Beispielangaben zu
OOP (Objekt-Orientierte Programmierung)

SS 2009

Beispiel 2

Relevante Themen:

- Untertyp-Beziehungen
- Zusicherungen
- Fehlerbehandlung

Design und Implementierung:
Schreiben Sie in C++ ein skriptgesteuertes Bildbearbeitungsprogramm imgsynth2. Wie der Name schon andeutet, handelt es sich hier um eine Erweiterung des Programms imgsynth der ersten Beispielangabe.

Die Synopsis des Programms ist unverändert: imgsynth2 -i <scriptfile>
Die Änderungen betreffen eine Erweiterung der zu unterstützenden Befehle und zu unterstützenden Bildformate.

So sollen zusätzlich zu den Befehlen read(),fillrect() und write() noch folgende Befehle zusätzlich unterstützt werden:

invert() … Invertiert das Bild Invertieren heißt, für jedes Pixel und allen Farbkomponenten (R,G,B) der aktuelle Wert vom Maximalwert subtrahiert wird:

$$val_{neu} = MAXVAL - val_{alt}$$

brightness(faktor) … Ändert die Helligkeit der Pixel. Für jedes Pixel wird jede Farbkomponente folgendermaßen aktualisiert:

$$val_{neu} = \min (MAXVAL, val_{alt} * faktor)$$

mirror_y() … Spiegelt das Bild um die Y-Achse (horizontales Spiegeln)

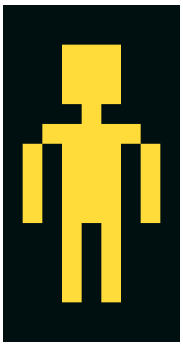
mirror_x() … Spiegelt das Bild um die X-Achse (vertikales Spiegeln)

Weiters sollen zusätzliche Bildformate unterstützt werden.
Zum einen soll die Unterstützung des „Windows Bitmap“ Formates der ersten Beispielangabe (Typ „BMP“) um ein weiteres Pixelformat erweitert werden. Mit dem Eintrag Pixelformat=16 soll das Format B5G5R5 unterstützt werden, welches fünf Bits für die blaue, fünf Bits für die grüne und fünf Bits für die rote Farbkomponente benutzt.

Damit man auf die 16 Bit kommt, wird die Farbkomponente Blau mit sechs Bits gespeichert, jedoch nur die niederwertigsten fünf Bits werden verwendet.

Weiters soll noch das Bildformat XPM unterstützt werden, welches ein ASCII-Textfile mit ANSI C-Syntax darstellt (dieses Format wurde ursprünglich für das X-Window System entwickelt und erlaubt die einfache Benutzung von Bildern in C Programmen). Obwohl dieses Bildformat in Textform gespeichert ist, ist das Einlesen nicht allzu schwer. Man muss aber beachten, dass dieses Bildformat im Gegensatz zu den beiden BMP Formaten mit Farbpaletten arbeitet. Mit Farbpaletten werden Farbwerte als Paare von Identifier und Farbwert definiert und die einzelnen Bildpunkte definieren nur den Identifier als Farbindex. Im Folgenden ist ein Beispiel gezeigt, wo die Definition der Farbpaletten Fett hervorgehoben wurde:

```
[0] man_yellow.xpm
/* XPM */
static char *man_yellow_xpm[]={
"9 17 2 1",
". c #001010",
"# c #ffdc3a",
".........",
".........",
"...###...",
"...###...",
"...###...",
"....#....",
"..#####..",
".#.###.#.",
".#.###.#.",
".#.###.#.",
".#.###.#.",
"...#.#...",
"...#.#...",
"...#.#...",
"...#.#...",
".........",
"........."};
```



Oberhalb der Farbpalettendefinition befindet sich eine Headerzeile mit den Werten X, Y, #C und #c, wobei X die Länge der Pixelzeilen definiert (9 im Beispiel), Y die Anzahl der Pixelzeilen definiert (17 im Beispiel), #C die Anzahl der Farben definiert (2 im Beispiel) und #c die Anzahl der Textzeichen pro Farbpixel definiert.

Der Einfachheit halber sei angenommen, dass ANSI C Kommentare nur am Zeilenanfang beginnen dürfen. Weiters braucht nur die Variante unterstützt werden, wo die Farben direkt als Binärwerte definiert werden (Varianten, wo Farben mit symbolischen Namen definiert werden, braucht daher nicht unterstützt werden). Weitere Informationen zum XPM Bildformat findet man beispielsweise im Internet unter folgender Adresse:

http://en.wikipedia.org/wiki/X_PixMap

Für dieses Beispiel ist es notwendig, dass man das Klassendesign gegenüber dem ersten Beispiel refaktorisiert. Die Klasse CBitmap soll nun eine abstrakte Interfaceklasse sein,

von welcher Klassen für die jeweiligen Dateiformate abgeleitet werden. Ebenso soll es für die Klasse CPixelFormat gemacht werden, welche nun abstrakte Interfaceklasse für die Klassen der unterschiedlichen Pixelcodierungen ist. In dem Beispiel werden zumindest drei Pixelcodierungen benötigt: RGB16, RGB24 und Indexed8. Das Pixelformat meint eine Farbkodierung mit Paletten und 8 Bit breiten Farbindizes.

Die in der Spezifikation nicht genauer ausgeführten Teile können selbst sinnvoll ausgestaltet werden, wobei jedoch die folgenden Anforderungen einzuhalten sind.

Bonusaufgabe (freiwillig, ermöglicht 5 Bonuspunkte)

- Unterstützung weiterer einfachen Dateiformaten (z.B. TGA)
-
- Skriptbefehle in dynamisch ladbaren Modulen implementiert (Plugin-Mechanismus für beliebige Erweiterungen).

Anforderungen:
Kommentieren Sie den Code, um das Verstehen des Codes zu vereinfachen. Zusätzlich soll zu Beginn jeder Klasse eine Klassenbeschreibung stehen. Am Beginn jeder Datei soll außerdem Name, MNr, KZ der Gruppenmitglieder sowie die Beispielnummer stehen. Ein nicht kommentierter Code wird negativ bewertet!

Ausnahmen (Exceptions) müssen immer abgefangen werden, sofern möglich. Überlegen Sie sich dabei aber auch, welche Ausnahmen zu einer Programmbeendigung führen müssen und in welchen Fällen es ausreicht, eine Warnung bzw. Fehlermeldung auszugeben. Bei jeder Ausnahmebehandlung ist auf jeden Fall eine Fehlermeldung auf die Fehlerausgabe (cerr) auszugeben. Als Orientierung seien einige mögliche Quellen für Ausnahmen aufgezählt: Speichermangel, inkorrekter Input, Schreiben auf Datei fehlgeschlagen, etc.

Schreiben Sie klare Zusicherungen an den passenden Stellen im Programm. Dort, wo man Zusicherungen kompakt als Code formulieren kann, wird empfohlen, die Zusicherungen zusätzlich als „assert()“ hinzuschreiben, damit sie auch vom Compiler geprüft werden können. Mit der Compileroption „-DNDEBUG“ kann man den fertig ausgetesteten Code generieren sodass „assert()“ keinen Overhead mehr verursacht.

Benutzen Sie zur Programmentwicklung auch ein Makefile, welches zumindest die Targets „clean“, „all“ und „run“ enthält. Mittels „make run“ soll das Programm direkt aufgerufen werden können (mit sinnvollen Argumenten versehen).

Es sind die Programmierrichtlinien der LVA einzuhalten.

Schreiben Sie zu dem Programm ein Protokoll, das folgende Informationen beinhaltet:

- Identifikation (Name, MNr, Kz)
- Aufgabenstellung (kann direkt von dieser Angabe übernommen werden, Beispiele mit Grafiken sind nicht zu übernehmen)
- Klassendiagramme (inkl. Beschreibung)
- Schematische Beschreibung der Allokation und Freigabe von Ressourcen
- Schematische Beschreibung der Ausnahmebehandlung (mögliche Ursachen von Exceptions, sowie die Strategie bei deren Behandlung (u.a., ob lokal oder eher global behandelt))
- Dokumentation des Arbeitsaufwandes
- Dokumentation von ev. aufgetretenen Problemen
- Kommentierter Programmcode

Der Umfang des Protokolls soll (ohne Mitzählen der Codeseiten) zwischen 5 bis 10 Seiten sein.

Abgabe:

Legen Sie ein Verzeichnis „Beispiel2“ an, in dem Sie alle Ihre Programmdateien hineingeben. Geben Sie auch das Protokoll in dieses Verzeichnis hinein.
Packen Sie das Verzeichnis folgendermaßen ein:

```
tar –zcvf Beispiel2.tgz Beispiel2
```

und geben Sie dieses Archivfile elektronisch ab. Ohne diese elektronische Abgabe sowie das Mitbringen eines schriftlichen Ausdruckes des Protokolls kann die Abgabe nicht positiv gewertet werden.

Der relevante Stoff für das Abgabegespräch ist das Kapitel 1 sowie Abschnitt 2.1 und 2.2 vom Kapitel 2 des Skriptums. Es ist dabei auch notwendig, dass Sie den Text, wo zutreffend, für die Lösung des Beispiels anwenden.