1 Testataufgaben

1.1 Projekt (Woche 3) **(7 Punkte)**

In dieser Woche soll die Implementierung aller Grundfunktionen beendet werden. Mit Ausnahme weniger optionaler Erweiterungen soll das Programm vollständig lauffähig sein.

1. Stellen Sie sicher, dass Ihr Programm alle Teile enthält, die Sie in Ihrem Design-Konzept aus Übungsblatt 08 angegeben haben und demonstrieren Sie dies Ihrem Tutor im Testat.

2. Planen Sie Tests, um Ihr Programm auf Fehler zu prüfen, zum Beispiel im Umgang mit falschen Eingaben oder nicht gefundenen Dateien. Führen Sie diese Tests durch und dokumentieren Sie diese.

3. Erklären Sie den Anderen in Ihrer Gruppe auch in den neuen Programmteilen die Stellen, die strukturierte Datentypen, Dateiverarbeitung und dynamische Speicherverwaltung verwenden.

4. Stellen Sie sicher, dass Ihr Programm alle Leistungsmerkmale realisiert, die Sie sich gemäß Spezifikation vorgenommen haben. Dies sollen Sie Ihrem Tutor abschließend und hauptsächlich im letzten Testat demonstieren. Dabei wird erwaret, dass jedes Gruppenmitglied dem Tutor auf Nachfrage jeden Programmabschnitt erklären kann.

1.2 Übungsaufgabe zur Vorlesung **(3 Punkte)**

Realisieren Sie die Übungsaufgabe zur Vorlesung vom Übungsblatt 09 in Form einer doppelt verketteten linearen Liste anstelle eines Feldes. Der Speicher für die Listenelemente soll dabei dynamisch alloziert werden. Fügen Sie zwei weitere Strukturinstanzen hinzu und löschen Sie das dritte Listenelement. Implementieren Sie zusätzlich eine Funktion, die dazu dient, die Reihenfolge der Elemente umzudrehen. Testen Sie deren Funktionalität im Hauptprogramm.

2 Präsenzaufgaben

1. Eine nicht weiter beschriebene Struktur `Spieler` besitze unter anderem einen Zeiger `next` auf die eigene Struktur. Schreiben Sie ein Funktion, die ein neues Element an den Anfang einer linearen Liste der Struktur `Spieler` einfügt. Die Eingabeparameter der Funktion sollen ein Zeiger auf den Listenanfang und ein Zeiger auf das einzufügende Element sein. Die Funktion soll einen Zeiger auf den neuen Listenanfang zurückgeben.

2. Definieren Sie eine Struktur mit dem Namen `Types`, welche Komponenten mit den Datentypen `char`, `short`, `long` und `double` strukturiert. Schreiben Sie ein Programm, das ein Element der Struktur `Types` erzeugt und die Anfangsadressen der einzelnen Komponenten auf dem Bildschirm ausgibt.

3. Definieren Sie eine Struktur zur Personenverwaltung. Gespeichert werden sollen Name, Alter, und Geschlecht (`'w'` für weiblich und `'m'` für männlich). Schreiben Sie nun eine Funktion, die in einer linearen Liste aus Elementen dieser Struktur den Namen der ältesten Person ausgibt.

4. Eine nicht weiter beschriebene Struktur `Person` besitze unter anderem einen Zeiger `next` auf die eigene Struktur und eine Komponente `Alter`. Schreiben Sie eine Funktion, die alle Personen, die älter als 56 Jahre sind, aus einer linearen Liste mit dieser Struktur entfernt. Diese Funktion hat den Listenanfang als Eingabeargument und gibt einen Zeiger auf den Anfang der geänderten Liste zurück.

5. Schreiben Sie eine Funktion, die eine $n \times n$-Matrix A auf Symmetrie (d.h. $A = A^T$) überprüft. Eingabeparameter sei ein Zeiger auf die Matrix, sowie die Matrixdimension. Die Funktion soll „1“ zurückgeben wenn A symmetrisch ist, sonst „0“.

6. Betrachten Sie folgende einfach verkettete lineare Liste

   mit folgender Listenelementdefinition:

```
struct liste {
  int value;
  struct liste *next;
};
```

   (a) Zwei Zeiger `p1` und `p2` zeigen jeweils auf den Anfang einer entsprechenden Liste. Schreiben Sie eine Funktion `listconcat`, welche die erste Liste an das Ende der zweiten Liste anhängt. Dabei sollen die beiden Zeiger `p1` und `p2` die einzigen Parameter der Funktion sein. Die Funktion soll einen Zeiger auf den Anfang der Gesamtliste zurückgeben. Es darf in der Funktion kein neuer Speicher angefordert werden. Berücksichtigen Sie auch, dass die beiden Listen leer sein können.

   (b) Erstellen Sie eine Funktion `listmax`, die das bezüglich der `value`-Komponente maximale Listenelement findet und seinen Inhalt in einen neuen Datensatz kopiert. Die Funktion soll einen Zeiger auf den neuen Datensatz zurückgeben. Der einzige Parameter der Funktion soll ein Zeiger auf den Listenanfang sein.

7. Zur Lagerhaltung sollen Artikel mit einer Bezeichnung von maximal 20 Zeichen und einer ganzzahligen Artikelnummer in einer doppelt verketteten Liste aufsteigend sortiert nach Artikelnummer gespeichert werden. Eine doppelt verkettete Liste unterscheidet sich von einer linearen Liste lediglich dadurch, dass in der zugehörigen Struktur nicht nur ein Zeiger auf den Nachfolger, sondern auch auf den Vorgänger eines Listenelementes enthalten ist. Dabei ist der Vorgänger des ersten Listenelementes `NULL`.

   (a) Definieren Sie eine Struktur `list` zur Speicherung der Elemente der doppelt verketteten Liste.

   (b) Schreiben Sie eine Funktion `list_insert`, die ein neues Listenelement in eine sortierte Liste an der richtigen Stelle einfügt. Eingabeparameter sind ein Zeiger auf den Listenanfang und ein Zeiger auf das einzufügende Element. Rückgabeparameter ist ein Zeiger auf den Listenanfang der geänderten Liste.

   (c) Schreiben Sie eine Funktion `list_delete`, die ein Listenelement aus einer sortierten Liste löscht. Eingabeparameter sind ein Zeiger auf den Listenanfang und die Artikelnummer des zu löschenden Elementes. Rückgabeparameter ist ein Zeiger auf den Listenanfang der geänderten Liste.