

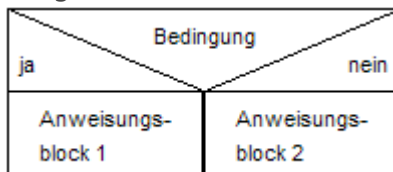
Struktogramme

- 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



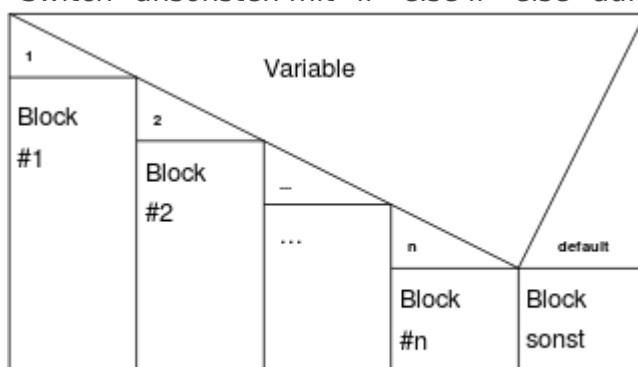
```
# Anweisungen in Python
a = input("Zahl eingeben") # Anweisung 1
b = 5 * a # Anweisung 2
print(b) # Anweisung 3
```

- 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.



// 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++;

}

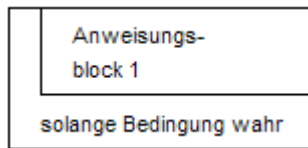
// Kopfgesteuerte for-Schleife

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.



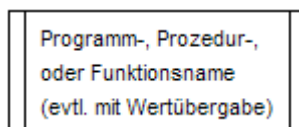
```
// Fussgesteuerte-Schleife
do { // Schleife wird betreten
    printf("%i\n", a); // Schleife wird ausgeführt
    a++;
} while (a <= 10); // Prüfung und sollte a größer 10 sein, wird Schleife beendet.
Ansonsten wird Schleife erneut ausgeführt
```

- 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  
}
```

- o `main` llt. 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