20190114 - Probeklausur

Test und Assessment – Druckansicht

20190114 - Probeklausur

Datum: Mon Jan 21 09:13:25 2019 Maximale Punktezahl: 61

Frage 1 - [a00] Skript "Programmierung mit ANSI-C" 20190114 (1 Punkt) [ID: 47997]

- programmierung_mit_ansi_c.pdf (884.05 KB)

Speichern Sie das oben in der Dateiliste hinterlegte Skript "Programmierung mit ANSI-C" (Datei `programmierung_mit_ansi_c.pdf`) im Ordner `~/klausur` .

Wie lautet die Überschrift von Kapitel 9.1?

- (•) Headerdateien *(1 Punkt)*
- ( ) Variablen *(0 Punkte)*
- ( ) Arithmetische Operatoren *(0 Punkte)*
- ( ) Das Modulkonzept *(0 Punkte)*
- ( ) Adressen und Zeiger *(0 Punkte)*

Frage 2 - [a01] Zeichen und Strings 20190114 (7 Punkte) [ID: 47998]

- a01-testing.c (495 B)

Programmieren Sie eine Funktion zum Säubern von Strings mit folgenden Prototyp.

```
int strclean(char *t);
```

Anforderungen

- Die Leerzeichen und die nicht druckbaren Zeichen werden aus dem übergebenen String entfernt.
- Wenn ein Fehler aufgetreten ist (z.B. wenn `t` der Null-Zeiger ist) wird ein negativer Wert zurückgeliefert, ansonsten wird die Anzahl der aus dem String entfernten Zeichen zurückgeliefert.
- Benutzen Sie zur Erkennung von Leerzeichen und nicht druckbaren Zeichen Funktionen der Standardbibliothek.

Hinweise

- Der übergebenen String wird u.U. kürzer, d.h. die Funktion muss sowohl die Stelle im String von der ein Zeichen gelesen wird, als auch die Stelle an die ein Zeichen geschrieben wird, kennen.
- Vergessen Sie nicht den String mit dem Null-Zeichen abzuschließen.

Lösen - Testen - Abgeben

- Erstellen Sie eine neue Datei, z.B. `a01.c` , mit der Lösung dieser Aufgabe.
- Kompilieren Sie Ihren Lösung.

  ```
  gcc -c -Wall -std=c11 a01.c
  ```

  Binden Sie die von Ihnen erstellte und die für das Testen bereitgestellte Objectcode-Datei zu einem ausführbaren Programm.

  ```
  gcc -o a01 a01.o a01-testing.o
  ```

  Hinweis. Die Minimalanforderung für diese Aufgabe ist, dass das Compilieren und Binden erfolgreich durchgeführt werden kann.
- Testen Sie Ihre Lösung.
- Zum Abgeben der Lösung laden Sie Ihre Datei, mit dem Büroklammer-Symbol, in den nachfolgen Editor.

0:0

Frage 3 - [a02] Iteration 20190114 (8 Punkte) [ID: 47999]

- a02-testing.c (237 B)

Sei x eine ganze Zahl. Ausgehend von x kann eine Folge x_0, x_1, x_2, ... mit folgender Vorschrift gebildet werden.

- Für n < 0 gilt x_n=0
- Für n = 0 gilt x_0=x
- Für n > 0 gilt
  - x_n= $3x_{n-1}$ + 1 falls x_{n-1} ungerade ist

- $x_n = x_{n-1}/2$ falls x_{n-1} gerade ist

Beispiel. Der Startwert x=3 liefert die Folge x_0=3, x_1=10, x_2=5, x_3=16, x_4=8, x_5=4, x_6=2, x_7=1, x_8=4, ...

Programmieren Sie eine Funktion mit Prototyp

```
int sequence(int x, int n);
```

die folgende Anforderungen erfüllt.

- Der Aufruf `sequence(x,n)` liefert den Wert des `n`-te Folgenglied (x_n) bei Startwert `x` (x_0= x).
  Beispiel. Der Aufruf `sequence(3, 4)` liefert den Wert 8, den Wert von x_4 im obrigen Beispiel.
- Die Berechnung erfolgt **iterativ**, d.h. die Funktion ruft sich nicht selbst auf.
- Mögliche Ganzzahlüberläufe werden erkannt und verhindert. Wird ein Überlauf erkannt ist der Funktionswert -1.

Lösen - Testen - Abgeben

- Erstellen Sie eine neue Datei, z.B. `a02.c`, mit der Lösung dieser Aufgabe.
- Kompilieren Sie Ihren Lösung.

  ```
  gcc -c -Wall -std=c11 a02.c
  ```

  Binden Sie die von Ihnen erstellte und die für das Testen bereitgestellte Objectcode-Datei zu einem ausführbaren Programm.

  ```
  gcc -o a02 a02.o a02-testing.o
  ```

  Hinweis. Die Minimalanforderung für diese Aufgabe ist, dass das Compilieren und Binden erfolgreich durchgeführt werden kann.
- Testen Sie Ihre Lösung.
- Zum Abgeben der Lösung laden Sie Ihre Datei, mit dem Büroklammer-Symbol, in den nachfolgen Editor.

0:0

Frage 4 - [a03-1] Rekursion 20190114 (27 Punkte) [ID: 48000]

- a03-testing.h (49 B)

Ein Feld kann mit folgendem Algorithmus absteigend sortiert werden.

Für den Algorithmus wird das Feld sowohl vorwärts (vom Anfang zum Ende des Feldes), als auch rückwärts (von Ende zum Anfang des Feldes) durchlaufen werden. Die Laufrichtung, mit der der Algorithmus beginnt, ist beliebig.

1. Ist das Feld leer oder besteht nur aus einer Feldkomponenten ist es sortiert und der Algorithmus ist beendet.
2. Durchlaufe das Feld in der aktuellen Laufrichtung, vergleiche jeweils die Inhalte zweier aufeinander folgender Feldkomponenten und vertausche die Inhalte, falls der Inhalt der vorderen Feldkomponente kleiner als der Inhalt der hinteren ist.
3. Hat im Schritt 2. kein Vertauschen von Inhalten stattgefunden ist das Feld sortiert und der Algorithmus ist beendet.
4. Wende den Algorithmus, mit **umgekehrter Laufrichtung**, erneut auf das um eine Komponente kleinere Feld an.
   - War die Laufrichtung **vorwärts** wird die **letzte** Komponente weggelassen.
   - War die Laufrichtung **rückwärts** wird die **erste** Komponente weggelassen.

Programmieren Sie eine **rekursive** Funktion, d.h. die Funktion muss sich selbst aufrufen, mit Prototyp

```
int recSort(int *start, int *end, int reverse);
```

die das oben beschriebene Sortierverfahren anwendet, um ein Feld von `int`-Werten zu sortieren. Die Deklaration der Funktion finden Sie in der Headerdatei `a03-testing.h` aus der Dateiliste.

Anforderungen

- Integrieren Sie folgende Zeile, z.B. als erste Zeile, in Ihre Lösung.

  ```
  #include "a03-testing.h"
  ```

- Der Zeiger `start` verweist auf die erste Feldkomponente der Zeiger `end` **hinter** die letzte Feldkomponente des zu sortierenden Felds.
- Der Wert `reverse` wird als Wahrheitswert behandelt und gibt die Laufrichtung an, gilt `reverse` ist die Laufrichtung rückwärts.
- Der Rückgabewert der Funktion ist die Anzahl der beim Sortieren durchgeführten Vertauschungen von benachbarten Feldkomponenten.
- Verwenden Sie **maximal eine Schleife**. Das ist möglich, weil die Bedingung, unter der die Schleife fortfährt, für beide Laufrichtungen gleich ist. Unabhängig von der Laufrichtungen müssen die betrachteten benachbarten Komponenten innerhalb des Feldes liegen.
- Abhängig von `reverse` wird der Startpunkt für das Durchlauf des Feldes durch eine lokale Variable repräsentiert.

Lösen - Testen - Abgeben

- Speichern Sie die oben in der Dateiliste hinterlegte Datei `a03-testing.h` im Ordner `~/klausur`.
- Erstellen Sie im Ordner `~/klausur` eine neue Datei, z.B. `a03-1.c`, mit der Lösung dieser Aufgabe.
- Kompilieren Sie Ihren Lösung.

```
gcc -c -Wall -std=c11 a03-1.c
```

Hinweis. Die Minimalanforderung für diese Aufgabe ist, dass das Compilieren erfolgreich durchgeführt werden kann.

- Testen Sie Ihre Lösung mit der Lösung von Aufgabe `[a03-2]`.
- Zum Abgeben der Lösung laden Sie **nur** die von Ihre erstellte Datei, z.B. `a03-1.c`, mit dem Büroklammer-Symbol in den nachfolgen Editor.

0:0

Frage 5 - [a03-2] Hauptprogramm 20190114 (18 Punkte) [ID: 48001]

- a03-testing.h (49 B)

Es steht eine Funktion mit Prototyp

```
int recSort(int *start, int *end, int reverse);
```

zur Verfügung, die das Feld, auf dessen erste Komponente der Zeiger `start` und hinter dessen letzte Komponente der Zeiger `end` verweist, absteigend sortiert, wobei der Wert von `reverse` beliebig ist. Die Deklaration der Funktion finden Sie in der Headerdatei `a03-testing.h` aus der Dateiliste.

In der der Standardheaderdatei `stdlib.h` ist die Funktion `atoi` mit folgendem Prototyp deklariert.

```
int atoi(const char *str);
```

Der Funktion `atoi` wird ein Zeiger auf einen `char`-Vektor übergeben, der eine Zeichenkette enthält. Falls möglich wird die Zeichenkette in einen `int`-Wert umgewandelt und dieser wird zurückgeliefert. Ist die Umwandlung nicht möglich, wird 0 zurückgeliefert.

Programmieren Sie eine `main`-Funktion (ohne Benutzereingaben), die auf der Kommandozeile Argumente übergeben bekommt. Alle übergebenen Argumente werden in `int`-Werte umwandelt und in einem Feld passender Länge gespeichert. Anschließend wird das Feld sortiert.

Anforderungen

- Integrieren Sie folgende Zeile, z.B. als erste Zeile, in Ihre Lösung.
  ```
  #include "a03-testing.h"
  ```
- Enthält die Kommandozeile keine Argumente wird die Funktion `recSort` mit einem Null-Zeigern für `start` und `end`, sowie einem beliebigen Wert für `reverse` aufgerufen und der Rückgabewert der Funktion wird ausgegeben.
- Für das Feld, das die umgewandelten Argumente der Kommandozeile aufnimmt, wird **Speicher auf dem Heap** reserviert, dabei wird nicht mehr Speicher als nötig angefordert.
- Für die Umwandlung der auf der Kommandozeile übergebenen Argumente wird die Funktion `atoi` verwendet. Es wird für jedes Argument den Rückgabewert von `atoi` als zugehörigen `int`-Wert benutzt. Es muss nicht überprüft werden, ob das Argument tatsächlich eine ganze Zahl repräsentiert.
- Das Feld, das die umgewandelten Argumente der Kommandozeile enthält, wird ausgegeben.
- Mit der Funktion `recSort` wird das Feld sortiert und der Rückgabewert wird ausgegeben.
- Das sortierte Feld wird ausgegeben.
- Speicher, der auf dem Heap reserviert wurde, wird auch wieder freigegeben.

Lösen - Testen - Abgeben

- Speichern Sie die oben in der Dateiliste hinterlegte Datei `a03-2-testing.h` im Ordner `~/klausur`.
- Erstellen Sie im Ordner `~/klausur` eine neue Datei, z.B. `a03-2.c`, mit der Lösung dieser Aufgabe.
- Kompilieren Sie Ihren Lösung.
  ```
  gcc -c -Wall -std=c11 a03-2.c
  ```

  Hinweis. Die Minimalanforderung für diese Aufgabe ist, dass das Compilieren erfolgreich durchgeführt werden kann.
- Testen Sie Ihre Lösung mit der Lösung von Aufgabe `[a03-1]`.
- Zum Abgeben der Lösung laden Sie **nur** die von Ihre erstellte Datei, z.B. `a03-2.c`, mit dem Büroklammer-Symbol in den nachfolgen Editor.

0:0