Aufgaben zu Grundlagen der Informatik II

Steffen Oesterwind

Sommersemester 2025

Übung 1

Wiederholung von Bitoperationen. (Hinweis: includiere `stdint.h` für die `uint*_t` Datentypen)

Aufgabe 1.1

In einer Variable vom Typ `uint32_t` soll an die Bits 3 bis 9 ein Wert geschrieben werden. Wie ist dabei vorzugehen? Implementiere eine Funktion mit der Signatur `void setbits(uint32_t *x, uint8_t value)`, wobei `x` auf die zu modifizierende Variable zeigt und `value` der zu setzende Wert ist.

Aufgabe 1.2

Eine Bitmaske bezeichnet eine Zahl, in der in binärer Darstellung eine bestimmte Anzahl an Bits ohne Lücke gesetzt sind. Schreibe eine Funktion, die Position und Anzahl zu setzender Bits erhält und die daraus resultierende Bitmaske als `uint32_t` zurück gibt. z.B. Pos: 3, Anzahl: 5 → `0b11111000`

Aufgabe 1.3

Erweitere die Fuktion aus 1.1 dergestalt, dass Position und Anzahl der zu manipulierenden Bits als Argumente übergeben werden können.

Übung 2

Rekursion und Komplexität.

Aufgabe 2.1

Schreibe eine Funktion, welche rekursiv die Fakultät einer Zahl berechnet.

Aufgabe 2.2

Schreibe eine Funktion, welche rekursiv die Fibonacci-Zahl `n` berechnet.

Aufgabe 2.3

Schreibe eine Funktion, die rekursiv und ohne Verwendung von Schleifen das größte Element in einem Array vom Typ `int` findet. Dazu bietet es sich an, das Array rekursiv als „Erstes Element“ und `Rest` aufzufassen, wobei der Rest wiederum aus erstem Element und Rest besteht. Die Rekursion endet, wenn der Rest die Länge 0 hat.

Aufgabe 2.4

Von welcher Komplexitätsklasse ist eine Funktion, welche die Summe aller Elemente eines Arrays bildet?

Aufgabe 2.5

Schreibe eine Funktion, die überprüft, ob in einem Array von Typ `int` ein Wert doppelt vorkommt. Welche Komplexitätsklasse hat diese Funktion?

Übung 3

Such- und Sortieralgorithmen.

Aufgabe 3.1

Implementiere eine Funktion, die mittels linearer Suche einen Wert vom Typ `int` in einem Array aus Elementen vom Typ `int` sucht und die Position, an der das Element gefunden wurde, zurück gibt.

Aufgabe 3.2

Implementiere eine Funktion, die rekursiv mittels binärer Suche einen Wert vom Typ `int` in einem Array aus Elementen vom Typ `int` sucht und die Position, an der das Element gefunden wurde, zurück gibt.

Aufgabe 3.3

Implementiere eine Funktion, die ein Array aus Elementen vom Typ `int` mittels Bubblesort sortiert.

Aufgabe 3.4

Implementiere eine Funktion, die ein Array aus Elementen vom Typ `int` mittels Selectionsort sortiert.

Aufgabe 3.5

Implementiere eine Funktion, die ein Array aus Elementen vom Typ `int` mittels Insertionsort sortiert.

Übung 4

Mehr Sortieralgorithmen und Dateioperationen.

Aufgabe 4.1

Implementiere eine Funktion, die ein Array aus Elementen vom Typ `int` mittels Mergesort sortiert. Der Sortieralogrithmus soll rekursiv implementiert werden.

Aufgabe 4.2

Implementiere eine Funktion, die ein Array aus Elementen vom Typ `int` mittels Quicksort sortiert. Der Sortieralogrithmus soll rekursiv implementiert werden.

Aufgabe 4.3

Schreibe ein Programm, das eine Datei mit vom Anwender interaktiv einzugebendem Dateinamen öffnet und deren gesamten Inhalt (als ASCII), sowie die Dateigröße ausgibt. Lege die Datei zunächst an.

Aufgabe 4.4

Schreibe ein Programm, das eine Datei mit vom Anwender einzugebenden Dateinamen öffnet und die hexadezimale Repräsentation des Dateiinhalts als ASCII in eine neue Datei schreibt, deren Namen aus dem der Quelldatei durch Anhängen von `.hex` gebildet wird, z.B.: `hallo.txt` → `hallo.txt.hex`
Dabei sollen 16 Oktette je Zeile ausgegeben werden, und jede Zeile soll mit der relativen Adresse in hexadezimaler Darstellung, bezogen auf den Anfang der Quelldatei, beginnen. Beispielhafte Ausgabe:

```
00000000 63 53 72 68 69 65 65 62 65 20 6e 69 50 20 6f 72
00000010 72 67 6d 61 2c 6d 64 20 73 61 65 20 6e 69 20 65
00000020 61 44 65 74 20 69 69 6d 20 74 6f 76 20 6d 6e 41
```

Übung 5

Aufgabe 5.1

Implementiere eine verkettete Liste, die Zahlen vom Typ `int` speichert. Deklariere dazu zunächst eine geeignete Datenstruktur und implementiere dann folgende Funktionen:

- Anlegen einer Liste mit einem Element
- Anhängen eines Elementes an eine Liste
- Entfernen eines Elementes
- Löschen der Liste
- Ausgeben aller Elemente der Liste
- Elemente zählen
- Element an Position `i` finden

Aufgabe 5.2

Implementiere einen Stapelspeicher, der Zahlen vom Typ `int` speichert. Deklariere dazu zunächst eine geeignete Datenstruktur und implementiere dann folgende Funktionen:

- Anlegen eines leeren Stapels
- Ein Element auf den Stapel legen
- Ein Element vom Stapel entfernen und dessen Wert zurückgeben
- Löschen des Stapels
- Löschen aller Elemente
- Elemente zählen

Aufgabe 5.3

Implementiere einen FIFO-Speicher (per Ringbuffer), der Zahlen vom Typ `int` speichert. Deklariere dazu zunächst eine geeignete Datenstruktur und implementiere dann folgende Funktionen:

- Anlegen eines FIFOs von angegebener Größe
- Ein Element in den FIFO schreiben
- Ein Element aus dem FIFO lesen
- Löschen des FIFOS
- Löschen aller Elemente
- Freie Speicherplätze zählen

Übung 6

Aufgabe 6.1

Erstelle eine Struktur zum Speichern von Prüfungsergebnissen. Dabei sollen Vorname, Nachname, Matrikelnummer und Note gespeichert werden können. Für die Namensfelder sollen jeweils 16 Zeichen zur Verfügung stehen. Schreibe eine Funktion, die ein Array aus diesen Strukturen nach Vor-/Nachname, Note, oder Matrikelnummer sortiert. Verwende hierfür die Funktion `qsort` aus der C-Standardbibliothek.

Übung 7

Aufgabe 7.1

Modifiziere deine Implementation der verketteten Liste aus Aufgabe 5.1 so, dass anstatt Daten vom Typ `int` Daten beliebigen Typs gespeichert werden können. Speichere dazu einen Voidpointer und erweitere die Funktionen zum Löschen eines Elementes um einen Funktionspointer, der den Speicher, auf den der Voidpointer zeigt, geeignet freigeben kann. Die Funktion zum Ausgeben aller Elemente soll ebenfalls einen Funktionspointer erhalten, der ein einzelnes Element ausgibt. Implementiere dies beispielhaft für den Datentyp `double`.

Aufgabe 7.2

Implementiere eine Klasse `Greeter` in Python, die Grüße ausgibt. Im Konstruktor soll der Gruß („Hi“, „Hallo“, „Guten Tag“, ...) übergeben werden. Die Methode `greet` soll als Parameter den Namen erhalten.

```
>>> greeter = Greeter("Hallo")
>>> greeter.greet("Archimedes")
Hallo Archimedes
```

Übung 8

Aufgabe 8.1

Implementiere ein Programm in Python, welches Zeilen aus einer vom Benutzer einzugebenden Datei einliest, und jede Zeile mit „`Hallo `“ vorangestellt in eine neue Datei schreibt.

Aufgabe 8.2

Schreibe ein Programm in Python, welches aus einer Textdatei (Format siehe unten) Koordinaten einliest, jede davon in ein Objekt der zu implementierenden Klasse `Vector` speichert und diese in einer Liste ablegt. Die Klasse `Vector` soll neben Subtraktion auch den Operator zur Bestimmung des Absolutwertes implementieren. Bei letzterem soll der Betrag des Vektors zurückgegeben werden ($\sqrt{x^2 + y^2}$).

```
1.2 0
-4 5
-2 3
1 5.6
```

Zur Überprüfung des Einlesens sollen anschließende alle Elemente der Liste ausgegeben werden. Abschließend ist die Gesamtlänge des durch die Vektoren beschriebenen Linienzuges auszugeben.