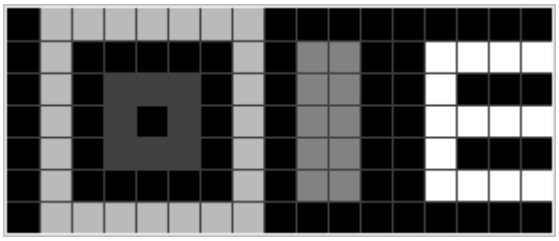
1 PGM-Bilder

In dieser Aufgabe lernen Sie die Verwendung von File-Streams.

In der nachfolgenden Abbildung sehen Sie ein Beispiel für eine PGM Bilddatei im ASCII-Format:

image_ascii.pgm

```
P2
# comment
17 7
63
0 46 46 46 46 46 46 46  0  0  0  0  0  0  0  0  0
0 46  0  0  0  0  0 46  0 32 32  0  0 63 63 63 63
0 46  0 16 16 16  0 46  0 32 32  0  0 63  0  0  0
0 46  0 16  0 16  0 46  0 32 32  0  0 63 63 63 63
0 46  0 16 16 16  0 46  0 32 32  0  0 63  0  0  0
0 46  0  0  0  0  0 46  0 32 32  0  0 63 63 63 63
0 46 46 46 46 46 46 46  0  0  0  0  0  0  0  0  0
```



PGM-Dateien können direkt in CLion betrachtet werden. Wenn Sie sie bearbeiten möchten, ändern Sie die Dateiendung auf `.txt`.

Die Datei besteht aus zwei Teilen: im oberen Teil ist der Bild-Header aufgeführt und im unteren Teil die Bildmatrix.

Der Bild-Header ist bei PGM immer im ASCII-Format, unabhängig davon ob es sich um PGM-ASCII oder PGM-Binary handelt. Nur die Bildmatrix ist bei PGM-Binary im Binärformat abgespeichert. Dadurch braucht das PGM-Binary üblicherweise viel weniger Speicherplatz und kann schneller gelesen und geschrieben werden.

Der Bild-Header muss ganz zu Beginn den Formatbezeichner P2 (für PGM-ASCII) bzw. P5 (für PGM-Binary) enthalten. Header-Zeilen die mit einem # beginnen werden als Kommentare überlesen. Nach dem Formatbezeichner müssen die Bildbreite gefolgt von der Bildhöhe in Pixel stehen. Die letzte Header-Information enthält den positiven Maximalwert, welcher in der Bildmatrix vorkommen darf.

Nach jeder Zeile steht ein `"\n"` Charakter. Auch nach der letzten Zeile.

Die genaue Spezifikation finden Sie auf sourceforge.net.

1.1 Aufgabe

Implementieren Sie die Klasse `PGM`, welche eine Bilddatei im Format PGM-ASCII einlesen und das gleiche Bild im Format PGM-Binary abspeichern kann. Verwenden Sie diese Klasse in einer eigenen App. Folgend ein Vorschlag für das Interface dieser Klasse:

pgm.h

```
1  #pragma once
2
3  #include <cstddef>
4
5  #include <vector>
6  #include <string>
7
8  class PGM {
9   public:
10   bool ReadASCII(const std::string& filename);
11   bool WriteBinary(const std::string& filename);
12
13  private:
14   size_t width_;
15   size_t height_;
16   int32_t max_value_;
17   std::vector<uint8_t> data_;
18 };
```