```
/* Examen Junio 2013*/
Class BigInteger{
    private:
        int *datos;
        int numeroDigitos;
        void reservarEspacio(int numeroDigitos);
        void liberarEspacio();
        friend ostream & operator<<(ostream & output,const BigInteger & objeto);
        bool hayAcarreo(const BigInteger & otro);
    public:
        BigInteger();
        ~BigInteger();
        BigInteger(const char * cadena);
        BigInteger(const BigInteger & otro);
        BigInteger(unsigned int valor);
        const BigInteger & operator=(const BigInteger & otro);
        const BigInteger & operator+(const BigInteger & otro);
}


//**********************************
//*************UTILS.CPP************
//**********************************
ostream & operator<<(ostream &output, const & BigInteger objeto){
    output<<endl;
    for(int i=objeto.numeroDigitos-1;i>0;i--)
        output<<objeto.datos[i]<<" ";
    output<<endl;
return output;
}


//**********************************
//*************UTILS.CPP************
//**********************************

//**********************************
//***********BigInteger.CPP**********
//**********************************
void BigInteger::reservarEspacio(int numeroDigitos){
    if(numeroDigitos>0){
        datos=new int[numeroDigitos];
        this->numeroDigitos=numeroDigitos;
    }
}
void BigInteger::liberarEspacio(){
    if(datos!=0){
        delete []datos;
        numeroDigitos=0;
    }
}

BigInteger::BigInteger(){
    datos=0;
    numeroDigitos=0;
}

~BigInteger(){
    liberarEspacio();
}
//La cadena se inserta al reves
BigInteger::BigInteger(const char * cadena){
    reservarEspacio(strlen(cadena));
    for(int i=0;i<numeroDigitos;i++){
        datos[numeroDigitos-i-1]=cadena[i]-'0
;   }
}

BigInteger::BigInteger(const BigInteger & otro){
    reservarEspacio(otro.numeroDigitos);

    for(int i=0;i<numeroDigitos;i++)
        datos[i]=otro.datos[i];
```

```
}

BigInteger::BigInteger(unsigned int valor){
    unsigned int copiaValor=valor;
    int potencia=0;
    while(copiaValor>0){
        copiaValor=copiaValor/10;
        potencia++;
    }
    reservarEspacio(potencia);
    for(int i=0;i<numeroDigitos;i++){
        datos[i]=valor%10;
        valor=valor/10;
    }

}

const BigInteger & BigInteger::operator=(const BigInteger & otro){
    if(this!=otro){
        if(datos!=0)
            liberarEspacio();
        reservarEspacio(otro.numeroDigitos);

        for(int i=0;i<numeroDigitos;i++)
            datos[i]=otro.datos[i];
    }

}

bool BigInteger::hayAcarreo(const BigInteger & otro){
        int potenciaMaximaThis=numeroDigitos;
        int potenciaMaximaOtro=otro.numeroDigitos;
        cont BigInteger * menor, *mayor;
        int menorPotencia;

        if(potenciaMaximaThis>potenciaMaximaOtro){
            menorPotencia=otro.numeroDigitos;
            menor=&otro;
            mayor=this;
        }
        else{
            menorPotencia=numeroDigitos;
            menor=this;
            mayor=&otro;
        }
        int digitoMayor=mayor->datos[menorPotencia-1];
        int digitoMenor=menor->datos[menorPotencia-1];

        bool acarreo=(digitoMayor+digitoMenor)>9;

        for(int i=menorPotencia;i<mayor->numeroDigitos&&acarreo;i++){
            digitoMayor=mayor->datos[i];
            if(digitoMayor<9)
                acarreo=false;
        }

    return acarreo;
}

const BigInteger & BigInteger::operator+(const BigInteger & otro){
    bool acarreo=hayAcarreo(otro);
    const BigInteger *menor,*mayor;
        if(potenciaMaximaThis>potenciaMaximaOtro){
            menor=&otro;
            mayor=this;
        }
        else{
            menor=this;
            mayor=&otro;
        }

    //reservar espacio
    int *nuevoDatos;
```

```
    int nuevoTam;

    if(acarreo){
        nuevoDatos=new int[mayor->numeroDigitos+1];
        nuevoTam=mayor->numeroDigitos+1;
    }
    else{
        nuevoDatos=new int[mayor->numeroDigitos];
        nuevoTam=mayor->numeroDigitos;
    }

    int digitoMenor,digitoMayor,suma;
    acarreo=false;

    for(int i=0;i<menor->numeroDigitos;i++){
        digitoMenor=menor->datos[i];
        digitoMayor=mayor->datos[i];
        suma=digitoMayor+digitoMenor;
        if(acarreo)
            suma++;

        if(suma>9){
            nuevoDatos[i]=suma%10;
            acarreo=true;
        }
        else{
            nuevoDatos[i]=suma;
            acarreo=false;
        }
    }

    for(int i=menor->numeroDigitos;i<mayor->numeroDigitos;i++){
        digitoMayor=mayor->datos[i];
        if(acarreo){
            digitoMayor++
        }

        if(digitoMayor>9){
            nuevoDatos[i]=digitoMayor%10;
            acarreo=true;
        }
        else{
            nuevoDatos[i]=digitoMayor;
            acarreo=false;
        }
    }

    if(acarreo)
        nuevoDatos[nuevoTam-1]=1;

    liberarEspacio();
    datos=nuevoDatos;
    numeroDigitos=nuevoTam;

    return *this;
}
//***********************************
//**********BigInteger.CPP**********
//***********************************

//***********************************
//**********suma.CPP**********
//***********************************
int main(int argc, char * argv[]){
    ifstream flujo;
    BigInteger suma,valor;

    //Llamada correcta
        if(argc<2){
            cerr<<"Uso : suma datos.xtx"<<endl;
            return -1;
        }
```

```
        flujo.open(argv[1);
        if(flujo){
            while(flujo>>valor)
                suma=suma+valor;
            cout << "La suma es:" << suma;
        }
        flujo.close();
        return 0;
}
//**********************************
//***********suma.CPP**********
//**********************************
```