Klasse pfc::bitmap

Kurzbeschreibung für die Version 1.02

Peter Kulczycki
peter.kulczycki<AT>fh-hagenberg.at

Department of Medical Informatics and Bioinformatics
University of Applied Sciences Upper Austria
Softwarepark 11, 4232 Hagenberg, Austria

Version 1·09·1006 – 14. Oktober 2015

Inhalt

1. Interface
2. Beispiel
3. Anmerkungen und Einschränkungen
4. Konzeptbilder

Interface / Konstruktoren

```
1   bitmap ()
2   bitmap (bitmap const & src)
3   bitmap (bitmap && src)
4
5   bitmap (size_t const width, size_t const height)
6   bitmap (size_t const width, size_t const height,
7           pfc::byte_t * p_image)
8
9   bitmap (char const * const p_filename)
10  bitmap (string const & filename)
11
12  bitmap (istream & in)
```

Der Default-Konstruktor erzeugt ein Bitmap der Größe 0×0 Pixel^2. Mit den Methoden `create`, `from_file` oder `from_stream` bzw. dem Zuweisungsoperator kann dessen Größe verändert werden. Die Parameter `width` und `height` geben die Größe (in Pixel) des zu konstruierenden Bitmaps an. Das mit `width` und `height` konstruierte Bitmap ist weiß. Mit `p_image` kann ein extern allokierter Speicher für die Bilddaten vorgegeben werden.

Interface / Initialisierung

```
1  void clear  ()
2  void create (size_t const width, size_t const height)
3  void create (size_t const width, size_t const height,
4               pfc::byte_t * p_image)
```

Die Methode `clear` ist ein Synonym für den Methodenaufruf `create(0,0)`. Mit der Methode `create` kann die Größe eines Bitmaps verändert werden. Die Parameter `width` und `height` geben die neue Größe eines Bitmaps (in Pixel) an. Das so veränderte Bitmap ist weiß. (Auch dann, wenn die neuen Dimensionen gleich den alten sind.) Mit `p_image` kann ein extern allokierter Speicher für die Bilddaten vorgegeben werden.

Interface / Operatoren und Getter

```
1  bitmap & operator = (bitmap const & rhs)
2  bitmap & operator = (bitmap && rhs)
```

```
1  size_t get_height       () const
2  size_t get_width        () const
3  size_t get_width_bytes  () const
4
5  size_t get_image_size () const
6  size_t get_num_bytes  () const
7  size_t get_num_pixels () const
8
9  pfc::byte_t *  get_image  ()
10 pfc::RGB_3_t * get_pixels ()
```

Die Methoden `get_image_size` bzw. `get_num_bytes` liefern die Anzahl der Bytes, die die geladenen Bilddaten im Speicher benötigen. Die Methode `get_num_pixels` liefert die Anzahl der Pixels, aus denen die geladenen Bilddaten bestehen. Die Methoden `get_image` bzw. `get_pixels` liefern Zeiger auf die Bilddaten (siehe dazu die Konzeptbilder).

Interface / Lesen und schreiben

```
1  bool from_file   (char const * const p_filename)
2  bool from_file   (string const & filename)
3  bool from_stream (istream & in)
```

```
1  bool to_file   (char const * const p_filename) const
2  bool to_file   (string const & filename) const
3  bool to_stream (ostream & out) const
```

Diese Methoden lesen bzw. schreiben ein Bitmap. Es können dabei Dateinamen (Parameter `p_filename` und `filename`) oder Dateiströme (Parameter `in` und `out`) angegeben werden. Dateinamen müssen die Erweiterung „bmp“ besitzen. Alle Funktionen liefern dann `true`, wenn die Lese- bzw. Schreiboperationen auf Stream-Ebene erfolgreich durchgeführt wurden.

Beispiel

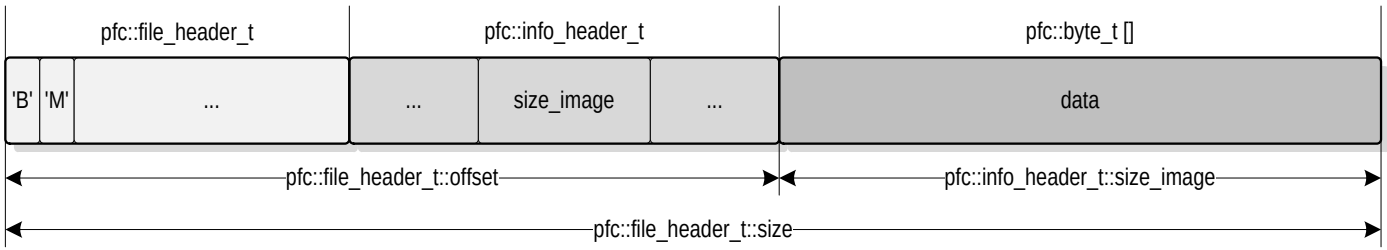
```
1  #include "pfc-bitmap.hpp"
2
3  int main () {
4      pfc::bitmap b1 (300, 100);
5      pfc::bitmap b2;
6      pfc::bitmap b3;
7
8      b2 = b1;
9
10     b3.from_file ("erythrocytes.bmp");
11
12     pfc::swap (b2, b3);
13
14     b1.to_file ("output-1.bmp");
15     b2.to_file ("output-2.bmp");
16     b3.to_file ("output-3.bmp");
17 }
```

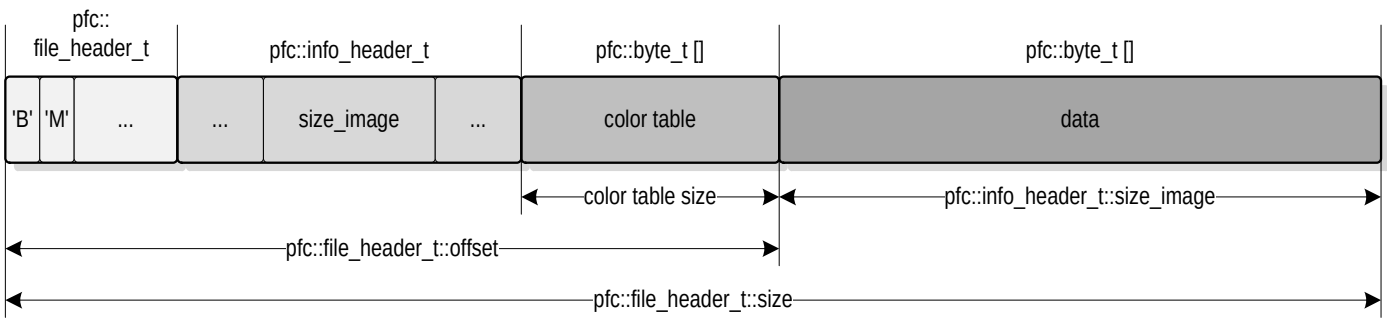
Anmerkungen und Einschränkungen

Derzeit (Version 1.02) gelten die folgenden Einschränkungen:

1. Die Klasse `pfc::bitmap` ist mit MS Visual-Studio 2010 bzw. 2012 sowie gcc 4.6 bzw. 4.7 unter Windows und Linux übersetzbar.
2. Im Namensraum `pfc` werden die folgenden Typen definiert: `bitmap`, `byte_t`, `dword_t`, `long_t`, `RGB_3_t` und `word_t`.
3. Im Namensraum `pfc` werden die folgenden Funktionen definiert: `abs`, `ccstr_empty`, `ceil_div`, `is_negative`, `is_zero`, `mem_copy`, `mem_copy_ptr`, `mem_reset`, `mem_set`, `mem_set_ptr`, `read`, `read_ptr`, `reset`, `swap`, `write` und `write_ptr`.
4. Es werden nur 24-Bit-Bitmaps unterstützt (lesend und schreibend).
5. Die dreifache Breite eines Bitmaps (in Pixel) muss durch vier teilbar sein (eine Scanline muss an einer 32-Bit-Grenze enden, siehe die Konzeptbilder).

Konzeptbilder





Konzeptbilder