DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Número 2

Algoritmos y Estructuras de Datos I

Grupo: 1

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	gciruelos@dc.uba.ar
Gatti, Mathias	477/14	mathigatti@gmail.com
Rabinowicz, Lucía	105/14	lu.rabinowicz@gmail.com
Weber, Andres	923/13	herr.andyweber@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1. Observaciones

1. a

2. Especificación

2.1. posicionesMasOscuras

problema posicionesMasOscuras $(i : Imagen) = result : [\langle \mathbb{Z}, \mathbb{Z} \rangle]$ {

asegura : $mismos(result, [(x, y) | x \leftarrow [0..ancho(i)), y \leftarrow [0..alto(i)), sumaColor(color(i, x, y)) == colorMinimo(i)])$;

aux sumaColor(p:Pixel):$\mathbb{Z}$ = red(p)+green(p)+blue(p);
aux colorMinimo(i:Imagen):$\mathbb{Z} = min([sumaColor(color(i, x, y)) | x \leftarrow [0..ancho(i)), i \leftarrow [0..alto(i))])$;
aux min $(l : [\mathbb{Z}]) : \mathbb{Z} = [l_i | (\forall i, j \leftarrow [0..|l|))l_i \leq l_j]_0$;
}

2.2. top10

problema top10 $(g : Galeria)$=result:$[Imagen]$

asegura : $0 \leq |Result| \leq 10$;

asegura : $if\ |imagenes(g)| \leq 10\ then\ mismos(result, imagenes(g))\ else\ (\forall i \leftarrow sacar(imagenes(g), result))(j \leftarrow result), votos(g, i) \leq votos(g, j)$;

asegura : $estaOrdenadaDecreciente([votos(g, result_i) | i \leftarrow [0..|result|)])$;

aux sacar $(L : [T], A : [T]) : [T] = [l | (l \leftarrow L), (\forall a \leftarrow A), l \neq a]$
aux estaOrdenadaDecreciente$(l : [\mathbb{Z}]) : Bool = (\forall i, j \leftarrow [0..|l|), i \geq j), l_j \geq l_i$
}

2.3. laMasChiquitaConPuntoBlanco

problema laMasChiquitaConPuntoBlanco $(g : Galeria) = result : Imagen$ {

requiere existeImagenConPuntoBlanco: $(\exists h \leftarrow imagenes(g))tieneBlanco(h)$;

asegura : $tieneBlanco(result)$;

asegura : $ancho(Result) * alto(Result) \leq ancho(j) * alto(j)(\forall j \leftarrow imagenesConBlanco(g))$;

aux imagenesConBlanco $(g : Galeria) : [Imagen] = [H | H \leftarrow imagenes(g), tieneBlanco(H)]$
aux tieneBlanco$(i : Imagen) : Bool = (\exists x \leftarrow [0..ancho(i)), y \leftarrow [0..alto(i)))sumaColor(color(i, x, y)) == 765$
aux sumaColor$(p : Pixel) : \mathbb{Z} = red(p) + green(p) + blue(p)$
}

2.4. agregarImagen

problema agergarImagen $(g : Galeria, i : Imagen)$ {

requiere : $i \notin imagenes(g)$;

modifica: g;

asegura : $mismos(imagenes(g), imagenes(pre(g)) + +[i])$;

asegura : $votos(g, i) == 0$;

asegura : $(\forall h \leftarrow imagenes(g), j \leftarrow imagenes(pre(g)), h == j)votos(g, h) == votos(pre(g)), j)$;

}

2.5. votar

problema votar $(g : Galeria, i : Imagen)$ {

requiere : $i \in imagenes(g)$;

modifica: g;

asegura : $mismos(imagenes(g), imagenes(pre(g)))$;

asegura : $(\forall h \leftarrow imagenes(g), j \leftarrow imagenes(pre(g)), h == j, h \neq i) votos(g, h) == votos(pre(g), j)$;

asegura : $votos(g, i) == votos(pre(g), i) + 1$;

}

2.6. eliminarMasVotada

problema eliminarMasVotada $(g : Galeria)$ {

requiere : $|imagenes(g)| \neq 0$;

modifica: g;

asegura : $|imagenes(pre(g))| == |imagenes(g)| + 1$;

asegura : $(\exists h \in imagenes(pre(g)))(\forall j \leftarrow imagenes(g)), h \notin imagenes(g)) votos(pre(g), h) \geq votos(g, j)$;

asegura : $(\forall j \leftarrow imagenes(pre(g)), not(esMasVotada(pre(g), j))), j \in imagenes(g)$;

asegura : $(\forall i \leftarrow imagenes(g) not(esMasVotada(pre(g), i)) votos(g, i) == votos(pre(g), i)$;

asegura : $(\forall i \leftarrow imagenes(g), esMasVotada(i)) votos(g, i) == votos(pre(g), i)$;

aux esMasVotada $(g : Galeria, i : Imagen) : Bool = (\forall j \leftarrow imagenes(g)) votos(g, i) \geq votos(g, j)$

}

3. Implementacion

Código fuente 1: pixel.cpp

```
1  #include "pixel.h"
2
3  Pixel::Pixel(int red, int green, int blue) {
4      if(red < 0 || red > 255) red = 0;
5      if(green < 0 || green > 255) green = 0;
6      if(blue < 0 || blue > 255) blue = 0;
7
8      intensidades[0] = red;
9      intensidades[1] = green;
10     intensidades[2] = blue;
11 }
12
13 void Pixel::cambiarPixel(int red, int green, int blue) {
14     if(red < 0 || red > 255) red = 0;
15     if(green < 0 || green > 255) green = 0;
16     if(blue < 0 || blue > 255) blue = 0;
17
18     intensidades[0] = red;
19     intensidades[1] = green;
20     intensidades[2] = blue;
21 }
22
23 int Pixel::red() const {
```

```
24      return intensidades[0];
25  }
26
27  int Pixel::green() const {
28      return intensidades[1];
29  }
30
31  int Pixel::blue() const {
32      return intensidades[2];
33  }
34
35  void Pixel::guardar(std::ostream& os) const {
36      os << "(" << intensidades[0] << ";" << intensidades[1] << ";" << intensidades[2] << ")";
37  }
38
39  void Pixel::cargar(std::istream& is) {
40      char parentesis, puntoYComaOParentesis;
41      is >> parentesis;
42
43      int colores = 0;
44      while(colores < 3) {
45          int este_color;
46          is >> este_color;
47          intensidades[colores] = este_color;
48
49          is >> puntoYComaOParentesis;
50          colores++;
51      }
52  }
```

Código fuente 2: imagen.cpp

```
1   #include "imagen.h"
2   #include <algorithm>
3
4   Imagen::Imagen(int alto_param, int ancho_param) {
5       Pixel negro(0, 0, 0);
6       Pixel1DContainer filaNegra(ancho_param, negro);
7
8       int i = 0;
9       while (i < alto_param) {
10          pixels.push_back(filaNegra);
11          i++;
12      }
13  }
14
15  Pixel Imagen::obtenerPixel(int fila, int columna) const {
16      return pixels[fila][columna];
17  }
18
19  void Imagen::modificarPixel(int fila, int columna, const Pixel &pixel) {
20      pixels[fila][columna] = pixel;
21  }
22
23  int Imagen::alto() const {
24      return pixels.size();
25  }
26
27  int Imagen::ancho() const {
28      return pixels[0].size();
29  }
30
31  int abs(int x) {
32      return x > 0 ? x : -x;
33  }
34  int max(int a, int b) {
35      return a > b ? a : b;
36  }
37  int min(int a, int b) {
38      return a > b ? b : a;
39  }
40
41  vector<Pixel> kVecinos(Pixel2DContainer pixels, int pixel_i, int pixel_j, int k) {
```

```
42      int alto = pixels.size();
43      int ancho = pixels[0].size();
44
45      vector<Pixel> resultado;
46      int i = max(0, pixel_i - k + 1);
47      while(i < min(alto, pixel_i + k)) {
48          int j = max(0, pixel_j - k + 1);
49          while(j < min(ancho, pixel_j + k)) {
50
51              //if(abs(pixel_i-i)<k && abs(pixel_j-j)<k)
52              resultado.push_back(pixels[i][j]);
53              j++;
54          }
55          i++;
56      }
57
58      return resultado;
59  }
60
61  Pixel promedio(vector<Pixel> pixels) {
62      int r = 0, g = 0, b = 0;
63      int cantidad = pixels.size();
64
65      int i = 0;
66      while (i < cantidad) {
67          r += pixels[i].red();
68          g += pixels[i].green();
69          b += pixels[i].blue();
70
71          i++;
72      }
73
74      Pixel resultado(r / cantidad, g / cantidad, b / cantidad);
75      return resultado;
76  }
77
78  void Imagen::blur(int k) {
79      int i = 0;
80
81      Pixel pixel_negro(0, 0, 0);
82
83      Pixel2DContainer nuevo_pixels;
84
85      Pixel1DContainer filaNegra(ancho(), pixel_negro);
86      int iter = 0;
87      while (iter < alto()) {
88          nuevo_pixels.push_back(filaNegra);
89          iter++;
90      }
91
92      int kVecCompletos = (2 * k - 1) * (2 * k - 1);
93      while (i < alto()) {
94          int j = 0;
95          while (j < ancho()) {
96              vector<Pixel> kVec = kVecinos(pixels, i, j, k);
97
98              if(kVec.size() == kVecCompletos)
99                  nuevo_pixels[i][j] = promedio(kVec);
100             else nuevo_pixels[i][j] = pixel_negro;
101
102             j++;
103         }
104
105         i++;
106     }
107
108     pixels = nuevo_pixels;
109 }
110
111 Pixel mediana(vector<Pixel> pixels) {
112     vector<int> reds, greens, blues;
113     int cantidad = pixels.size();
114
```

```
115     int i = 0;
116     while (i < cantidad) {
117         reds.push_back(pixels[i].red());
118         greens.push_back(pixels[i].green());
119         blues.push_back(pixels[i].blue());
120 
121         i++;
122     }
123 
124     sort(reds.begin(), reds.end());
125     sort(greens.begin(), greens.end());
126     sort(blues.begin(), blues.end());
127 
128     Pixel resultado(reds[cantidad / 2], greens[cantidad / 2], blues[cantidad / 2]);
129     return resultado;
130 }
131 
132 void Imagen::acuarela(int k) {
133     Pixel pixel_negro(0, 0, 0);
134 
135     Pixel2DContainer nuevo_pixels;
136 
137     Pixel1DContainer filaNegra(ancho(), pixel_negro);
138     int iter = 0;
139     while (iter < alto()) {
140         nuevo_pixels.push_back(filaNegra);
141         iter++;
142     }
143 
144     int i = 0;
145     while (i < alto()) {
146         int j = 0;
147         while (j < ancho()) {
148             vector<Pixel> kVec = kVecinos(pixels, i, j, k);
149 
150             if(kVec.size() == (2 * k - 1) * (2 * k - 1))
151                 nuevo_pixels[i][j] = mediana(kVec);
152             else nuevo_pixels[i][j] = pixel_negro;
153 
154             j++;
155         }
156         i++;
157     }
158     pixels = nuevo_pixels;
159 }
160 
161 
162 
163 vector<pair<int, int> > Imagen::posicionesMasOscuras() const {
164     int colorMasOscuro = 255 * 3;
165 
166     vector<pair<int, int> > resultado;
167 
168     int i = 0, j;
169     while (i < alto()) {
170         j = 0;
171         while (j < ancho()) {
172             Pixel estePixel = pixels[i][j];
173             int colorPixel = estePixel.red() + estePixel.green() + estePixel.blue();
174 
175             if (colorPixel < colorMasOscuro) colorMasOscuro = colorPixel;
176         }
177     }
178 
179 
180     i = 0;
181     while (i < alto()) {
182         j = 0;
183         while (j < ancho()) {
184             Pixel estePixel = pixels[i][j];
185             int colorPixel = estePixel.red() + estePixel.green() + estePixel.blue();
186 
187             if (colorPixel < colorMasOscuro) resultado.push_back(make_pair(i, j));
```

```
188         }
189     }
190
191
192
193     return resultado;
194 }
195
196
197 bool Imagen::operator==(const Imagen &otra) const {
198     bool resultado = true;
199
200     if(alto() != otra.alto() || ancho() != otra.ancho())
201         resultado = false;
202     else {
203         int i = 0;
204         while (i < alto()) {
205             int j = 0;
206             while (j < ancho()) {
207                 Pixel p1 = pixels[i][j];
208                 Pixel p2 = otra.obtenerPixel(i, j);
209                 if (p1.red() != p2.red() || p1.green() != p2.green() || p1.blue() != p2.blue()) resultado
210             }
211         }
212     }
213
214     return resultado;
215 }
216
217 void Imagen::guardar(std::ostream& os) const {
218     os << alto() << " " << ancho() << " ";
219     int i = 0;
220     os << "[";
221     while (i < alto()) {
222         int j = 0;
223         while (j < ancho()) {
224             if(i != 0 || j != 0) os << ",";
225             pixels[i][j].guardar(os);
226
227             j++;
228         }
229         i++;
230     }
231     os << "]";
232 }
233
234 void Imagen::cargar(std::istream& is) {
235     int alto, ancho;
236     is >> alto;
237     is >> ancho;
238
239     char charMolesto;
240     is >> charMolesto; //corchete o coma
241
242     vector<vector<Pixel> > nueva_imagen;
243
244     int i = 0;
245     while(i < alto) {
246         int j = 0;
247         vector<Pixel> fila;
248         while(j < ancho) {
249             Pixel este_pixel(0, 0, 0);
250             este_pixel.cargar(is);
251
252             is >> charMolesto;
253
254             fila.push_back(este_pixel);
255             j++;
256         }
257
258         nueva_imagen.push_back(fila);
259         i++;
260     }
```

```
261
262     pixels = nueva_imagen;
263 }
```

Código fuente 3: galeria_imagenes.cpp

```
1  #include "galeria_imagenes.h"
2
3
4  Imagen GaleriaImagenes::laMasChiquitaConPuntoBlanco () const {
5      vector<Imagen> imagenes_blancas;
6      vector<int> tamanio_imagenes;
7
8      int k = 0;
9      while (k < imagenes.size()) {
10         int n = imagenes[k].alto();
11         int m = imagenes[k].ancho();
12         int i = 0;
13         while (i < n) {
14             int j = 0;
15             while (j < m) {
16                 Pixel este_pixel = imagenes[k].obtenerPixel(i, j);
17                 if (este_pixel.red() == 255 && este_pixel.blue() == 255 && este_pixel.green() == 255) {
18                     imagenes_blancas.push_back(imagenes[k]);
19                     tamanio_imagenes.push_back(m * n);
20                     j = m;  // para que termine el ciclo
21                     i = n;  // porque ya encontramos el pixel
22                 }
23                 j++;
24
25             }
26             i++;
27         }
28         k++;
29     }
30
31     int mas_chica = tamanio_imagenes[0];
32     int i_mas_chica = 0;
33     int i = 1;
34     while(i < imagenes_blancas.size()) {
35         if(tamanio_imagenes[i] < mas_chica) {
36             i_mas_chica = i;
37             mas_chica = tamanio_imagenes[i];
38         }
39         i++;
40     }
41     return imagenes_blancas[i_mas_chica];
42
43 }
44
45
46 void GaleriaImagenes::agregarImagen(const Imagen &imagen) {
47     vector<Imagen> imagenes_nuevas;
48     vector<int> votos_nuevos;
49
50     imagenes_nuevas.push_back(imagen);
51     votos_nuevos.push_back(0);
52
53     int i = 0;
54     while(i < imagenes.size()) {
55         imagenes_nuevas.push_back(imagenes[i]);
56         votos_nuevos.push_back(votos[i]);
57
58         i++;
59     }
60 }
61
62 void GaleriaImagenes::votar(const Imagen &imagen) {
63     int i = 0;
64     std::vector<Imagen> imagenes_nuevas;
65     std::vector<int> votos_nuevos;
66
67     int votos_imagen;
```

```
68      while (i < imagenes.size() && !(imagenes[i] == imagen)) {
69          imagenes_nuevas.push_back(imagenes[i]);
70          votos_nuevos.push_back(votos[i]);
71
72          i++;
73      }
74      votos_imagen = votos[i];
75      i++;
76      // requerimos que la imagen este, entonces en este estado i<imagenes.size() e imagenes[i]==imagen
77      while (votos[i] == votos_imagen && i < imagenes.size()) {
78          imagenes_nuevas.push_back(imagenes[i]);
79          votos_nuevos.push_back(votos[i]);
80
81          i++;
82      }
83      imagenes_nuevas.push_back(imagen);
84      votos_nuevos.push_back(votos_imagen + 1);
85
86      while (votos[i] == votos_imagen && i < imagenes.size()) {
87          imagenes_nuevas.push_back(imagenes[i]);
88          votos_nuevos.push_back(votos[i]);
89
90          i++;
91      }
92
93      imagenes = imagenes_nuevas;
94      votos = votos_nuevos;
95  }
96
97  void GaleriaImagenes::eliminarMasVotada() {
98      if(imagenes.size() > 0) {
99          imagenes.pop_back();
100         votos.pop_back();
101     }
102 }
103
104
105 vector<Imagen> GaleriaImagenes::top10 () const {
106     int i = 0;
107     vector <Imagen> result;
108     while (i < 10 && imagenes.size() - i - 1 < imagenes.size()) {
109         result.push_back(imagenes[imagenes.size() - i - 1]);
110         i++;
111     }
112     return result;
113 }
114
115
116 vector<Imagen> dividir (const Imagen &img, int n, int m) {
117     vector<Imagen> resultado;
118
119     int cols  = img.ancho() / n;
120     int filas = img.alto() / m;
121
122     int i = 0;
123
124     while(i < img.alto()) {
125         int j = 0;
126         while(j < img.ancho()) {
127
128             Imagen esta_imagen(filas, cols);
129
130             int i_img = 0;
131             while (i_img < filas) {
132                 int j_img = 0;
133                 while(j_img < cols) {
134                     Pixel este_pixel = img.obtenerPixel(i + i_img, j + j_img);
135                     esta_imagen.modificarPixel(i_img, j_img, este_pixel);
136
137                     j_img++;
138                 }
139                 i_img++;
140             }
```

```
141             resultado.push_back(esta_imagen);
142             j += cols;
143         }
144         i += filas;
145     }
146
147     return resultado;
148 }
149
150
151
152 void GaleriaImagenes::dividirYAgregar(const Imagen &imagen, int n, int m) {
153     int alto = imagen.alto();
154     int ancho = imagen.ancho();
155
156     if (alto % m == 0 && ancho % n == 0) {
157         vector<Imagen> dividida = dividir(imagen, n, m);
158
159         int i = 0;
160         while(i < dividida.size()) {
161             this->agregarImagen(dividida[i]);
162         }
163     }
164 }
165
166 void GaleriaImagenes::guardar(std::ostream& os) const {
167     os << imagenes.size() << endl;
168     int i = 0;
169     while(i < imagenes.size()) {
170         imagenes[i].guardar(os);
171         os << endl;
172     }
173 }
174
175 void GaleriaImagenes::cargar(std::istream& is) {
176     Imagen im(1, 1);
177
178     vector<Imagen> imagenes_nuevas;
179     vector<int> votos_nuevos;
180
181     int cantidad_imagenes;
182     is >> cantidad_imagenes;
183     int i = 0;
184     while(i < cantidad_imagenes) {
185         im.cargar(is);
186
187         imagenes_nuevas.push_back(im);
188         votos_nuevos.push_back(0);
189
190         i++;
191     }
192     imagenes = imagenes_nuevas;
193     votos = votos_nuevos;
194 }
```

Código fuente 4: main.cpp

```
1  #include "pixel.h"
2  #include "imagen.h"
3  #include "galeria_imagenes.h"
4
5  #include <string>
6  #include <fstream>
7
8  int main() {
9      string input = "";
10     GaleriaImagenes galeriaCargada;
11     string archivo_galeria;
12
13     while(input != "x") {
14
15
16         cout << "_____" << endl;
```

```
17          cout << "|                                      |" << endl;
18          cout << "| 1. blur                              |" << endl;
19          cout << "| 2. acuarela                          |" << endl;
20          cout << "| 3. cargar galeria                    |" << endl;
21          cout << "| 4. dividir y agregar                 |" << endl;
22          cout << "| 5. posiciones mas oscuras            |" << endl;
23          cout << "| 6. top 10                            |" << endl;
24          cout << "| 7. la mas chiquita con punto blanco  |" << endl;
25          cout << "| 8. agregar imagen                    |" << endl;
26          cout << "| 9. votar                             |" << endl;
27          cout << "| 10. eliminar mas votada              |" << endl;
28          cout << "| 11. guardar galeria                  |" << endl;
29          cout << "|______________________________________|" << endl;
30          cout << "Que desea hacer? ";
31          cin >> input;
32          cout << endl;
33          if (input == "1") {
34              int k;
35              string archivo_in;
36              string archivo_out;
37
38              cout << "Ingrese un k, el nombre del archivo de entrada y el de salida" << endl;
39              cin >> k >> archivo_in >> archivo_out;
40
41              ifstream ifs(archivo_in.c_str());
42              ofstream ofs(archivo_out.c_str());
43
44              Imagen im(1, 1);
45              im.cargar(ifs);
46
47              cerr << "Cargo imagen" << endl;
48              im.blur(k);
49              cerr << "Hizo blur" << endl;
50
51              im.guardar(ofs);
52          }
53          else if(input == "2") {
54              int k;
55              string archivo_in;
56              string archivo_out;
57
58              cout << "Ingrese un k, el nombre del archivo de entrada y el de salida" << endl;
59              cin >> k >> archivo_in >> archivo_out;
60
61              ifstream ifs(archivo_in.c_str());
62              ofstream ofs(archivo_out.c_str());
63
64              Imagen im(1, 1);
65              im.cargar(ifs);
66
67              im.blur(k);
68
69              im.guardar(ofs);
70          }
71          else if(input == "3") {
72              string archivo_in;
73
74              cout << "Ingrese el nombre del archivo" << endl;
75              cin >> archivo_in;
76
77              ifstream ifs(archivo_in.c_str());
78              galeriaCargada.cargar(ifs);
79              archivo_galeria = archivo_in;
80          }
81          else if(input == "4") {
82              string archivo_in, archivo_in2;
83              int n, m;
84
85              cout << "Ingrese el nombre del archivo de la galeria, el nombre del archivo de la imagen, el n
86
87              cin >> archivo_in >> archivo_in2 >> n >> m;
88
89              ifstream ifs_galeria(archivo_in.c_str());
```

```
90              ifstream ifs_imagen(archivo_in2.c_str());
91
92              galeriaCargada.cargar(ifs_galeria);
93
94              Imagen im(1, 1);
95              im.cargar(ifs_imagen);
96
97              galeriaCargada.dividirYAgregar(im, n, m);
98          }
99          else if(input == "5") {
100             string archivo_in;
101
102             cout << "Ingrese el nombre del archivo de la imagen" << endl;
103
104             ifstream ifs(archivo_in.c_str());
105
106             Imagen im(1, 1);
107             im.cargar(ifs);
108
109             vector<pair<int, int> > resultado = im.posicionesMasOscuras();
110             int i = 0;
111             while(i < resultado.size()) {
112                 cout << "(" << resultado[i].first << "," << resultado[i].second << ")";
113             }
114             cout << endl;
115
116             galeriaCargada.agregarImagen(im);
117         }
118         else if(input == "6") {
119             string archivo_out;
120
121             cout << "Ingrese el nombre del archivo para el output" << endl;
122
123             ofstream ofs(archivo_out.c_str());
124
125             vector<Imagen> top = galeriaCargada.top10();
126
127             int i = 0;
128             ofs << "[";
129             while(i < top.size()) {
130                 if (i != 0) ofs << ",";
131                 top[i].guardar(ofs);
132             }
133             ofs << "]";
134         }
135         else if(input == "7") {
136             string archivo_out;
137
138             cout << "Ingrese el nombre del archivo para el output" << endl;
139
140             ofstream ofs(archivo_out.c_str());
141
142             Imagen resultado = galeriaCargada.laMasChiquitaConPuntoBlanco();
143             resultado.guardar(ofs);
144         }
145         else if(input == "8") {
146             string archivo_in;
147
148             cout << "Ingrese el nombre del archivo de la imagen" << endl;
149
150             ifstream ifs(archivo_in.c_str());
151
152             Imagen im(1, 1);
153             im.cargar(ifs);
154
155             galeriaCargada.agregarImagen(im);
156         }
157         else if(input == "9") {
158             string archivo_in;
159
160             cout << "Ingrese el nombre del archivo de la imagen" << endl;
161
162             ifstream ifs(archivo_in.c_str());
```

```
163
164             Imagen im(1, 1);
165             im.cargar(ifs);
166
167             galeriaCargada.votar(im);
168         }
169         else if(input == "10") {
170             galeriaCargada.eliminarMasVotada();
171         }
172         else if(input == "11") {
173             ofstream ofs(archivo_galeria.c_str());
174             galeriaCargada.guardar(ofs);
175         }
176         else {
177             if(input != "x") cout << "Opcion invalida, por favor seleccione una opcion valida (1-10, x)."
178         }
179
180     }
181
182     return 0;
183 }
```

4. Demostraciones