Teil 2: Grundlagen der Programmierung

Nutzen Sie bitte ausschließlich die Programmierumgebung VSC

Algorithmen

Aufgabe 2.1: Algorithmus mit PAP darstellen und analysieren

Entwickeln Sie für das `Eierkochen` (5, 7 und 10 Minuten Eier) **einen** Algorithmus, stellen diesen als PAP dar, markieren Anweisungen, Operatoren und Kontrollstrukturen und benennen diese.

Aufgabe 2.2: Algorithmus mit Pseudocode darstellen und analysieren

Nehmen Sie an, dass Sie eine Messwertreihe in einer geeigneten Variablen (genauer einem *Array*) `measSeries` vorliegen haben. Die Variable hat dabei die Länge `numberMeas` (entspricht der Anzahl der Messwerte) und Sie sollen für die Messwertreihe den Mittelwert und den Maximalwert bestimmen. Um einzelne Messwerte aus der Variable auszulesen, können Sie diese im Pseudocode per `measSeries[i]` ansprechen, wobei `i = 0,…, (numberMeas – 1)`. Entwickeln Sie hierfür einen Algorithmus, stellen diesen mit Pseudocode dar, markieren Anweisungen, Operatoren und Kontrollstrukturen und benennen diese.

Aufgabe 2.3: Algorithmus mit Struktogramm darstellen

Überlegen Sie sich für ein alltägliches Problem einen Algorithmus zur Problemlösung und stellen diesen als Struktogramm dar, markieren Anweisungen, Operatoren und Kontrollstrukturen und benennen diese. Ihre Kommilitonen sollen herausfinden, welches alltägliche Problem Ihr Algorithmus löst. Daher verwenden Sie bitte nur Probleme, die auch alle anderen kennen und die nicht bereits in der Vorlesung besprochen wurden oder Teil dieses Übungsblatts sind.

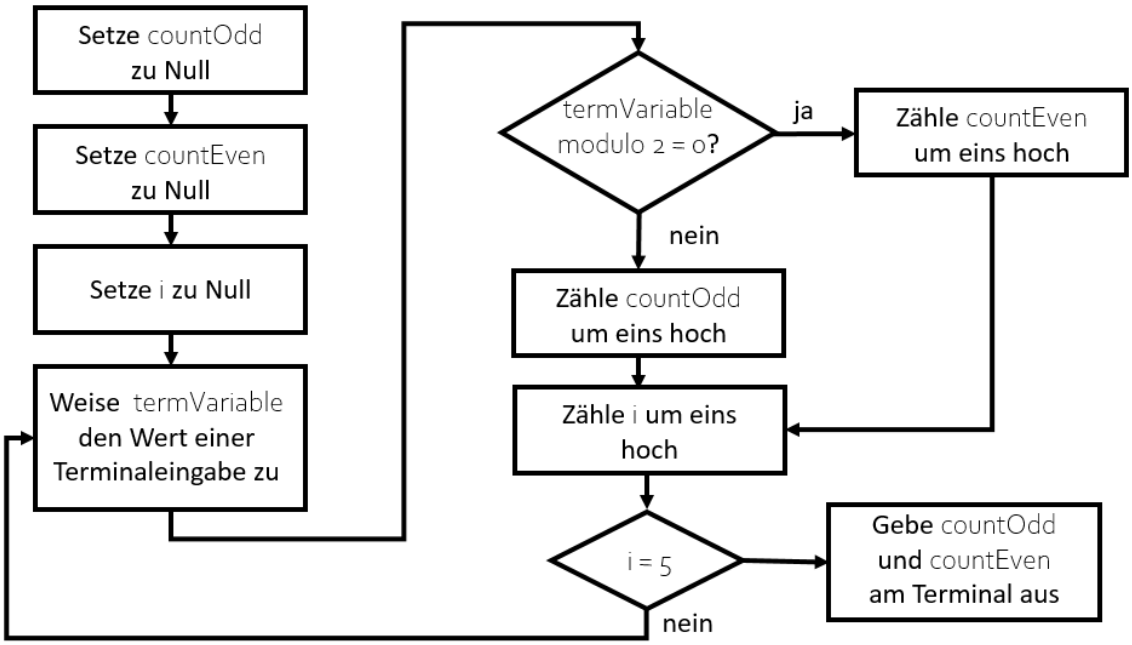
Werkzeuge der Programmierung

Aufgabe 2.4: Multiplikation auf Ebene des Maschinencodes

Multiplizieren Sie die Dezimalzahlen "42" und "18" mit den Operationen der x86 Prozessorgruppe. Entwerfen Sie hierzu einen passenden RAM Auszug und erläutern basierend daran den Programmablauf. Recherchieren Sie die hierzu notwendigen Operationen im Internet (z.B. ChatGPT) und verwenden eine RAM Darstellung, wie in Kapitel 2 "Werkzeuge der Programmierung / Der Computer – ein einfaches Beispiel" dargestellt.

Aufgabe 2.5: Kompilierung eines einfachen Programms

Setzen Sie den folgenden PAP als C Programm um, kompilieren Ihren Code und zeigen sowohl Terminalein-/-ausgabe als auch die Hinweise des Compilers.



Aufgabe 2.6: Header Datei untersuchen und einbinden

Öffnen Sie die Header Datei myHeader26.h (im selben Ordner wie dieses Aufgabenblatt) Fügen Sie ein eigenes Makro ein, binden die HeaderDatei in die Quellcode Datei main26.c ein, rufen dort Ihr erzeugtes Makro auf, kompilieren die Dateien und führen sie aus. Zeigen Sie das Vorgehen.

Aufgabe 2.7: Standardbibliotheken über Header Datei einbinden

Entwerfen Sie einen Algorithmus, der `nRoll` mal einen Würfelwurf simuliert und die Summe der Augenzahlen summiert und am Ende (wenn `nRoll` mal gewürfelt wurde) die Summe ausgibt. Recherchieren Sie im Internet nach einer geeigneten Bibliothek (bzw. einer Funktion daraus), um eine Zufallszahl zu generieren. Setzen Sie diesen Algorithmus in C um. Die Variable `nRoll` kann entweder im Programm definiert werden oder aber über `scanf()` eingelesen werden.

Programmiersprache C

Aufgabe 2.8: ASCI/ANSI

Weisen Sie in einem C Programm nach, dass das Zeichen "g" tatsächlich als 0b1100111 abgelegt wird. Erweitern Sie dieses Programm, um zu bestimmen, wie "G" abgespeichert wird und überprüfen anhand einer weiteren Kleinbuchstaben-Großbuchstaben-Paarung, ob auch hier der Abstand 32 beträgt.

Aufgabe 2.9: Endlosschleife umschreiben

Wie kann man folgenden Programmteil so umschreiben, dass keine Endlosschleife verwendet wird. Erweitern Sie den Code dann so, dass nach Verlassen der while-Schleife ausgegeben wird, wie oft diese durchlaufen wurde.

```
int zahl, summe = 0;
printf ("Summenberechnung\nBeenden der Eingabe mit -1\n");
while (1)
{
    printf("Bitte Wert eingeben > ");
    scanf("%d", &zahl);
    if (zahl == -1)
       break;
    else
       summe += zahl;
}
printf("Die Summe aller Werte beträgt: %d\n", summe);
```

Aufgabe 2.10: Zweierpotenzen

Entwickeln Sie einen Algorithmus (Darstellung bleibt Ihnen überlassen), der in einer Schleife 2er Potenzen 2^n ausgibt für n=0,...,10. Verwenden Sie dabei nicht die `pow()` Funktion.

Erwartete Ausgabe:
$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
...
$2^9 = 512$
$2^{10} = 1024$

Aufgabe 2.11: Fakultät berechnen

Schreiben Sie ein Programm, das eine Zahl n einliest, dazu die Fakultät n! berechnet und das Ergebnis am Terminal ausgibt. Bis zu welcher Zahl n liefert das Programm das richtige Ergebnis, ab wann liefert es nicht mehr das richtige Ergebnis und woran liegt das?

Testdaten:
Eingabe n: 10

Erwartete Ausgabe:
10! = 3.628.800

Aufgabe 2.12: n bei gegebener n! bestimmen

Schreiben Sie ein Programm, das eine Zahl einliest und überprüft, ob diese Zahl das Ergebnis einer Fakultätsberechnung n! ist, mit n=0,…,10? Geben Sie in dem Fall n aus, andernfalls -1.

Testdaten:
Eingabe 1: myFak: 3.628.800

Eingabe 2: myFak: 3.628.799

Erwartete Ausgabe 1: n = 10

Erwartete Ausgabe 2: n = -1

Aufgabe 2.13: Dezimalzahlzerlegung

Gegeben ist eine Zahl z vom Datentyp int, deren Wert zwischen 0 und 999 liegt.

Ermitteln Sie den Wert der Hunderter-, Zehner- und Einerziffer von z und speichern Sie diese in den drei Variablen zH, zZ und zE ab. Geben Sie die jeweiligen Werte auf dem Bildschirm aus.

Testdaten:
Eingabe z: 15

Erwartete Ausgabe:
15 = 0 x 100 + 1 x 10 + 5 x 1

Aufgabe 2.14: Umwandlung Dezimal- in Hexadezimalzahl

Entwickeln sie einen Algorithmus zur Umrechnung einer positiven Dezimalzahl in eine Hexadezimalzahl und setzen sie diesen in der Programmiersprache C um.

- Lesen Sie die umzurechnende Zahl als integer-Wert von der Tastatur ein.
- Ermitteln sie die Hexadezimalziffern und speichern sie diese als ASCII-Zeichen in den vier char-Variablen HexZiffer3, HexZiffer2, HexZiffer1 und HexZiffer0 ab
- Geben sie die Ziffern auf dem Bildschirm aus.