2. Image Klasse

Compilieren

Größere Projekte bestehen sinnvollerweise aus mehreren Dateien. In diesem Fall müssen dem Compiler alle Source-Dateien angegeben werden, also z.B.
`g++ main.cpp image.cpp -o output.exe`

Erstellen einer Image Klasse

Implementieren Sie eine Klasse mit dem folgenden Interface (Datei `image.h`), d.h. implementieren Sie die fehlenden Funktionen `Init`, `Load` und `save` in der Datei `image.cpp`.

```
1  class Image
2  {
3    public:  enum TYPE{UNDEF,BINARY,GRAY,RGB};
4    private: TYPE type = TYPE::UNDEF;
5    private: string fileName = "";
6    private: int width = 0;
7    private: int height = 0;
8    public:  unsigned char* Data = 0;
9
10 public:
11
12   /** Standard Constructor: do nothing */
13   Image() { }
14
15   /** Constructor: Sets Width, Height and Type and allocates memory. */
16   Image(int w, int h, TYPE t) { Init(w,h,t); }
17
18   /** Constructor: Create and load an image with the given filename*/
19   Image(string name) { Load(name); }
20
21   /** Destructor */
22   ~Image() {
23     if(Data)
24     {delete Data;}
25     Data = 0;
26   }
27
28   /** Sets Width, Height and Type and allocates memory. */
29   void Init(int w, int h, TYPE t);
30
31   /** Loads an image with the given filename */
32   void Load(string filename);
33
34   /** Saves image under its fileName */
35   void Save(){
36     switch(type){
37       case BINARY: cerr<<"ERROR! Unable to save image of type BINARY!"<<endl;
38       case GRAY:   save(fileName, "P5", width, height, 255, Data); break;
39       case RGB:    save(fileName, "P6", width, height, 255, Data); break;
40       case UNDEF:  cerr<<"ERROR! Unable to save image of type UNDEF!"<<endl;
```

```
41      }
42    }
43
44    /** Sets fileName and saves image under this fileName */
45    void SaveAs(string name) {
46      fileName=name;
47      Save();
48    }
49
50    int Width() const { return width; }  // returns number of columns
51    int Height()const { return height; } // returns number of rows
52    TYPE Type() const { return type; }   // returns image type, e.g. RGB
53    int Count() const { return width*height; } // returns number of pixels
54
55  private:
56    /** name: filename without extension
57    mode: magic number, describes the image format, must be P1,P2,...P6
58    w:    width of image, number of columns
59    h:    height of image, number of rows
60    max:  max gray value or max value of rgb channel, normally 255
61    data: array of pixel data */
62    void save(string name, string mode, int w, int h, int max,
63              unsigned char* data);
64
65  };
```

Hinweise:

- Die Klasse enthält bewusst keine Zugriffsfunktionen auf einzelne Pixelwerte der Art `unsigned char getValue(int x, int y);`
  Überlegen Sie, warum eine solche Funktion problematisch ist
  (a) in Bezug auf das Laufzeitverhalten und
  (b) in Bezug auf die Implementierung für RGB- und Graubilder.

- Für die `Load`-Funktion genügt es zunächst, nur das Laden von Grauwertbildern (P5) zu implementieren.

- Die public-Funktionen `Save` und `SaveAs` basieren auf der privaten `save`-Funktion, die Sie bereits in der ersten Übung programmiert haben.

Zum Testen können sie z.B. folgende `main`-Funktion verwenden:

```
1  #include <iostream> // cin, cout
2  #include "image.h"
3  using namespace std;
4
5  int main (int argc, char *argv[]){ // image class
6    clog << "02 Image Class: Create and save an image, load and save it again"
7    Image* img1 = new Image(256,100,Image::GRAY);
8    int i = 0;
9    for(int y=0;y<100;++y)
10     for(int x=0;x<256;++x)
11       { img1->Data[i++] = x; }
12   img1->SaveAs("02_image");
13   delete img1;
14
15   Image* img2 = new Image("02_image.pgm");
16   img2->SaveAs("02_test");
17   delete img2;
18   return 0;
19 }
```

Optional

- Ergänzen Sie Ihre Klasse um einen Copy-Constructor.

```
1 /** Copy Constructor: Creates a deep copy of an image */
2 Image(const Image &orig);
```

  Dieser Constructor bekommt ein Bild übergeben und kopiert den gesamten Bildinhalt inkl. Data-Array in das neue Bild. Nutzen Sie zum Kopieren der Pixelwerte den Befehl `memcpy` in `<string.h>`, das ist schneller als eine `for`-Schleife.

- Erweitern Sie die Funktion `Load` um die Möglichkeit, Farbbilder (P6) zu laden
- Erweitern Sie die Funktion `Load` um die Möglichkeit, binär gespeicherte Grau- und Farbbilder (P2 und P3) zu laden