1 Übungsaufgabe Mondlandung

Entwickeln Sie eine einfache Simulation einer Mondlandung. Nehmen Sie an, Ihre Raumfähre befindet sich in 50.000m Höhe über der Mondoberfläche und hat eine Abwärtsgeschwindigkeit von derzeit 3600km/Std = 1000 m/s. Wenn Sie keinen Schub geben, so werden Sie mit 1.63m/s^2 vom Mond angezogen. Die Antriebe Ihrer Raumfähre vermögen bei vollem Schub jedoch eine Gegenbeschleunigung von 12m/s^2 zu erzeugen und verbrauchen dabei pro Sekunde 100 Liter Treibstoff. Anfangs verfügen Sie über 10.000 Liter Treibstoff. Ihre Raumfähre kann eine Aufsetzgeschwindigkeit von 10m/s aushalten, ansonsten ist das Landemanöver gescheitert. Hierfür ist eine entsprechende Meldung auszugeben.

Einige Hilfestellungen

- Es gelten folgende Formeln
  $v = a * t$ (v-Geschwindigkeit, a-Beschleunigung, t-Zeit)
  $s = v * t$ (s-Zurückgelegte Wegstrecke bei konst. Geschwindigkeit ~ Höhenverlust)
  $s = ½ a t^2$ (zurückgelegte Wegstrecke unter Einwirken einer konstanten Beschleunigung)

- Es ist sinnvoll, die Simulationsgeschwindigkeit einstellbar zu gestalten (entweder über eine entsprechende Tastaturabfrage oder über die Definition einer Konstanten). Dann können Sie das Spiel entweder als Strategiespiel laufen lassen (z.B. ein Simulationsschritt pro 10 Sek. Aktion in Realität) oder aber eine feinere Auflösung wählen (z.B. 1/10 Sek, wenn Sie eine geeignete Tastaturabfrage verwenden, s.u.), um mehr in die Richtung Actionspiel zu gehen.

  Vorschlag: Realisieren Sie zuerst eine Variante, in der in jedem Zyklus mittels scanf abgefragt wird, ob Schub gegeben werden soll oder nicht (Strategievariante). Wenn Sie dann noch unternehmungslustig genug sind, können Sie anschließend z.B. in Visual C mit Hilfe der folgenden Funktionen abfragen lassen, ob eine Taste gedrückt wurde:
  _kbhit() - liefert true, wenn eine Taste gedrückt ist, sonst false
  _getch() - liefert das gerade gedrückte Zeichen
  (Für die Verwendung dieser beiden Funktionen wird ein #include <conio.h> am Dateianfang benötigt!)
  Achtung: In anderen C-Versionen als Visual C können diese Funktionen anders heißen!!!

- Es ist sicher hilfreich, die wesentlichen Kenndaten, wie Geschwindigkeit, Höhe, Treibstoffvorrat kontinuierlich auf dem Bildschirm auszugeben.

P.S.: Sie können das Spiel auch abwechslungsreicher gestalten, indem Sie es nicht nur eindimensional wie oben ausführen, sondern zusätzlich eine Horizontalgeschwindigkeit vorgeben, das Raumschiff lenkbar gestalten und dann den Schub entsprechend in einen horizontalen und vertikalen Anteil aufteilen.

Lerninhalte:

– *Verwendung grundlegender Sprachelemente wie z.B. Ausdrücke, Abfragen, Schleifen*

2 Übungsaufgabe Annuitätentilgung

Durch diese Übungsaufgabe soll einem Bankkunden ein Tilgungplan für ein beantragtes Darlehen erstellt werden. Es wurde eine Annuitätentilgung vereinbart. Weitere Darlehenskenngrößen:

> Darlehen: 100.000 €, Zins: 5% p.a. (eff), Tilgung: 3% p.a.
> (Bitte lassen Sie diese Größen über die Tastatur abfragen!)

Geben Sie bitte eine Tabelle aus, in der jeweils in einer Zeile die folgenden Daten stehen, bis das Darlehen vollständig zurückgezahlt wurde.

Lfd. Jahr | Restdarlehen | Zinsen in diesem Jahr | Tilgung in diesem Jahr

Bitte berechnen Sie auch den Gesamtbetrag, der zurückbezahlt werden muß (also Darlehen+Zinsen) und die monatliche Belastung

Hilfestellung (Annuitätentilgung)

Annuität (Jahresrate) = Darlehen * (Zins+Tilgung) / 100 *(Diese Rate bleibt über die gesamte Laufzeit konstant!)*

Zinsen pro Jahr = Restdarlehen * Zins/100

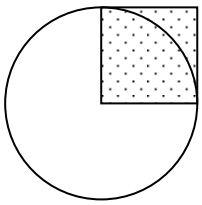
Tilgung pro Jahr = Annuität – Zinsen pro Jahr

Bitte probieren Sie aus, welche Auswirkungen eine Variierung der Tilgungsrate auf 1% bzw. 5% auf Laufzeit und Gesamtrückzahlungsbetrag hätte.

Lerninhalte:

- *Verwendung grundlegender Sprachelemente wie z.B. Ausdrücke, Abfragen, Schleifen*

3 Bestimmung der Zahl Pi



Bestimmen Sie die Zahl Pi mittels der Monte Carlo-Methode. Hierbei werden jeweils zwei Zahlen x und y im Bereich von [0..1] gewürfelt. Dieses Zahlenpaar entspricht einer Koordinate im Einheitsquadrat. Für jedes Zahlenpaar ist nun zu testen, ob sich die Koordinate innerhalb oder außerhalb des 1. Quadranten des Einheitskreises befindet. Dieser Test kann leicht mittels einer Abstandsberechnung (Pythagoras!) vorgenommen werden.

Das Verhältnis der Punkte im Einheitskreis zur Gesamtanzahl der gewürfelten Punkte entspricht der Zahl Pi/4 (Warum?). Lassen Sie hinreichend viele Zahlen würfeln, um Pi mit zufriedenstellender Genauigkeit zu erhalten.

Hinweis: Zum Würfeln einer Zahl können Sie die Funktion rand() verwenden. Diese Funktion ist im Header stdlib.h definiert. Diese Funktion liefert eine positive Ganzzahl im Bereich 0 .. RAND_MAX. Diese Ganzzahl muß dann noch auf den Bereich [0.0 ... 1.0] umskaliert werden!

<u>*Alternative*</u>: Sie können ebenfalls versuchen, die Zahl Pi durch in den Kreis eingeschriebene oder umschreibende Vielecke anzunähern. (Welche Variante ermittelt schneller die Zahl Pi?)

Lerninhalte:

- *Verwendung grundlegender Sprachelemente wie z.B. Ausdrücke, Abfragen, Schleifen*

4 Min, Max und Durchschnittsbestimmung

Erstellen Sie ein Programm, welches fortwährend Zahlen von der Tastatur abfragt, bis schließlich die Zahl –1 eingegeben wird. Für die eingegebenen Zahlen sollen minimaler Wert, maximaler Wert und Durchschnittswert bestimmt werden. Die Zahl –1, welche ja die Abbruchbedingung der Eingabe darstellt, soll dabei nicht mitgezählt werden.

Lerninhalte:

- *Verwendung grundlegender Sprachelemente wie z.B. Ausdrücke, Abfragen, Schleifen*

5 Zeichen, Zeilen und Wörter zählen

Schreiben Sie ein Programm, welches für auf der Tastatur eingegebene Texte die Anzahl der Zeichen, Zeilen und Wörter bestimmt.

Zur Abfrage eines Zeichens können Sie gut die Funktion getchar() verwenden. Hierbei erhalten Sie jeweils das nächste Zeichen. getchar() liefert den Wert EOF, wenn die Eingabe beendet wurde. Das Eingabeende kann unter MSDOS, Win durch die Tastenkombination CTRL-Z und unter UNIX durch die Tastenkombination CTRL-D erzeugt werden.

Tip: Sie können unter Unix, MSDOS, etc. auch Dateien an ein Programm umleiten, so daß diese dann dem Programm wie Eingaben von der Tastatur erscheinen. Somit können Sie dann mit Ihrem Programm die Anzahl Zeichen, Zeilen, Wörter in jeder beliebigen Datei zählen lassen. Hierzu müssen Sie unter Windows ein DOS-Fenster öffnen, dort in das Verzeichnis wechseln, in dem sich Ihr ausführbares Programm befindet und dieses dann durch

prog < datei

starten. Das '<' bewirkt, daß die Tastatureingaben dann aus der nachfolgend angegebenen Datei gelesen werden. Unter Unix können Sie ebenfalls ein Programm in der Form prog < datei aufrufen. Sie brauchen hierzu jedoch kein zusätzliches Fenster öffnen.

Tip2: Ebenso können Sie die Bildschirmausgaben in eine Datei leiten: prog > datei.out. Alle Bildschirmausgaben finden sich jetzt in der Datei datei.out wieder.

Lerninhalte:

- *Verwendung grundlegender Sprachelemente wie z.B. Ausdrücke, Abfragen, Schleifen*
- *Umgang mit Zeichenfolgen, z.B. Test auf Leerräume, Zeilenende, Eingabeende*

6 Spiel "Hangman"

Realisieren Sie das bekannte Wortratespiel auf dem Computer. Zeigen Sie zunächst von einem zu ratenden Wort für die einzelnen Zeichen jeweils nur Striche an. Anschließend fragen Sie nach Buchstaben. Sollten die Buchstaben im zu ratenden Wort vorkommen, so werden die entsprechenden Positionen aufgedeckt, d.h. an dieser Stelle werden die tatsächlichen Zeichen angezeigt. Durch Eingabe eines Zeichens (z.B. '#' – welches dann in keinem Wort vorkommen darf) kann das Raten abgebrochen werden und die Eingabe des vollständigen Worts ist nun möglich. Stimmt das eingegebene Wort mit dem verdeckten Wort überein, ist das Spiel zu Ende, ansonsten wird wieder in den obigen Buchstabenratemodus zurückgekehrt.

Beispiel: Zu ratendes Wort: "Hangman" (verdeckt)

```
- - - - - - - -     Eingabe (a)
- a - - - a -       Eingabe (g)
- a - g - a -       Eingabe (e)
- a - g - a -       Eingabe (n)
- a n g - a n       Eingabe (#) ~ Ende Buchstabenraten
```

Eingabe: Hangman → richtig geraten / bzw. Zurückkehren zum Buchstabenraten

Tip: Sie können das Spiel abwechslungsreicher gestalten, wenn Sie sich eine Tabelle von zu ratenden Wörtern im Programm anlegen und aus dieser Tabelle zufällig einen Eintrag auswählen, anstatt diesen vorher einzugeben.

7 Übungsaufgabe Zufallsgütetest

C verfügt über eine Bibliotheksfunktion, die Zufallszahlen generiert. Wir wollen in dieser Übungsaufgabe untersuchen, wie gut (d.h. zufällig) diese Funktion tatsächlich ist.

Hierzu wollen wir die folgenden Eigenschaften betrachten

- Gleichverteilung der Zufallszahlen
- Verteilung der Abstände zweier nacheinander generierter Zahlen

Für beide Verteilungen soll ein Histogramm erstellt werden, in dem die einzelnen Ereignisse gezählt werden. Anschließend sollen die Histogramme hinsichtlich folgender Eigenschaften ausgewertet werden:

Minimum, Maximum, Durchschnitt, Standardabweichung

Diese Daten sind auf dem Bildschirm auszugeben

Hinweis:

- Verwenden Sie für die Auswertung sinnvollerweise **eine** Funktion, an die Sie einmal das erste und dann einmal das zweite Histogramm übergeben.
- Die Formel für die Standardabweichung σ_n lautet $= \mathrm{sqrt}\left(\frac{\sum_{i=1}^{n} (x_i - xd)^2}{n} \right)$

  xd – sei der Durchschnitt über alle x

Mögliche Erweiterung:

Geben Sie das Histogramm auf dem Bildschirm aus. Hierbei haben Sie die Möglichkeit das Histogramm um 90° nach rechts zu drehen, so daß eine Histogrammspalte in einer Bildschirmzeile dargestellt wird (leichtere Variante) oder Sie stellen es "ungedreht" dar (bessere Übung!)

Lerninhalte:

- *Verwendung von Feldern und Funktionen*
- *Operationen auf Feldern (z.B. Durchlauf, Summenbildung, Minimumsuche)*

8 Übungsaufgabe Schiffe versenken

In dieser Übungsaufgabe soll das bekannte "Schiffe versenken"-Spiel auf einem 10x10 Kästchen großen Spielfeld realisiert werden. Hierbei darf der Computer auf einem Spielfeld Schiffe plazieren (ein Schiff, welches 4 Kästchen groß ist, zwei 3er-Schiffe und drei 2er-Schiffe). Die Schiffe müssen von dem Verfahren zufällig plaziert werden, jedoch natürlich so, daß sich zwei Schiffe nicht berühren, es muß also mindestens ein Kästchen zwischen zwei Schiffen frei bleiben).

Im folgenden dürfen Sie raten, wo sich die Schiffe befinden könnten. Hierzu können Sie Koordinaten eingeben und der Computer hat die Aufgabe, diese Position folgendermaßen zu bewerten:

- Wasser
- knapp daneben
- Treffer
- Treffer, versenkt

Neben dieser Information soll nach jedem Raten ein aktualisiertes Spielfeld angezeigt werden, in dem schon geratene Positionen, getroffene Schiffe, etc. geeignet dargestellt werden.

Hinweis:

Überlegen Sie sich bitte zuerst eine geeignete Repräsentation für die Schiffe (welche Informationen werden für ein Schiff benötigt, um alle Zustände beschreiben zu können, z.B. belegte Felder, vollständig versenkt?, ...)

Lerninhalte:

- *Verwendung von Feldern und Funktionen*
- *Operationen auf zweidimensionalen Feldern (z.B. Suchen nach freien Stellen, Umgebungstest, ...)*

9 Übungsaufgabe Zeile wortweise zerlegen

Schreiben Sie eine Funktion, die eine übergebene Zeile, die auch Interpunktionszeichen, etc. enthalten darf, wortweise zerlegen kann. Diese Funktion soll die folgende Schnittstelle besitzen:

```
char *ZerlegeInWorte( char **cp);
```

*cp ist dabei ein Zeiger auf die übergebene Textzeile.

Die Funktion liefert als Returnwert einen Zeiger auf den Anfang des nächsten Worts im String zurück oder NULL, falls kein Wort mehr vorhanden ist.

Darüber hinaus wird *cp derart verändert, daß es nach dem Funktionsaufruf auf das erste Zeichen hinter dem gelesenen Wort zeigt. Falls die Funktion den Wert NULL liefert, ist jedoch der Wert von *cp unerheblich. Der Inhalt der ursprünglichen Zeile darf in der Funktion nicht verändert werden!

Der Aufruf dieser Funktion könnte etwa wie folgt aussehen:

```
char buf[100], *bufp, *wordp;
gets(buf);
bufp = buf;
while (wordp = ZerlegeInWorte(&bufp))
      printf("Wort %s\n", wordp);
printf("Ende des Zerlegens\n");
```

Hinweis:

- Da die Eingabezeile nicht verändert werden darf, muß ein Puffer für ein Wort innerhalb der Funktion angelegt werden, in den dann das jeweilige Wort kopiert wird. Damit dieser Puffer über den Aufruf der Funktion hinaus Bestand hat, ist er folgendermaßen zu definieren: `static char tmpbuf[20];`
- Es gibt eine Reihe von Zeichenklassifiziermakros (wie z.B. ISALPHA, siehe Hilfetexte), die hier hilfreich sind.

Lerninhalte:

- *Arbeit mit Zeichenketten, Zeiger auf Zeichen*
- *veränderliche Rückgabeparameter von Funktionen*

10 Worthäufigkeit in Dateien zählen

Der Inhalt einer beliebigen Textdatei soll hinsichtlich der Häufigkeit der verwendeten Worte analysiert werden. Verwenden Sie zum Zählen ein großes Feld (z.B. 5000 Elemente) vom folgenden WordCount-Strukturtyp

```
struct WordCount {
   char     word[20];
   int      count;
};
```

Gehen Sie dabei wie folgt vor:

Nach dem Programmstart ist der Name der zu analysierenden Datei abzufragen.

Diese Datei soll geöffnet (Fehlerkontrolle!) und dann zeilenweise abgearbeitet werden. Die Zerlegung einer Zeile in einzelne Worte soll mit der Funktion ZerlegeInWorte aus der vorherigen Übungsaufgabe geschehen.

Suchen Sie dann im Feld, ob dieses Wort schon vorkommt. Falls dies der Fall ist, so wird der Wortzähler erhöht, ansonsten wird dieses Wort ans Ende der bisherigen Liste angehängt.

Konvertieren Sie alle Wörter in Groß- oder Kleinbuchstaben, um eine Trefferstreuung durch Groß-/ Kleinschreibung zu vermeiden.

Achten Sie bitte darauf, daß Ihr Programm auch dann noch sinnvoll weiterarbeitet, wenn angenommene Dimensionierungen nicht ausreichen sollten (max. Wortlänge = 20 Zeichen, max. Anzahl unterschiedlicher Wörter=5000)

Nach vollständigem Einlesen der Datei, sind dann die 20 am häufigsten verwendeten Wörter zusammen mit ihrer Häufigkeit auszugeben (Tip: Funktion qsort – siehe Hilfetexte!)

Tip

Sie können ja einmal ein altes Word-Dokument von Ihnen im Text-Format (SaveAs ...) abspeichern und dieses analysieren lassen.

Lerninhalte:

- *Arbeit mit Dateien*
- *Weiterverwendung von früher geschriebenen Funktionen*
- *Arbeit mit Feldern von einem Strukturtyp*
- *Suchen von Zeichenketten*

11 Übungsaufgabe Labyrinthsuche

Gegeben Sei ein geschlossenes Labyrinth in Form eines zweidimensionalen Feldes. Hindernisse und freie Flächen seien geeignet kodiert dargestellt.

Entwickeln Sie einen Algorithmus, der zwischen zwei anzugebenden Positionen des Labyrinths Weg bestimmt, bzw. meldet, daß kein Weg existiert. Beispiel:

```
Start, Ziel: 1 2 45 10

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X                              >>>>>>>>>>>>>>>>>>>vX
XS>>>>>>>vX    X      X    XXX^<XXXX    XXXv<<<<<X
Xv<<<<<<<<X    X             X>^...XXXXXX  >>>>>vX
X>>>>>>>>vX    X    XXXXXX   X^<<<<<<<<<<<XXXXXv<X
Xv<<<<<<<<XXXXXXX      X >>>>>>>>>>>>>>>>>^X    >vX
X>vXXXXX      X        X>^XXX  XXXXXXX....X   XXvX
X >>>>vX      XXXXX    X^<<<<<<<<<<<X.....Xv<<<<<X
Xv<<<<<XXXX     X      X.........>>>^X.....X>>>>>vX
X>>>>>>>>>>>>>>>vX      XXXXXXXXX^<<<XXXXXXXXXv<<<X
X  XXXXXXXXXv<<<X             >>>>>^X....X   Z   X
X        X  >>>vX        XXXXX^<XXXXXX...X   XXX X
X        X     >>>>>>>>>>>>>vX>^.........X       X
X        X     XXXXXXXXXv<<<<X^<<<<<<<<<<X       X
X        X          X   >>>>vX....XXXXX>^XXXXXXX X
X        X          Xv<<<<<<<X.........X^<<<<<<<<<X
X                    >>>>>>>>>>>>>>>>>>>>>>>>>>>^X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Weitere Suche?
```

Im einzelnen:

1. Das Labyrinth sei als Textdatei abgelegt, ’X‘ könnte z.B. ein Hindernis und ‘ ‘ einen Freiraum kennzeichnen. Die Größe des Labyrinths (x/y-Abmessungen) sei in den ersten beiden Zeilen der Datei angegeben. Diese Datei soll bei Programmstart eingelesen und angezeigt werden.
2. Anschließend sollen Start- und Zielpunkt für die Wegesuche abgefragt werden
3. Danach ist der Weg zwischen Start und Ziel zu bestimmen und geeignet anzuzeigen
4. Schließlich ist abzufragen, ob eine erneute Wegeplanung gewünscht wird. In diesem Fall ist das Verfahren ab Punkt 2) zu wiederholen.

Lerninhalte:

- *Rekursion*
- *Arbeit mit Dateien*

12 Übungsaufgabe Adreßverwaltung

Konzipieren und implementieren Sie ein Programm für die Verwaltung von persönlichen Adreßdaten (z.B. Name, Vorname, Telefonnummer, Straße, Ort und PLZ) und evtl. weitere zugehörige Daten. Hierbei sind die folgenden Aspekte zu beachten:

- Der Programmstart kann mit einem optionalen Parameter erfolgen, der eine Datei mit schon gespeicherten Adreßdaten beschreibt. Falls dieser Parameter angegeben wird, ist diese Datei einzulesen.
- Danach geht das Programm in einen Kommandomodus. Nun sollen die folgenden Befehle zur Verfügung stehen:
  - 'E' <RET> – Eingabe eines neuen Datensatzes. Die einzelnen Elemente des Datensatzes sollen von der Tastatur abgefragt werden. Anschließend wird der Datensatz in der Liste abgespeichert.
  - 'S <Name>' <RET>. Nun soll der Adreßdatensatz zu dem angegebenen Namen gesucht und auf dem Bildschirm angezeigt werden.
  - 'L <Name>' <RET>. Auch hier soll zunächst der Datensatz gesucht und angezeigt werden; Anschließend soll abgefragt werden, ob dies der für das Löschen gewünschte Datensatz ist? Bei Verneinung der Frage soll weitergesucht werden, ansonsten ist der Datensatz aus der Liste zu löschen.
  - 'W <Dateiname>' <RET>. Der gesamte Inhalt der Liste ist in die angegebene Datei zu sichern; Dateiformat s.u.
  - 'A' <RET>. Der Inhalt der Liste wird datensatzweise auf dem Bildschirm ausgegeben und jeweils auf eine Bestätigung gewartet.
  - 'X' <RET>. Beendigung des Programms. Sollten noch nicht alle Daten abgespeichert worden sein, so ist noch eine Bestätigung einzuholen bzw. sind die Daten entsprechend abzuspeichern.

Dateiaufbau

- Die Adreßdaten befinden sich in einer Textdatei, wobei in jeder Zeile genau ein Datensatz steht
- Die einzelnen Elemente eines Datensatzes sind durch " " (z.B. "Neue Str.") gekennzeichnet, so daß jedes Element eindeutig zugeordnet werden kann.

Implementierungsvarianten

1. Implementieren Sie die obige Aufgabenstellung derart, daß Sie für die Verwaltung der Daten ein großes **Feld fester Größe** (z.B. 1000 Elemente) einer geeigneten Struktur verwenden. Überlegen Sie bei dieser Variante, wie Sie eine Löschoperation am elegantesten ausführen können.
2. Implementieren Sie die Datenhaltung in Form einer **linearen Liste** und nicht mehr durch ein Feld fester Größe! Überlegen Sie, wie Sie hier Operationen wie Einfügen und Löschen realisieren können.

Randaspekte

- Achten Sie auf eine "saubere" Programmierung, d.h. insbesondere
  - strukturierte Programmierung
  - Trennung von "allgemeinen" Abfragefunktionen mit der Datenverwaltung, und Datenablagefunktionen.
  - Konsequentes Einhalten von Einrückregeln, welche Sie auch immer wählen
- Versuchen Sie, die lineare Liste mit ihren Zugriffsfunktionen in einer eigenen Datei und entsprechende Prototypen in einer .h – Datei zu implementieren

Lerninhalte:
- *Umgang/Konzeption/Implementierung von etwas größeren Programmieraufgaben (mehrere Programmdateien)*
- *Dynamische Speicheranforderung*
- *Lineare Liste mit einigen Zugriffsfunktionen*
- *Entwurf einer einfachen Kommandoschnittstelle*