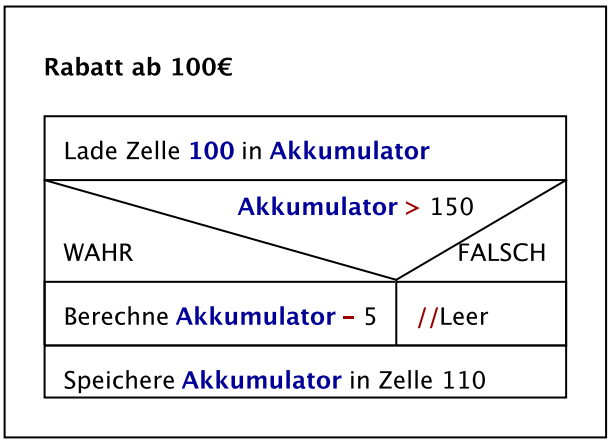
3.3.2 Die bedingte Anweisung

Nicht jedes Programm läuft linear ab. Oft muss zur Laufzeit eine Entscheidung getroffen werden, wie der Programmablauf weitergeht. Unser Grundbaustein für diese Art der Anweisung ist die **bedingte Anweisung**. Wir betrachten zuerst die einseitige Bedingte Anweisung, die keinen „Sonst"-Teil hat.
Wir nehmen uns als Beispiel einen Rabattrechner. Wenn wir für mehr als 150€ bestellen, soll 5€ Rabatt abgerechnet werden.

Struktogramm:



Wir sehen, dass es eine Befehlszeile gibt, die nur ausgeführt werden soll, wenn die Bedingung

```
Akkumulator > 150
```

wahr ist. Ansonsten muss dieser Befehl übersprungen werden. Damit ist klar, dass wir für die bedingte Anweisung die Sprung-Befehle des Assemblers brauchen.

Zusätzlich zum Sprung-Befehl muss vorher natürlich überprüft werden, ob der Sprung durchgeführt werden muss. Daher benötigen wir auch einen Vergleichs-Befehl. Betrachten wir nochmal den Befehlssatz der Minimaschine:

Arithmetische Befehle				
Code	Daten	PC nach Befehl	SR	Beschreibung
CMP	Adresse	+2	Beeinflusst	Vergleicht Akku mit Inhalt der Adresse und setzt SR:[1] Adresse/Wert < Akku -> kein Flag Adresse/Wert = Akku -> Zero Flag Adresse/Wert > Akku -> Negativ Flag
CMPI	Wert	+2	Beeinflusst	

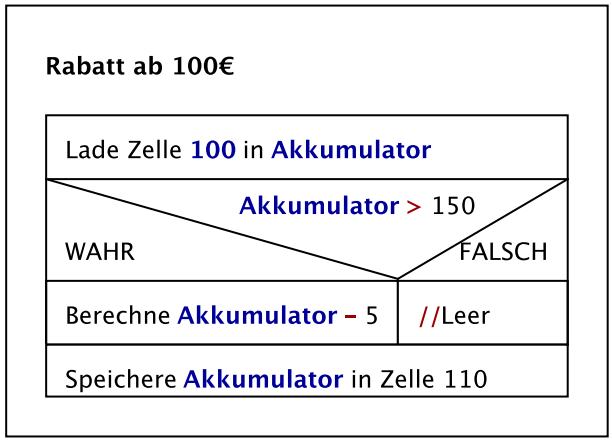
Steuerbefehle				
Code	Daten	PC nach Befehl	SR	Beschreibung
JMP	Adresse	Adresse		Springe zur Adresse (neuer PC)
JMPZ	Adresse	+2 oder Adresse		Springe zu Adresse wenn Zero-Flag gesetzt
JMPNZ	Adresse	+2 oder Adresse		Springe zu Adresse wenn **kein** Zero-Flag gesetzt
JMPN	Adresse	+2 oder Adresse		Springe zu Adresse wenn Negativ-Flag gesetzt
JMPNN	Adresse	+2 oder Adresse		Springe zu Adresse wenn **kein** Negativ-Flag gesetzt
JMPP	Adresse	+2 oder Adresse		Springe zu Adresse wenn weder Zero- noch Negativ-Flag gesetzt
JMPNP	Adresse	+2 oder Adresse		Springe zu Adresse wenn **entweder** Zero- oder Negativ-Flag gesetzt

[1] Anmerkung: Der Vergleich ist eigentlich ein Akku-Speicherzelle/Wert ohne Speichern des Ergebnisses im Akku.

Die Flags in Kombination mit dem Sprung-Befehlen ermöglichen uns nun die bedingte Anweisung durchzuführen.

Im Beispiel sieht das dann so aus:

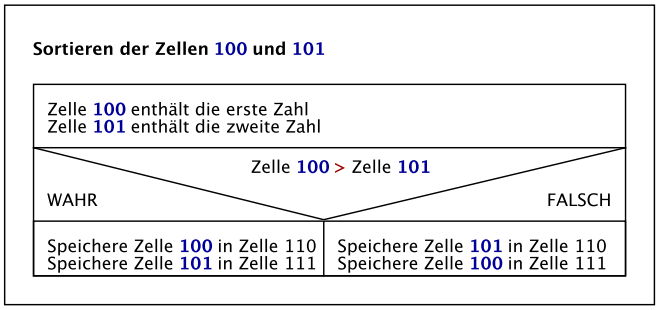
```
Start:      LOAD        100
            CMPI        150
            JMPN        weiter
            SUBI        5
weiter:     STORE       100
            HOLD
```



JMPN sorgt dafür, dass bei einem Negativ-Flag der Sprung zur Marke „weiter“ durchgeführt wird. Das führt dazu, dass wir den nächsten Befehl überspringen und somit den „WAHR“ Teil nur ausführen, wenn die Bedingung erfüllt ist.

Zweiseitige Bedingte Anweisung

Die zweiseitige bedingte Anweisung funktioniert sehr ähnlich. Nur müssen wir hier in beiden Fällen etwas überspringen. Nehmen wir das Sortieren zweier Zahlen.



Wir nutzen wieder Sprungbefehle, um die bedingte Anweisung umzusetzen. Diesmal brauchen beide Teile des Struktogramms einen Sprungbefehl.

```
Start:      LOAD        100         #Lädt Zelle 100
            CMP         101         #Vergleiche mit Zelle 101
            JMPN        Sonst       #Z.101 >= Z.100 springe zu sonst
Dann:[2]    STORE       110         #Z.100 kommt in Z.110
            LOAD        101         #Lade Zelle 101
            STORE       111         #Speichern in Zelle 111
            JMP         Weiter      #Überspringe „Sonst“-Teil
Sonst:      STORE       111         #Speicher kleinere Z.100 in Z.111
            LOAD        101         #Lade andere Zelle
            STORE       110         #Speicher größeren Wert
Weiter:     HOLD                    #Ende der bedingten Anweisung
```

[2] Die Sprungmarke „Dann“ kann weggelassen werden und dient nur der Veranschaulichung

Aufgaben:

a) Erstellen Sie ein Programm, das überprüft, ob die Zelle [100] eine Zahl größer als 50 enthält. Speichern Sie „1“ für Ja und „0“ für Nein in die Zelle 110.
b) Erstellen Sie ein Programm, das auf den Wert der Zelle 100 einen Rabatt von 10% berechnet. Speichern Sie den fertigen Betrag in Zelle [110] und den Rabatt in Zelle [111].
c) Erstellen Sie ein Programm, das auf den Wert der Zelle [100] einen Rabatt in Höhe von [101]% gibt, sobald der Wert von [100] den Wert von [102] überschreitet.
d) Erstellen Sie eine Tabelle, mit welchen Kombinationen von JMP- und CMP-Befehlen Sie die Vergleichsoperationen „==“, „<“, “>“, „<=“ und „>=“ abbilden können.
e) Erstellen Sie ein Programm, das den Betrag der Zahl in Zelle [100] in Zelle [110] speichert.
f) Bestimmen Sie mit einem Programm, das Minimum der Zellen [100], [101] und [102].
g) Sortieren Sie die drei Zahlen in den Zellen [100], [101] und [102].
h) Erweitern Sie das Programm aus g) so, dass man mit dem Inhalt der Speicherzelle [103] angeben kann, ob aufsteigend (1) oder absteigend (0) sortiert werden soll.
i) (schwer!) Sortieren Sie drei Zellen wie in g) ohne zusätzliche Speicherzellen zu benutzen.