Prof. Dr. A. Weber
Dr. Hassan Errami

Universität Bonn
Institut für Informatik II
14.11.2018

Wintersemester 2018/2019

Übungen zur Vorlesung
Algorithmisches Denken und imperative Programmierung (BA-INF-014)
Aufgabenblatt 5

Zu bearbeiten bis: 30.11.2018

Hinweis: In der Woche vom 26.11. - 29.12. findet die erste Online-Aufgabe zu Ihren gewohnten Tutorienzeiten statt, bei der Sie eine vor Ort gestellte Aufgabe lösen müssen. Daher haben Sie für die Bearbeitung dieses Übungszettels zwei Wochen Zeit, die Abgabefrist ist der 30.11.18.

Aufgabe 1 *(Umgang mit Strings - 3+4=7 Punkte)*

a) Palindrome sind Wörter, die vorwärts und rückwärts gelesen gleich sind, z.B. Ebbe.

- Schreiben Sie eine Funktion, die als Argument einen String erhält und diesen umgekehrt ausgibt.
- Schreiben Sie eine Funktion, die zurückgibt, ob ein als Argument übergebener String ein Palindrom ist.

b) Die Caesar-Verschlüsselung ist ein einfaches symmetrisches Verschlüsselungsverfahren.

- Implementieren Sie die Funktionen *char*encrypt(char*s)* bzw. *char*decrypt(char*s)*, die einen übergebenen String mittels ROT13 verschlüsseln bzw. entschlüsseln. Dabei wird jeweils jeder Buchstabe durch seinen dreizehnten Nachfolger im Alphabet ersetzt. Sie müssen dabei nur das lateinische Alphabet mit seinen 26 Buchstaben betrachten, alle anderen Zeichen lassen Sie unverändert. **Tipp:** Betrachten Sie für die Lösung der Aufgabe unbedingt eine ASCII-Tabelle. Testen Sie Ihre Funktion, indem Sie einen vom Benutzer Ihres Programms eingegebenen String einlesen, diesen verschlüsseln, ausgeben und direkt wieder entschlüsseln.
- Schreiben Sie nun eine Funktion *char*encrypt(char*s, int k)* sowie *char*decrypt(char*s, int k)*, die um eine beliebige Stelle k, $k \in \mathbb{N}_0, k < 27$ verschiebt.

Aufgabe 2 *(Dynamische Arrays - 5+5+2+1=13 Punkte)*

Ein dynamisches Array hat die Eigenschaft, dass man flexibel am Ende des Arrays neue Elemente anfügen kann. Solange noch Platz vorhanden ist kann der Speicher des intern vorhandenen statischen Arrays verwendet werden. Wenn wir an ein Array ein Element anhängen wollen und kein freier Platz mehr vorhanden ist, müssen wir neuen Speicher allokieren, der genug Platz bietet. Die Werte aus dem alten Array müssen dann in den neuen Speicher umkopiert werden. Danach kann das neue Element hinten angefügt werden.
Bei der naiven Implementierung des dynamischen Arrays wird jeweils Speicher der exakt benötigten Größe allokiert; eine andere Strategie ist es, die Größe des statischen Arrays jeweils zu verdoppeln, wenn der Speicherplatz nicht ausreichend war.
Beachten Sie im Folgenden, dass Sie nicht nur Speicher für ein neues Array allokieren, sondern den Speicher für das alte auch wieder freigeben, sobald dieses nicht mehr benötigt wird.

a) Schreiben Sie einen Datentyp *DynArray* zur Speicherung von dynamischen int-Arrays, welches auch die aktuelle und maximale Größe des Arrays speichert. Schreiben Sie eine Funktion *dyn_array_add*, welche die oben skizzierte Verdopplungsstrategie beim Einfügen eines Elements realisiert, wenn nicht mehr genug Speicher vorhanden ist.
b) Schreiben Sie analog einen Datentyp *DynArrayMin* und eine Funktion *dyn_array_min_add*, die die Strategie realisiert, dass jeweils nur der benötigte Speicherplatz allokiert wird.
c) Schreiben Sie eine Methode, die alle Elemente eines übergebenen DynArrays/DynArrayMins auf der Konsole ausgibt.
d) Testen Sie Ihre Implementierung. Welche Vorteile bieten die verschiedenen Implementierungen in welchen Fällen?