

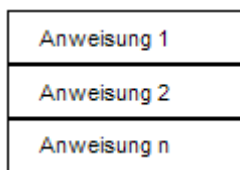
SAE

Zusammenfassungen für das Fach SAE vorbereitend für die Abschlussprüfung Teil 1.

- Struktogramme
- JSON / XML
- C
- Pseudocode
- Übungsaufgabe 1
- Übungsaufgabe 2

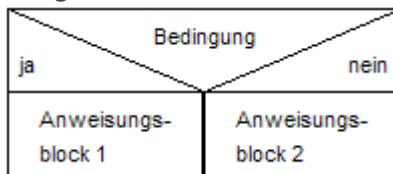
Struktogramme

- auch als Nassi-Shneiderman-Diagramme bekannt
- Ziel: Darstellung eines Programms unabhängig von der Programmiersprache
- Symbole
 - mit Hilfe dieser Symbole lässt sich der Ablauf eines Programms beschreiben
 - fast alle Symbole lassen sich beliebig ineinander verschachteln
 - Prozess Symbol / Process Symbol: Anweisungen werden nacheinander von oben nach unten durchlaufen



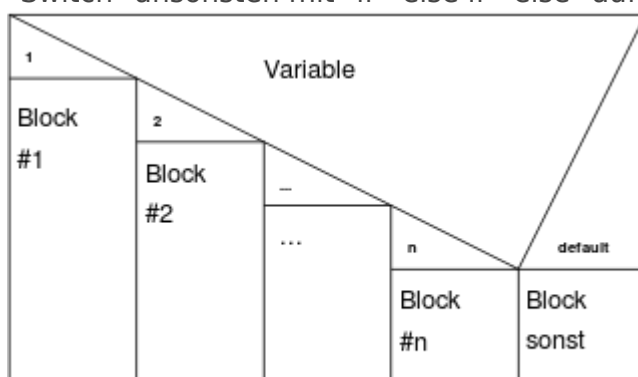
```
# Anweisungen in Python
a = input("Zahl eingeben") # Anweisung 1
b = 5 * a # Anweisung 2
print(b) # Anweisung 3
```

- Verzweigung / Decision Symbol: Bedingung wird geprüft, wenn Sie zu trifft wird "ja" ausgeführt, andernfalls "nein". Kann verschachtelt sein.



```
# Verzweigung in Python
if (a == 5): # Prüfung der Variablen a -> Ergebnis True oder False
    print("a ist fünf") # Wenn Prüfung True, wird dieser Block ausgeführt
else:
    print("a ist nicht fünf") # Wenn Prüfung False, wird dieser Block ausgeführt
```

- Sonderfall: Case-Statement: Inhalt der Variablen wird geprüft und entsprechender Fall wird ausgeführt. Manche Programmiersprachen haben "Switch" ansonsten mit "if - else if - else" auflösbar.



// Mit Switch in C aufgelöst

case (a) {
// Angabe welche Variable geprüft werden soll

1: printf("a hat den Wert eins");
// Auswahl entsprechend der Prüfung

2: printf("a hat den Wert zwei");

3: printf("a hat den Wert drei");

default: print("a ist größer drei");
// Sollte keine Prüfung zutreffen wird dieser Fall ausgeführt

};

// Mit if - else if - else aufgelöst

if (a == 1) {
// Prüfung der Variable a -> Ergebnis True / False

printf("a hat den Wert eins");
// Block der True als Ergebnis hat wird ausgeführt

} else if (a == 2) {

printf("a hat den Wert zwei");

} else if (a == 3) {

printf("a hat den Wert drei");

} else {

printf("a ist größer drei");
// Wird ausgeführt sollte kein Block True sein

};

- Schleifen oder Wiederholungsstruktur: Struktur die solange durchlaufen wird, bis Endbedingung erfüllt ist
 - Kopfgesteuerte - Schleifen (z.B. while und for): Bedingung wird vor ersten durchlauf geprüft und nur dann betreten, wenn Bedingung zutrifft. Kann also auch nicht durchlaufen werden, wenn Endbedingung direkt zutrifft.

solange Bedingung wahr

Anweisungs-
block 1

// Kopfgesteuerte while-Schleife

while (i <= 10) {
// Die Variable i wird geprüft, sollte sie bereits größer 10 sein, wird Schleife nicht ausgeführt

printf("%i\n", i);
// Wird ausgeführt sollte i kleiner oder gleich 10 sein

i++;

}

// Kopfgesteuerte for-Schleife

for (i = 0; i <= 10; i++) {
// Sonderfall i läuft von 0 bis 10

printf("%i\n", i);

}

- Fußgesteuerte - Schleife (z.B. do-while): Endbedingung wird nach dem ersten Durchlauf geprüft und wenn diese Zutrifft, wird die Schleife beendet. Sollte sie nicht zu treffen, wird die Schleife solange durchlaufen bis die Bedingung zutrifft.



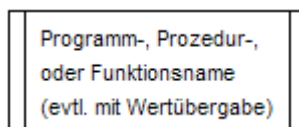
```
// Fussgesteuerte-Schleife
do { // Schleife wird betreten
    printf("%i\n", a); // Schleife wird ausgeführt
    a++;
} while (a <= 10); // Prüfung und sollte a größer 10 sein, wird Schleife beendet.
Ansonsten wird Schleife erneut ausgeführt
```

- Funktion und Funktionsaufruf: Um nicht mehrfach den gleichen Code schreiben zu müssen, kann man Funktionen schreiben die man immer wieder aufrufen kann. Dazu muss man zwei Dinge beachten.

Zum Einen muss die Funktion aufgerufen werden ggf. Parameter übergeben werden. Zum Anderen wird die Funktion als eigenes Struktogramm geschrieben und am Anfang die möglichen Übergabeparamet genannt.

Sollte die Funktion Rückgabeparameter haben sind diese mit **Rückgabe** zu markieren (nicht Ausgabe).

- Funktionsaufruf



```
// Möglicher Funktionsaufruf in C

// Funktionsdefinition
int berechne(int zahl1, int zahl2) { // Funktion bekommt zwei Integer übergeben und gibt
    einen Integer zurück
    int ergebnis; // lokale Variable
    ergebnis = zahl1 + zahl2; // Berechnung von ergebnis
    return ergebnis; // Rückgabewert
}

//Hauptfunktion
void main(void) {
    int a; // Deklaration von Variablen
    int b;
```

```
int c;  
a = 4; [ ][ ][ ][ ][ ][ ][ ][ ] // Zuweisung von Wert an Variable  
b = 5;  
c = ergebnis(a, b); [ ][ ][ ][ ] // Aufruf von Funktion "berechne" Übergabeparameter sind Werte  
von a, b. Rückgabe von Funktion wird zugewiesen  
printf("%i\n", c); [ ][ ][ ][ ] // Ausgabe von c  
}
```

- main

Ilt. Hauptfunktion (main) und

main
Deklaration von a
Deklaration von b
Deklaration c
a <- 4
b <- 5
c <- berechne(a, b)
Ausgabe von c

berechne Parameter zahl1, zahl2
Deklariere ergebnis
ergebnis <- zahl1 + zahl2
Rückgabe ergebnis

JSON / XML

JavaScript Object Notation und Extensible Markup Language

- beides sind Vereinbarungen (Notationen) die dem Austausch von Daten zwischen Anwendungen und Computersystemen
 - Webseiten bekommen z.B. die Informationen die an bestimmten Stellen über JSON oder XML, damit nicht immer der Code der Webseite angefasst werden muss, sondern einfach die erhaltene Übertragung ausgelesen werden muss und die "Webseite" dann weiß wo sie was eintragen muss
- Vorteil dabei ist, dass es sich um ein Datenformat handelt, das sowohl vom Computer als auch Menschen lesbar ist
- für beide Formate gibt es Parser in den üblichen Programmiersprachen
 - Parser sind Programme oder Programmteile die eine Datei auslesen und interpretieren können
- XML arbeitet mit Feldern die geöffnet werden `<feldöffnung>` und geschlossen werden `</feldöffnung>`
 - zwischen solchen Feldern können weitere Felder geöffnet werden, dabei entwickelt sich eine Hierarchie der Daten. Felder die zwischen zwei anderen Feldern liegen, sind diesem untergeordnet
- JSON erinnert an die Dictionary-Notation aus Python. Dabei werden Bereiche mit geschweiften Klammern geöffnet und Feldnamen durch "Feldname" gekennzeichnet. Mehrere solcher Felder werden durch Komma getrennt

Exkurs: Notation

- Notation: Art und Weise Informationen darzustellen. Dabei werden bestimmten Zeichen bestimmte Eigenschaften zu geordnet. Eine Notation die wir alle kennen ist die Infix Notation. Also die Art und Weise wie wir Rechnungen darstellen:
 - Infix Notation: $3 + 4$
- Alternativ dazu existieren auch folgende Notationen:
 - Prefix Notation: $+ 3 4$
 - Postfix Notation: $3 4 +$
- Wer sich wundert warum es so was gibt, die Postfix Notation kommt gerne bei Berechnungen die mit Hilfe des Computers gemacht werden zum Einsatz, da sie schneller zu verarbeiten ist und damit die Rechenzeit signifikant verkürzt.

- Beispiel für XML:

```
<?xml version="1.0" standalone="yes" ?>
- <shop location="Birmingham" size="Large">
- <food>
  <Name>Apple</Name>
  <type>fruit</type>
  <cost>15</cost>
</food>
- <food>
  <Name>Carrot</Name>
  <type>vegetable</type>
  <cost>10</cost>
</food>
</shop>
```

- Beispiel für JSON:

```
{
  "@context": "http://schema.org/",
  "@type": "SearchAction",
  "object": {
    "@type": "Event",
    "location": {
      "@type": "Place",
      "geo": {
        "@type": "GeoCoordinates",
        "latitude-input": "optional",
        "longitude-input": "optional"
      }
    },
    "organizer": {
      "@type": "Organization",
      "identifier-input": "optional"
    },
    "isAccessibleForFree-input": "optional"
  },
  "result": {
    "@type": "Event",
    "name-output": "required",
    ...
  },
  "query-input": "optional"
}
```

C

C Grundbefehle

Befehl	Gebrauch	Beispiel
#include	Dient zum einbinden von Bibliotheken und Code aus anderen Dateien	#include <stdio.h>
printf	Dient zur Ausgabe von Text und Variableninhalten	printf("%i",test);
scanf	Dient zum einlesen von Variablen	scanf("%i",&test);
switch (){ case : break; }	Dient zum vergleichen von einer Variablen mit vorgegebenen Werten. Wird gerne zur Menüsteuerung verwendet.	switch (test) { case 1: (); break; case 2: (); break; default:(); }
if (){ Aktion; } else{ Aktion; }	Das If Else Statement dient zum Prüfen von Variablen gegen Werte oder andere Variablen. Wird auch Wenn Dann genannt. Es Können auch mehrere Vergleiche verschachtelt werden.	if (max_punkte >= err_punkte) { /*Berechnung*/ } else { /*Fehlermeldung*/ }

Pseudocode

Pseudocode ist eine detaillierte und dennoch lesbare Beschreibung dessen, was ein Computerprogramm oder ein Algorithmus machen soll. Pseudocode wird in einer formal gestalteten, natürlichen Sprache und nicht in einer Programmiersprache ausgedrückt.

Fallunterscheidungen

- `if ... then ... else ... end if/exit`
- `wenn ... dann ... sonst ... wenn_ende`
- `falls ... dann ... falls_nicht ... falls_ende`

Schleifen

- `wiederhole ... solange/bis ... wiederhole_ende`
- `while ... do ...`
- `repeat ... until ...`
- `for ... to ... step Schrittweite ... next`

Kommentare

- `// kommentar`
- `# kommentar`
- `/* kommentar */`

Beispiel

WENN die Pizza in Folie eingepackt ist

□Entferne Folie

schalte Ofen ein

gib Pizza auf Blech in Ofen

SOLANGE Pizza noch nicht fertig

□warte eine Minute

entnimm Pizza aus dem Ofen

Tiefkühlpizza backen

öffne Verpackung

die Pizza ist in Folie eingepackt

WAHR

FALSCH

entferne Folie

tue nichts

Pizza noch nicht fertig

warte eine Minute

entnimm Pizza aus dem Ofen

Übungsaufgabe 1

Zeiterfassungssystem

Das Systemhaus bietet ein System zur Arbeitszeiterfassung an, welches lokal auf den Arbeitsplatzrechnern installiert wird. Dieses steht mit einem Server im LAN in Verbindung und sammelt dort die Daten. Fällt die Netzwerkverbindung aus, so werden die Daten lokal in einer XML-Datei vorgehalten und bei nächster Gelegenheit zum Server übertragen.

Die XML-Datei liegt im lokalen Profil des jeweiligen Benutzers, ist jedoch nicht verschlüsselt und kann mit einem Editor im Klartext gelesen werden. Die Informationen und deren Struktur können Sie der Anlage 6.pdf entnehmen.

1. Um die Sicherheit der Informationen zu gewährleisten sollen Sie eine Schutzbedarfsanalyse durchführen.
 - Kennzeichnen Sie den Schutzbedarf der Informationen unter Betrachtung der Schutzziele Vertraulichkeit, Verfügbarkeit und Integrität. Informationen zu den Schutzzielen finden Sie in der Anlage 5.pdf.
 - Begründen Sie Ihre Klassifizierung. Verwenden Sie für Ihre Lösung das Vorgabeblatt Anlage 11.pdf.
2. Die Informationen der XML-Datei zur Arbeitszeiterfassung sollen nicht mehr in einer XMLDatei, sondern in einer kleinen Datenbank lokal und verschlüsselt zwischengespeichert werden.

Ermitteln Sie auf Basis der dargestellten Informationen aus der XML-Datei (Anlage 6) die Struktur der dafür nötigen Tabellen. Verwenden Sie für Ihre Lösung das Vorgabeblatt Anlage 12.pdf mit der Anlage 12.1.
3. Geben Sie an, welches Beziehungsmuster zwischen den Informationen UserAccount und LogEntry besteht. Begründen Sie Ihre Antwort. Verwenden Sie für Ihre Lösung das Vorgabeblatt Anlage 12.pdf mit der Anlage 12.2.

Übungsaufgabe 2

Sämtliche Geräte wie PCs und Handhelds werden über ein Management-Tool verwaltet. An dieses Tool übertragen alle Geräte ihre aktuellen Betriebssystemparameter, wie Geräte-ID, Betriebssystemname, Version, letztes Update, freier Permanentpeicher, Größe des RAM und Datum des letzten Logins.

Das Management-Tool besitzt eine Datenbankschnittstelle, über die SQL-Statements abgesetzt werden können (Anlage 8.pdf). Die IT-Abteilung möchte eine aktuelle Übersicht der Betriebszustände der Geräte.

Erstellen Sie für die IT-Abteilung folgende Abfragen per SQL:

1. Anzahl aller Geräte
2. Anzahl der Geräte gruppiert nach Betriebssystem
3. alle Geräte deren Update älter als 30 Tage ist
4. welche User haben welche Geräte benutzt