Hochschule RheinMain
Fachbereich DCSM - Informatik
Marcus Thoss, M.Sc.
Prof. Dr. Robert Kaiser
Christian Neugebauer

Hardwarenahe Programmierung I

WS 2021/22
LV 1512

Übungsblatt 4

Aufgabe 4.1 (Blackjack):

Sie entwickeln eine Umsetzung des Kartenspiels Blackjack („17+4“). Für den Anfang wird das Spiel als Kommandozeilenprogramm in einer Linux-Shell gespielt werden. Informieren Sie sich über die Regeln des Spiels. Verwenden Sie möglichst durchgängig englische Ausgaben und Variablennamen und kommentieren Sie Ihren Code auf Englisch. Legen Sie für alle in Header-Dateien deklarierten Funktionen Kommentare über den Deklarationen an, die jeweils beschreiben, was die Funktion tut.

a) Legen Sie die C-Quelldateien `blackjack.c`, `game.c`, `game.h`, `ui.c` (für „User Interface“) und `ui.h` an. `blackjack.c` soll Ihre `main()`-Funktion enthalten.

   Schreiben Sie hierzu ein Makefile, das das Programm `blackjack` unter Verwendung aller drei Dateien erstellt.

   In `ui.c` und `game.c` sollen Funktionen definiert werden, die `blackjack.c` verwendet, fügen Sie deshalb passende `#include`-Anweisungen in `blackjack.c` ein und schützen Sie ihre `.h`-Dateien gegen rekursives `#include`.

   Versionieren Sie alle Quelldateien und das Makefile fortlaufend in Gitlab. Versehen Sie das Verzeichnis, das Sie dafür in Gitlab verwenden, auch mit einer sinnvollen Readme-Datei.

b) Spielkarten besitzen eine Farbe und einen Wert. Beides soll in Ihrem Programm über Zahlenwerte vom Typ `int` repräsentiert werden. Verwenden Sie die Zahlen 1-13 für Ass-9, Bube, Dame und König. Verwenden Sie die Zahlen 1-4 für Karo, Herz, Pik und Kreuz. Die 0 soll als Wert jeweils „nicht definiert“ bedeuten.

   Definieren Sie in `game.h` Makros für die genannten Zahlenkonstanten (z.B. `#define KARO 1`).

c) Definieren Sie eine Funktion `print_player()` in `ui.c`, die folgende Argumente erhält: ein Character-Array für einen Namen `char name[]`, zwei weitere `int[]`-Arrays, die Farbe und Wert der Karten des Spielers darstellen und einen `int`-Parameter, der die Anzahl der Karten des Spielers darstellt. Die Funktion soll mittels `printf()` die Kartenhand des Spieler ausgeben (für Strings, also das Character-Array, verwenden Sie `%s` in `printf()`).

   Hinweis: Sie müssen nun auch `game.h` in `ui.c` mit `#include` einbinden.

   Testen Sie Ihre Funktion `print_player()` mit einer Funktion `test_print_player()`, die in `blackjack.c` implementiert ist und passende Array-Variablen für einen Spieler anlegt, diese mit *festen Testwerten* füllt (es soll noch nicht „richtig“ gespielt werden) und die Ausgabefunktion

aufruft. Beachten Sie, dass das Array für den Speilernamen als letzten Wert ASCII 0 enthalten muss, was das Ende des String kennzeichnet. Rufen sie `test_print_player()` in Ihrer `main()`-Funktion auf.

d) Implementieren Sie die Funktionen `get_value()` und `get_colour()` in `game.c`, die einen zufälligen Kartenwert bzw. -farbe *zurück*geben. Verwenden Sie `random()`, um Wert und Farbe auszuwählen.

e) Implementieren Sie eine Funktion `check()` in `game.c`, die prüft, ob die Hand des Spielers bereits 21 Punkte überschritten hat und das Ergebnis der Prüfung zurückgibt. Denken Sie daran, dass Bube, Dame und König jeweils 10 Punkte wert sind. Ass kann einen oder 11 Punkte wert sein; Sie können für Ass aber auch mit der Annahme von stets 11 Punkten vereinfachen. Überlegen Sie sich sinnvolle Funktionsparameter.

   Testen Sie Ihre Funktionen mit einer Funktion `test_single_round()`, die wie bei `test_print_player()` zunächst Arrays für Namen, Werte und Farben anlegt und solange `get_value()`, `get_colour()`, `check()` und `print_player()` aufruft, bis entweder fünf Karten gezogen wurden oder 21 Punkte überschritten sind. Benutzen Sie eine `int`-Variable für den Index der aktuell gezogenen Karte.

   Entwerfen Sie `test_single_round()` zunächst als Flussdiagramm (auf Papier oder mit PlantUML), bevor Sie die Funktion implementieren.

   Rufen Sie `test_single_round()` in `main()` auf.

f) Scheiben Sie in `ui.c` die Funktion `enter_name()`, die ein `char`-Array für den Namen bekommt, einen Text ausgibt, der zur Eingabe eines Spielernamens auffordert und die Eingabe durch Einlesen von Zeichen mit `getchar()` entgegennimmt und im Array abspeichert. Die Eingabe soll durch die Return-Taste beendet werden und hinter dem Ende des Namens soll im `buf`-Array 0 als Wert eingetragen sein.

   Schreiben Sie eine Funktion `ask_move()`, die zur Eingabe von `h` oder `s` für *hit* (weitere Karte) bzw. *stay* (keine Karte mehr) auffordert und die Eingabe als Rückgabewert der Funktion zurückliefert (Typ der Rückgabe ist Ihnen überlassen).

g) Programmieren Sie nun statt der Aufrufe der Testfunktionen eine einfache Spielelogik in `main()`, die mit die neuen Funktionen nutzt, um einen Spieler seinen Namen eingeben zu lassen und anschließend bis zu fünf Karten (falls möglich, testen Sie mit `check()`!) ziehen und jeweils den aktuellen Stand ausgeben zu lassen.

h) Die Weiterentwicklung zu einen Spiel gegen einen oder mehere Computergegener oder menschliche Mitspieler ist sicherlich reizvoll aber im Rahmen dieses Praktikumsblatts freiwillig.