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

aux imagenesConBlanco $(g : Galeria) : [Imagen] = [H | H \leftarrow imagenes(g), tieneBlanco(H)]$
aux tieneBlanco$(i : Imagen) : Bool = (\exists x \leftarrow [0..ancho(i)), y \leftarrow [0..alto(i)))sumaColor(color(i, x, y)) == 255 * 3$
aux sumaColor$(p : Pixel) : \mathbb{Z} = red(p) + green(p) + blue(p)$
}

2.4. agregarImagen

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

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 : $(\forall h \leftarrow imagenes(pre(g))) h \notin imagenes(g) \implies (\forall j \leftarrow imagenes(pre(g))) votos(pre(g), h) \geq gotos(pre(g), j)$;

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

}

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 charMolesto; //punto y coma o parentesis
41     is >> charMolesto;
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 >> charMolesto;
50         colores++;
51     }
52 }
```

Código fuente 2: imagen.cpp

```
1  #include "imagen.h"
2
3  void sort(vector<int> &v) {
4      //algoritmo de burbujeo
5      int i = 0;
6      while(i < v.size()) {
7          int j = i;
8          while(j < v.size() - 1) {
9              if (v[j] > v[j + 1]) {
10                 int temp = v[j];
11                 v[j] = v[j + 1];
12                 v[j + 1] = temp;
13             }
14             j++;
15         }
16         i++;
17     }
18 }
19
20 Imagen::Imagen(int alto_param, int ancho_param) {
21     Pixel negro(0, 0, 0);
22     Pixel1DContainer filaNegra(ancho_param, negro);
23
24     int i = 0;
25     while (i < alto_param) {
26         pixels.push_back(filaNegra);
27         i++;
28     }
29 }
30
31 Pixel Imagen::obtenerPixel(int fila, int columna) const {
32     return pixels[fila][columna];
33 }
34
35 void Imagen::modificarPixel(int fila, int columna, const Pixel &pixel) {
36     pixels[fila][columna] = pixel;
37 }
38
39 int Imagen::alto() const {
40     return pixels.size();
41 }
42
43 int Imagen::ancho() const {
44     return pixels[0].size();
45 }
46
```

```
47  int max(int a, int b) {
48      return a > b ? a : b;
49  }
50  int min(int a, int b) {
51      return a > b ? b : a;
52  }
53
54  vector<Pixel> kVecinos(Pixel2DContainer pixels, int pixel_i, int pixel_j, int k) {
55      int alto = pixels.size();
56      int ancho = pixels[0].size();
57
58      vector<Pixel> resultado;
59      int i = max(0, pixel_i - k + 1);
60      while(i < min(alto, pixel_i + k)) {
61          int j = max(0, pixel_j - k + 1);
62          while(j < min(ancho, pixel_j + k)) {
63              resultado.push_back(pixels[i][j]);
64              j++;
65          }
66          i++;
67      }
68
69      return resultado;
70  }
71
72  Pixel promedio(vector<Pixel> pixels) {
73      int r = 0, g = 0, b = 0;
74      int cantidad = pixels.size();
75
76      int i = 0;
77      while (i < cantidad) {
78          r += pixels[i].red();
79          g += pixels[i].green();
80          b += pixels[i].blue();
81
82          i++;
83      }
84
85      Pixel resultado(r / cantidad, g / cantidad, b / cantidad);
86      return resultado;
87  }
88
89  void Imagen::blur(int k) {
90      int i = 0;
91
92      Pixel pixel_negro(0, 0, 0);
93
94      Pixel2DContainer nuevo_pixels;
95
96      Pixel1DContainer filaNegra(ancho(), pixel_negro);
97      int iter = 0;
98      while (iter < alto()) {
99          nuevo_pixels.push_back(filaNegra);
100         iter++;
101     }
102
103     int kVecCompletos = (2 * k - 1) * (2 * k - 1);
104     while (i < alto()) {
105         int j = 0;
106         while (j < ancho()) {
107             vector<Pixel> kVec = kVecinos(pixels, i, j, k);
108
109             if(kVec.size() == kVecCompletos)
110                 nuevo_pixels[i][j] = promedio(kVec);
111             else nuevo_pixels[i][j] = pixel_negro;
112
113             j++;
114         }
115
116         i++;
117     }
118
119     pixels = nuevo_pixels;
```

```
120 }
121
122 Pixel mediana(vector<Pixel> pixels) {
123     vector<int> reds, greens, blues;
124     int cantidad = pixels.size();
125
126     int i = 0;
127     while (i < cantidad) {
128         reds.push_back(pixels[i].red());
129         greens.push_back(pixels[i].green());
130         blues.push_back(pixels[i].blue());
131
132         i++;
133     }
134
135     sort(reds);
136     sort(greens);
137     sort(blues);
138
139     Pixel resultado(reds[cantidad / 2], greens[cantidad / 2], blues[cantidad / 2]);
140     return resultado;
141 }
142
143 void Imagen::acuarela(int k) {
144     Pixel pixel_negro(0, 0, 0);
145
146     Pixel2DContainer nuevo_pixels;
147
148     Pixel1DContainer filaNegra(ancho(), pixel_negro);
149     int iter = 0;
150     while (iter < alto()) {
151         nuevo_pixels.push_back(filaNegra);
152         iter++;
153     }
154
155     int i = 0;
156     while (i < alto()) {
157         int j = 0;
158         while (j < ancho()) {
159             vector<Pixel> kVec = kVecinos(pixels, i, j, k);
160
161             if(kVec.size() == (2 * k - 1) * (2 * k - 1))
162                 nuevo_pixels[i][j] = mediana(kVec);
163             else nuevo_pixels[i][j] = pixel_negro;
164
165             j++;
166         }
167         i++;
168     }
169     pixels = nuevo_pixels;
170 }
171
172
173
174 vector<pair<int, int> > Imagen::posicionesMasOscuras() const {
175     int colorMasOscuro = 255 * 3;
176
177     vector<pair<int, int> > resultado;
178
179     int i = 0, j;
180     while (i < alto()) {
181         j = 0;
182         while (j < ancho()) {
183             Pixel estePixel = pixels[i][j];
184             int colorPixel = estePixel.red() + estePixel.green() + estePixel.blue();
185
186             if (colorPixel < colorMasOscuro) colorMasOscuro = colorPixel;
187         }
188     }
189
190
191     i = 0;
192     while (i < alto()) {
```

```
193         j = 0;
194         while (j < ancho()) {
195             Pixel estePixel = pixels[i][j];
196             int colorPixel = estePixel.red() + estePixel.green() + estePixel.blue();
197 
198             if (colorPixel < colorMasOscuro) resultado.push_back(make_pair(i, j));
199         }
200     }
201 
202 
203 
204     return resultado;
205 }
206 
207 
208 bool Imagen::operator==(const Imagen &otra) const {
209     bool resultado = true;
210 
211     if(alto() != otra.alto() || ancho() != otra.ancho())
212         resultado = false;
213     else {
214         int i = 0;
215         while (i < alto()) {
216             int j = 0;
217             while (j < ancho()) {
218                 Pixel p1 = pixels[i][j];
219                 Pixel p2 = otra.obtenerPixel(i, j);
220                 if (p1.red() != p2.red() || p1.green() != p2.green() || p1.blue() != p2.blue()) resultado
221             }
222         }
223     }
224 
225     return resultado;
226 }
227 
228 void Imagen::guardar(std::ostream& os) const {
229     os << alto() << " " << ancho() << " ";
230     int i = 0;
231     os << "[";
232     while (i < alto()) {
233         int j = 0;
234         while (j < ancho()) {
235             if(i != 0 || j != 0) os << ",";
236             pixels[i][j].guardar(os);
237 
238             j++;
239         }
240         i++;
241     }
242     os << "]";
243 }
244 
245 void Imagen::cargar(std::istream& is) {
246     int alto, ancho;
247     is >> alto;
248     is >> ancho;
249 
250     char charMolesto;
251     is >> charMolesto; //corchete o coma
252 
253     vector<vector<Pixel> > nueva_imagen;
254 
255     int i = 0;
256     while(i < alto) {
257         int j = 0;
258         vector<Pixel> fila;
259         while(j < ancho) {
260             Pixel este_pixel(0, 0, 0);
261             este_pixel.cargar(is);
262 
263             is >> charMolesto;
264 
265             fila.push_back(este_pixel);
```

```
266             j++;
267         }
268
269         nueva_imagen.push_back(fila);
270         i++;
271     }
272
273     pixels = nueva_imagen;
274 }
```

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      imagenes.pop_back();
99      votos.pop_back();
100 }
101
102
103 vector<Imagen> GaleriaImagenes::top10 () const {
104     int i = 0;
105     vector <Imagen> result;
106     while (i < 10 && imagenes.size() - i - 1 < imagenes.size()) {
107         result.push_back(imagenes[imagenes.size() - i - 1]);
108         i++;
109     }
110     return result;
111 }
112
113
114 vector<Imagen> dividir (const Imagen &img, int n, int m) {
115     vector<Imagen> resultado;
116
117     int cols  = img.ancho() / n;
118     int filas = img.alto() / m;
119
120     int i = 0;
121
122     while(i < img.alto()) {
123         int j = 0;
124         while(j < img.ancho()) {
125
126             Imagen esta_imagen(filas, cols);
127
128             int i_img = 0;
129             while (i_img < filas) {
130                 int j_img = 0;
131                 while(j_img < cols) {
132                     Pixel este_pixel = img.obtenerPixel(i + i_img, j + j_img);
133                     esta_imagen.modificarPixel(i_img, j_img, este_pixel);
134
```

```
135                     j_img++;
136                 }
137                 i_img++;
138             }
139             resultado.push_back(esta_imagen);
140             j += cols;
141         }
142         i += filas;
143     }
144 
145     return resultado;
146 }
147 
148 
149 
150 void GaleriaImagenes::dividirYAgregar(const Imagen &imagen, int n, int m) {
151     int alto = imagen.alto();
152     int ancho = imagen.ancho();
153 
154     if (alto % m == 0 && ancho % n == 0) {
155         vector<Imagen> dividida = dividir(imagen, n, m);
156 
157         int i = 0;
158         while(i < dividida.size()) {
159             this->agregarImagen(dividida[i]);
160         }
161     }
162 }
163 
164 void GaleriaImagenes::guardar(std::ostream& os) const {
165     os << "[";
166 
167     int i = 0;
168     while(i < imagenes.size()) {
169         if (i!=0) os << ",";
170         os << "(";
171         imagenes[i].guardar(os);
172         os << "," << votos[i] << ")";
173     }
174     os << "]" << endl;
175 }
176 
177 void GaleriaImagenes::cargar(std::istream& is) {
178     Imagen im(1, 1);
179     int numero_de_votos;
180 
181     vector<Imagen> imagenes_nuevas;
182     vector<int> votos_nuevos;
183 
184     char charMolesto; // parentesis, coma o chorchete
185     is >> charMolesto; // '['
186 
187     while(charMolesto != ']') {
188         is >> charMolesto; // '('
189         im.cargar(is);
190 
191         is >> charMolesto; // ','
192         is >> numero_de_votos;
193         is >> charMolesto; // ')'
194 
195         imagenes_nuevas.push_back(im);
196         votos_nuevos.push_back(numero_de_votos);
197 
198         is >> charMolesto; // ',' o ']' y se termina el ciclo
199     }
200     imagenes = imagenes_nuevas;
201     votos = votos_nuevos;
202 }
```

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             im.blur(k);
48
49             im.guardar(ofs);
50         }
51         else if(input == "2") {
52             int k;
53             string archivo_in;
54             string archivo_out;
55
56             cout << "Ingrese un k, el nombre del archivo de entrada y el de salida" << endl;
57             cin >> k >> archivo_in >> archivo_out;
58
59             ifstream ifs(archivo_in.c_str());
60             ofstream ofs(archivo_out.c_str());
61
62             Imagen im(1, 1);
63             im.cargar(ifs);
64
65             im.acuarela(k);
66
67             im.guardar(ofs);
68         }
69         else if(input == "3") {
70             string archivo_in;
71
72             cout << "Ingrese el nombre del archivo" << endl;
73             cin >> archivo_in;
74
75             ifstream ifs(archivo_in.c_str());
```

```
76              galeriaCargada.cargar(ifs);
77              archivo_galeria = archivo_in;
78          }
79          else if(input == "4") {
80              string archivo_in, archivo_in2;
81              int n, m;
82
83              cout << "Ingrese el nombre del archivo de la galeria, el nombre del archivo de la imagen, el n
84
85              cin >> archivo_in >> archivo_in2 >> n >> m;
86
87              ifstream ifs_galeria(archivo_in.c_str());
88              ifstream ifs_imagen(archivo_in2.c_str());
89
90              galeriaCargada.cargar(ifs_galeria);
91
92              Imagen im(1, 1);
93              im.cargar(ifs_imagen);
94
95              galeriaCargada.dividirYAgregar(im, n, m);
96          }
97          else if(input == "5") {
98              string archivo_in;
99
100             cout << "Ingrese el nombre del archivo de la imagen" << endl;
101
102             ifstream ifs(archivo_in.c_str());
103
104             Imagen im(1, 1);
105             im.cargar(ifs);
106
107             vector<pair<int, int> > resultado = im.posicionesMasOscuras();
108             int i = 0;
109             while(i < resultado.size()) {
110                 cout << "(" << resultado[i].first << "," << resultado[i].second << ")";
111             }
112             cout << endl;
113
114             galeriaCargada.agregarImagen(im);
115         }
116         else if(input == "6") {
117             string archivo_out;
118
119             cout << "Ingrese el nombre del archivo para el output" << endl;
120
121             ofstream ofs(archivo_out.c_str());
122
123             vector<Imagen> top = galeriaCargada.top10();
124
125             int i = 0;
126             ofs << "[";
127             while(i < top.size()) {
128                 if (i != 0) ofs << ",";
129                 top[i].guardar(ofs);
130             }
131             ofs << "]";
132         }
133         else if(input == "7") {
134             string archivo_out;
135
136             cout << "Ingrese el nombre del archivo para el output" << endl;
137
138             ofstream ofs(archivo_out.c_str());
139
140             Imagen resultado = galeriaCargada.laMasChiquitaConPuntoBlanco();
141             resultado.guardar(ofs);
142         }
143         else if(input == "8") {
144             string archivo_in;
145
146             cout << "Ingrese el nombre del archivo de la imagen" << endl;
147
148             ifstream ifs(archivo_in.c_str());
```

```
149
150             Imagen im(1, 1);
151             im.cargar(ifs);
152
153             galeriaCargada.agregarImagen(im);
154         }
155         else if(input == "9") {
156             string archivo_in;
157
158             cout << "Ingrese el nombre del archivo de la imagen" << endl;
159
160             ifstream ifs(archivo_in.c_str());
161
162             Imagen im(1, 1);
163             im.cargar(ifs);
164
165             galeriaCargada.votar(im);
166         }
167         else if(input == "10") {
168             galeriaCargada.eliminarMasVotada();
169         }
170         else if(input == "11") {
171             ofstream ofs(archivo_galeria.c_str());
172             galeriaCargada.guardar(ofs);
173         }
174         else {
175             if(input != "x") cout << "Opcion invalida, por favor seleccione una opcion valida (1-10, x)."
176         }
177
178     }
179
180     return 0;
181 }
```

4. Demostraciones