCount ()
  getArrayCount Return the number of Array objects instantiated

Private Attributes

- int * ptr
- int size

Static Private Attributes

- static int arrayCount = 0

Friends

- istream & operator>> (istream &, Array &)
  operator>> Overloaded input operator for class Array; inputs values for entire array.
- ostream & operator<< (ostream &, const Array &)
  operator<< Overloaded output operator for class Array

3.1.1 Detailed Description

Definition at line 29 of file array.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Array() [1/2]

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

Default constructor.

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

Definition at line 35 of file array.cpp.

3.1.2.2 Array() [2/2]

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

Copy constructor.

Precondition

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

Postcondition

init is copied into ∗this, arrayCount is incremented

Definition at line 50 of file array.cpp.

3.1.2.3 ∼Array()

```
~Array ( )
```

Destructor.

Precondition

ptr points to memory on the heap

Postcondition

Array for ptr is deallocated, arrayCount is decremented

Definition at line 64 of file array.cpp.

3.1.3 Member Function Documentation

3.1.3.1 getArrayCount()

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

getArrayCount Return the number of Array objects instantiated

Precondition

None

Postcondition

Returns the number of arrays

Definition at line 143 of file array.cpp.

3.1.3.2 getSize()

```
int getSize ( ) const
```

getSize Get the size of the array

Precondition

None

Postcondition

Returns the size of the array

Definition at line 75 of file array.cpp.

3.1.3.3 operator"!=()

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

operator!= Determine if two arrays are not equal

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

Definition at line 122 of file array.cpp.

3.1.3.4 operator=()

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

operator=

Precondition

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

Postcondition

∗this is assigned the same array as right

Definition at line 82 of file array.cpp.

3.1.3.5 operator==()

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

operator== Determine if two arrays are equal

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

Definition at line 104 of file array.cpp.

3.1.3.6 operator[]()

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

operator[] Overloaded subscript operator, terminates if subscript out of range error

Precondition

0 <= subscript < size

Postcondition

Returns the array value at position "subscript"

Definition at line 132 of file array.cpp.

3.1.4 Friends And Related Function Documentation

3.1.4.1 operator<<

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

operator<< Overloaded output operator for class Array

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

Definition at line 166 of file array.cpp.

3.1.4.2 operator>>

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

operator>> Overloaded input operator for class Array; inputs values for entire array.

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

Definition at line 153 of file array.cpp.

3.1.5 Member Data Documentation

3.1.5.1 arrayCount

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

Definition at line 150 of file array.h.

3.1.5.2 ptr

```
int* ptr  [private]
```

Definition at line 148 of file array.h.

3.1.5.3 size

```
int size  [private]
```

Definition at line 149 of file array.h.

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

- array.h
- array.cpp

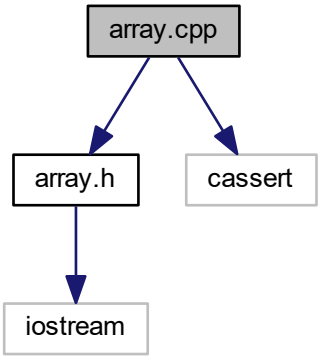
Chapter 4

File Documentation

4.1 array.cpp File Reference

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

Include dependency graph for array.cpp:



Functions

- istream & operator>> (istream &input, Array &a)
- ostream & operator<< (ostream &output, const Array &a)

4.1.1 Function Documentation

4.1.1.1 operator<<()

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

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

Definition at line 166 of file array.cpp.

4.1.1.2 operator>>()

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

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

Definition at line 153 of file array.cpp.

4.2 array.cpp

```
00001 //--------------------------------------------------------------------------
00002 // ARRAY.CPP
00003 // Member function definitions for class Array
00004 // Author:  Deitel/Deitel  (Additional comments by Olson and Zander)
00005 //--------------------------------------------------------------------------
00006 // Array class:  like an int array (retains all functionality) but also
00007 //   includes additional features:
00008 //   -- allows input and output of the whole array
00009 //   -- allows for comparison of 2 arrays, element by element
00010 //   -- allows for assignment of 2 arrays
00011 //   -- size is part of the class (so no longer needs to be passed)
00012 //   -- includes range checking, program terminates for out-of-bound subscripts
00013 //
00014 // Assumptions:
00015 //   -- size defaults to a fixed size of 10 if size is not specified
00016 //   -- array elements are initialized to zero
00017 //   -- user must enter valid integers when using >>
00018 //   -- in <<, integers are displayed 10 per line
00019 //--------------------------------------------------------------------------
00020
00021 #include "array.h"
00022 #include <cassert>
```

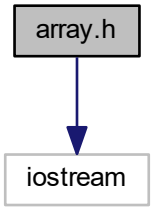
```
00023
00024 // Initialize static data member at file scope
00025 int Array::arrayCount = 0;
00026
00027
00028 //--------------------------------------------------------------------------
00029 // Default constructor
00030 // Preconditions:   None
00031 // Postconditions:  ptr points to an array of size arraySize and all
00032 //        elements of the array have been initialized to zero.
00033 //        arrayCount is incremented.
00034 //        Negative input values result in the default size of 10
00035 Array::Array(int arraySize) {
00036     ++arrayCount;
00037     size = (arraySize > 0 ? arraySize : 10);
00038     ptr = new int[size];
00039     assert(ptr != NULL);
00040
00041     for (int i = 0; i < size; i++)
00042         ptr[i] = 0;
00043 }
00044
00045
00046 //--------------------------------------------------------------------------
00047 // Copy constructor
00048 // Preconditions:   init.ptr points to an array of size at least init.size
00049 // Postconditions:  init is copied into *this, arrayCount is incremented
00050 Array::Array(const Array &init) {
00051     ++arrayCount;
00052     size = init.size;
00053     ptr = new int[size];
00054     assert(ptr != NULL);
00055
00056     for (int i = 0; i < size; i++)
00057         ptr[i] = init.ptr[i];
00058 }
00059
00060 //--------------------------------------------------------------------------
00061 // Destructor
00062 // Preconditions:   ptr points to memory on the heap
00063 // Postconditions:  Array for ptr is deallocated, arrayCount is decremented
00064 Array::~Array() {
00065     --arrayCount;
00066     delete[] ptr;
00067 }
00068
00069
00070 //----------------------------- getSize ------------------------------------
00071 // getSize
00072 // Get the size of the array
00073 // Preconditions:   None
00074 // Postconditions:  Returns the size of the array
00075 int Array::getSize() const { return size; }
00076
00077
00078 //--------------------------------------------------------------------------
00079 // operator=
00080 // Preconditions:   right.ptr points to an array of size at least right.size
00081 // Postconditions:  *this is assigned the same array as right
00082 const Array& Array::operator=(const Array& right) {
00083     if (&right != this) {
00084         delete[] ptr;
00085         size = right.size;
00086         ptr = new int[size];
00087         assert(ptr != NULL);
00088
00089         for (int i = 0; i < size; i++)
00090             ptr[i] = right.ptr[i];
00091     }
00092
00093     return *this;
00094 }
00095
00096
00097 //--------------------------------------------------------------------------
00098 // operator==
00099 // Determine if two arrays are equal.
00100 // Preconditions:  ptr and right.ptr point to arrays with size at least
00101 //          size and right.size, respectively
00102 // Postconditions:  true is returned if the arrays have the same size and
00103 //          elements false is return otherwise
00104 bool Array::operator==(const Array& right) const {
00105     if (size != right.size)
00106         return false;
00107
00108     for (int i = 0; i < size; i++)
00109         if (ptr[i] != right.ptr[i])
```

```
00110             return false;
00111     return true;
00112 }
00113
00114
00115 //--------------------------------------------------------------------------
00116 // operator!=
00117 // Determine if two arrays are not equal.
00118 // Preconditions:   ptr and right.ptr point to arrays with size at least
00119 //          size and right.size, respectively
00120 // Postconditions:  false is returned if the arrays have the same size and
00121 //          elements true is return otherwise
00122 bool Array::operator!=(const Array& right) const {
00123     return !(*this == right);
00124 }
00125
00126
00127 //--------------------------------------------------------------------------
00128 // operator[]
00129 // Overloaded subscript operator, terminates if subscript out of range error
00130 // Preconditions:   0 <= subscript < size
00131 // Postconditions:  Returns the array value at position "subscript"
00132 int& Array::operator[](int subscript) {
00133     assert(0 <= subscript && subscript < size);
00134     return ptr[subscript];
00135 }
00136
00137
00138 //--------------------------------------------------------------------------
00139 // getArrayCount
00140 // Return the number of Array objects instantiated
00141 // Preconditions:   None
00142 // Postconditions:  Returns the number of arrays
00143 int Array::getArrayCount() { return arrayCount; }
00144
00145
00146
00147 //--------------------------------------------------------------------------
00148 // operator>>
00149 // Overloaded input operator for class Array; inputs values for entire array.
00150 // Preconditions:   a.ptr must point to an array with size at least a.size
00151 // Postconditions:  The first a.size elements of a.ptr are filled with
00152 //           integers read from the input istream
00153 istream& operator>>(istream &input, Array &a) {
00154     for (int i = 0; i < a.size; i++)
00155         input >> a.ptr[i];
00156     return input;
00157 }
00158
00159
00160 //--------------------------------------------------------------------------
00161 // operator<<
00162 // Overloaded output operator for class Array
00163 // Preconditions:   a.ptr must point to an array with size at least a.size
00164 // Postconditions:  The first a.size elements of a.ptr are sent to the
00165 //            output istream 10 per line with a trailing endl
00166 ostream& operator<<(ostream &output, const Array &a) {
00167     int i;
00168     for (i = 0; i < a.size; i++) {
00169         output << a.ptr[i] << ' ';
00170         if ((i + 1) % 10 == 0)
00171             output << endl;
00172     }
00173
00174     if (i % 10 != 0)
00175         output << endl;
00176     return output;
00177 }
```

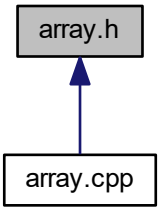
4.3 array.h File Reference

```
#include <iostream>
```

Include dependency graph for array.h:



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



Classes

- class Array

4.4 array.h

```
00001 //-----------------------------------------------------------------------
00002 // ARRAY.H
00003 // Simple class Array (for integers)
00005
00006 //-----------------------------------------------------------------------
00019
00020 //-----------------------------------------------------------------------
00021
00022
00023 #ifndef ARRAY_H
00024 #define ARRAY_H
00025
00026 #include <iostream>
00027 using namespace std;
00028
00029 class Array {
00030     //-------------------------------------------------------------------------
00032
00034
00037
```

```
00038
00039     friend istream& operator>>(istream &, Array &);
00040
00041     //--------------------------------------------------------------------------
00044
00046
00049
00050     friend ostream& operator<<(ostream &, const Array &);
00051
00052 public:
00053     //--------------------------------------------------------------------------------
00055
00057
00062
00063     Array(int = 10);
00064
00065     //--------------------------------------------------------------------------
00067
00069
00071
00072
00073     Array(const Array &);
00074
00075     //--------------------------------------------------------------------------
00077
00079
00081
00082     ~Array();
00083
00084     //--------------------------------------------------------------------------
00087
00089
00091
00092     int getSize() const;
00093
00094     //--------------------------------------------------------------------------
00096
00098
00100
00101     const Array& operator=(const Array &);
00102
00103     //--------------------------------------------------------------------------
00106
00109
00112
00113     bool operator==(const Array &) const;
00114
00115     //--------------------------------------------------------------------------
00118
00121
00124
00125     bool operator!=(const Array &) const;
00126
00127     //--------------------------------------------------------------------------
00130
00132
00134
00135     int& operator[](int);
00136
00137     //--------------------------------------------------------------------------
00140
00142
00144
00145     static int getArrayCount();
00146
00147 private:
00148     int* ptr;                         // pointer to first element of array
00149     int size;                         // size of the array
00150     static int arrayCount;            // # of Arrays instantiated
00151 };
00152
00153 #endif
```

Index