# 1. Bilder speichern im PNM-Format



## Installation

(a) Installieren Sie sich folgende Tools:

- einen C++-Compiler, z.B. *g++*,
- einen Editor mit Syntax-Highlighting, z.B. *notepad++* und
- das kostenfreie Bildbearbeitungstool *Gimp*.

(b) Schreiben Sie ein einfaches „Hello World"-Programm in C++.

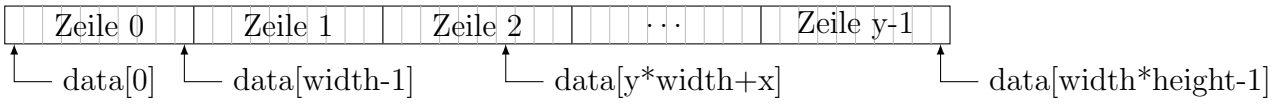
(c) Compilieren und testen Sie Ihr Programm.

```
g++ main.cpp -o output.exe
```

---

## Allokieren von Speicher für Bilder

Natürlich gibt es unzählige Möglichkeiten, den notwendigen Speicherplatz für Bilder zu allokieren, z.B. als eindimensionales Array (sehr effizient):




```
1 int width = ...
2 int height = ...
3 unsigned char* data = new unsigned char [width*height];
```

Der Datenzugriff erfolgt dann mit

```
1 for(int y=0;y<height;++y)
2   for(int x=0;x<width;++x)
3     { data[y*width+x] = ... }
```

oder

```
1 for(int i=0;i<width*height;++i)
2   { data[i] = ... }
```

---

## Das PNM-Format

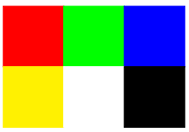
Das Portable-Pixmap-Format wurde um 1980 entwickelt, um Bilder als einfacher ASCII Text z.B. per E-Mail zu übermitteln. Eine PNM-Datei beginnt mit einem Identifier (Magic Number P1...P6), der Bildgröße (Breite und Höhe), dem Maximalwert und einer linearen Folge der Bilddaten (unkomprimiert). Abhängig von der Magic Number werden die Pixeldaten als ASCII-Folge oder binär gespeichert.

| Typ | ASCII | Binär | Extension | Wertebereich |
|---|---|---|---|---|
| Portable BitMap | P1 | P4 | .pbm | 0–1 (schwarz & weiß) |
| Portable GrayMap | P2 | P5 | .pgm | 0–255 (grauwerte) |
| Portable PixMap | P3 | P6 | .ppm | 0–255 (RGB) |

Magic Number, Breite, Höhe und Maximalwert werden ASCII-Dezimal angegeben, jeweils gefolgt von einem Whitespace (\n, \r, \t, space). Beim binären PBM-Format entfällt die Angabe des Maximalwertes.

### Beispiel

| | |
|---|---|
| RGB, ASCII | `P3` |
| Breite & Höhe | `3 2` |
| Maximalwert | `255` |
| Bilddaten RGB-Tupel | `255 0 0 0 255 0 0 0 255`<br>`255 255 0 255 255 255 0 0 0` |

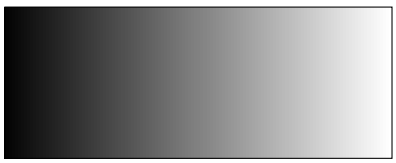


Siehe auch: https://en.wikipedia.org/wiki/Netpbm_format

---

### Speichern von Bildern im PNM-Format

(a) Allokieren Sie *dynamisch* Speicher (mit dem Befehlt `new`) für ein 8-Bit-Grauwertbild der Größe 256×100 und weisen Sie jedem Pixel als Grauwert seine eigene x-Koordinate zu, es entsteht also ein Grauwertverlauf von links schwarz nach rechts weiß.



100 Zeilen

256 Spalten

```
1 unsigned char* pixel = new ...
2 for(...){
3   pixel[...] = ... ;
4 }
5 delete pixel; // nicht vergessen: Speicher freigeben ;-)
```

(b) Schreiben Sie eine Funktion

```
1 void Save(string name, string mode, int w, int h, int max,
2   unsigned char* data);
```

zum Speichern von Bildern im PNM-Format und speichern Sie Ihr Bild aus (a). Sie können es dann mit einem beliebigen Bildbearbeitungsprogramm betrachten, z.B. Gimp oder Irfanview. Implementieren Sie mindestens die Formate P5 und P6 (Grau und RGB binär gespeichert).

Tipp: Nutzen Sie die Funktion `write(const char* s, streamsize n)` eines `ofstream` (≙output file stream, #include <fstream>),
siehe http://www.cplusplus.com/reference/fstream/ofstream/

---

### Optional

1. Implementieren Sie auch das Speichern von Grauwert- und Farbbildern als ASCII-Datei (Formate P2 und P3).
2. Implementieren Sie auch das Speichern von binären Bildern (Formate P1 und P4).