Aufgabe 1

1 Lernziel

- Kennenlernen der Entwicklungsumgebung
- Erste praktische Erfahrungen mit der Programmiersprache C++ sammeln
- Erste hardwarenahe Programmierung

2 Aufgabenbeschreibung

Zum Überprüfen späterer Implementierungen und zur Erleichterung der Fehlersuche soll das Übungsbetriebssystem CoStuBS als erstes eine Ausgabemöglichkeit erhalten. Mit dieser werden Zeichen auf dem Bildschirm ausgegeben. Desweiteren wird eine Abstraktion eingeführt, die spätere Ausgaben auf andere Geräte, wie die serielle Schnittstelle, ermöglicht.

3 Theoretische Aufgaben

3.1 Vorbereitung

Informiert euch über den CGA Bildschirm. Unterstützung erhaltet ihr z.B. hier:

- Informationen zum **CGA Bildschirm**

3.2 Kontrollfragen

Versucht, die folgenden Fragen zu beantworten **bevor** ihr mit der Implementierung beginnt.

- Was ist ein CGA-Bildschirm?

- Wie viele Zeichen passen auf einen CGA-Bildschirm und in welcher Anordnung (Zeilen und Spalten)?
- Wie viel Platz (in Byte) benötigt ein Zeichen auf dem CGA-Bildschirm (inklusive aller Attribute)?
- Wie kann man den CGA-Bildschirm ansprechen?
- Wie sieht der Ablauf des Schreibens eines Zeichens skizzenhaft aus?
- Wie sieht der Ablauf des Setzens des Cursors skizzenhaft aus?
- Was ist eine Initialisierungsliste in C++?

4 Praktische Aufgaben

4.1 Kennenlernen der Entwicklungsumgebung

1. Der Emulator wird später ein iso-Image zum Start des Systems benutzen. Zum Übersetzen von CoStuBS und der Erstellung des Images wechselt ihr in das `bin`-Verzeichnis und führt dort den Befehl `make clean bootdisk` aus. Das `clean` ist hierbei wichtig, wenn ihr nur kleine Änderungen gemacht habt. Führt ihr `make` ohne den Parameter `clean` aus, werdet ihr zeitnah auf das Problem stoßen, dass eine kleine Änderung, die ihr gemacht habt, keine Wirkung zeigt.
   An dieser Stelle wird das Übersetzen scheitern - ihr habt ja noch nichts implementiert. Im Makefile könnt ihr sehen, dass für das Übersetzen mehrere C++-Dateien benötigt werden. Was bei diesen zu implementieren ist, wird in den nachfolgenden Teilaufgaben beschrieben. Nachdem ihr diese gelöst habt, fahrt mit dem nächsten Schritt hier fort.

2. Nachdem ihr mit dem obigen Befehl (vorausgesetzt es hat ohne Fehler geklappt) das Ganze übersetzt habt, könnt ihr nun den Befehl `bochs` aufrufen. Dieser wird die virtuelle Maschine starten, vorausgesetzt die Konfiguration in der Datei `bochsrc` ist korrekt. Im Anschluss könnt ihr sehen, ob euer Code funktioniert.

4.2 Einarbeitung in die Vorgabe

Wie in jeder Vorgabe sind neue Dateien hinzugekommen oder bereits bestehende wurde verändert. Schaut euch die vorgegebenen Dateien an, damit ihr deren Inhalt versteht. Einzige Ausnahme sind die Dateien im Ordner machine/boot - diese müsst ihr nicht verstehen.
Zur Unterstützung werden manchmal UML-Diagramme zur Verfügung gestellt. Bitte beachtet, dass diese nicht immer vollständig sind.

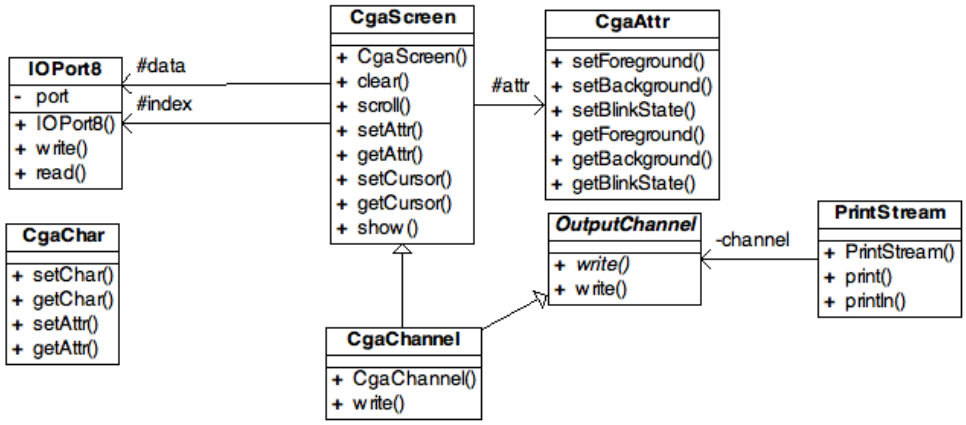

Abbildung 1: Klassendiagramm

Gegeben ist das Klassendiagramm in Abbildung 1. Außerdem geben wir für den Anfang Skizzen einiger Klassen vor. Ferner wird die Klasse IOPort8.h vorgegeben, die einen 8 Bit breiten Ein-/Ausgabeport beschreibt und uns erlaubt, von einer Hochsprache aus mit der Hardware zu kommunizieren.

4.3 CgaScreen

Hier soll die Klasse CgaScreen.h implementiert und in einer kleinen Anwendung getestet werden. Diese Klasse ist unser Zugang zum CGA-Bildschirm und sie soll uns erlauben, Zeichen mit den zugehörigen Darstellungsattributen (Farbe etc.) auf dem Bildschirm darzustellen. Diese Klasse benötigt die Hilfsklasse CgaAttr.h zur Beschreibung der Darstellungsattribute. Die Klasse CgaChar.h dient als Softwareprototyp für die Darstellung eines Zeichens in dem Format, wie es von der CGA-Hardware gefordert wird. Vervollständigt die Implementierung dieser Klassen, erstellt gegebenenfalls neue Quelldateien. Erstellt Testfälle, mit denen ihr die Korrektheit eurer Implementierung testen könnt.

Hinweis:

Bitte verwendet folgende Farbbezeichner:

BLACK, BLUE, BROWN, CYAN, GRAY, GREEN, LIGHT_BLUE, LIGHT_CYAN, LIGHT_GRAY, LIGHT_GREEN, LIGHT_MAGENTA, LIGHT_RED, MAGENTA, RED, WHITE, YELLOW.

Bitte denkt daran, dass neu erstellte *.cc Dateien im Makefile eingetragen werden *müssen*. Achtet hierbei ganz besonders darauf, dass ihr nicht Zeichen, im Besonderen Leerzeichen, einfügt, die an dieser Stelle keinen Zweck haben.

4.4 CgaChannel

Hier soll die Klasse CgaChannel.h implementiert und in einer kleinen Anwendung getestet werden. Diese Klasse hat die Aufgabe, einen hardwareunabhängigen Ausgabekanal auf die vorhandene CGA-Ausgabehardware abzubilden. Somit stellt diese Klasse eine Brücke zwischen einem abstrakten Konzept (einem Ausgabekanal, beschrieben durch die Klasse OutputChannel.h) und einer konkreten Ausgabehardware (beschrieben durch die Klasse CgaScreen.h) dar. Erstellt Testfälle, mit denen ihr die Korrektheit eurer Implementierung testen könnt.

Hinweis:

Der CGA-Bildschirm fängt auf der Position (0,0) an. Dementsprechend sollte auch der Aufruf der Methode setCursor mit den entsprechenden Parametern dafür sorgen, dass der Cursor oben links in der Ecke steht.

4.5 PrintStream

Zu guter letzt brauchen wir noch eine Klasse, die es uns erlaubt, z.B. Zahlen in menschenlesbarer Form auszugeben. Dazu müßt Ihr eine Klasse PrintStream .h vollständig implementieren, die Texte, Zahlen und dergleichen auf einem beliebigen Ausgabekanal ausgeben kann. Diese Klasse ist hardwareunabhängig und kann deshalb in Kombination mit vielen unterschiedlichen Ausgabekanälen benutzt werden. Hier zeigt es sich, wie sinnvoll es war, Ausgabekanäle als ein abstraktes BS-Konzept einzuführen.

Hinweis:

Zahlen zur Basis 2 sowie zur Basis 16 sollten mit einem entsprechenden Präfix ausgegeben werden, sodass die Gefahr eliminiert wird, später Adressen mit Dezimalzahlen zu verwechseln. Darüberhinaus reicht es, wenn negative Zahlen mit Vorzeichen ausgegeben werden. Eine Umwandlung in das Zweierkomplement ist nicht nötig aber erlaubt.

4.6 Vorbereitung der Abnahme

Testet eure Implementierung ausführlich und vervollständigt Kommentare. Bereitet euch auf die Abnahme vor. Informationen zu deren Ablauf findet ihr auf der Lehrstuhl-Webseite.

Viel Erfolg!