

Übungen zu Grafikprogrammierung in C++

Was für die Abgabe der Lösungen zu beachten ist:

- Nur Quelldateien, also z.B. *.cpp, *.cxx, *.h, Makefile, *.sln oder *.vcproj sollten ins SVN
- Versionskontrolle von Binärdateien, wie z.B. *.exe, *.o, *.obj, macht keinen Sinn
- Die Lösungen müssen mit g++ bzw. dem Visual Studio C++ Compiler (.NET 2005) zu kompilieren und linken sein

Aufgabe 5 (H) Funktionen & Kontrollstrukturen

In dieser Aufgabe soll die Programmierung von Funktionen und die Integration von Kontrollstrukturen geübt werden. Dafür wird eine Funktion zur Berechnung der Fakultät als Beispiel verwendet.

- Schreiben Sie eine Applikation, die über eine Funktion `Factorial` verfügt, die die Fakultät einer vom Benutzer eingegebenen Zahl berechnet und ausgibt. Erstellen Sie hierzu lediglich eine Datei `Factorial.cpp` ohne header-Datei mit nur zwei Funktionen, einer `main`-Funktion und der oben genannten. Verwenden Sie für die Berechnung der Fakultät Rekursion. Definieren Sie zuerst die `main`-Funktion, dann die Funktion `Factorial` und verzichten Sie auf eine explizite Deklaration der Funktionen.
- Drehen Sie nun die Definition der beiden Funktionen um, so dass zuerst `Factorial`, dann `main` definiert wird. Versuchen Sie die Quelldatei zu kompilieren.
- Fügen Sie vor den Funktionsdefinitionen nun die Deklarationen ein, also die Funktionsprototypen. Kompilieren Sie erneut die Quelldatei. Liefert der Übersetzer ein anderes Ergebnis im Vergleich zur letzten Teilaufgabe? Wenn ja, erklären Sie kurz weshalb.
- Erstellen Sie nun eine header-Datei `Factorial.h` und setzen Sie dort die Funktionsprototypen ein. Löschen Sie diese aus der `.cpp`-Datei und fügen Sie stattdessen ein `# include`-Statement ein, das die header-Datei einbindet.
- Versuchen Sie jetzt die Funktion `Factorial` so umzuschreiben, dass keine Rekursion mehr verwendet wird. Schreiben Sie diesen Code in eine Funktion `FactorialNonRecursive`. Übersetzen Sie das Programm und testen Sie die Laufzeit für beide Funktionen (schreiben Sie einfach `time`) in der Kommandozeile vor das Executable, also das ausführbare Programm. Gibt es einen Unterschied?

Aufgabe 6 (H) Weiteres zu Kontrollstrukturen

Schreiben Sie ein Programm, das den Wochentag für ein vom Benutzer eingegebenes Datum ausgibt. Erstellen Sie dafür eine Datei `Weekday.cpp` mit den entsprechenden Funktionen. Falls nötig, erstellen Sie ebenfalls einen header-Datei. Für die Berechnung des Wochentages sollen die Regeln des gregorianischen Kalenders gelten:

- Die Zeitrechnung beginnt drei Tage nach Christi Geburt, am 1. Januar 0001. Dies ist ein Samstag.
- Schaltjahre werden folgendermaßen berechnet: Bis 1599 ist jedes vierte Jahr ein Schaltjahr, wie es im julianischen Kalender festgelegt wurde. Ab 1600 ist jedes vierte Jahr ein Schaltjahr, es sei denn, es handelt sich um ein Jahrhundert, das nicht durch 400 teilbar ist. Also ist 1600 oder 2000 ein Schaltjahr, aber 1700 oder 1900 nicht. Natürlich sind aber z.B. 1904, 1724 usw. wiederum schon Schaltjahre.
- Der zusätzliche Tag bei Schaltjahren ist der 29. Februar.
- Auf den 4. Oktober 1582 folgte gleich der 15. Oktober 1582, wobei die Abfolge der Wochentage dabei unverändert blieb.¹

Diese Regeln wurden von Papst Gregor in einer päpstlichen Bulle am 24. Februar 1582 erlassen. Versuchen Sie, für Wochentage und Monatsnamen *enumerations* zu verwenden. Sollte es dabei zu Konvertierungsschwierigkeiten kommen, versuchen Sie ein type-casting anzuwenden, wie Sie es aus Java kennen.

- a) An welchem Wochentag hat Papst Gregor die Bulle zur Einführung des gregorianischen Kalenders erlassen?
- b) An welchem Wochentag wurde die Bastille² gestürmt und damit der Prozess zur Demokratisierung Frankreichs eingeleitet?
- c) An welchem Wochentag ist die ehemalige DDR der Bundesrepublik beigetreten?³
- d) An welchem Wochentag hat Kurt Cobain Selbstmord begangen⁴?
- e) Was war Silvester 2000 für ein Wochentag?
- f) An welchem Wochentag werden Sie ihrer Meinung nach ihr Diplom bekommen (dies ist jeweils am 15. jedes Monats möglich)?

Aufgabe 7 (H) Makefile

Im Falle von mehreren Dateien, die zu kompilieren und linken sind, kann dies ein aufwendiger Prozess sein. Besonders wenn die Programme eine kritische Größe erreichen.

Hierfür stellt GCC sogenannte Makefiles zur Verfügung. In Ihnen wird spezifiziert, was und in welcher Reihenfolge kompiliert und gelinkt werden soll. Dies geschieht dann durch einen einfachen Aufruf von

¹Um die Frühlings-Tagundnachtgleiche wieder mit dem 21. März in Übereinstimmung zu bringen und Ostern wieder am richtigen Tage feiern zu können, folgte der Papst dem Vorschlag des Mediziners und Hobby-Astronomen Aloisius Lilius und bestimmte, dass im Jahre 1582 zehn Tage übersprungen werden sollten.

²14.Juli 1789

³3.10.1990

⁴8.April 1994

make

In einer Datei mit dem Namen `Makefile` wird die Reihenfolge der Kompilier- und Linkschritte angegeben. Die Datei ist wie folgt aufgebaut.

```
target: dependencies
[tab] system command
```

In dieser Aufgabe soll nun das Programm aus der vorhergehenden Aufgabe durch ein geeignetes `Makefile` kompiliert werden. In einem `Makefile` sollen nun die Objektdateien durch das Kompilieren entstehen, bevor das Projekt zu einem ausführbaren Programm mit dem Namen `Aufgabe7` gelinkt wird.

Ein einfaches `Makefile` wäre folgendes:

```
all:
    g++ -c -o testfile.o testfile.cpp
    g++ -c -o main.o main.cpp
    g++ testfile.o main.o -o MyApplication

clean:
    rm -f MyApplication
    rm -f *.o
```

Eine ausführliche Einführung zu `Makefiles` ist unter <http://mrbook.org/tutorials/make/> zu finden oder mit `man make` abrufbar.

Aufgabe 8 (H) IDEs

IDEs (Integrated Development Environments) sind Programme, die das Entwickeln von Software erleichtern sollen. In einer grafischen Benutzeroberfläche werden verschiedene Werkzeuge angeboten, die das Programmieren in großen Projekten vereinfachen. Neben Fenstern, die die einzelnen Dateien strukturiert anzeigen und Dialogen, die Compilerflags komfortabel konfigurieren lassen gibt es auch fortgeschrittene Werkzeuge wie Debugger zum Fehlerfinden und Profiler zum Aufspüren von ressourcenintensiven Programmteilen. Prinzipiell könnte man alles auch per Kommandozeile bedienen, jedoch kann man mit einer IDE effizienter und übersichtlicher Arbeiten. Eine der häufig verwendeten IDEs zur Entwicklung von C++-Programmen ist Visual Studio .NET 2005. Jeder Student der TU München kann eine kostenlose Version dieser IDE über den Maniac-Server beziehen

(<https://prod.maniac.tum.de/ManiacGUI>).

Im Visual Studio bilden mehrere Dateien ein Projekt (*.vcproj). Jedes Projekt hat ein Ziel, das kompiliert wird. In unserem Fall ist das eine exe-Datei. Man kann mehrere Projekte in einer Solution bündeln.

- a) Portieren Sie Aufgabe 5 in eine VS solution. Öffnen Sie dazu das Visual Studio und wählen Sie `File` → `New` → `Project`. Als Vorlage dient ihnen das `Win32 Console Project`. Geben Sie ihrem Projekt einen Namen und bestätigen Sie mit `OK` und drücken Sie `Next`. Aktivieren Sie unter den `Project Settings` das Kontrollkästchen `Empty Project`.
- b) Nun haben sie eine Solution mit einem Projekt erstellt. Im `Solution Explorer` können Sie die Dateien aus Aufgabe 5 mittels Rechtsklick auf einen Ordner und dann mit `Add` → `Add Existing Item` zu Ihrem Projekt hinzufügen. Mit `Add New Item` fügen Sie eine neue Datei ein.

- c) Erstellen Sie eine ausführbare Datei, indem Sie auf Build → Build Solution klicken, oder drücken sie Ctrl-Alt-B. Die ausführbare Datei liegt dann in einem Unterordner des Solution-Ordners mit Namen Debug.
- d) Sie können das Programm mit F5 starten (und automatisch kompilieren, wenn nötig). Sie werden nun beobachten, dass kurz ein Konsolenfenster öffnet, sich dann aber wieder schließt, da das Programm terminiert. Sie können ihr Programm mittendrin anhalten, indem Sie einen Breakpoint kurz vor dem Verlassen der main-Methode setzen (in dieser Zeile kurz auf den grauen Balken links vom Code klicken). Das Programm kann im Debugmodus Zeile für Zeile durchschritten werden.

Die Dateien müssen so in das auf den SVN Server eingechekkt sein, dass wir mit einem einzigen Klick das Programm kompilieren können. Bitte also die .sln, .vcproj, .cpp und .h Dateien einchecken, jedoch keine anderen!

Aufgabe 9 (H) Zeiger (Pointer) und Felder (Arrays)

Ein Zeiger vom Typ T wird durch T* ausgedrückt. Ein einfaches Beispiel hierfür ist:

```
int no = 23;
int* p = &no; // p holds the address of no
int no2 = *p; // no2 == 23
```

Dieses einfache Programm verdeutlicht den Unterschied zwischen einem Objekt von einem bestimmten Typ T und einem Pointer auf das Objekt. Der Pointer gibt die Speicheradresse an, in der das Objekt dieses Datentyps liegt.

Mit Hilfe von Pointern soll die Funktion

```
bool sortList(int* list, int length){ ... }
```

geschrieben werden, die eine Liste von Integerwerten aufsteigend sortiert. Der einfachste Algorithmus hierfür ist der BubbleSort. Er vergleicht jeweils zwei benachbarte Listenelemente. Falls der linke Wert kleiner ist, so muss nichts gemacht werden. Falls der linke Wert größer ist, so müssen die beiden Werte vertauscht werden.

Das Programm soll mit Beispieldaten getestet werden. Eine Liste mit 7 Elementen kann z.B. durch

```
int listToSort [] = {3,7,2,4,1,9,5};
```

erstellt werden. Nach dem Aufruf der Funktion sortList soll genau diese Liste sortiert sein.

Aufgabe 10 (H) Einfache Bildmanipulation

Ein Bild wird im Folgenden als eindimensionales Feld von Integerwerten dargestellt. Zusätzlich werden Variablen für die Dimensionen des Bildes benötigt. Jeder Wert zwischen 0 und 255 drückt die Intensität des Grauwertes aus, wobei 0 schwarz und 255 weiß ist. Z.B. stellt

```
int width = 5; // Breite des Bildes
int height = 4; // Höhe des Bildes
```

```
int* image = {2 , 7 , 8 , 3 , 5 ,  
              6 , 100, 120, 121, 4 ,  
              222, 212, 122, 120, 34,  
              56 , 78 , 67 , 89 , 101};
```

ein kleines 5x4 Pixel großes Bild dar.

In zwei Funktionen soll das Bild horizontal und vertikal geflippt werden.

Aufgabe 11 (H) Bildmanipulation am echten Beispiel

Nachdem das Programm mit dem einfachen Zahlenbeispiel getestet wurde, sollen die beiden implementierten Funktionen auf echte Bitmaps angewendet werden.

Im SVN ist unter `aufgaben/Aufgabe11` ein Beispielprogramm (`Aufgabe11.cpp`), das auf der Basis von `aufgaben/imagelib` ein `.bmp` Bild einliest und dieses wieder speichert. Es ist sowohl ein Makefile als auch eine VS solution vorhanden, womit das Beispielprogramm und die lib erzeugt werden können. Der Inhalt von `aufgaben` soll in das jeweilige Teamverzeichnis kopiert werden und die Datei `Bitmap.cpp` angepasst werden.

Die Datei `Bitmap.cpp` muss um die flip-Funktionen `flipH()` sowie `flipV()` erweitert werden. Dabei ist zu beachten, dass die Bild-Daten in der Membervariablen `*m_data` gespeichert werden, welche manipuliert werden muss. Der Datentyp eines Bitmaps ist `byte`. Die Höhe und Breite des Bildes stehen in der Membervariable `m_dim`. Dabei ist `m_dim[0]` die Breite und `m_dim[1]` die Höhe des Bildes.