Aufgabenblatt 1: Wiederholung

Kapitel 1: Praktikumsaufgaben

1.1.1: Analyse Codefragmente

Betrachten Sie die folgenden Codefragmente:

```
if( a && b && c)
    x = 1;
else
    {
    if( !b)
        x = 2;
    }
```

```
if( a)
    {
    if( b && !c)
        x = 1;
    else
        {
        if( !a && b)
            x = 2;
        }
    }
else
    x = 3;
```

```
if( a)
    {
    if( b)
        x = 1;
    else
        {
        x = 2;
        if( !c)
            x = 3;
        }
    }
else
    {
    if( b || c)
        x = 4;
    }
```

Bei a, b, c und x soll es sich dabei um int-Variablen handeln. Erstellen Sie für jedes der Codefragmente eine Tabelle, die für alle relevanten Fälle den sich ergebenden Wert der Variablen x angibt. Beachten Sie, dass es Fälle geben kann, in denen der Wert von x undefiniert ist.

1.1.2: Dualdarstellung

Schreiben Sie ein C-Programm, das alle Zahlen zwischen 1 und 4711 ausgibt, in deren Dualdarstellung genau drei Einsen vorkommen. Erstellen Sie dazu u.a. eine Funktion, die zu einer als Parameter übergebenen Zahl die Anzahl der Einsen in der Dualdarstellung ermittelt und an das rufende Programm zurückgibt.

1.1.3: ggT (iterativ und rekursiv)

Den größten gemeinsamen Teiler (ggT) von zwei natürlichen Zahlen können Sie berechnen, indem Sie so lange die kleinere Zahl von der größeren Zahl abziehen, bis beide Zahlen gleich (= ggT) sind.
Erstellen Sie eine

a) iterative
b) rekursive

C-Funktion, die den ggT von zwei natürlichen Zahlen berechnet und das Ergebnis an das rufende Programm zurückgibt.

1.1.4: Die meisten geraden Zahlen

Sie haben einen Array: `int zahlen[200][300]`. Schreiben Sie ein Programm, das den Index der Zeile (Zeilenindex = erster Index) ermittelt, in der die meisten geraden Zahlen stehen.

1.1.5: Datum-String zerlegen

Erstellen Sie eine C-Funktion, die ein Kalenderdatum im Format „TT.MM.JJJJ“, also zum Beispiel „31.01.2011“, als String übergeben bekommt und Tag, Monat und Jahr als Zahlenwerte zurück gibt.

Schreiben Sie ein Programm, das diese Funktion verwendet, um zu berechnen, wie viele Tage zwischen zwei vom Benutzer eingegebenen Daten liegen. Gehen Sie dabei vereinfachend davon aus, dass ein Jahr 360 und ein Monat 30 Tage hat.

Verwenden Sie keine Hilfsfunktionen aus der Runtime-Library.

1.1.6: String vervielfachen

Erstellen Sie eine Funktion:

```
void strmult(int a, char *s)
```

Die Funktion soll einen String `s` entsprechend der Anzahl `a` im Puffer `s` vervielfachen.

Beispiel: a = 3, s = „INF1“; Ergebnis: s = „INF1INF1INF1"

Sie können davon ausgehen, dass das rufende Programm den Zeichenpuffer in der erforderlichen Größe bereitstellt.

1.1.7: Einlesen von der Tastatur (Wdh.)

Erstellen Sie ein Programm, das 10 ganze Zahlen von der Tastatur einliest und die Zahlen anschliessend aufsteigend sortiert ausgibt.

1.1.8: Dynamische Arrays

Verändern Sie Ihr Programm Einlesen von der Tastatur (Wdh.) so, dass der Benutzer eine beliebige Anzahl von Zahlen eingeben kann. Die Eingabe wird mit der Eingabe eines durch Sie festgelegten Zahlenwertes beendet. Das Programm verwaltet die Daten weiterhin in einem Array und vergrößert den Speicher in sinnvollen Schrittgrößen, wenn keine neuen Werte mehr aufgenommen werden können.

1.1.9: Verkettete Liste

Erstellen Sie ein Programm, das von der Tastatur jeweils den Namen einer Person und deren Alter einliest und in einer verketteten Liste nach deren Name sortiert speichert. Die Eingabe wird durch den Benutzer beendet. Nach der Eingabe wählt der Benutzer einen Namen seiner Wahl und es erfolgt die entsprechende Ausgabe der Person und deren zugehöriges Alter.

1.1.10: Klausurtermine

Sie haben eine Liste, in der Sie ihre Klausurtermine abgelegt haben, mit Hilfe der folgenden Datenstrukturen aufgebaut:

```
struct datum
    {
    int tag;
    int monat;
    int jahr;
    };
struct klausurliste
    {
    struct klausurliste *next;
    struct datum termin;
    char fach[30];
    char *pruefer;
    float note;
    };
```

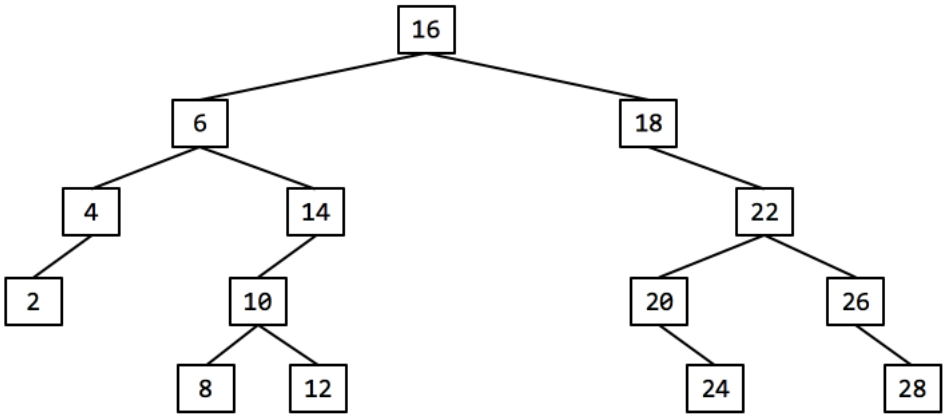
1. Erstellen Sie eine Funktion, die eine Note für ein bestimmtes Fach in die Klausurliste einträgt und den bisher erzielten Notenschnitt zurückgibt.
2. Erstellen Sie eine Funktion, die prüft, ob es Terminüberschneidungen (zwei Klausuren am gleichen Tag) gibt.
3. Erstellen Sie eine Funktion, die eine vollständige Kopie der Liste in umgekehrter Reihenfolge erstellt.

Sorgen Sie dafür, dass alle zur Bearbeitung der Aufgaben erforderlichen Informationen durch die Schnittstellen der Funktionen fließen.

Kapitel 2: Vertiefung und Selbsttest

1.2.1: Aufbau eines Baumes (statisch)

Erstellen Sie ein Programm, dass den folgenden Baum im Speicher aufbaut:



1.2.2: Prüfung sortierter Baum

Sie haben einen Baum mit folgender Knotenstruktur:

```
struct node
    {
    int zahl;
    struct node *left;
    struct node *right;
    };
```

Erstellen Sie eine Funktion, die feststellt, ob der Baum bezüglich des Feldes `zahl` aufsteigend sortiert ist.

1.2.3: Baum

Erstellen Sie ein Programm, das Namen von der Tastatur einliest und in einem Baum sortiert verwaltet. Nach der Eingabe eines Namens wir der Benutzer gefragt, ob der Name zum Baum hinzugefügt werden oder aus dem Baum gelöscht werden soll. Nach Abschluss der jeweiligen Aktion wird der Baum sortiert ausgegeben.