4. Praktikum zum Programmieren II

Ablauf:

a) Bearbeiten Sie die Aufgaben in Teams zu zwei Personen. Sie erzeugen aber eine Lösung pro Team.
b) Achten Sie bei der Teamzusammenstellung darauf, dass **Beide** an der Lösung der Aufgabe tatsächlich arbeiten und sich gegenseitig kontrollieren und ergänzen. Nicht gewünscht ist es, dass eine Person die Aufgabe(n) erledigt und die zweite Person wohlwollend aber passiv danebensitzt.
c) Wenn Sie die Aufgaben erledigt haben, geben Sie dem Praktikumsleiter Nachricht. Die Arbeit wird dann **im Termin** besprochen und abgenommen.
d) Lassen Sie sich mit der Fertigstellung nicht bis zum Ende des Praktikumstermins Zeit, da Sie ggf. noch Nacharbeiten an den Aufgaben durchführen müssen. Sie können das Praktikum erst verlassen, wenn die Aufgaben abgenommen sind und gegen Ende des Termins wird es erfahrungsgemäß zeitlich eng werden.
e) Bevor Sie den Praktikumsleiter rufen, machen Sie Qualitätstests anhand folgender Checkliste:
    i) Jedes Programm ist kommentiert.
    ii) Jede Datei enthält im Kopf einen Kommentar. Als Autor des Programmes führen Sie alle Team-Mitglieder auf.
    iii) Der Programmcode lässt sich übersetzen und zwar fehlerfrei und **ohne Warnings**. Prüfen sie das mit dem Befehl „Projektmappe neu erstellen“.
    iv) Sie haben das Programm ausprobiert und auch einmal im Debugger Schritt-für-Schritt verfolgt?
    v) Sie halten die Codierungsrichtlinien ein?

Einstimmung ins Thema

In diesem Praktikum soll geübt werden:

1. Umfang mit File I/O
2. Dynamische Speicherverwaltung und Umgang mit Listen
3. Datenumwandlung

Aufgabe 4.1 – Viewer für BMP (Bitmap) Dateien.

Einführung:
Sie haben eine Liste von schönen Fotos und möchten diese anschauen. Die Fotos liegen alle im Format BMP (Bitmap) vor. Schreiben Sie dazu einen Viewer, mit dessen Hilfe Sie BMP-Bilder auf dem Bildschirm anzeigen können[1].

Jede Bitmap Datei hat zunächst einen sogenannten „header", d.h. eine gewisse Anzahl Bytes, die Informationen über das Bild an sich enthalten, z.B. die Breite und Höhe in Pixeln, die Farbtiefe usw.

Die entsprechenden Datenstrukturen wurden schon eingerichtet, in „`main.h`" finden Sie die Typen dazu:

- `TBitmapFileHeader` und
- `TBitmapInfoHeader`

Wir beschränken uns in dieser Aufgabe auf eine Teilmenge, d.h.

Header	**Feld**	**Was tun wir in dieser Aufgabe damit**
`TBitmapFileHeader`	`kKennung`	Dieser Wert wird geprüft. Wenn er nicht vorliegt, ist es keine BMP Datei und wir bearbeiten sie nicht weiter und zeigen sie auch nicht an.
	`dateiGroesseInBytes`	Diesen Wert ignorieren wir
	`reserved1zero`	Wir prüfen, ob der Wert 0 ist. Wenn nicht 0, dann ist es keine BMP Datei und wir zeigen sie nicht an.
	`offsetInBytes`	Diesen Wert benötigen wir, damit wir wissen, wo in der Datei die Bilddaten anfangen. Der Wert zählt ab Anfang der Datei von 0 … an.

`TBitmapInfoHeader`	`biSize`	Wir prüfen, ob der Wert == sizeof(TBitmapInfoHeader) ist. Wenn nicht, dann ist es keine BMP Datei oder eine andere Version und wir zeigen sie nicht an.
	`biWidth`	Diesen Wert benötigen wir, er sagt uns wie viele Pixel eine Zeile enthält.
	`biHeight`	Diesen Wert benötigen wir, abs(biHeight) sagt uns, wie viele Zeilen das Bild hat. Wenn der Wert negativ ist, dann „steht das Bild auf dem Kopf" und wir müssen es umdrehen (siehe 4.3)
	`biPlanes`	Wir unterstützen nur Bilder mit einer Ebene (plane), d.h. wenn dieser Wert != 1 ist, zeigen wir das Bild nicht an.
	`biBitCount`	Wir unterstützen in dieser Version nur echte Farbbilder, d.h. dieser Wert muss == 24 sein. Andere Bilder unterstützen wir nicht.
	`biCompression`	Wir unterstützen nur Bilder ohne Compression,

[1] Eine Übersicht über das Dateiformat BMP finden Sie z.B. bei Wikipedia.
https://de.wikipedia.org/wiki/Windows_Bitmap

		d.h. dieser Wert muss == 0 sein. Andere Bilder unterstützen wir nicht.
	`biSizeImage`	Diesen Wert ignorieren wir.
	`biXPelsPerMeter`	Diesen Wert ignorieren wir.
	`biYPelsPerMeter`	Diesen Wert ignorieren wir.
	`biClrUsed`	Diesen Wert ignorieren wir.
	`biClrImportant`	Diesen Wert ignorieren wir.

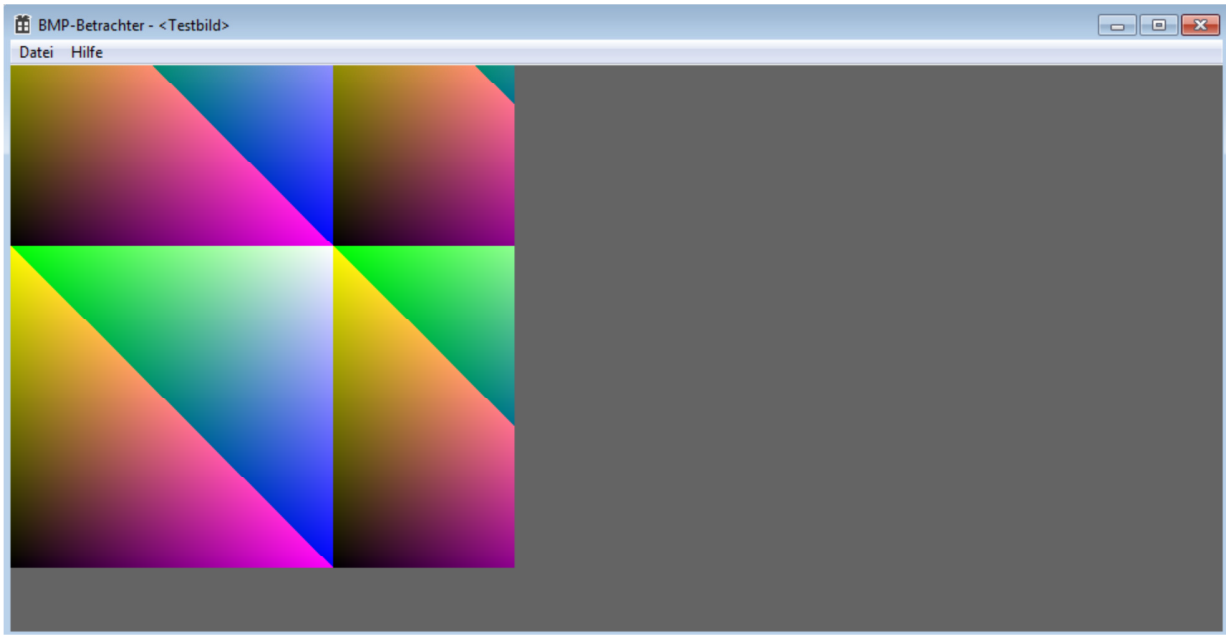
4.1.1) Vorbeitungen:

a) Laden Sie PR2-4.1(Framework).zip herunter und packen Sie die Dateien aus.

b) Legen Sie die Dateien in Ihrem `source/repros/…` - Verzeichnis ab, da wo Sie auch Ihre anderen Projekte abgelegt haben.

c) Starten Sie Visual Studio und öffnen Sie das Projekt `PR2-4.1.sln`

d) Übersetzen Sie das Projekt und lassen Sie es laufen. Sie sehen das Windows Fenster mit einem Testbild. Das sollte so aussehen:



Wenn Sie das Framework (ohne Änderungen) nicht übersetzen können, prüfen Sie, ob Sie Ihr Visual Studio richtig installiert haben.

e) Als Hilfestellung finden Sie auch das fertig kompilierte Programm `PR2-4.1.exe` in der Aufgaben Sammlung. Dieses können Sie zum Vergleich immer mal wieder heranziehen und sehen, wie ihr Programm arbeiten **sollte**.

f) In der Aufgabensammlung ist eine Beispielmenge von BMP Bildern, mit denen Sie testen können. Sie erfüllen alle die oben genannten Eigenschaften, d.h. jedes davon soll sich mit dem Bildbetrachter öffnen lassen, auch wenn einige je nach Monitorauflösung möglicherweise nicht vollständig darstellbar sind (Dateien vorher aus dem ZIP auspacken).

g) Das Projekt kommt mit **zwei C-Dateien**.

- **main.c** – Diese Datei sollten Sie **nicht** verändern, jedenfalls so lange nicht, bis Sie die Aufgabe vollständig gelöst haben. Danach können Sie natürlich mit den Funktionen herumspielen und eigene Dinge ausprobieren.

Wundern Sie sich nicht, dass es hier keine main() Funktion gibt, diese heißt in der Windows-API `wWinMain()`.

- **Aufgabe4.1.c** – Hier tragen Sie Ihre Änderungen und Programme ein.

4.1.2) Implementation der Funktion „ladeDatei“

Implementieren Sie nun die Funktion `ladeDatei(const char *dateiName, TBild *pBild)`.

Lösungsskizze:

a) Öffnen Sie die Datei „dateiName“ als Binärdatei zum Lesen.
b) Löschen Sie das aktuelle Bild mit der entsprechenden Funktion.
c) Lesen Sie den Dateiheader aus und prüfen Sie, ob es sich um eine im Sinne dieser Aufgabenstellung gültige BMP-Datei handelt.
d) Positionieren Sie den File-Pointer an die Stelle, an der die Bilddaten beginnen.
e) Berechnen Sie, wie viele Bytes eine Bildzeile in der BMP Datei lang ist.
   Dabei beachten:
   - Jeder Pixel benötigt genau drei Bytes.
   - Das Ende einer Zeile ist so lange mit 0-Bytes aufgefüllt, bis dass die Anzahl der Bytes durch 4 teilbar ist, d.h. die nächste Zeile fängt immer an einer durch 4 teilbaren Adresse an.
f) Berechnen Sie, wie viele Zeilen das Bild enthält, beachten Sie die Angaben zu `biHeight` bezüglich des Vorzeichens.
g) Erzeugen Sie dynamisch einen Zwischenpuffer, der eine ganze Zeile aus der Datei aufnehmen kann.
h) Berechnen Sie, wie viele Bytes eine Zeile im TBild benötigt. Dieser Wert ist anders als der vorige des Zwischenpuffers! Die Pixel im TBild sind alle vom Typ COLORREF mit sizeof(COLORREF) == 4.
i) Erzeugen Sie eine verkettete Liste von Zeilen (TBmpZeilenElement) und erzeugen Sie für jede Zeile mit malloc() dynamisch einen Speicherbereich, der eine solches Zeilenelement aufnehmen kann.
j) Lesen Sie zeilenweise die Bilddaten aus der Datei und legen Sie sie zunächst im Zwischenpuffer ab.
k) Übertragen Sie die Pixel aus dem Zwischenpuffer in die TBild-Zeile.
   Dabei beachten:
   - Die Pixel werden durch Tupel von drei Bytes dargestellt. Ein Byte liefert den rot-Anteil, eines den grün-Anteil und eines den blau-Anteil. Hohe Werte bedeuten hohe Helligkeit, d.h. das Tupel(0,0,0) bedeutet schwarz und das Tupel(255,255,255) bedeutet weiß.
   - In der BMP Datei sind die Werte blau/grün/rot sortiert, d.h. erst kommt immer der blau-Anteil, dann der grüne, dann der rote.
   - Im Windows-System verwenden Sie für die Umwandlung das Makro „RGB(…), wobei hier die Reihenfolge rot/grün/blau ist. Mit anderen Worten:
     Die Pixel-Zeilen für Windows bestehen aus COLORREF-Werten, die Sie wie folgt aus den RGB Werten erzeugen können:

```
…
int rot = …; // aus dem Zwischenpuffer
int gruen = …;
int blau = …;

COLORREF cr = RGB(rot, gruen, blau);
…
```

l) Am Ende sorgen Sie dafür, dass alle Felder des TBild mit den richtigen Werten ausgefüllt sind und return mit TRUE;
   Wenn Sie alles richtig gemacht haben, wir das Bild dann angezeigt.
m) Wenn Sie während des Einlesens einen Fehler entdecken, dann return FALSE.

n) Sorgen Sie im Fehlerfall dafür, dass alle nicht mehr benötigten dynamischen Speicherbereiche wieder freigegeben werden und alle Dateien auf jeden Fall geschlossen werden !

4.1.3) Implementation der Funktion „spiegelnBild“
Implementieren Sie die Funktion spiegelnBild(). Sie „stellt das Bild auf den Kopf“.
Dazu wird die bestehende Liste der Zeilen Z1->Z2->Z3->Z4->…->Zn (pBild->pBitmap zeigt auf Z1) umgedreht, d.h. am Ende sieht die Liste so aus: Zn->…->Z4->Z3->Z2->Z1 und pBild->pBitmap zeigt auf Zn.

Prüfen Sie in ladeDatei() den Wert von `TBitmapInfoHeader.biHeight`.
Wenn er negativ ist, so heißt das, dass Sie das Bild spiegeln müssen; machen Sie das mit spiegelnBild().
Zum Ausprobieren können Sie die Datei „Rathaus-Nacht(Flip).bmp“ öffnen.

Kommentare:

- Das Pixel-weise Arbeiten ist wie man leicht erkennt nicht sonderlich performant. Es gibt natürlich schnellere Methoden auch unter Windows, aber hier geht es darum, den Umgang mit Dateien und dynamischem Speicher kennenzulernen.
- Diese Darstellung von Fenstern und Bildern unter Windows ist recht alt, es handelt sich um das sogenannte Win32-API. Es gibt mittlerweile modernere Varianten, z.B. openGL, Direct-X oder .NET
  Diese Toolsets bzw. APIs eignen sich aber nicht so gut für die C-Programmierung und ich wollte Ihnen das Vergnügen einer „echten“ Windows-Anwendung nicht vorenthalten, so ein Text-Bildschirm ist auf Dauer auch langweilig.
- **Bildnachweis**:
  Die Bilder entstammen entweder meiner Kamera oder wurden von der NASA (nasa.org) bereitgestellt oder entstammen dem public domain Pool von google.

Aufgabe 4.2

Diese Aufgabe ist von besonderer Wichtigkeit.
Sie werden während der Prüfung keine Gelegenheit haben, dieses zu korrigieren. Daher müssen diese Vorbereitungen bzw. Überprüfungen unbedingt bis zum Beginn der Prüfung abschlossen sein.

a) Loggen Sie sich auf dem Laborrechner ein. Verwenden Sie **NICHT Ihren privaten** Laptop
b) Bereiten Sie Ihren Arbeitsplatz im PC-Pool für das siebte Praktikum so vor, wie Sie es für die Laborprüfung machen würden. Sie sollten demnach alle relevanten Unterlagen, wie z.B. Ihre Programme und die Vorlesungsfolien, auf Ihrem Netzlaufwerk abspeichern. Zudem bringen Sie bitte alle weiteren Unterlagen wie Bücher, Ausdrucke, etc. mit, die Sie ebenfalls in der Laborprüfung verwenden möchten. Schauen Sie sich zur Vorbereitung das Skript und Ihre Programme nochmals ausführlich an.