```cpp
#include <vector>
#include <fstream>
#include <iostream>
#include <ctime>
using namespace std;

#include "BmpImage.h"

#pragma region "konstruktør/dekonstruktør"
BmpImage::BmpImage()
{

}


BmpImage::~BmpImage()
{
    delete[] header;
}
#pragma endregion

#pragma region "hovedfunksjoner"
bool BmpImage::lesBitmapFil(string filename)
{
    fstream infile;
    infile.open(filename.c_str(), ios::binary | ios::in);
    if (infile.fail())
    {
        cout << "Feil ved aapning av fil\n";
        return false;
    }
    else
    {
        infile.seekg(10, ios::beg);
        infile.read((char*)&this->HEADERSIZE, 4);
        infile.seekg(10, ios::beg);
        infile.read((char*)&this->offset, 4);

        this->header = new unsigned char[this->HEADERSIZE];

        infile.seekg(0, ios::beg);
        infile.read((char*)this->header, this->HEADERSIZE);

        infile.seekg(18, ios::beg);
        infile.read((char*)&this->width, 4);
        infile.read((char*)&this->heigth, 4);

        infile.seekg(offset, ios::beg);
        this->stuffing = width % 4;

        for (unsigned int i = 0; i < this->heigth; i++)
        {
            vector<Pixel> rad;

            for (unsigned int j = 0; j < this->width; j++)
            {
```

```
57                  infile.read((char*)this->enPixel, 3);
58
59                  Pixel nyPixel((int)this->enPixel[2], (int)this->enPixel[1],
                      (int)this->enPixel[0]);
60                  rad.push_back(nyPixel);
61              }
62              int p = infile.tellp();
63
64              infile.seekg(p + stuffing, ios::beg);
65              this->pixelData.push_back(rad);
66          }
67          infile.close();
68
69          return true;
70      }
71  }
72
73  bool BmpImage::lagreBitmapFil(string filename)
74  {
75      fstream outfile;
76      outfile.open(filename.c_str(), ios::binary | ios::out);
77
78      if (outfile.fail())
79      {
80          cout << "Feilet aa aapne utfil\n";
81
82          return false;
83      }
84      else
85      {
86          unsigned char utdata[3];
87          unsigned char stuffBytes[3] = { 0, 0, 0 };
88
89          outfile.write((char*)this->header, this->HEADERSIZE);
90
91          outfile.seekp(offset, ios::beg);
92
93          for (int i = 0; i < this->heigth; i++)
94          {
95              vector<Pixel> rad = this->pixelData[i];
96              for (int j = 0; j < this->width; j++)
97              {
98                  Pixel *p = &(rad.at(j));
99                  utdata[0] = p->getB();
100                 utdata[1] = p->getG();
101                 utdata[2] = p->getR();
102                 outfile.write((char*)utdata, 3);
103             }
104             outfile.write((char*)stuffBytes, this->stuffing);
105         }
106         outfile.close();
107
108         return true;
109     }
110
111 }
```

```
112
113 bool BmpImage::setPixel(int x, int y, Pixel &ny_pixel)
114 {
115     this->pixelData[x][y] = ny_pixel;
116
117     return true;
118 }
119
120 void BmpImage::graaskala()
121 {
122     int grey = 0;
123     //gjennomløper alle rader:
124     for (int i = 0; i < this->heigth; i++)
125     {
126         //henter rad nr i:
127         vector<Pixel> *rad = &(this->pixelData[i]);
128         //gjennomløper alle pikslene i rad nr i:
129         for (int j = 0; j < this->width; j++)
130         {
131             Pixel *p = &(rad->at(j)); // henter referanse til piksel
132
133             grey = (p->getR() + p->getG() + p->getB()) / 3; // gråfarge basert ↵
                    på snitt av rgb fargen
134             //setter grå pikselfarge
135             p->edit(grey, grey, grey); //endrer piksel
136         }
137     }
138 }
139
140 void BmpImage::invert()
141 {
142     int red = 0, green = 0, blue = 0;
143
144     for (int i = 0; i < this->heigth; i++)
145     {
146         vector<Pixel> *rad = &(this->pixelData[i]);
147
148         for (int j = 0; j < this->width; j++)
149         {
150             Pixel *p = &(rad->at(j));
151
152             red = (255 - p->getR());
153             green = (255 - p->getG());
154             blue = (255 - p->getB());
155
156             p->edit(red, green, blue);
157         }
158     }
159 }
160 #pragma endregion
161
162 #pragma region "get funksjoner"
163 int BmpImage::getwidth()
164 {
165     return this->width;
166 }
```

```
167
168 int BmpImage::getheigth()
169 {
170     return this->heigth;
171 }
172
173 int BmpImage::getheadersize()
174 {
175     return this->HEADERSIZE;
176 }
177
178 int BmpImage::getoffset()
179 {
180     return this->offset;
181 }
182
183 int BmpImage::getstuffing()
184 {
185     return this->stuffing;
186 }
187 #pragma endregion
188
189 #pragma region "oppgave 2b funksjoner"
190 void BmpImage::rammInnBilde(Pixel &Farge, int rammeBredde)
191 {
192     for (int i = 0; i < rammeBredde; i++)
193     {
194         for (int j = 0; j < this->width; j++)
195         {
196             this->pixelData[i][j] = Farge;
197             this->pixelData[(this->heigth) - i - 1][j] = Farge;
198         }
199         for (int j = 0; j < this->heigth; j++)
200         {
201             this->pixelData[j][i] = Farge;
202             this->pixelData[j][(this->width) - i - 1] = Farge;
203         }
204     }
205 }
206
207 int BmpImage::antallPiksler(Pixel &Farge)
208 {
209     int R = 0, G = 0, B = 0, pix = 0;
210     int RR = Farge.getR(), GG = Farge.getG(), BB = Farge.getB();
211     for (int i = 0; i < this->heigth; i++)
212     {
213         vector<Pixel> *rad = &(this->pixelData[i]);
214         for (int j = 0; j < width; j++)
215         {
216             Pixel *p = &(rad->at(j));
217             int R = p->getR();
218             int G = p->getG();
219             int B = p->getB();
220
221             if (R == RR && G == GG && B == BB)
222             {
```

```
223                 pix++;
224             }
225         }
226     }
227
228
229     return pix;
230 }
231
232 void BmpImage::tegnTilfeldigePunkter(int antall, Pixel &Farge)
233 {
234     srand(time(0));
235     for (int i = 0; i < antall; i++)
236     {
237         int x = rand() % (this->heigth - 1);
238         int y = rand() % (this->width - 1);
239
240         this->pixelData[x][y] = Farge;
241     }
242 }
243
244 void BmpImage::flip()
245 {
246     for (int i = 0; i < ((this->heigth)/2); i++)
247     {
248         for (int j = 0; j < this->width; j++)
249         {
250             Pixel midlertidig =  pixelData[i][j];
251             pixelData[i][j] = pixelData[(this->heigth) - i - 1][j];
252             pixelData[(this->heigth) - i - 1][j] = midlertidig;
253
254         }
255     }
256 }
257
258 void BmpImage::mirror()
259 {
260     for (int i = 0; i < ((this->width) / 2); i++)
261     {
262         for (int j = 0; j < this->heigth; j++)
263         {
264             Pixel midlertidig = pixelData[j][i];
265             pixelData[j][i] = pixelData[j][(this->width) - i - 1];
266             pixelData[j][(this->width) - i - 1] = midlertidig;
267
268         }
269     }
270 }
271 #pragma endregion
```