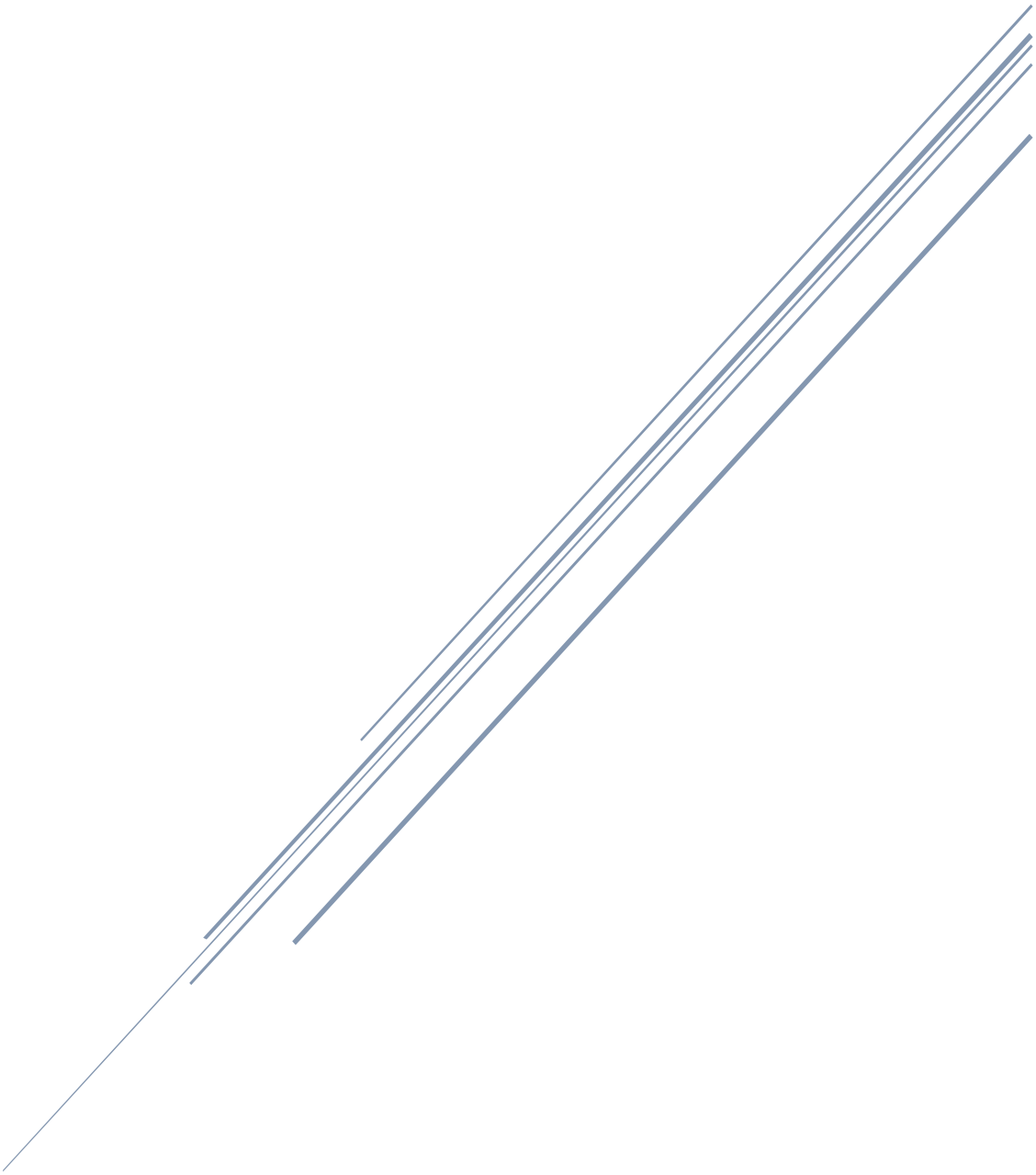
PRÁCTICA FINAL

Metodología de la Programación



Víctor Urquiza Hinojosa – 15474852S
Rafael LaChica Garrido – 76655442F

Tabla de contenido

- Ejercicio Vector

Clase Lista con destructor, constructor de copia y sobrecarga del operador de asignación.

Destructor de lista:

```
Lista::~Lista(){
    if(cabecera!=0) {
        destruir();
    }
}
```

```
void Lista::destruir(){
    if(cabecera!=0) {
        Celda *actual=cabecera;
        while(actual->siguiente!=0) {
            Celda *sig = actual->siguiente;
            //TODO Hace falta est en destruir
            //cerr << actual << "  " << actual->datos << " " << actual->siguiente << endl;
            delete actual;
            actual=sig;
        }
            //cerr << actual << "  " << actual->datos << " " << actual->siguiente << endl;
        delete actual;
        cabecera=0;
        num_elementos=0;
    }
}
```

El destructor lo que hace es siempre y cuando el puntero de tipo Celda que es cabecera sea distinto de cero, nos vamos a crear un puntero que apunte a cabecera y luego vamos a recorrer el vector mientras el puntero creado apunte a elementos de la lista. Lo que se hace es ir borrando esa posición de la lista e ir incrementando hacia la siguiente posición hasta que se borran todos los elementos de la lista. Borramos el puntero creado para que no quede memoria y establecemos a cero la cabecera y el número de elementos.

Constructor de copia de Lista:

```
Lista::Lista(const Lista & otra){
  this->cabecera=0; //Llamamos desde el constructor inicializamos
  this->copiar(otra);
}
```

```
void Lista::copiar(const Lista & otra){
    if (otra.cabecera != 0){
        this->destruir();
        this->inicializar();
        for (int i=0;i<otra.longitud();i++){
            this->insertar(otra.getCelda(i));
        }
    }
}
```

El constructor de copia lo que hace es establecer la cabecera a cero, comprobar que la lista que queremos copiar sea distinta de cero y en ese caso, borramos la original, la inicializamos y la copiamos con respecto a la otra lista, quedando dos listas iguales.

Operador de asignación de Lista:

```
Lista & Lista::operator = (const Lista & otra){
    this->copiar(otra);
    return *this;
}
```

Hace lo mismo que lo comentado anteriormente, menos lo de establecer la cabecera a cero.

Operador + de Lista:

```
Lista & Lista::operator + (const string str){
  insertar(str);
  return *this;
}
```

```
void Lista::insertar(string valor){
    if (cabecera == 0){ //Comprobamos si la lista est
        this->inicializar();
        cabecera->datos=valor;
    }
    else{
        Celda *nueva=new Celda;
        Celda *ultima=cabecera;
        nueva->datos=valor;
        nueva->siguiente=0;
        if (ultima->siguiente!=0)
            while(ultima->siguiente!=0) {
                ultima=ultima->siguiente;
        }
        ultima->siguiente=nueva;
    }
    num_elementos++;
}
```

Empezamos comprobando si la lista está vacía, en ese caso se inicializa y la insertamos. Si no está vacía, nos creamos dos punteros, uno vacío de tipo Celda y otro que apunta a la lista de cabecera. En el puntero vacío apuntamos el puntero hacia la variable valor que es un string y a siguiente que es el elemento siguiente de la lista lo ponemos a cero. Hacemos una serie de comprobaciones con el fin de que el puntero que habíamos creado que apuntaba a la lista cabecera, vaya guardando la siguiente posición de la lista. Para finalizar, incrementamos el número de elementos de la lista en la variable entera.

Destructor de la clase Imagen:

```
//Destructor
Imagen::~Imagen(){
        //cout<<"\nDESTRUCTOR"<<endl;
        destruir();
}

//Funcion auxliar al destructor
void Imagen::destruir(){
    nfilas = ncolumnas = 0;
    if (datos!=0) {
        delete [] datos;
    }
    datos = 0;
}
```

En el destructor de la clase imagen, ponemos las variables de las filas y columnas a cero, comprobamos que el vector datos no esté vacío, y en todo caso, lo borramos. Por ultimo establecemos datos a cero.

Constructor de copia de la clase Imagen:

```
Imagen::Imagen(const Imagen & copia){
        datos = 0; //Inicializamos la variable datos para que no haya problemas
        //se esta llamando desde el constructor por lo que no se ha creado antes
        //y no hay que destruir nada
        this->copiar(copia.datos,copia.nfilas,copia.ncolumnas);
}

void Imagen::copiar(byte * data,int f, int c){
        this->nfilas = f;
        this->ncolumnas = c;
        crear(nfilas,ncolumnas);

        for (int i=0; i<nfilas*ncolumnas; i++) {
            this->datos[i]=data[i];
        }
}
```

Esta función que es el constructor de copia, lo único que hace es copiar una imagen en otra.

Operador de asignación de la clase Imagen:

```
Imagen & Imagen::operator = (const Imagen & copia){
        this->copiar(copia.datos,copia.nfilas,copia.ncolumnas);
        return *this;
}
```

Utiliza la función vista en el apartado anterior.

Operador + de la clase Imagen:

```
Imagen operator + (const Imagen imagenA, const Imagen imagenB){
    if (imagenA.filas()>0 && imagenB.filas()>0) {
        int nuevaFila = imagenA.filas() > imagenB.filas() ? imagenA.filas() : imagenB.filas();
        int nuevaColumna = imagenA.columnas()+imagenB.columnas(); //Sumamos el numero de column
        Imagen nueva (nuevaFila,nuevaColumna);
        // //Escribimos la imagen A
        for (int i=0; i<imagenA.filas(); i++)
            for (int j=0;j<imagenA.columnas();j++)
                nueva.datos[i*nueva.columnas()+j]=imagenA.get(i,j);

            //Escribimos la imagen B
            for (int i=0; i<imagenB.nfilas; i++)
                for (int j=0;j<imagenB.ncolumnas;j++)
                    nueva.datos[i*nueva.columnas()+j+imagenA.columnas()]=imagenB.get(i,j);

                return nueva;
    }
    else{
        cout<<"\nDevolviendo imagen nula, una de las 2 imagenes no existe:"<<endl;
        Imagen nueva;
        return nueva;
    }
}
```

Esta función lo que hace es juntar dos imágenes en una, para ello, comprobamos que las dos imágenes tienen más de cero filas. Una vez comprobado, pasamos a crear una variable entera donde vamos a guardar la del valor de filas de la imagen más grande, y otra variable para las columnas en la que se suman las columnas de las dos imágenes. Creamos la imagen y empezamos a escribir las dos imágenes en la una imagen creada.

Programa Suma.cpp.

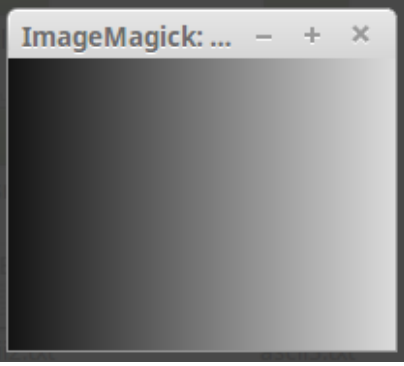
```
int main(int argc,char * argv[]){
    if (argc == 5){
        Imagen lecturaA, lecturaB;
    //Leemos ImagenA
    if (!lecturaA.leerImagen(argv[1])) {
        cerr << "error leyendo "<<argv[1]<<endl;
        return 1;
    }
    //Leemos ImagenB
    if (!lecturaB.leerImagen(argv[2])) {
        cerr << "error leyendo "<<argv[2]<<endl;
        return 1;
    }
    //Probamos a crear la imagen concatenada
        try{
            Imagen suma = (lecturaA+lecturaB);
            if (argv[4]==string("b"))
                suma.escribirImagen(argv[3],true);
            else
                suma.escribirImagen(argv[3],false);
        }
        catch (int e)
        {
            cout << "An exception occurred. Exception Nr. " << e << '\n';
        }

    }else{
        cout<<"\nError en el paso de argumentos, falta alguno de ellos....."<<endl;
        cout<<"\nFormato: ./suma img1.pgm img2.pgm img_out.pgm t"<<endl;
    }
    return 0;
}
```

Ejemplos de Ejecución:

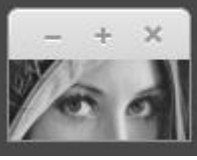
Testimagen:

```
portatil@portatil-VirtualBox ~/Escritorio/vector $ ./bin/testimagen
degradado.pgm guardado correctamente
usa: display degradado.pgm para ver el resultado
trozo.pgm guardado correctamente
usa: display trozo.pgm para ver el resultado
```

```
portatil@portatil-VirtualBox ~/Escritorio/vector $ display degradado.pgm
```

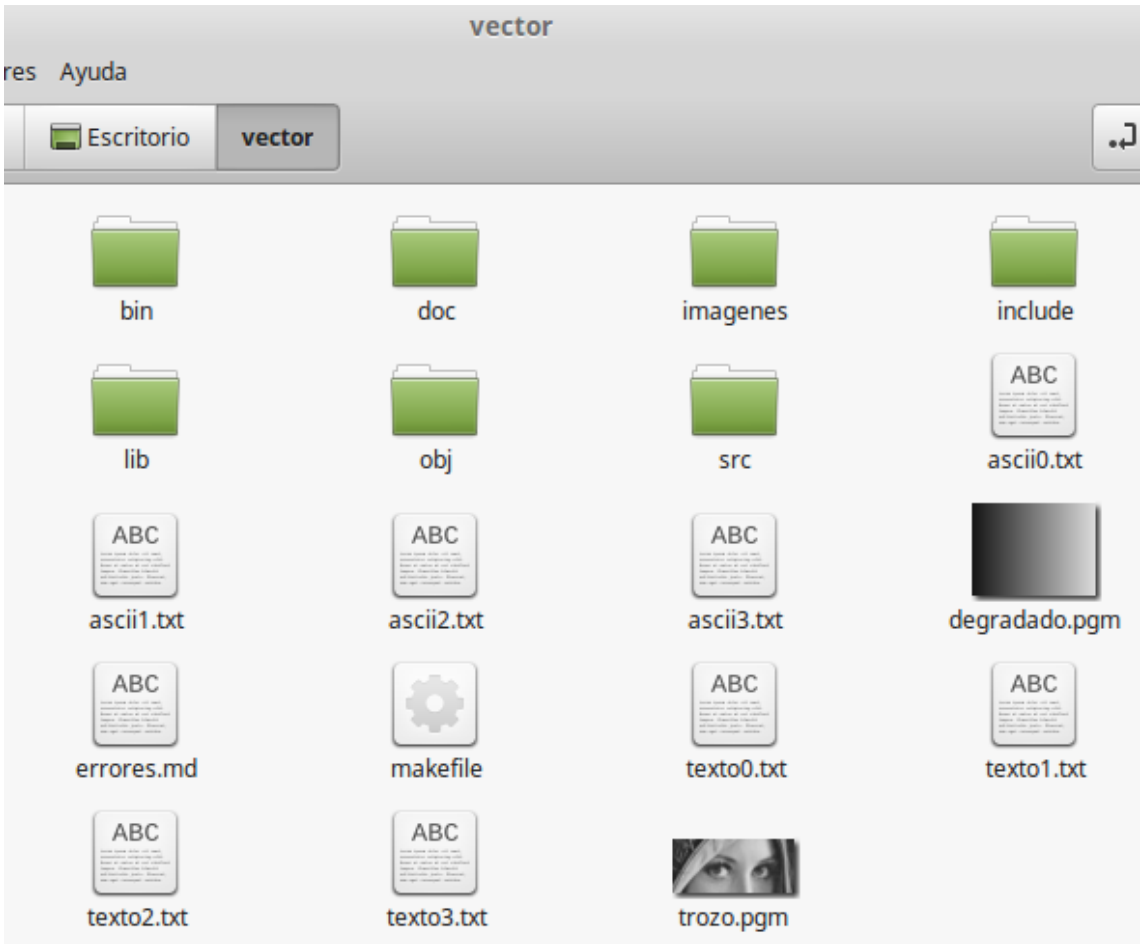


```
portatil@portatil-VirtualBox ~/Escritorio/vector $ display trozo.pgm
```



TestARTEASCII2:

```
portatil@portatil-VirtualBox ~/Escritorio/vector $ ./bin/arteASCII2 imagenes/gato.pgm grises.txt

++++++++++++++++++++++++++++

Imprimiendo lista de grises

Lista de longitud : 4
#$:.
@mQt]?'
@%#*+=-:.
$@B%8&WM#*oahkbdp\qwmZ00QLCJUYXzcvunxrjft/|()1{}[]?-_+<>i!lI;:,"^'`.

++++++++++++++++++++++++++++

 ascii0.txt
 ascii1.txt
 ascii2.txt
 ascii3.txt
Exito en la conversion.
Chequee los ficheros asciiX.txt
```



Suma:


portatil@portatil-VirtualBox ~/Escritorio/vector $ ./bin/suma imagenes/gio.pgm imagenes/gato.pgm sumafoto.pgm t
portatil@portatil-VirtualBox ~/Escritorio/vector $ display sumafoto.pgm
ImageMagick: sumafoto.pgm


portatil@portatil-VirtualBox ~/Escritorio/vector $ ./bin/suma imagenes/gato.pgm imagenes/lena.pgm sumafoto2.pgm t
portatil@portatil-VirtualBox ~/Escritorio/vector $ display sumafoto2.pgm
ImageMagick: sumafoto2.pgm