Object-Oriented Programming II

객체지향프로그래밍 II

Lecture 4

11장 연산자 오버로딩 (Part 3)

1. Case Study: Array Class

박인규 교수

1. Case Study: Array Class

C++ 배열의 단점

- ✓ 배열의 범위를 체크하지 않는다. (no range checking)
- ✓ == 로 두 개의 배열을 비교할 수 없다.
- ✓ 대입 연산자로 다른 배열로 대입될 수 없다.
  - ✓ 배열의 이름은 const 포인터이므로
- ✓ 만약 배열이 함수 인자로 전달될 때는 배열의 크기도 인수로 전달되어야 한다.
  - ✓ 배열의 이름으로는 배열 크기를 알 수 없음

C++ 배열의 개선

- ✓ 다음의 기능을 포함하는 새로운 배열을 구현하기 위한 클래스
- ✓ 범위 체크(range checking)가 가능
- ✓ 하나의 배열 객체를 다른 배열 객체에 대입할 수 있다.
- ✓ 별도의 인수로 배열의 크기를 함수에 전해줄 필요가 없다.
- ✓ 배열 전체를 `<<`과 `>>`를 통해 입출력 할 수 있다.
- ✓ `==` 과 `!=` 을 통한 배열 비교가 가능하다.

(추가할 만한 유용한 기능이 또 무엇이 있을까?)

Array 클래스의 복사 생성자 (Copy Constructor)

- ✓ 객체의 복사가 필요할 때마다 사용:
  - 객체가 값으로 함수에 전달될 때 (함수에서 값으로 객체를 반환할 때)
  - 같은 클래스의 다른 객체를 복사하여 초기화 할 때 다음과 같이 사용
    - ✓ `Array newArray( oldArray );` 또는
    - ✓ `Array newArray = oldArray`

수행후 newArray 는 oldArray 의 복사본이 됨

Array 클래스의 복사 생성자 (Copy Constructor)

✓ 복사 생성자 선언

- `Array( const Array & );`
- 반드시 참조(&)를 가져와야 한다.

그렇지 않으면, 인수는 값으로 전달되어지면서 매개변수 객체 생성

➡ 매개변수 객체 생성을 위해 복사 생성자 다시 호출

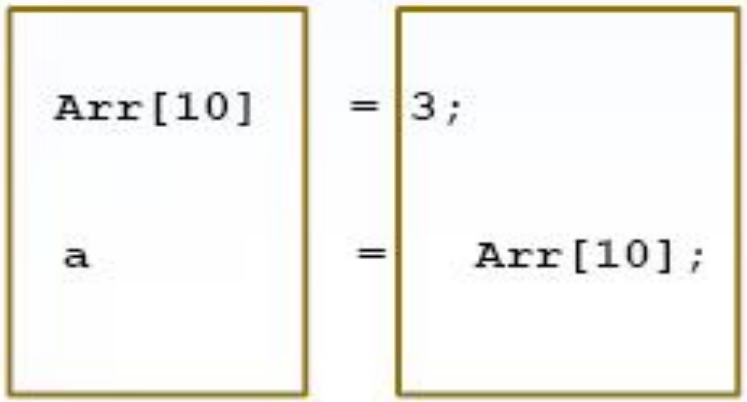
➡ 또다시 매개변수 객체 생성을 위해 복사 생성자 호출 ➡ 무한 루프

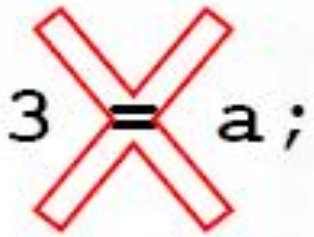
연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.h)

```
1  // Fig. 11.6: Array.h
2  // Array class for storing arrays of integers.
3  #ifndef ARRAY_H
4  #define ARRAY_H
5
6  #include <iostream>
7  using std::ostream;
8  using std::istream;
9
10 class Array
11 {
12    friend ostream &operator<<( ostream &, const Array & );
13    friend istream &operator>>( istream &, Array & );
14 public:
15    Array( int = 10 ); // default constructor
16    Array( const Array & ); // copy constructor
17    ~Array(); // destructor
18    int getSize() const; // return size
19
20    const Array &operator=( const Array & ); // assignment operator
21    bool operator==( const Array & ) const; // equality operator
22
23    // inequality operator; returns opposite of == operator
24    bool operator!=( const Array &right ) const
25    {
26       return ! ( *this == right ); // invokes Array::operator==
27    } // end function operator!=
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.h)

```
28
29    // subscript operator for non-const objects returns modifiable lvalue
30    int &operator[]( int );
31
32    // subscript operator for const objects returns rvalue
33    int operator[]( int ) const;
34 private:
35    int size; // pointer-based array size
36    int *ptr; // pointer to first element of pointer-based array
37 }; // end class Array
38
39 #endif
```





연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.cpp)

```
1  // Fig 11.7: Array.cpp
2  // Member-function definitions for class Array
3  #include <iostream>
4  using std::cerr;
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  #include <iomanip>
10 using std::setw;
11
12 #include <cstdlib> // exit function prototype
13 using std::exit;
14
15 #include "Array.h" // Array class definition
16
17 // default constructor for class Array (default size 10)
18 Array::Array( int arraySize )
19 {
20    size = ( arraySize > 0 ? arraySize : 10 ); // validate arraySize
21    ptr = new int[ size ]; // create space for pointer-based array
22
23    for ( int i = 0; i < size; i++ )
24       ptr[ i ] = 0; // set pointer-based array element
25 } // end Array default constructor
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.cpp)

```
26
27 // copy constructor for class Array;
28 // must receive a reference to prevent infinite recursion
29 Array::Array( const Array &arrayToCopy )
30    : size( arrayToCopy.size )
31 {
32    ptr = new int[ size ]; // create space for pointer-based array
33
34    for ( int i = 0; i < size; i++ )
35       ptr[ i ] = arrayToCopy.ptr[ i ]; // copy into object
36 } // end Array copy constructor
37
38 // destructor for class Array
39 Array::~Array()
40 {
41    delete [] ptr; // release pointer-based array space
42 } // end destructor
43
44 // return number of elements of Array
45 int Array::getSize() const
46 {
47    return size; // number of elements in Array
48 } // end function getSize
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.cpp)

```
49
50 // overloaded assignment operator;
51 // const return avoids: ( a1 = a2 ) = a3
52 const Array &Array::operator=( const Array &right )
53 {
54    if ( &right != this ) // avoid self-assignment
55    {
56       // for Arrays of different sizes, deallocate original
57       // left-side array, then allocate new left-side array
58       if ( size != right.size )
59       {
60          delete [] ptr; // release space
61          size = right.size; // resize this object
62          ptr = new int[ size ]; // create space for array copy
63       } // end inner if
64
65       for ( int i = 0; i < size; i++ )
66          ptr[ i ] = right.ptr[ i ]; // copy array into object
67    } // end outer if
68
69    return *this; // enables x = y = z, for example
70 } // end function operator=
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.cpp)

```
71
72 // determine if two Arrays are equal and
73 // return true, otherwise return false
74 bool Array::operator==( const Array &right ) const
75 {
76    if ( size != right.size )
77       return false; // arrays of different number of elements
78
79    for ( int i = 0; i < size; i++ )
80       if ( ptr[ i ] != right.ptr[ i ] )
81          return false; // Array contents are not equal
82
83    return true; // Arrays are equal
84 } // end function operator==
85
86 // overloaded subscript operator for non-const Arrays;
87 // reference return creates a modifiable lvalue
88 int &Array::operator[]( int subscript )
89 {
90    // check for subscript out-of-range error
91    if ( subscript < 0 || subscript >= size )
92    {
93       cerr << "\nError: Subscript " << subscript
94          << " out of range" << endl;
95       exit( 1 ); // terminate program; subscript out of range
96    } // end if
97
98    return ptr[ subscript ]; // reference return
99 } // end function operator[]
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.cpp)

```
100
101// overloaded subscript operator for const Arrays
102// const reference return creates an rvalue
103int Array::operator[]( int subscript ) const
104{
105   // check for subscript out-of-range error
106   if ( subscript < 0 || subscript >= size )
107   {
108      cerr << "\nError: Subscript " << subscript
109         << " out of range" << endl;
110      exit( 1 ); // terminate program; subscript out of range
111   } // end if
112
113   return ptr[ subscript ]; // returns copy of this element
114} // end function operator[]
115
116// overloaded input operator for class Array;
117// inputs values for entire Array
118istream &operator>>( istream &input, Array &a )
119{
120   for ( int i = 0; i < a.size; i++ )
121      input >> a.ptr[ i ];
122
123   return input; // enables cin >> x >> y;
124} // end function
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (Array.cpp)

```
125
126// overloaded output operator for class Array
127ostream &operator<<( ostream &output, const Array &a )
128{
129   int i;
130
131   // output private ptr-based array
132   for ( i = 0; i < a.size; i++ )
133   {
134      output << setw( 12 ) << a.ptr[ i ];
135
136      if ( ( i + 1 ) % 4 == 0 ) // 4 numbers per row of output
137         output << endl;
138   } // end for
139
140   if ( i % 4 != 0 ) // end last line of output
141      output << endl;
142
143   return output; // enables cout << x << y;
144} // end function operator<<
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (driver)

```
1  // Fig. 11.8: fig11_08.cpp
2  // Array class test program.
3  #include <iostream>
4  using std::cout;
5  using std::cin;
6  using std::endl;
7
8  #include "Array.h"
9
10 int main()
11 {
12    Array integers1( 7 ); // seven-element Array
13    Array integers2; // 10-element Array by default
14
15    // print integers1 size and contents
16    cout << "Size of Array integers1 is "
17       << integers1.getSize()
18       << "\nArray after initialization:\n" << integers1;
19
20    // print integers2 size and contents
21    cout << "\nSize of Array integers2 is "
22       << integers2.getSize()
23       << "\nArray after initialization:\n" << integers2;
24
25    // input and print integers1 and integers2
26    cout << "\nEnter 17 integers:" << endl;
27    cin >> integers1 >> integers2;
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (driver)

```
28
29    cout << "\nAfter input, the Arrays contain:\n"
30       << "integers1:\n" << integers1
31       << "integers2:\n" << integers2;
32
33    // use overloaded inequality (!=) operator
34    cout << "\nEvaluating: integers1 != integers2" << endl;
35
36    if ( integers1 != integers2 )
37       cout << "integers1 and integers2 are not equal" << endl;
38
39    // create Array integers3 using integers1 as an
40    // initializer; print size and contents
41    Array integers3( integers1 ); // invokes copy constructor
42
43    cout << "\nSize of Array integers3 is "
44       << integers3.getSize()
45       << "\nArray after initialization:\n" << integers3;
46
47    // use overloaded assignment (=) operator
48    cout << "\nAssigning integers2 to integers1:" << endl;
49    integers1 = integers2; // note target Array is smaller
50
51    cout << "integers1:\n" << integers1
52       << "integers2:\n" << integers2;
53
54    // use overloaded equality (==) operator
55    cout << "\nEvaluating: integers1 == integers2" << endl;
```

연산자 오버로딩으로 개선된 Array 클래스 예제 (driver)

```
56
57    if ( integers1 == integers2 )
58       cout << "integers1 and integers2 are equal" << endl;
59
60    // use overloaded subscript operator to create rvalue
61    cout << "\nintegers1[5] is " << integers1[ 5 ];
62
63    // use overloaded subscript operator to create lvalue
64    cout << "\n\nAssigning 1000 to integers1[5]" << endl;
65    integers1[ 5 ] = 1000;
66    cout << "integers1:\n" << integers1;
67
68    // attempt to use out-of-range subscript
69    cout << "\nAttempt to assign 1000 to integers1[15]" << endl;
70    integers1[ 15 ] = 1000; // ERROR: out of range
71    return 0;
72 } // end main
```

실행 결과

```
Size of Array integers1 is 7
Array after initialization:
          0          0          0          0
          0          0          0

Size of Array integers2 is 10
Array after initialization:
          0          0          0          0
          0          0          0          0
          0          0

Enter 17 integers:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

After input, the Arrays contain:
integers1:
          1          2          3          4
          5          6          7
integers2:
          8          9         10         11
         12         13         14         15
         16         17

Evaluating: integers1 != integers2
integers1 and integers2 are not equal
```

실행 결과

```
Size of Array integers3 is 7
Array after initialization:
           1           2           3           4
           5           6           7

Assigning integers2 to integers1:
integers1:
           8           9          10          11
          12          13          14          15
          16          17
integers2:
           8           9          10          11
          12          13          14          15
          16          17

Evaluating: integers1 == integers2
integers1 and integers2 are equal

integers1[5] is 13

Assigning 1000 to integers1[5]
integers1:
           8           9          10          11
          12        1000          14          15
          16          17

Attempt to assign 1000 to integers1[15]

Error: Subscript 15 out of range
```

Object-Oriented Programming II

객체지향프로그래밍 II

Lecture 4

11장 연산자 오버로딩 (Part 4)

1. Overloading ++ and --
2. Case Study: A Date Class
3. Standard Library Class string

박인규 교수

1. Overloading ++ and --

증가/감소 연산자의 오버로딩

- ✓ Date 클래스 객체 d1에 1을 더하려고 함
- ✓ Prototype (member function을 이용한 오버로딩)
  - Date &operator++();
  - ++d1 은 d1.operator++()과 동일
- ✓ Prototype (global function을 이용한 오버로딩)
  - Date &operator++(Date &);
  - ++d1 은 d1.operator++(d1)과 동일

접두 (prefix), 접미 (postfix) 증가의 구분

- ✓ 예) a++, ++a
- ✓ 접미 증가의 경우 공(空) 매개변수(dummy parameter)를 이용
  - 정수 0
- ✓ Prototype (global function을 이용한 오버로딩)
  - Date operator++(int);
  - d1++ becomes d1.operator++(0) // C++의 약속
- ✓ Prototype (member function을 이용한 오버로딩)
  - Date operator++(Date &, int);
  - d1++ becomes operator++(d1, 0)

접두 (prefix), 접미 (postfix) 증가의 반환값

- 반환값
  - 접두 증가 (prefix increment)
    - 참조형 반환 (Date &)
    - C++은 반환값을 lvalue로 취급B → assign 가능
  - 접미 증가 (postfix increment)
    - 값에 의한 반환 (Returns by value)
    - 이전 값을 가진 임시 객체 반환
    - C++은 반환값을 rvalue로 취급 → assign 불가능
- 감소 연산자 (--) 의 경우도 같은 방식으로 적용

2. Case Study: A Date Class

Date 클래스 예제 (개요)

- ✓ 증가 연산자 오버로드
  - 일, 월, 년을 변화
- ✓ += 연산자 오버로드
- ✓ 윤년 (leap years) 테스트를 위한 함수
- ✓ 일년 마지막 날임을 판단하는 함수

Date 클래스 예제 (date.h)

```
1  // Fig. 11.12: Date.h
2  // Date class definition.
3  #ifndef DATE_H
4  #define DATE_H
5
6  #include <iostream>
7  using std::ostream;
8
9  class Date
10 {
11    friend ostream &operator<<( ostream &, const Date & );
12 public:
13    Date( int m = 1, int d = 1, int y = 1900 ); // default constructor
14    void setDate( int, int, int ); // set month, day, year
15    Date &operator++(); // prefix increment operator
16    Date operator++( int ); // postfix increment operator
17    const Date &operator+=( int ); // add days, modify object
18    bool leapYear( int ) const; // is date in a leap year?
19    bool endOfMonth( int ) const; // is date at the end of month?
20 private:
21    int month;
22    int day;
23    int year;
24
25    static const int days[]; // array of days per month
26    void helpIncrement(); // utility function for incrementing date
27 }; // end class Date
28
29 #endif
```

Date 클래스 예제 (date.cpp)

```
1  // Fig. 11.13: Date.cpp
2  // Date class member-function definitions.
3  #include <iostream>
4  #include "Date.h"
5
6  // initialize static member at file scope; one classwide copy
7  const int Date::days[] =
8     { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
9
10 // Date constructor
11 Date::Date( int m, int d, int y )
12 {
13    setDate( m, d, y );
14 } // end Date constructor
15
16 // set month, day and year
17 void Date::setDate( int mm, int dd, int yy )
18 {
19    month = ( mm >= 1 && mm <= 12 ) ? mm : 1;
20    year = ( yy >= 1900 && yy <= 2100 ) ? yy : 1900;
21
22    // test for a leap year
23    if ( month == 2 && leapYear( year ) )
24       day = ( dd >= 1 && dd <= 29 ) ? dd : 1;
25    else
26       day = ( dd >= 1 && dd <= days[ month ] ) ? dd : 1;
27 } // end function setDate
```

Date 클래스 예제 (date.cpp)

```
28
29 // overloaded prefix increment operator
30 Date &Date::operator++()
31 {
32    helpIncrement(); // increment date
33    return *this; // reference return to create an lvalue
34 } // end function operator++
35
36 // overloaded postfix increment operator; note that the
37 // dummy integer parameter does not have a parameter name
38 Date Date::operator++( int )
39 {
40    Date temp = *this; // hold current state of object
41    helpIncrement();
42
43    // return unincremented, saved, temporary object
44    return temp; // value return; not a reference return
45 } // end function operator++
46
47 // add specified number of days to date
48 const Date &Date::operator+=( int additionalDays )
49 {
50    for ( int i = 0; i < additionalDays; i++ )
51       helpIncrement();
52
53    return *this; // enables cascading
54 } // end function operator+=
55
```

Date 클래스 예제 (date.cpp)

```
56 // if the year is a leap year, return true; otherwise, return false
57 bool Date::leapYear( int testYear ) const
58 {
59    if ( testYear % 400 == 0 ||
60       ( testYear % 100 != 0 && testYear % 4 == 0 ) )
61       return true; // a leap year
62    else
63       return false; // not a leap year
64 } // end function leapYear
65
66 // determine whether the day is the last day of the month
67 bool Date::endOfMonth( int testDay ) const
68 {
69    if ( month == 2 && leapYear( year ) )
70       return testDay == 29; // last day of Feb. in leap year
71    else
72       return testDay == days[ month ];
73 } // end function endOfMonth
74
```

Date 클래스 예제 (date.cpp)

```
75 // function to help increment the date
76 void Date::helpIncrement()
77 {
78    // day is not end of month
79    if ( !endOfMonth( day ) )
80       day++; // increment day
81    else
82       if ( month < 12 ) // day is end of month and month < 12
83       {
84          month++; // increment month
85          day = 1; // first day of new month
86       } // end if
87       else // last day of year
88       {
89          year++; // increment year
90          month = 1; // first month of new year
91          day = 1; // first day of new month
92       } // end else
93 } // end function helpIncrement
94
95 // overloaded output operator
96 ostream &operator<<( ostream &output, const Date &d )
97 {
98    static char *monthName[ 13 ] = { "", "January", "February",
99       "March", "April", "May", "June", "July", "August",
100      "September", "October", "November", "December" };
101   output << monthName[ d.month ] << ' ' << d.day << ", " << d.year;
102   return output; // enables cascading
103} // end function operator<<
```

Date 클래스 예제 (driver)

```
1  // Fig. 11.14: fig11_14.cpp
2  // Date class test program.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include "Date.h" // Date class definition
8
9  int main()
10 {
11    Date d1; // defaults to January 1, 1900
12    Date d2( 12, 27, 1992 ); // December 27, 1992
13    Date d3( 0, 99, 8045 ); // invalid date
14
15    cout << "d1 is " << d1 << "\nd2 is " << d2 << "\nd3 is " << d3;
16    cout << "\n\nd2 += 7 is " << ( d2 += 7 );
17
18    d3.setDate( 2, 28, 1992 );
19    cout << "\n\n  d3 is " << d3;
20    cout << "\n++d3 is " << ++d3 << " (leap year allows 29th)";
21
22    Date d4( 7, 13, 2002 );
23
24    cout << "\n\nTesting the prefix increment operator:\n"
25       << "  d4 is " << d4 << endl;
26    cout << "++d4 is " << ++d4 << endl;
27    cout << "  d4 is " << d4;
```

Date 클래스 예제 (driver 및 수행결과)

```
29     cout << "\n\nTesting the postfix increment operator:\n"
30        << "  d4 is " << d4 << endl;
31     cout << "d4++ is " << d4++ << endl;
32     cout << "  d4 is " << d4 << endl;
33     return 0;
34 } // end main
```

```
d1 is January 1, 1900
d2 is December 27, 1992
d3 is January 1, 1900

d2 += 7 is January 3, 1993

  d3 is February 28, 1992
++d3 is February 29, 1992 (leap year allows 29th)

Testing the prefix increment operator:
  d4 is July 13, 2002
++d4 is July 14, 2002
  d4 is July 14, 2002

Testing the postfix increment operator:
  d4 is July 14, 2002
d4++ is July 14, 2002
  d4 is July 15, 2002
```