EUROPEAN UNIVERSITY OF LEFKE
Faculty of Engineering
Department of Computer Engineering

COMP218
OBJECT-ORIENTED PROGRAMMING

Lab Work No. 8

Prepared by Seward Richard Mupereri (20140175)

Submitted to Dr. Ferhun Yorgancıoğlu

Task (1)

```
#include <iostream>

using namespace std;

class Array {
   friend ostream& operator<< ( ostream&, const Array& ); //overload “stream insertion” operator
   friend istream& operator>> ( istream&, Array& ); //overload “stream extraction” operator
public:
   Array(); //default constructor
   Array( const int*, int ); //parameterized constructor
   Array( const Array& ); //copy constructor
   ~ Array(); //destructor
   int getSize() const; //a constant member function
   void setSize( int ); //a non-constant member function
   bool operator==( const Array& ) const; //overload “is equal” operator
   bool operator!=( const Array& ) const; //overload “is not equal” operator
   int& operator[]( int ); //overload “subscript” operator as a non-constant l-value
   int operator[]( int ) const; //overload “subscript” operator as a constant r-value
   void operator()( int, int ); //overload “parenthesis” operator (passing index and value to be stored)
   Array operator++(); //overload “pre-increment” operator
   Array operator++(int ); //overload “post-increment” operator

private:
   int *list;
   int size;
};

ostream& operator<<( ostream& out, const Array& right )
{
   out << "SIZE: " << right.size << endl;
   out << "DATA: ";

   for (int i = 0; i < right.size; ++i)
   {
      out << right.list[i];
      out << "  ";
   }

   return out;
}

istream& operator>>( istream& in, Array& right )
{
   for (int i = 0; i < right.size; ++i)
   {
      in >> right.list[i];
   }

   return in;
}

Array::Array()
{
   size = 0;
   list = new int(size);
}

Array::Array( const int *_list, int size )
```

```
{
   this->size = size;
   this->list = new int[size];

   for(int i = 0; i< size; ++i)
   {
      list[i] = _list[i];
   }
}
Array::Array( const Array& Array )
{
   this->size = Array.size;
   this->list = new int[size];

   for(int i=0;i<Array.size;++i)
   {
      list[i] = Array.list[i];
   }
}

Array::~Array()
{
   for(int i = 0; i < this->size; ++i)
   {
      delete(&list[i]);
   }
}

int Array::getSize() const
{
   return size;
}

void Array::setSize( int s )
{
   size = s;
}

bool Array::operator==( const Array& right ) const
{
   if( this->size != right.size )
   {
      return false;
   }
   for( int i = 0; i < right.size; ++i )
   {
      if( list[i] != right.list[i] )
      {
         return false;
      }
   }

   return true;
}

bool Array::operator!=( const Array& right ) const
{
   if( this->size != right.size )
   {
      return true;
   }
```

```
    for(int i = 0; i < right.size;++i )
    {
        if( list[i] != right.list[i] )
        {
            return true;
        }
    }

    return false;
}

int& Array::operator[]( int i )
{
    return list[i];
}

int Array::operator[]( int i ) const
{
    return list[i];
}

void Array::operator()( int i, int x )
{
    list[i] = x;
}

Array Array::operator++()
{
    Array a;
    a.size= size;
    a.list= new int[size];

    for(int i=0;i< size;++i)
    {
        a.list[i]= ++list[i];
    }

    return a;
}

Array Array::operator++( int )
{
    Array a;
    a.size= size;
    a.list= new int[size];

    for(int i=0;i<size;++i)
    {
        a.list[i]= list[i]++;
    }

    return a;
}

int main()
{
    int size, value, index, x;

    cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
    cout << "|       OPERATOR OVERLOADING PROGRAM        |" << endl;
```

```
cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl << endl;
cout << "ENTER ARRAY SIZE: ";
cin >> size;
cout << endl;

int *list = new int(size);

cout << "ENTER " << size << " VALUES:" << endl;

for (int i = 0; i < size; ++i)
{
   cin >> list[i];
}

Array Arr1(list, size), Arr2, Arr3;

cout << Arr1 << endl << endl;
cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
cout << "|         ARRAY OBJECT OPERATIONS          |" << endl;
cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
cout << "1. CHANGE SIZE" << endl;
cout << "2. IS EQUAL? ( == )" << endl;
cout << "3. IS NOT EQUAL? ( != )" << endl;
cout << "4. SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )" << endl;
cout << "5. SUBSCRIPT - CONSTANT ( ARRAY[ index ] )" << endl;
cout << "6. PARENTHESIS ( ARRAY( index, value ) )" << endl;
cout << "7. PRE-INCREMENT ( ++size )" << endl;
cout << "8. POST-INCREMENT ( size++ )" << endl;
cout << "0. QUIT" << endl;
cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
cout << "ENTER A NUMBER FROM THE MENU: " << endl;

while ( ( cin >> value ) && ( value != 0 ) )
{
   switch ( value )
   {
      case 1:
      {
         cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
         cout << "|              CHANGE SIZE                 |" << endl;
         cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
         cout << "ENTER A NEW SIZE:";
         cin >> size;
         Arr1.setSize(size);
         cout << Arr1;
         cout << endl;

         break;
      }

      case 2:
      {
         cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
         cout << "|             IS EQUAL?( == )              |" << endl;
         cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
         cout << "/An identical object is created/" << endl;
         Arr2 = Arr1;
         cout << "ARE THE TWO OBJECTS EQUAL: ";
         cout << ( Arr1 == Arr2 ? "YES" : "NO" ) << endl;

         break;
```

```
    }

    case 3:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "|            IS NOT EQUAL? ( != )           |" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "/The same objects are compared again/" << endl;
       cout << "ARE THE TWO OBJECTS NOT EQUAL: ";
       cout << ( Arr1 != Arr2 ? "YES" : "NO" ) << endl;

       break;
    }

    case 4:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "| SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )|" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "Enter index to access:";
       cin >> index;
       if ( index < Arr1.getSize() )
       {
          cout << "Value at index " << index << " is " << Arr1[index] << endl;
       }
       else
       {
          cout << "INVALID INDEX" << endl;
       }

       break;
    }

    case 5:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "|   SUBSCRIPT - CONSTANT ( ARRAY[ index ] )  |" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "Enter index to access:";
       cin >> index;
       if ( index < Arr1.getSize() )
       {
          cout << "Value at index " << index << " is " << Arr1[index] << endl;
       }
       else
       {
          cout << "INVALID INDEX" << endl;
       }

       break;
    }

    case 6:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "|    PARENTHESIS ( ARRAY( index, value ) )   |" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "Enter index to change:";
       cin >> index;
       cout << "Enter a value:";
       cin >> x;
```

```
            Arr1.operator()( index, x );

            cout << Arr1 << endl;

            break;
        }

        case 7:
        {
            ++Arr1;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|        PRE-INCREMENT ( ++ARRAY )         |" << endl;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "ARRAY after pre-increment" << endl;
            cout << Arr1 << endl;

            break;
        }

        case 8:
        {
            Arr1++;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|        POST-INCREMENT ( ARRAY++ )        |" << endl;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "ARRAY after post-increment" << endl;
            cout << Arr1 << endl;

            break;
        }

        default:
        {
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|              INVALID OPTION              |" << endl;

            break;
        }

    }

    cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
    cout << "ENTER A NUMBER FROM THE MENU: " << endl;
  }

  cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
  cout << "|              QUITTING....                |" << endl;
  cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;

  return 0;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++
|        OPERATOR OVERLOADING PROGRAM        |
+++++++++++++++++++++++++++++++++++++++++++++

ENTER ARRAY SIZE:4

ENTER 4 VALUES:
12 45 78 56
SIZE: 4
DATA: 12  45  78  56

+++++++++++++++++++++++++++++++++++++++++++++
|          ARRAY OBJECT OPERATIONS           |
+++++++++++++++++++++++++++++++++++++++++++++
1. CHANGE SIZE
2. IS EQUAL? ( == )
3. IS NOT EQUAL? ( != )
4. SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )
5. SUBSCRIPT - CONSTANT ( ARRAY[ index ] )
6. PARENTHESIS ( ARRAY( index, value ) )
7. PRE-INCREMENT ( ++size )
8. POST-INCREMENT ( size++ )
0. QUIT
+++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
7
+++++++++++++++++++++++++++++++++++++++++++++
|         PRE-INCREMENT ( ++ARRAY )          |
+++++++++++++++++++++++++++++++++++++++++++++
ARRAY after pre-increment
SIZE: 4
DATA: 13  46  79  57
+++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
0
+++++++++++++++++++++++++++++++++++++++++++++
|               QUITTING....                |
+++++++++++++++++++++++++++++++++++++++++++++

Process finished with exit code 0
```

```
+++++++++++++++++++++++++++++++++++++++++++++
|        OPERATOR OVERLOADING PROGRAM        |
+++++++++++++++++++++++++++++++++++++++++++++

ENTER ARRAY SIZE:3

ENTER 3 VALUES:
7 5 6
SIZE: 3
DATA: 7  5  6

+++++++++++++++++++++++++++++++++++++++++++++
|          ARRAY OBJECT OPERATIONS           |
+++++++++++++++++++++++++++++++++++++++++++++
1. CHANGE SIZE
2. IS EQUAL? ( == )
3. IS NOT EQUAL? ( != )
4. SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )
5. SUBSCRIPT - CONSTANT ( ARRAY[ index ] )
6. PARENTHESIS ( ARRAY( index, value ) )
7. PRE-INCREMENT ( ++size )
8. POST-INCREMENT ( size++ )
0. QUIT
+++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
4
+++++++++++++++++++++++++++++++++++++++++++++
| SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )|
+++++++++++++++++++++++++++++++++++++++++++++
Enter index to access:1
Value at index 1 is 5
+++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
0
+++++++++++++++++++++++++++++++++++++++++++++
|               QUITTING....                |
+++++++++++++++++++++++++++++++++++++++++++++
```

Task (2)

HEADER FILE – operatorOverloading.h

```
#ifndef LABWORK8_OPERATOROVERLOADING_H
#define LABWORK8_OPERATOROVERLOADING_H

#include <iostream>

using namespace std;

class Array {
   friend ostream& operator<< ( ostream&, const Array& ); //overload “stream insertion” operator
   friend istream& operator>> ( istream&, Array& ); //overload “stream extraction” operator
public:
   Array(); //default constructor
   Array( const int*, int ); //parameterized constructor
   Array( const Array& ); //copy constructor
   ~ Array(); //destructor
   int getSize() const; //a constant member function
   void setSize( int ); //a non-constant member function
   bool operator==( const Array& ) const; //overload “is equal” operator
   bool operator!=( const Array& ) const; //overload “is not equal” operator
   int& operator[]( int ); //overload “subscript” operator as a non-constant l-value
   int operator[]( int ) const; //overload “subscript” operator as a constant r-value
   void operator()( int, int ); //overload “parenthesis” operator (passing index and value to be stored)
   Array operator++(); //overload “pre-increment” operator
   Array operator++(int ); //overload “post-increment” operator

private:
   int *list;
   int size;
};

#endif //LABWORK8_OPERATOROVERLOADING_H
```

IMPLEMENTATION FILE – operatorOverloading.cpp

```cpp
#include "operatorOverloading.h"

ostream& operator<<( ostream& out, const Array& right )
{
   out << "SIZE: " << right.size << endl;
   out << "DATA: ";

   for (int i = 0; i < right.size; ++i)
   {
      out << right.list[i];
      out << "  ";
   }

   return out;
}

istream& operator>>( istream& in, Array& right )
{
   for (int i = 0; i < right.size; ++i)
   {
      in >> right.list[i];
   }

   return in;
}

Array::Array()
{
   size = 0;
   list = new int(size);
}

Array::Array( const int *_list, int size )
{
   this->size = size;
   this->list = new int[size];

   for(int i = 0; i< size; ++i)
   {
      list[i] = _list[i];
   }
}
Array::Array( const Array& Array )
{
   this->size = Array.size;
   this->list = new int[size];

   for(int i=0;i<Array.size;++i)
   {
      list[i] = Array.list[i];
   }
}

Array::~Array()
{
   for(int i = 0; i < this->size; ++i)
   {
      delete(&list[i]);
   }
```

```
}

int Array::getSize() const
{
   return size;
}

void Array::setSize( int s )
{
   size = s;
}

bool Array::operator==( const Array& right ) const
{
   if( this->size != right.size )
   {
      return false;
   }
   for( int i = 0; i < right.size; ++i )
   {
      if( list[i] != right.list[i] )
      {
         return false;
      }
   }

   return true;
}

bool Array::operator!=( const Array& right ) const
{
   if( this->size != right.size )
   {
      return true;
   }

   for(int i = 0; i < right.size;++i )
   {
      if( list[i] != right.list[i] )
      {
         return true;
      }
   }

   return false;
}

int& Array::operator[]( int i )
{
   return list[i];
}

int Array::operator[]( int i ) const
{
   return list[i];
}

void Array::operator()( int i, int x )
{
   list[i] = x;
}
```

```
Array Array::operator++()
{
   Array a;
   a.size= size;
   a.list= new int[size];

   for(int i=0;i< size;++i)
   {
      a.list[i]= ++list[i];
   }

   return a;
}

Array Array::operator++( int )
{
   Array a;
   a.size= size;
   a.list= new int[size];

   for(int i=0;i<size;++i)
   {
      a.list[i]= list[i]++;
   }

   return a;
}
```

DRIVER FILE – main.cpp

```
#include <iostream>
#include "operatorOverloading.h"

using namespace std;

int main()
{
   int size, value, index, x;

   cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
   cout << "|       OPERATOR OVERLOADING PROGRAM       |" << endl;
   cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl << endl;
   cout << "ENTER ARRAY SIZE: ";
   cin >> size;
   cout << endl;

   int *list = new int(size);

   cout << "ENTER " << size << " VALUES:" << endl;

   for (int i = 0; i < size; ++i)
   {
      cin >> list[i];
   }

   Array Arr1(list, size), Arr2, Arr3;

   cout << Arr1 << endl << endl;
   cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
   cout << "|         ARRAY OBJECT OPERATIONS          |" << endl;
   cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
   cout << "1. CHANGE SIZE" << endl;
   cout << "2. IS EQUAL? ( == )" << endl;
   cout << "3. IS NOT EQUAL? ( != )" << endl;
   cout << "4. SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )" << endl;
   cout << "5. SUBSCRIPT - CONSTANT ( ARRAY[ index ] )" << endl;
   cout << "6. PARENTHESIS ( ARRAY( index, value ) )" << endl;
   cout << "7. PRE-INCREMENT ( ++size )" << endl;
   cout << "8. POST-INCREMENT ( size++ )" << endl;
   cout << "0. QUIT" << endl;
   cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
   cout << "ENTER A NUMBER FROM THE MENU: " << endl;

   while ( ( cin >> value ) && ( value != 0 ) )
   {
      switch ( value )
      {
         case 1:
         {
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|              CHANGE SIZE                 |" << endl;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "ENTER A NEW SIZE:";
            cin >> size;
            Arr1.setSize(size);
            cout << Arr1;
            cout << endl;

            break;
```

```
    }

    case 2:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "|              IS EQUAL?( == )              |" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "/An identical object is created/" << endl;
       Arr2 = Arr1;
       cout << "ARE THE TWO OBJECTS EQUAL: ";
       cout << ( Arr1 == Arr2 ? "YES" : "NO" ) << endl;

       break;
    }

    case 3:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "|            IS NOT EQUAL? ( != )           |" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "/The same objects are compared again/" << endl;
       cout << "ARE THE TWO OBJECTS NOT EQUAL: ";
       cout << ( Arr1 != Arr2 ? "YES" : "NO" ) << endl;

       break;
    }

    case 4:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "| SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )|" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "Enter index to access:";
       cin >> index;
       if ( index < Arr1.getSize() )
       {
          cout << "Value at index " << index << " is " << Arr1[index] << endl;
       }
       else
       {
          cout << "INVALID INDEX" << endl;
       }

       break;
    }

    case 5:
    {
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "|   SUBSCRIPT - CONSTANT ( ARRAY[ index ] )  |" << endl;
       cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
       cout << "Enter index to access:";
       cin >> index;
       if ( index < Arr1.getSize() )
       {
          cout << "Value at index " << index << " is " << Arr1[index] << endl;
       }
       else
       {
          cout << "INVALID INDEX" << endl;
       }
```

```
            break;
        }

        case 6:
        {
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|    PARENTHESIS ( ARRAY( index, value ) )   |" << endl;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "Enter index to change:";
            cin >> index;
            cout << "Enter a value:";
            cin >> x;

            Arr1.operator()( index, x );

            cout << Arr1 << endl;

            break;
        }

        case 7:
        {
            ++Arr1;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|        PRE-INCREMENT ( ++ARRAY )          |" << endl;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "ARRAY after pre-increment" << endl;
            cout << Arr1 << endl;

            break;
        }

        case 8:
        {
            Arr1++;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|        POST-INCREMENT ( ARRAY++ )         |" << endl;
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "ARRAY after post-increment" << endl;
            cout << Arr1 << endl;

            break;
        }

        default:
        {
            cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << "|              INVALID OPTION               |" << endl;

            break;
        }

    }

    cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
    cout << "ENTER A NUMBER FROM THE MENU: " << endl;
}

cout << "+++++++++++++++++++++++++++++++++++++++++++++" << endl;
cout << "|              QUITTING....                 |" << endl;
```

```
    cout << "++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

    return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++
|        OPERATOR OVERLOADING PROGRAM        |
++++++++++++++++++++++++++++++++++++++++++++++

ENTER ARRAY SIZE:5

ENTER 5 VALUES:
100 200 300 400 500
SIZE: 5
DATA: 100  200  300  400  500

++++++++++++++++++++++++++++++++++++++++++++++
|          ARRAY OBJECT OPERATIONS           |
++++++++++++++++++++++++++++++++++++++++++++++
1. CHANGE SIZE
2. IS EQUAL? ( == )
3. IS NOT EQUAL? ( != )
4. SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )
5. SUBSCRIPT - CONSTANT ( ARRAY[ index ] )
6. PARENTHESIS ( ARRAY( index, value ) )
7. PRE-INCREMENT ( ++size )
8. POST-INCREMENT ( size++ )
0. QUIT
++++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
4
++++++++++++++++++++++++++++++++++++++++++++++
| SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )|
++++++++++++++++++++++++++++++++++++++++++++++
Enter index to access:3
Value at index 3 is 400
++++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
0
++++++++++++++++++++++++++++++++++++++++++++++
|                QUITTING....                |
++++++++++++++++++++++++++++++++++++++++++++++
```

```
++++++++++++++++++++++++++++++++++++++++++++++
|        OPERATOR OVERLOADING PROGRAM        |
++++++++++++++++++++++++++++++++++++++++++++++

ENTER ARRAY SIZE:3

ENTER 3 VALUES:
852
963
456
SIZE: 3
DATA: 852  963  456

++++++++++++++++++++++++++++++++++++++++++++++
|          ARRAY OBJECT OPERATIONS           |
++++++++++++++++++++++++++++++++++++++++++++++
1. CHANGE SIZE
2. IS EQUAL? ( == )
3. IS NOT EQUAL? ( != )
4. SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )
5. SUBSCRIPT - CONSTANT ( ARRAY[ index ] )
6. PARENTHESIS ( ARRAY( index, value ) )
7. PRE-INCREMENT ( ++size )
8. POST-INCREMENT ( size++ )
0. QUIT
++++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
1
++++++++++++++++++++++++++++++++++++++++++++++
|               CHANGE SIZE                  |
++++++++++++++++++++++++++++++++++++++++++++++
ENTER A NEW SIZE:1
SIZE: 1
DATA: 852
++++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
0
++++++++++++++++++++++++++++++++++++++++++++++
|                QUITTING....                |
++++++++++++++++++++++++++++++++++++++++++++++
```

```
+++++++++++++++++++++++++++++++++++++++++++++
|        OPERATOR OVERLOADING PROGRAM        |
+++++++++++++++++++++++++++++++++++++++++++++

ENTER ARRAY SIZE:2

ENTER 2 VALUES:
15
26
SIZE: 2
DATA: 15  26

+++++++++++++++++++++++++++++++++++++++++++++
|          ARRAY OBJECT OPERATIONS           |
+++++++++++++++++++++++++++++++++++++++++++++
1. CHANGE SIZE
2. IS EQUAL? ( == )
3. IS NOT EQUAL? ( != )
4. SUBSCRIPT - NON CONSTANT ( ARRAY[ index ] )
5. SUBSCRIPT - CONSTANT ( ARRAY[ index ] )
6. PARENTHESIS ( ARRAY( index, value ) )
7. PRE-INCREMENT ( ++size )
8. POST-INCREMENT ( size++ )
0. QUIT
+++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
2
+++++++++++++++++++++++++++++++++++++++++++++
|               IS EQUAL?( == )              |
+++++++++++++++++++++++++++++++++++++++++++++
/An identical object is created/
ARE THE TWO OBJECTS EQUAL: YES
+++++++++++++++++++++++++++++++++++++++++++++
ENTER A NUMBER FROM THE MENU:
0
+++++++++++++++++++++++++++++++++++++++++++++
|                QUITTING....                |
+++++++++++++++++++++++++++++++++++++++++++++
```