Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim Geisenheim

Fachbereich Design Informatik Medien
Studiengang Bachelor Allgemeine Informatik

Programmieren 3 AI – Praktikum

Übungsblatt 11

2 + 0,5 Punkte

Sebastian Flothow

2012-01-16

Abnahme: 2012-01-23
Zur Abnahme ist die persönliche Anwesenheit erforderlich. Bearbeiten Sie die Aufgaben so, dass Sie nicht nur funktionierenden Code vorzuweisen haben, sondern diesen auch erläutern und Fragen dazu beantworten können. Zusätzlich sind die Lösungen am Tag der Abnahme per Email an `sebastian@flothow.de` einzusenden; dies dient u.a. der Prüfung auf eventuelle Plagiate.

Senden Sie Ihre Abgabe als tar.gz-Archiv(e). Diese müssen so gepackt sein, dass jedes Archiv genau ein Verzeichnis enthält, dessen Name (bis auf das Suffix) mit dem des Archivs übereinstimmt; ein solches Archiv kann durch einen Aufruf der Form `tar -czf` *`verzeichnis`*`.tar.gz` *`verzeichnis`* erzeugt werden.

Beim Kompilieren der Programme dürfen keine Fehler oder Warnungen auftreten.

Aufgabe 1 (1 Punkt)

Übernehmen Sie den stackbasierten Rechner von Blatt 10. Sofern die Bedeutung der Ein- und Ausgabefelder im Programmfenster noch nicht durch Beschriftungen (`QLabel`) kenntlich gemacht wurde, fügen Sie solche hinzu.

Verwenden Sie die I18N-Mechanismen von Qt, um das Programm mehrsprachig zu machen. Ändern Sie dazu das Programm so dass alle für Benutzer sichtbaren Texte durch `tr()` oder `QApplication::translate()` verarbeitet werden. Erzeugen Sie dann eine TS-Datei für eine Sprache Ihrer Wahl, übersetzen Sie die Strings und erzeugen daraus eine QM-Datei. Verwenden Sie den Ressourcenmechanismus von Qt, um die QM-Datei in das Programm einzubetten, und ergänzen Sie die `main()` so, dass sie eine zur Laufzeitumgebung passende Übersetzungsdatei lädt.

Testen Sie das Programme, indem sie die Umgebungsvariable `LANG` auf verschiedene Werte setzen (`locale -a` gibt alle vom System unterstützten Einstellungen aus). Prüfen Sie, dass die Übersetzung auch geladen wird, wenn das kompilierte Programm in ein anderes Verzeichnis kopiert wird, das die QM-Datei nicht enthält.

Statten Sie Ihr Projektverzeichnis mit einer Qt-Projektdatei aus, die Compiler-Warnungen aktiviert. Rufen Sie vor dem Packen `make distclean` auf.

Aufgabe 2 (1 Punkt)

Das Archiv `bmp.tar.gz` enthält eine kleine C-Bibliothek zum Erzeugen von BMP-Dateien.[1] Verändern Sie die Dateien `BMP.h` und `BMP.c` nicht. Machen Sie sich mit der Bibliothek soweit vertraut, dass sich Ihnen die Verwendung der in `BMP.h` aufgeführten öffentlichen Funktionen erschließt.

Schreiben Sie in C++ eine Wrapper-Klasse `BMPxx`, d.h. eine Klasse die intern die Funktionen aus `BMP.h` verwendet, nach außen hin jedoch eine objektorientierte Schnittstelle bietet. Ein Programm dass diese Klasse verwendet soll also `BMPxx`-Objekte regulär per `new` und `delete` anlegen und zerstören können und die Methoden `setPoint` und `writeToFile` darauf aufrufen können, ohne dabei mit der Struktur `Bmp` aus der vorgegebenen Bibliothek bzw. darauf zeigenden Pointern in Berührung zu kommen.

Schreiben Sie zum Testen in einer separaten C++-Datei eine `main`-Funktion, die mittels der Klasse `BMPxx` eine simple BMP-Datei schreibt.

Statten Sie das Abgabeverzeichnis mit einem Makefile aus. Dieses muss ein Default-Target haben, dass Ihr Programm mit Warnungen (`-Wall -Wextra -pedantic`) kompiliert und linkt, sowie das Target „`clean`“, dass die erzeugten Dateien wieder entfernt. Achten Sie dabei darauf, dass die Datei `BMP.c` mit `gcc` kompiliert wird, für das Kompilieren der C++-Quelltexte und das Linken `g++` verwendet wird. Rufen Sie vor dem Packen `make clean` auf.

Aufgabe 3 (optional, 1/2 Punkt)

Erweitern Sie das Makefile von Aufgabe 2 so, dass damit die vorgegebenen Quelltexte und Ihre `BMPxx`-Klasse zu einer statischen und einer dynamischen Bibliothek gelinkt werden können (`libbmpxx.a` bzw. `libbmpxx.so`). Testen Sie diese Bibliotheken.

[1] Diese Bibliothek wurde von Prof. Dr. Panitz für das Praktikum Programmieren 1 im WS 2006/07 geschrieben.