Operators Overloading

حان وقت الانطلاق

قبل أن تبدأ إضافة أدوارك العليا تأكد أن أساسك يتحمل.

(Operator Overloading)

Given the following main function, write the:

1 - Default & copy constructors.
2 - Destructor.
3 - Conditional operators (==) and (!=).
4 - Arithmetic unary (++) and binary (+) operators.
5 - Assignment operators (=) and (+=).
6 - Const & non-const object subscript operators ([ ]).
7 - Insertion (<<) and extraction (>>) operators.

```
# include <iostream>
# include "Array.h"
  using namespace std;

int main ( )
{
      Array  a ( 7 ), b;                    // default size = 10

      cout  <<  "enter seven elements for array a:" << endl;
      cin  >>  a;

      b = a;

      if (a == b)
            cout  <<  "both arrays are equal"  <<  endl;

      a ++;

      if (a != b)
            cout  <<  "now, both arrays are not equal"  <<  endl;

      Array  c ( a );

      c += a;

      cout  <<  c;

      c [ 3 ] ++;

      cout  <<  "value of c [ 3 ] =  "  <<  c [ 3 ]  <<  endl;

      return 0;
}
```

إذا أردت أن تختلف
مع إنسان قل له ما
يجب عمله

(Operator Overloading)

```
# include <iostream>
using namespace std;
class  Array
{
      friend void operator<< ( ostream&, const Array & );

      friend void operator>> ( istream&,  Array & );

      private:
        int *x;
        int size;

      public:
        Array ( int = 10 );

        Array ( const Array & );

        ~Array ( );

        bool  operator==  ( const Array & ) const;

        bool  operator!=   ( const Array & a ) const
        {
              return   ! (*this == a);
        }

        const Array & operator= ( const Array & );

        void operator++ (    );

        const Array operator+ ( const Array & );

        void operator+= ( const Array & a )
        { *this = *this + a; }

        int operator[]    ( int ) const;

        int & operator[] ( int );
};
```

(Operator Overloading)

```
Array::Array (int  s)                    // default constructor
{
    if  ( s <= 0 )
         s = 10;


    size = s;

    x = new int [size];

    for (int i = 0;  i < size;   i ++ )
                x [ i ] = 0;

}


Array:: Array ( const Array  & a )              // copy constructor
{
    size = a.size;

    x = new int [ size ];

    for ( int i = 0;   i < size;   i ++ )
              x [ i ] = a.x [ i ];

}


Array::~Array ( )                        // destructor
{
    delete  [ ]    x;
}
```

غالبا تكون النهاية
حسب الغاية

(Operator Overloading)

```
bool   Array::operator== ( const array  & a ) const
{
    if (size !=  a.size)
            return false;

    for ( int i = 0;   i < size;   i ++ )
            if ( x [ i ]  !=  a.x [ i ] )
                    return false;

    return true;

}
```

```
void  Array::operator++ (    )
{
   for ( int i = 0;   i < size;   i ++ )

                    x [ i ] ++;
}
```

هل تعلم أنك إذا نويت خيراً يكتب لك،
الآن تخيل أنك حصلت على
مليون دينار؟ بكم ستتبرع؟

```
// const   return prevents      ( a1 = a2 )  =  a3
const  Array Array::operator= (const array & a)
{
     if ( this ==  & a )
            return  * this;                  // self assignment

     if  ( size != a.size )
     {
          delete  [ ]  x ;

          size =  a.size;

          x = new int [ size ];
     }

     for ( int i = 0;   i < size;   i ++ )

               x [ i ] =  a.x [ i ];


    return  *this;
}


Array Array::operator+ (const array  & a )
{
   Array   c ( size );

   for ( int i = 0 ;   i < size;   i ++ )

         c.x [ i ] = x [ i ] + a.x [ i ];

   return   c;

}
```

(Operator Overloading)

```
int  Array::operator [ ] ( int  i ) const
{

    if ( i < 0   ||    i >= size )

    {
        cout <<  "invalid index\n";

        exit ( 1 );
     }

     return  x [ i ];

}
```

```
int &  Array::operator[ ] ( int  i )
{

     if ( i < 0   ||    i >= size )

     {
          cout << "invalid  index\n";

          exit ( 1 );
      }

      return   x [ i ];

}
```

أساس النجاح في الإدارة
أن تستفيد أقصى ما يمكن
من قوة الآخرين وتتضرر
أقل ما يمكن من ضعفهم

(Operator Overloading)

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

    output  << "array elements are:\n";

    for  ( int i = 0 ;   i < a.size ;  i ++ )

            output << a.x [ i ] << "    ";

    output << endl;

}


void operator>> ( istream &input,  Array &a )
{

     for  ( int i = 0;   i < a.size;   i ++ )

             input >> a.x [ i ];

}
```

Note:
The upper two functions could return reference to ostream and istream types respectively as follows:

ostream & operator << (ostream &, const Arry &);
instream & operator << (istream &, const Array &);

to enable multiple outputs and inputs as:
cout << a << b << endl;
cin >> a >> b;

يمكنك أن تولد من جديد،
تماما من جديد