WBS TRAINING

Programmierung(2)

UNSER ♥ SCHLÄGT FÜR BILDUNG

Agenda

- Verschachtelte Verzweigungen
  - **Definition, Motivation** und Beispiel
  - Darstellung im **PAP**
  - Darstellung im **Struktogramm**
  - Darstellung im **Pseudocode**
  - Syntax in **ANSI C**
- Switch-Case
  - **Definition, Motivation**
  - Darstellung im **Struktogramm**
  - Darstellung im **Pseudocode**
  - Syntax in **ANSI C**
- Komplexe Bedingungen
  - **Definition, Motivation** und Beispiel
  - Und-Operator
  - Oder-Operator
  - Nicht-Operator
- Fachpraktische Anwendungen

Verschachtelte Verzweigungen – Definition und Motivation

- Falls innerhalb eines if- und/oder else-Blockes eine weitere Verzweigung eingetragen wird, so spricht man von einer „**Verschachtelten Verzweigung**“.
- Verzweigungen (einfache wie auch verschachtelte) können mit einer Frage verglichen werden:

  Ist eine bestimmte Bedingung erfüllt? Ja oder nein? (soll also der if- oder else-Block ausgeführt werden?)
- Oft müssen aber in einem Programm mehrere solche Fragen gestellt werden, und nicht selten sind Folgefragen sinnlos, wenn eine vorangegangene Frage bereits mit „ja“ beantwortet wurde.
- Dies ist ein wichtiges Einsatzgebiet für „verschachtelte Verzweigungen“. Die Motivation besteht dann in diesem Fall darin, unnötige Fragen zu vermeiden, um die Performance zu verbessern.

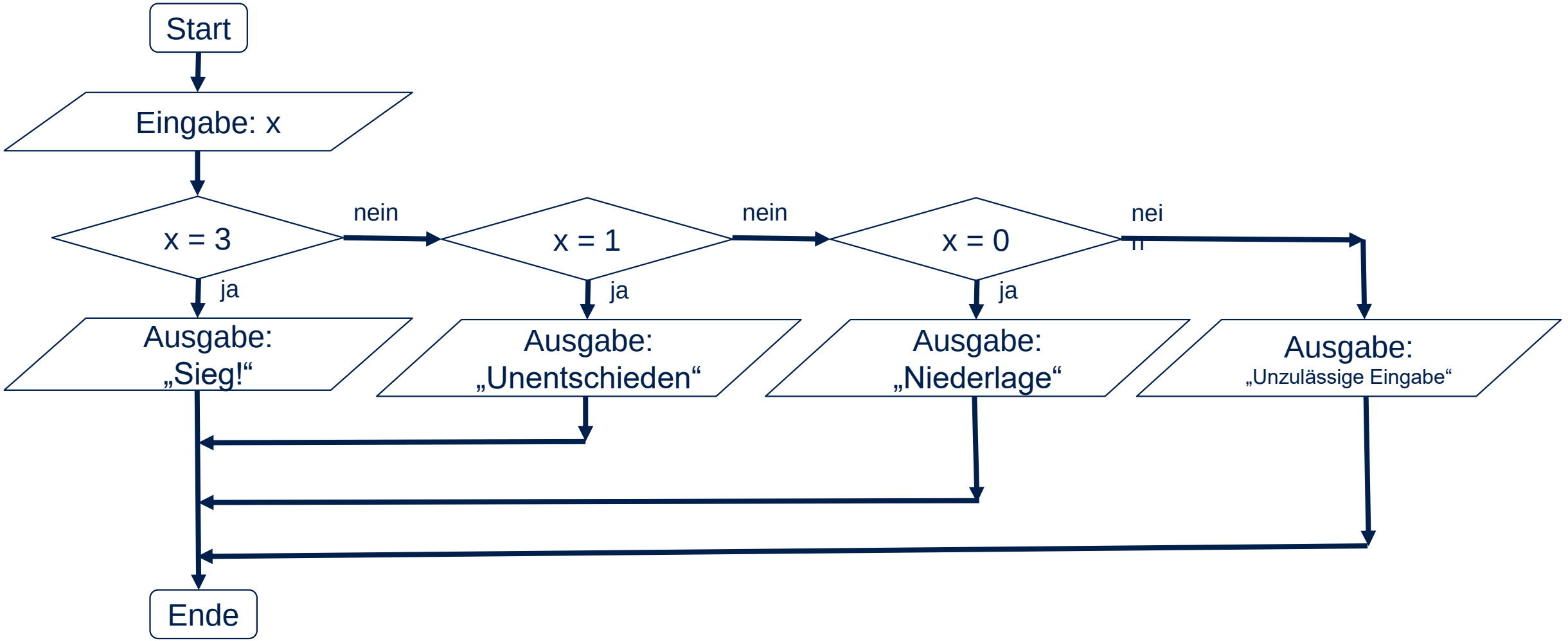
Verschachtelte Verzweigungen – **Beispielaufgabe**

Aufgabenstellung:

- Das Programm startet mit der Abfrage, wie viele Punkte der User beim letzten Fußballspiel erzielte.
- Anschließend folgt eine erste Verzweigung, die überprüft, ob der User den Wert 3 eingab.
- Falls dies mit „ja“ beantwortet wird, so erscheint „Sieg!“ und das Programm endet.
- Nur(!) wenn mit „nein“ geantwortet wurde, folgt eine zweite Verzweigung, die überprüft, ob 1 eingegeben wurde.
- Falls dies mit „ja“ beantwortet wird, so erscheint „Unentschieden“ und das Programm endet.
- Nur(!) wenn mit „nein“ geantwortet wurde, folgt eine dritte Verzweigung, die überprüft, ob 0 eingegeben wurde.
- Falls dies mit „ja“ beantwortet wird, so erscheint „Niederlage“ und das Programm endet.
- Nur(!) wenn mit „nein“ geantwortet wurde, ist klar, dass weder 0, 1 noch 3 eingegeben wurde. Es muss daher keine weitere Verzweigung gestartet (bzw. Bedingung überprüft) werden, sondern es kann sofort zur Fehlermeldung kommen: Es wird „Unzulässige Eingabe“ ausgegeben und das Programm endet.

Auch für diese Aufgabe wollen wir zunächst **PAP**, **Struktogramm** und **Pseudocode** erstellen, um erst daraufhin den entsprechenden **Quellcode** in ANSI C zu codieren.

Verschachtelte Verzweigung – **Beispielaufgabe** – PAP

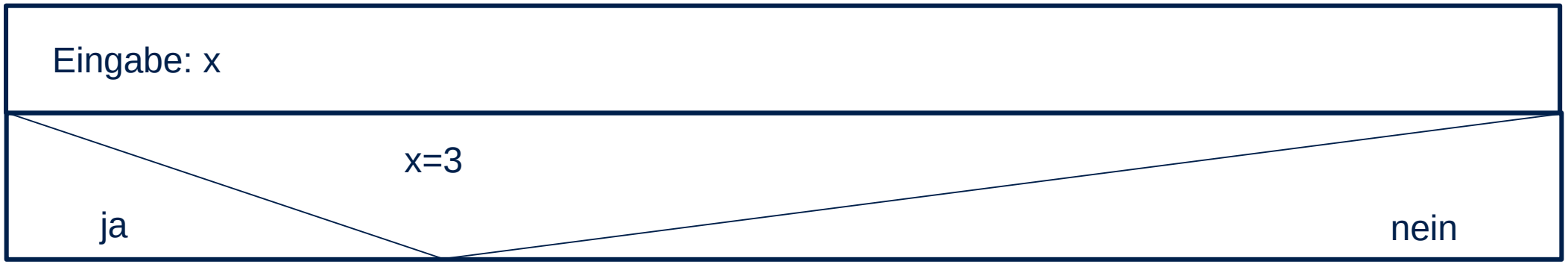


Verschachtelte Verzweigung – **Beispielaufgabe** – Struktogramm

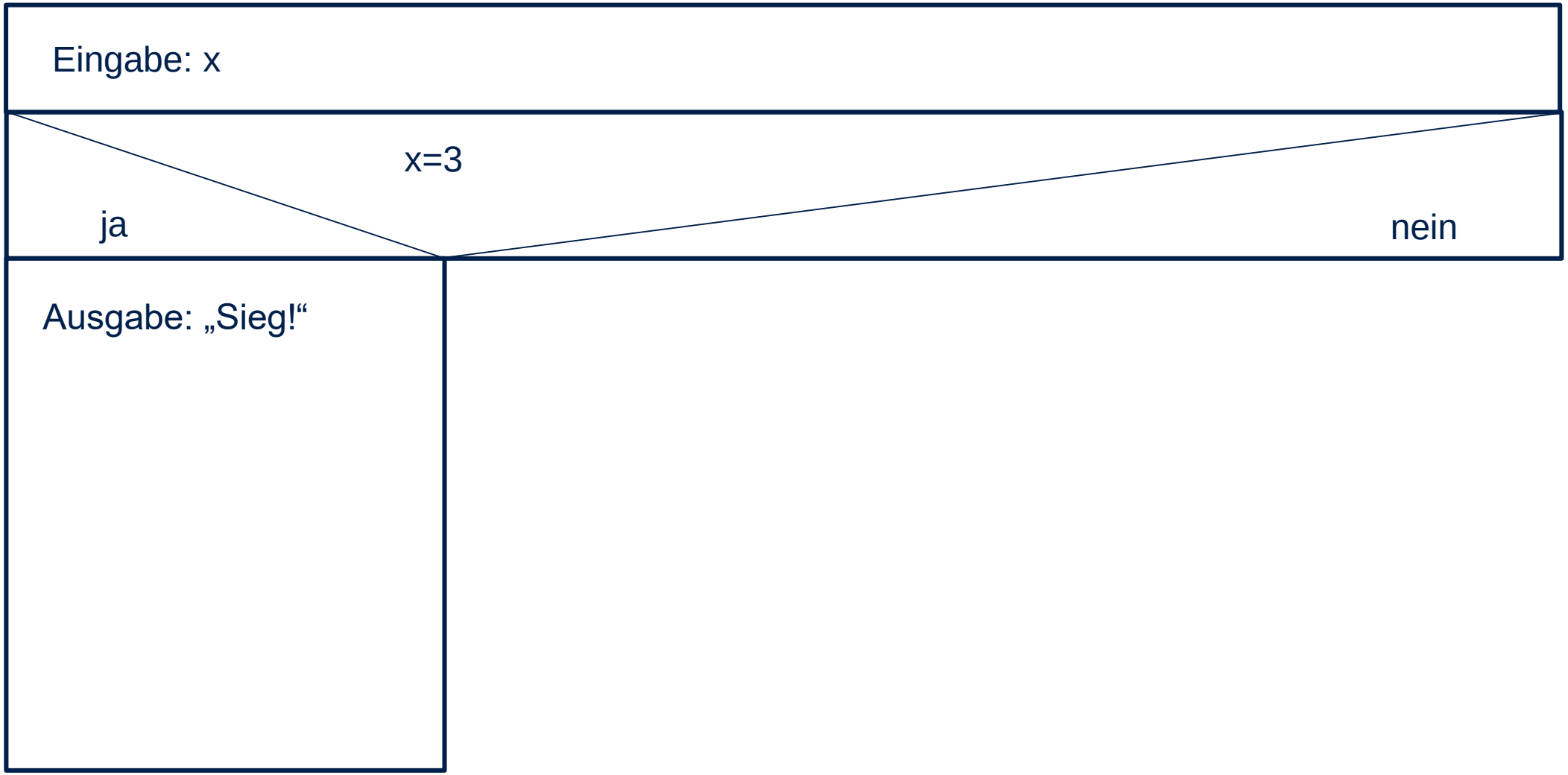
Verschachtelte Verzweigung – **Beispielaufgabe** – Struktogramm

Eingabe: x

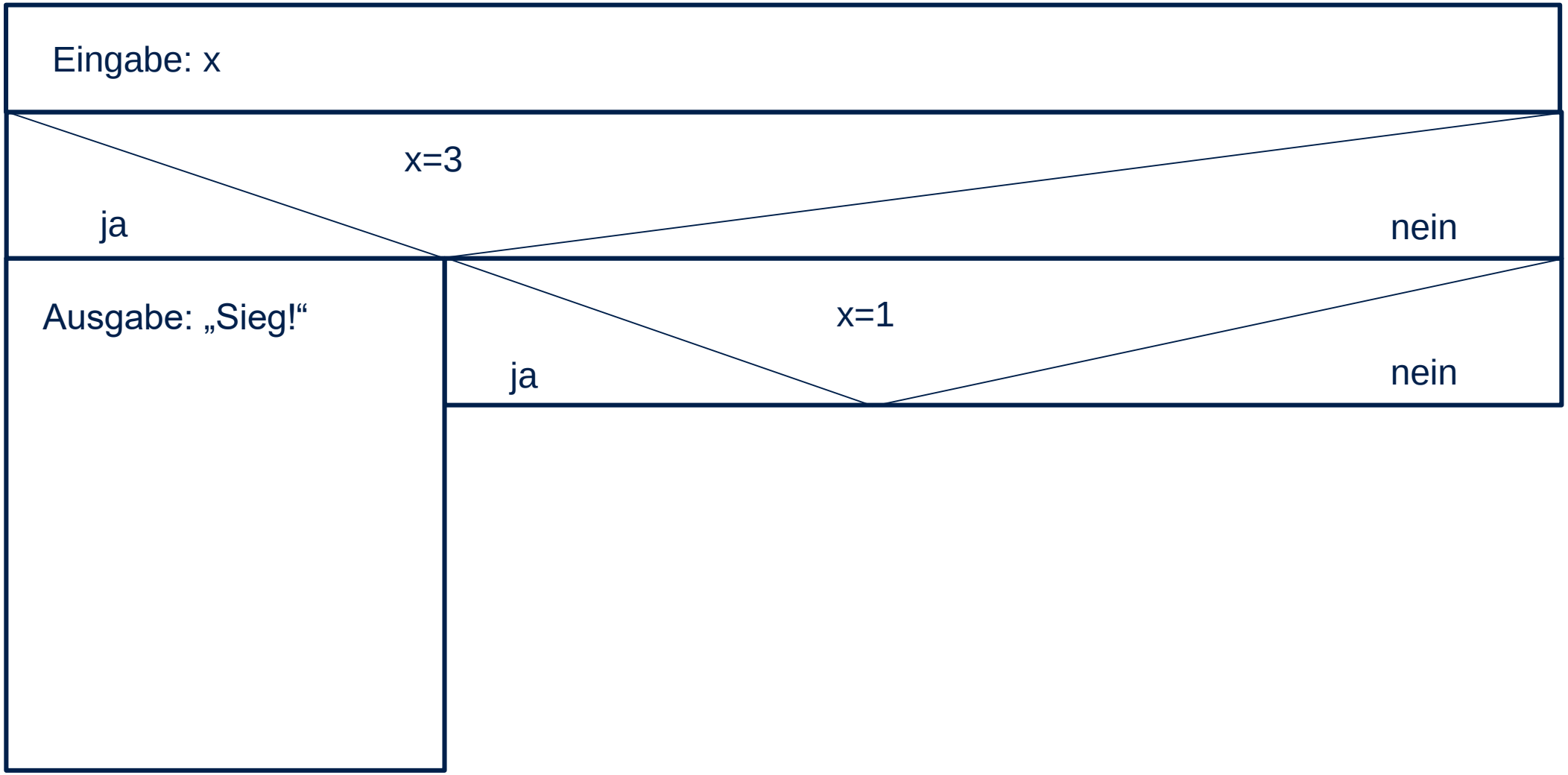
Verschachtelte Verzweigung – **Beispielaufgabe** – Struktogramm



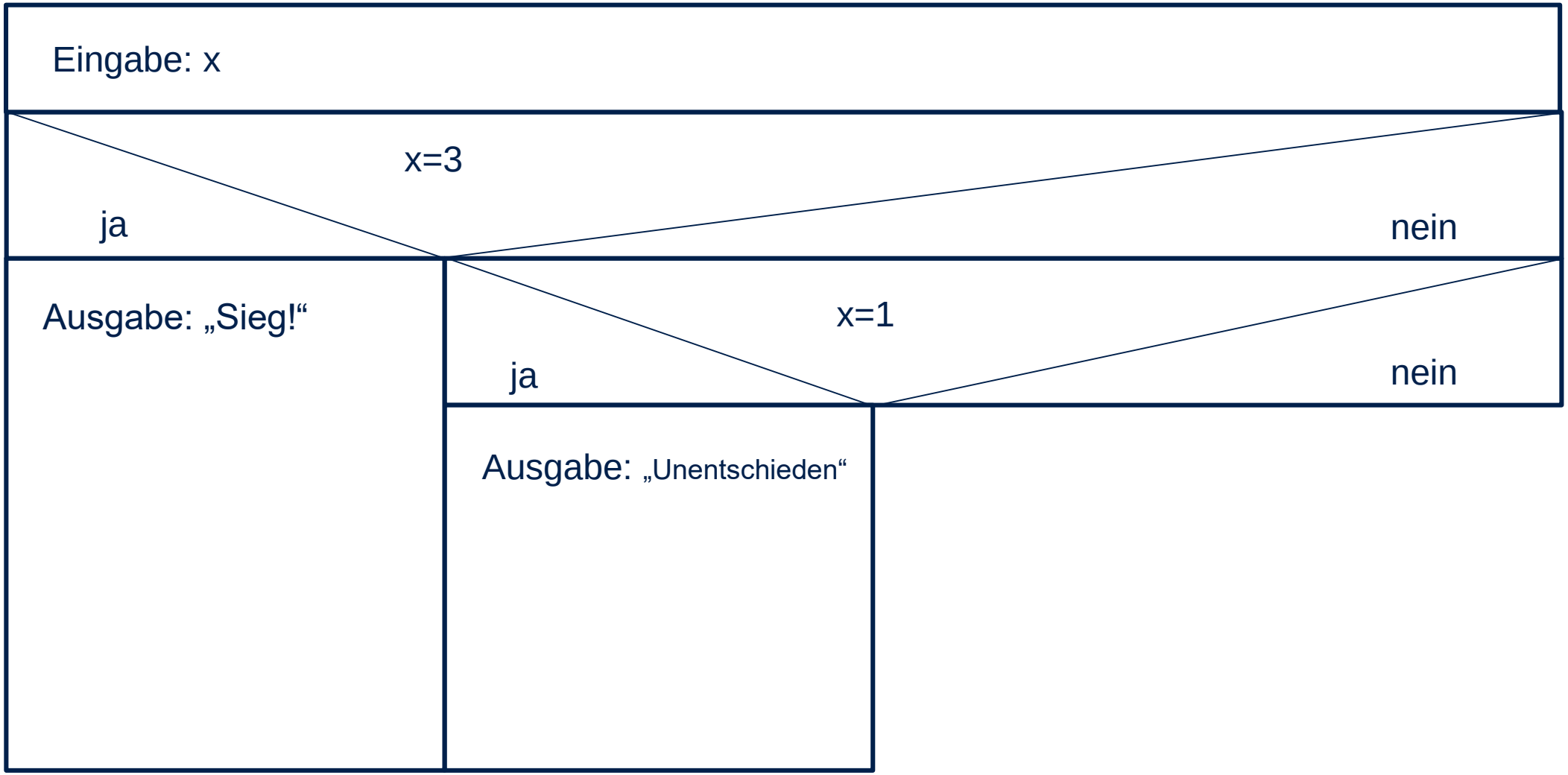
Verschachtelte Verzweigung – **Beispielaufgabe** – Struktogramm



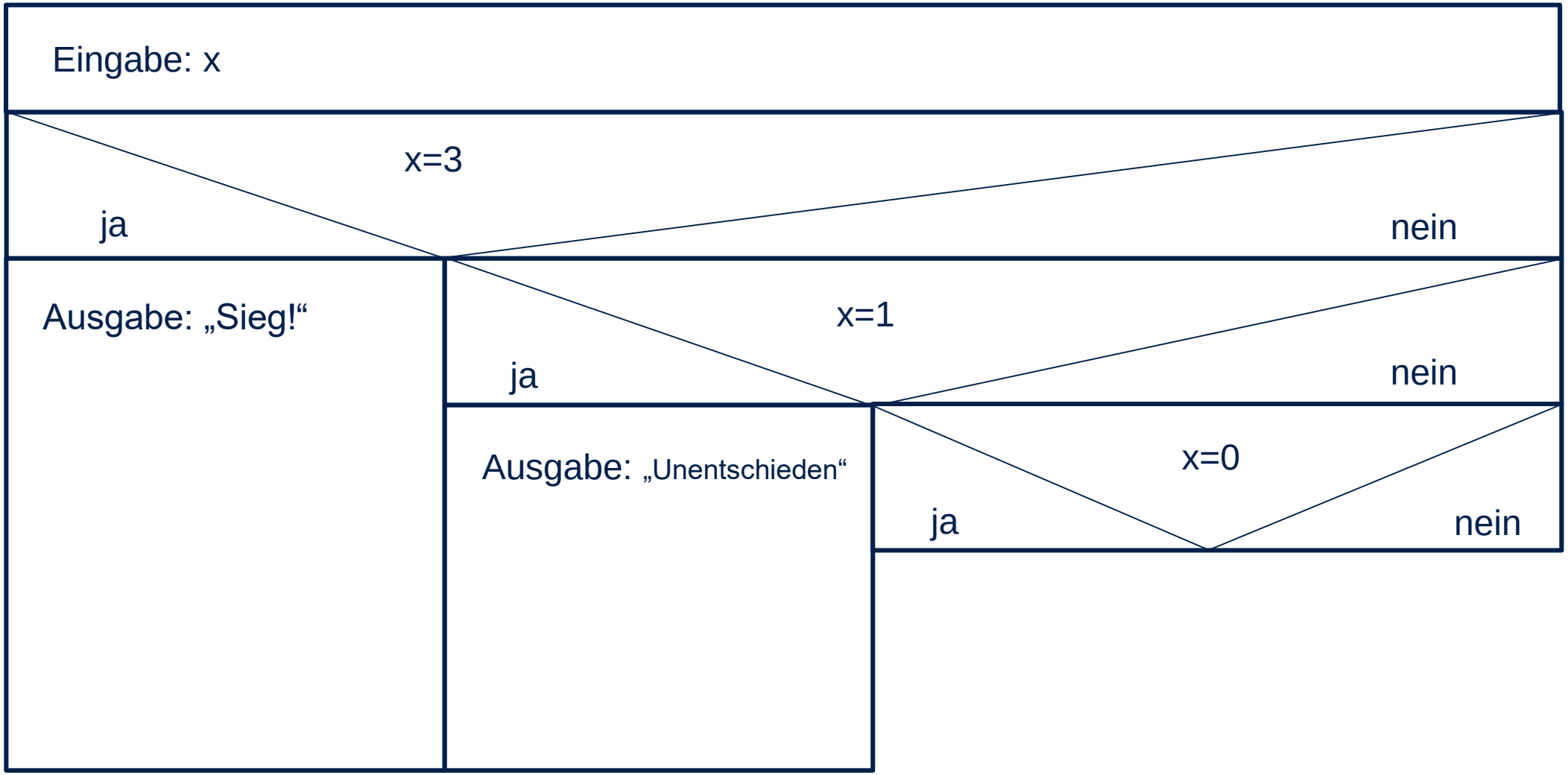
Verschachtelte Verzweigung – **Beispielaufgabe** – Struktogramm



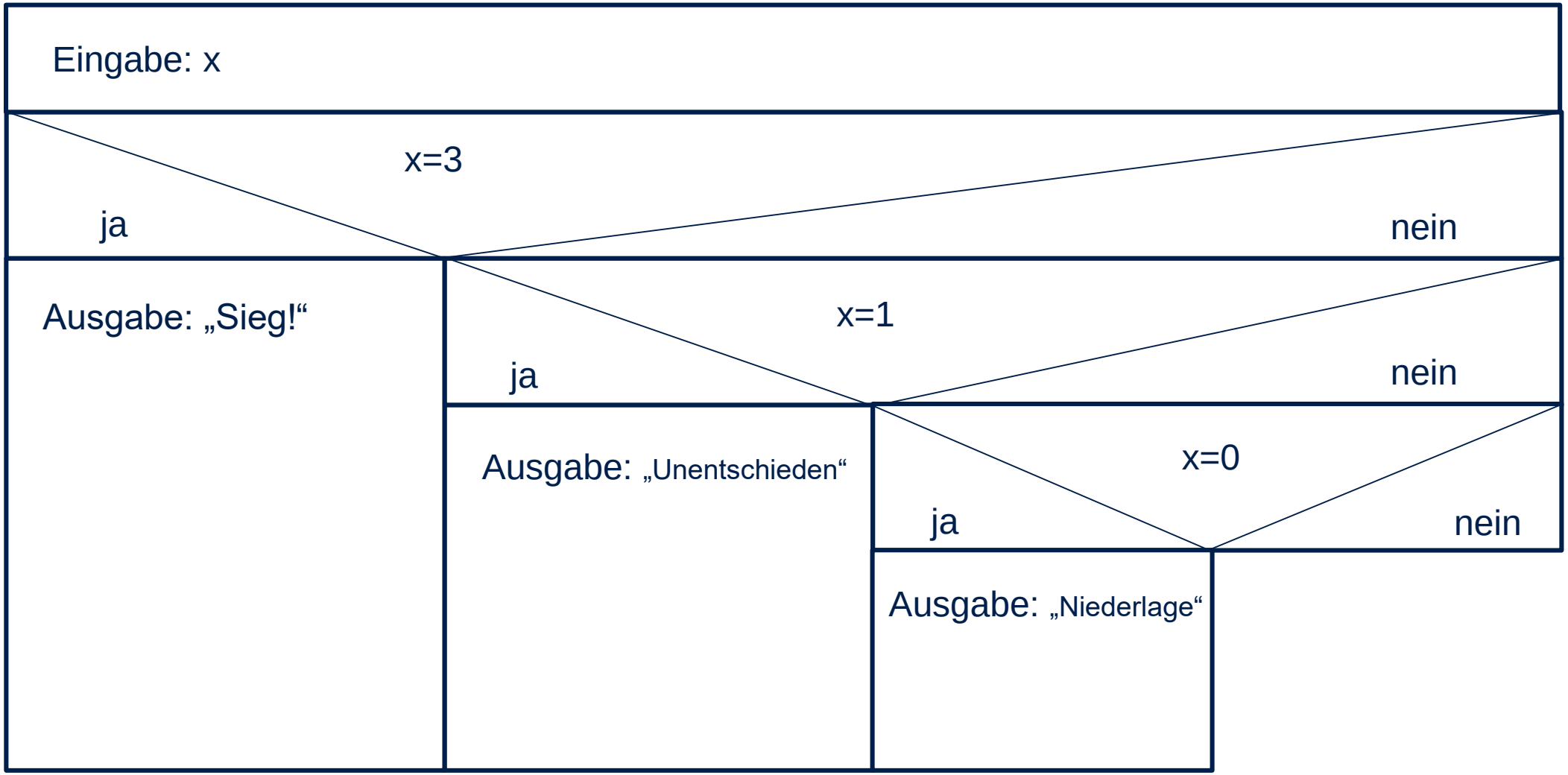
Verschachtelte Verzweigung – **Beispielaufgabe** – **Struktogramm**



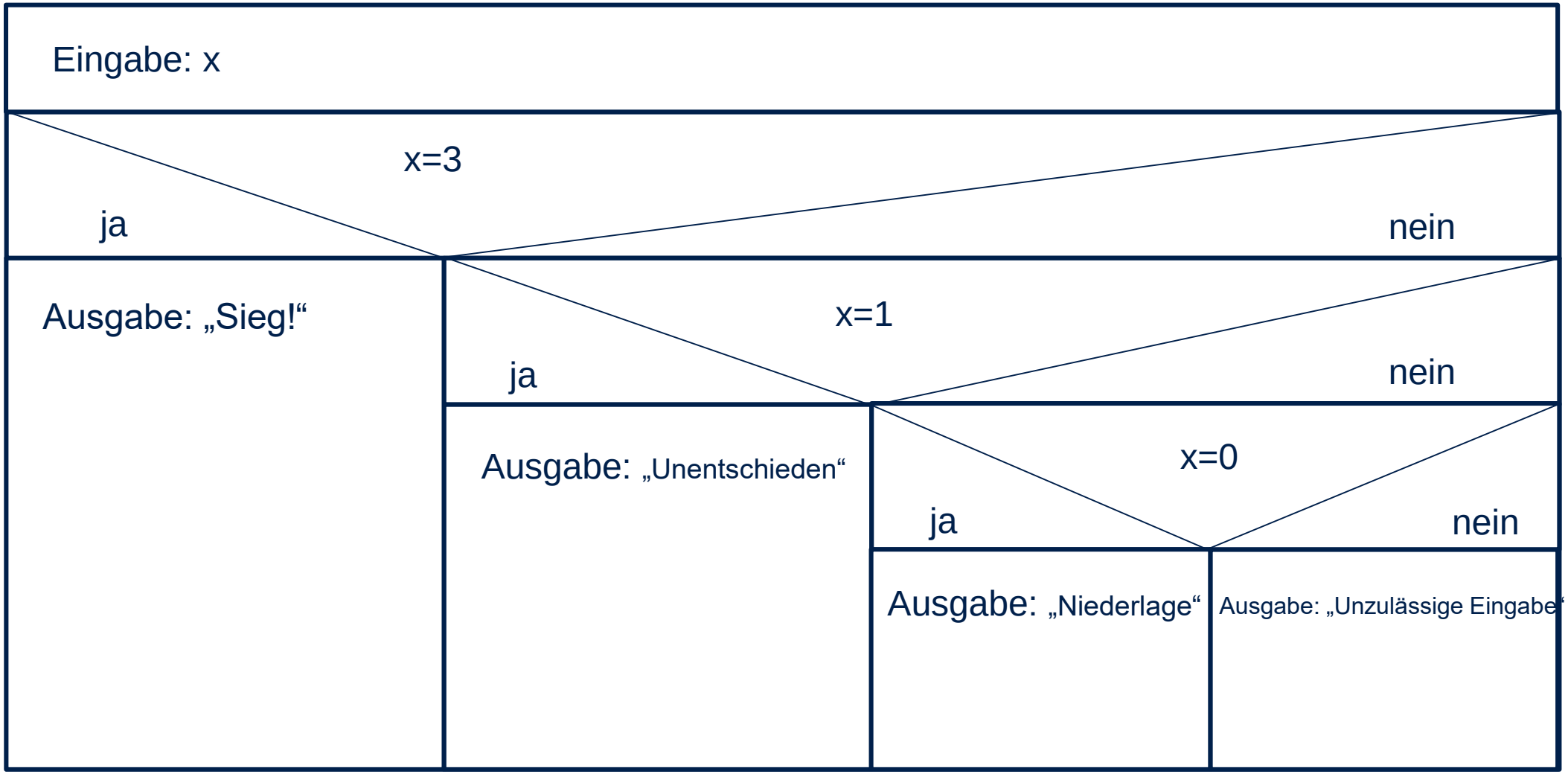
Verschachtelte Verzweigung – **Beispielaufgabe** – Struktogramm



Verschachtelte Verzweigung – **Beispielaufgabe** – **Struktogramm**



Verschachtelte Verzweigung – **Beispielaufgabe** – **Struktogramm**



Verschachtelte Verzweigung – **Beispielaufgabe** – **Pseudocode**

```
Programm „Verschachteltes-Verzweigungs-Beispiel“
{
        Eingabe: x
        Wenn(x==3)
        {
                Ausgabe: „Sieg!“
        }
        Sonst
        {
                Wenn(x==1)
                {
                        Ausgabe: „Unentschieden“
                }
                Sonst
                {
                        Wenn(x==0)
                        {
                                Ausgabe: „Niederlage“
                        }
                        Sonst
                        {
                                Ausgabe: „Unzulässige Eingabe“
                        }
                }
        }
}
```

Verschachtelte Verzweigung – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
		int x;

		printf(“Geben Sie bitte die erreichten Punkte ein: “);
		scanf(“%d“,&x);

		if(x==3)
		{
				printf(“Sieg!“);
		}
		else
		{
				if(x==1)
				{
						printf(“Unentschieden“);
				}
				else
				{
						if(x==0)
						{
								printf(“Niederlage“);
						}
						else
						{
								printf(“Unzulässige Eingabe“);
						}
				}
		}
}
```

Switch-Case – Definition und Motivation

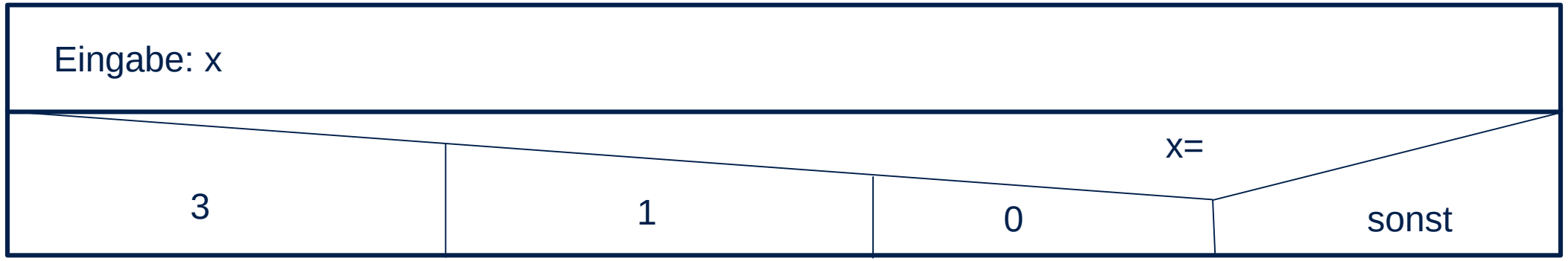
- Die bei (einfachen oder verschachtelten) Verzweigungen verwendeten Bedingungen sind natürlich beliebig. Es kann also z.B. auch mit den „>“(größer) oder „<“(kleiner) Operatoren gearbeitet werden.
- Der in unserer Beispielaufgabe vorliegende Fall, bei dem die Bedingungen aller Verzweigungen mit einem „==“(Vergleich) arbeiten, ist allerdings recht gebräuchlich.
- Da zudem verschachtelte Verzweigungen bei einer größeren Verschachtelungstiefe schnell unübersichtlich werden können, wurde eine abkürzende Schreibweise eingeführt.
- Diese Abkürzung wird als „**Switch-Case**“ bezeichnet, ist aber (zumindest nach ANSI-Standard) nur für den Vergleich auf Identität zulässig.
- Wir werden nun im Folgenden unsere Beispielaufgabe auch mittels Switch-Case-Anweisung lösen. Dies wird aber nur beim Struktogramm, Pseudocode und Quellcode sinnvoll sein, denn im PAP gibt es für den Switch-Case kein entsprechendes Symbol.

Switch-Case – **Beispielaufgabe** – **Struktogramm**

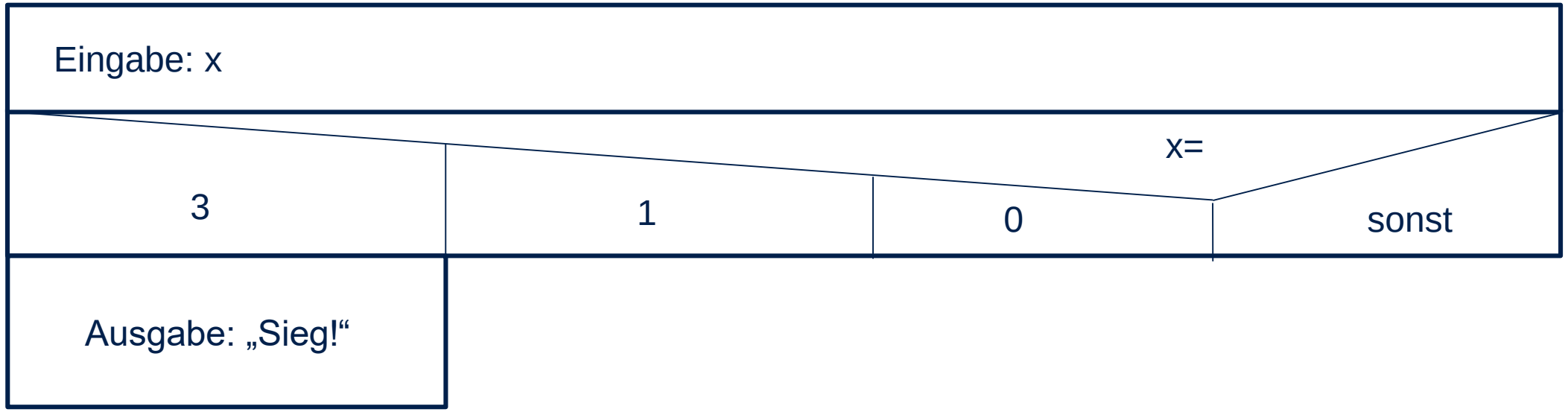
Switch-Case – **Beispielaufgabe** – **Struktogramm**

Eingabe: x

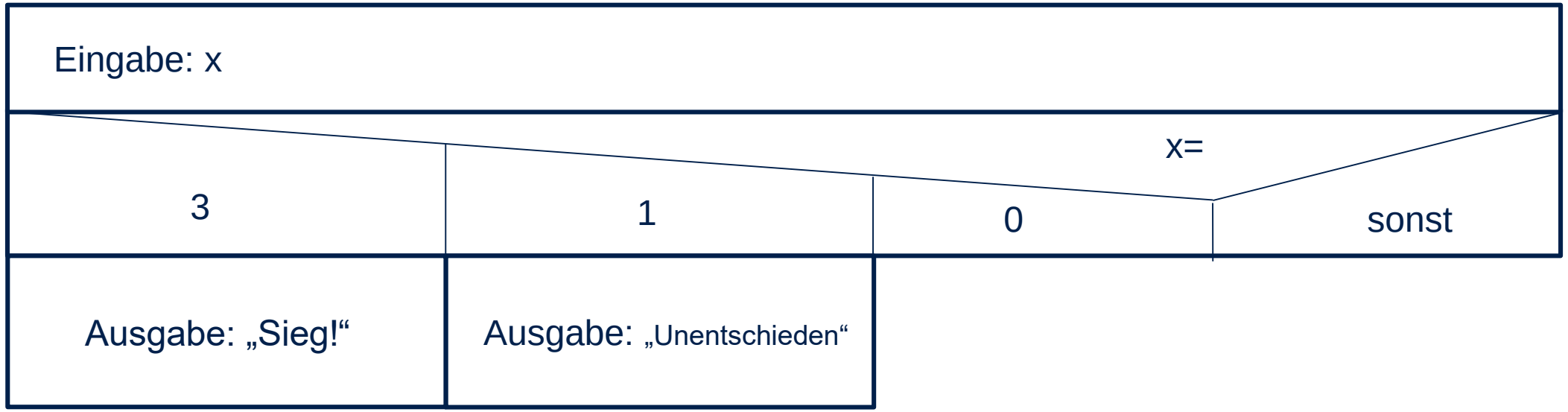
Switch-Case – **Beispielaufgabe** – **Struktogramm**



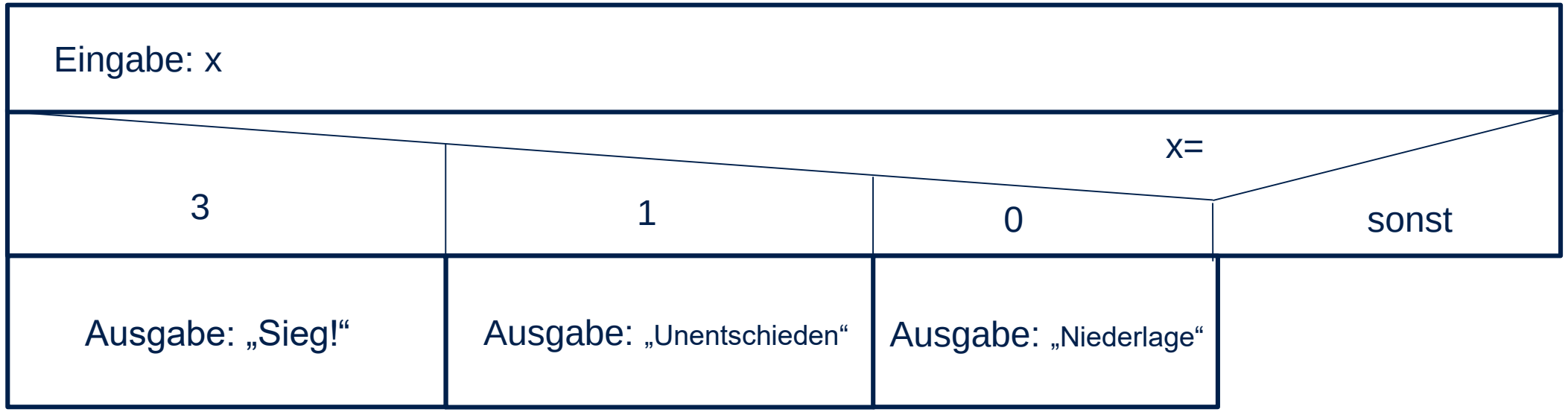
Switch-Case – **Beispielaufgabe** – **Struktogramm**



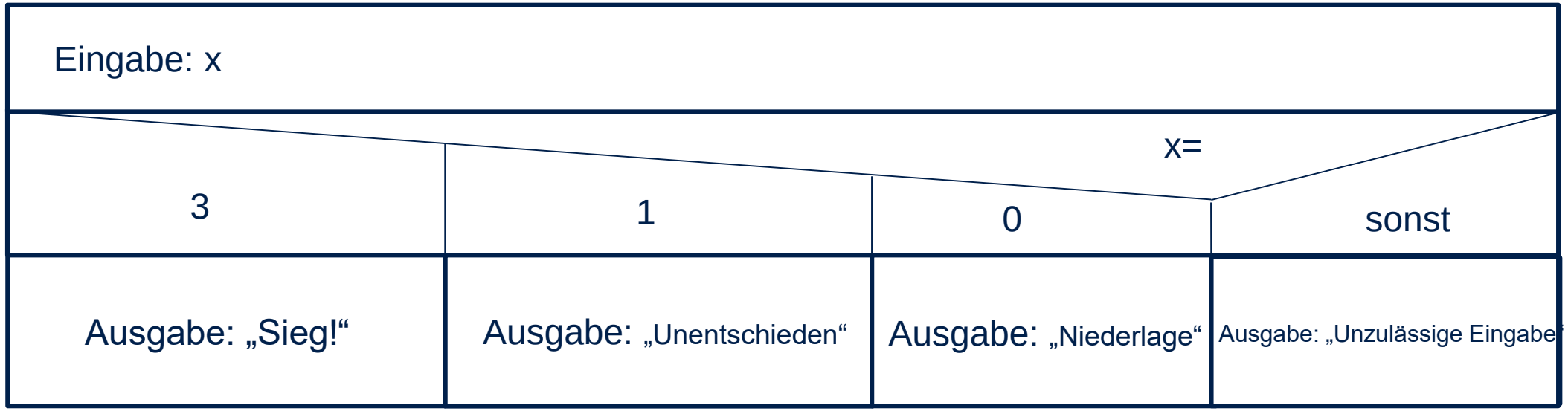
Switch-Case – **Beispielaufgabe** – **Struktogramm**



Switch-Case – **Beispielaufgabe** – **Struktogramm**



Switch-Case – **Beispielaufgabe** – **Struktogramm**



Switch-Case – **Beispielaufgabe** – **Pseudocode**

```
Programm „Verschachteltes-Verzweigungs-Beispiel“
{
        Eingabe: x
        switch(x)
        {
                3:      Ausgabe: „Sieg!“
                1:      Ausgabe: „Unentschieden“
                0:      Ausgabe: „Niederlage“
                sonst:  Ausgabe: „Unzulässige Eingabe“
        }
}
```

Switch-Case – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
        int x;

        printf(“Geben Sie bitte die erreichten Punkte ein: “);
        scanf(“%d“,&x);

        switch(x)
        {
                case 3 : printf(“Sieg!“); break;
                case 1 : printf(“Unentschieden“); break;
                case 0 : printf(“Niederlage“); break;
                default: printf(“Ungültige Eingabe“);
        }
}
```

Switch-Case – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
        int x;

        printf("Geben Sie bitte die erreichten Punkte ein: ");
        scanf("%d",&x);

        switch(x)
        {
                case 3 : printf("Sieg!"); break;
                case 1 : printf("Unentschieden"); break;
                case 0 : printf("Niederlage"); break;
                default: printf("Ungültige Eingabe");
        }
}
```

Switch-Case – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
        int x;

        printf("Geben Sie bitte die erreichten Punkte ein: ");
        scanf("%d",&x);

        switch(x)
        {
                case 3 : printf("Sieg!"); break;
                case 1 : printf("Unentschieden"); break;
                case 0 : printf("Niederlage"); break;
                default: printf("Ungültige Eingabe");
        }
}
```

Switch-Case – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
        int x;

        printf(“Geben Sie bitte die erreichten Punkte ein: “);
        scanf(“%d“,&x);

        switch(x)
        {
                case 3 : printf(“Sieg!“); break;
                case 1 : printf(“Unentschieden“); break;
                case 0 : printf(“Niederlage“); break;
                default: printf(“Ungültige Eingabe“);
        }
}
```

Komplexe Bedingung – Definition und Motivation

- Sie haben in vorangegangenen Bausteinen bereits die logischen (boolschen) Operatoren „UND“, „ODER“ und „NICHT“ kennengelernt.
- Mit diesen Operatoren können einfache Aussagen zu komplexen Aussagen verknüpft werden.
  - Als Auffrischung:
    - **A UND B** ist nur genau dann wahr, wenn beide Aussagen (A,B) wahr sind
    - **A ODER B** ist wahr, wenn mindestens einer der beiden Aussagen (A,B) wahr ist
    - **NICHT A** ist wahr, wenn A falsch ist
    - Ähnlich wie bei „Punkt vor Strich“ gilt die Regel „**UND vor ODER**“
    - **„NICHT“** bezieht sich immer nur auf die unmittelbar nachfolgende Aussage (falls keine Klammern eingesetzt werden)
    - Wie aus der Mathematik bekannt, können **Klammern** die Regel „UND vor ODER“ aufheben. (In den Klammern wird „zuerst gerechnet“)
- Komplexe Aussagen, die wir als Bedingung in Verzweigungen (oder Schleifen) verwenden, werden entsprechend als „**Komplexe Bedingungen**“ bezeichnet.
- Komplexe Bedingungen sind in der Programmierung dann bedeutsam, wenn diese erfüllt seien müssen, damit ein if-Block (oder Schleifendurchlauf) ausgeführt werden kann => Beispielaufgabe

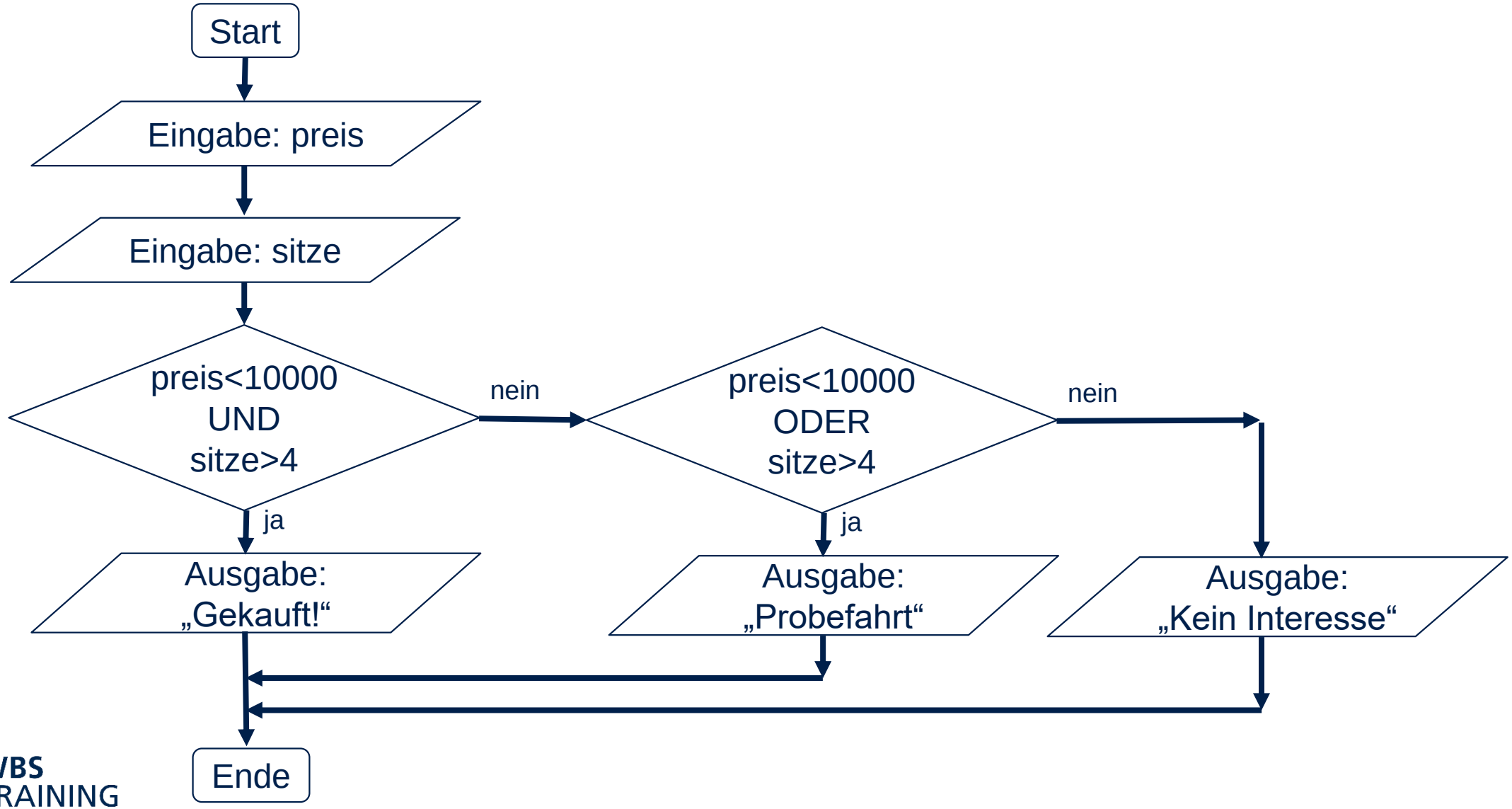
Komplexe Bedingung – **Beispielaufgabe**

Aufgabenstellung:

- Das Programm startet mit 2 Abfragen:
  - a) Einkaufspreis (eines Autos)
  - b) Anzahl der Sitzplätze (des selben Autos)
- Anschließend folgt eine erste Verzweigung, die überprüft, ob (Einkaufspreis<10.000 **UND** Anzahl Sitze>4)
- Falls dies mit „ja“ beantwortet wird, so erscheint „Gekauft!“ und das Programm endet.
- Bei „nein“ folgt eine zweite Verzweigung, die überprüft, ob (Einkaufspreis<10.000 **ODER** Anzahl Sitze>4)
- Falls dies mit „ja“ beantwortet wird, so erscheint „Probefahrt“ und das Programm endet.
- Bei „nein“ ist klar, dass weder der Preis<10.000 noch die Anzahl der Sitze>4 ist. Daher ist keine weitere Verzweigung notwendig und es erscheint die Ausgabe: „Kein Interesse“ bevor das Programm endet.

Auch für diese Aufgabe wollen wir zunächst **PAP**, **Struktogramm** und **Pseudocode** erstellen, um erst daraufhin den entsprechenden **Quellcode** in ANSI C zu codieren.

Komplexe Bedingung – **Beispielaufgabe** – **PAP**



Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**

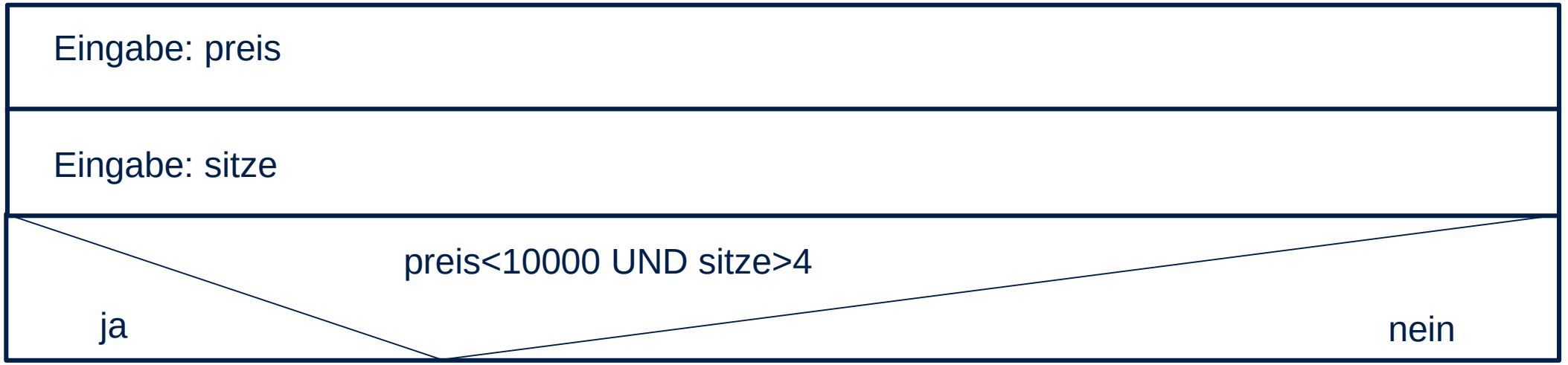
Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**

Eingabe: preis

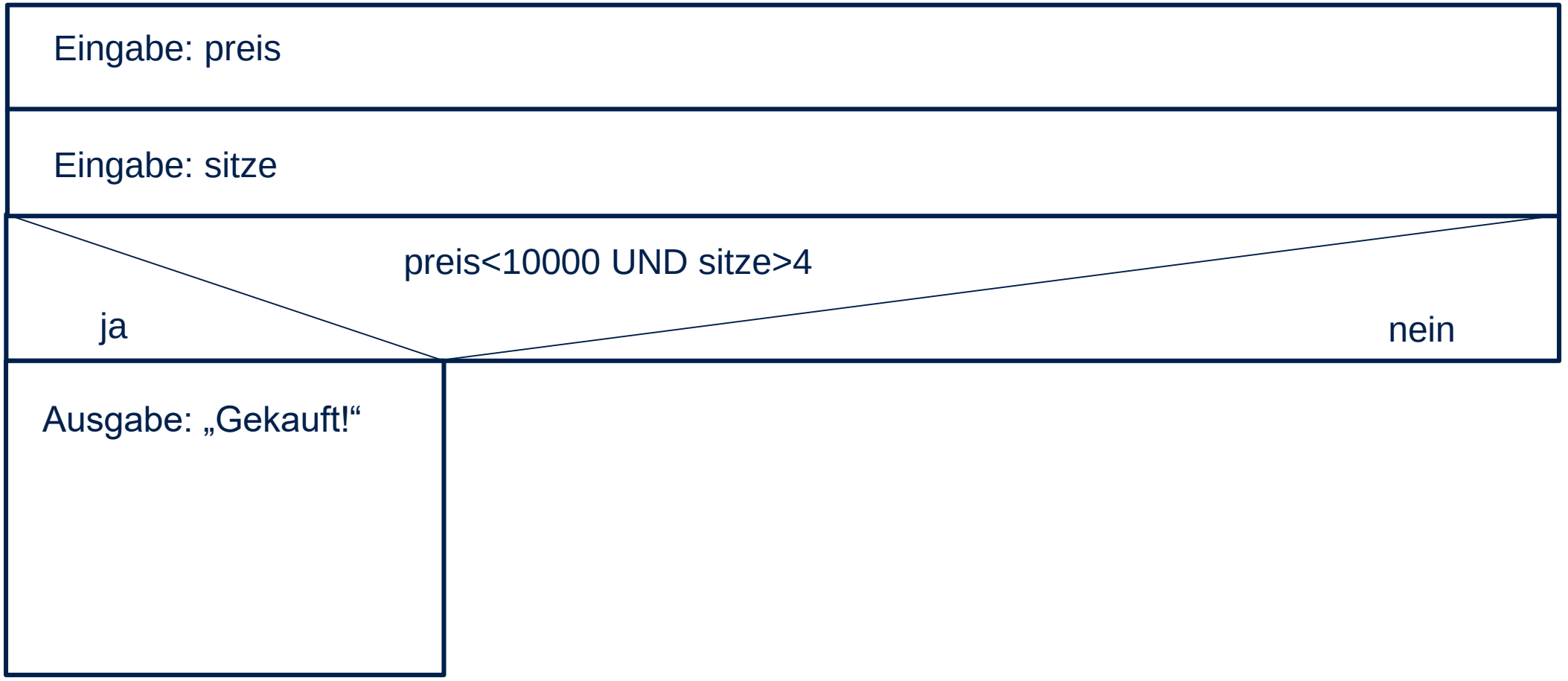
Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**

Eingabe: preis
Eingabe: sitze

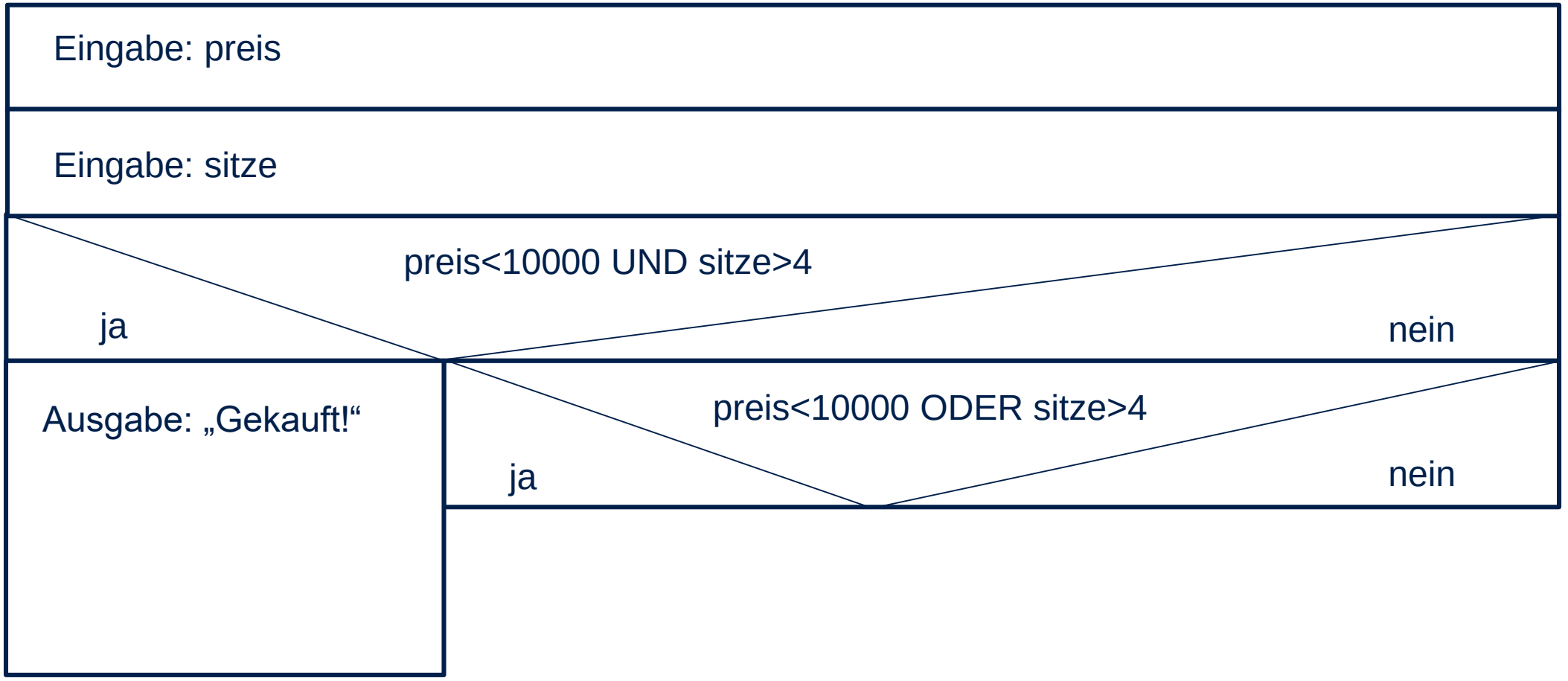
Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**



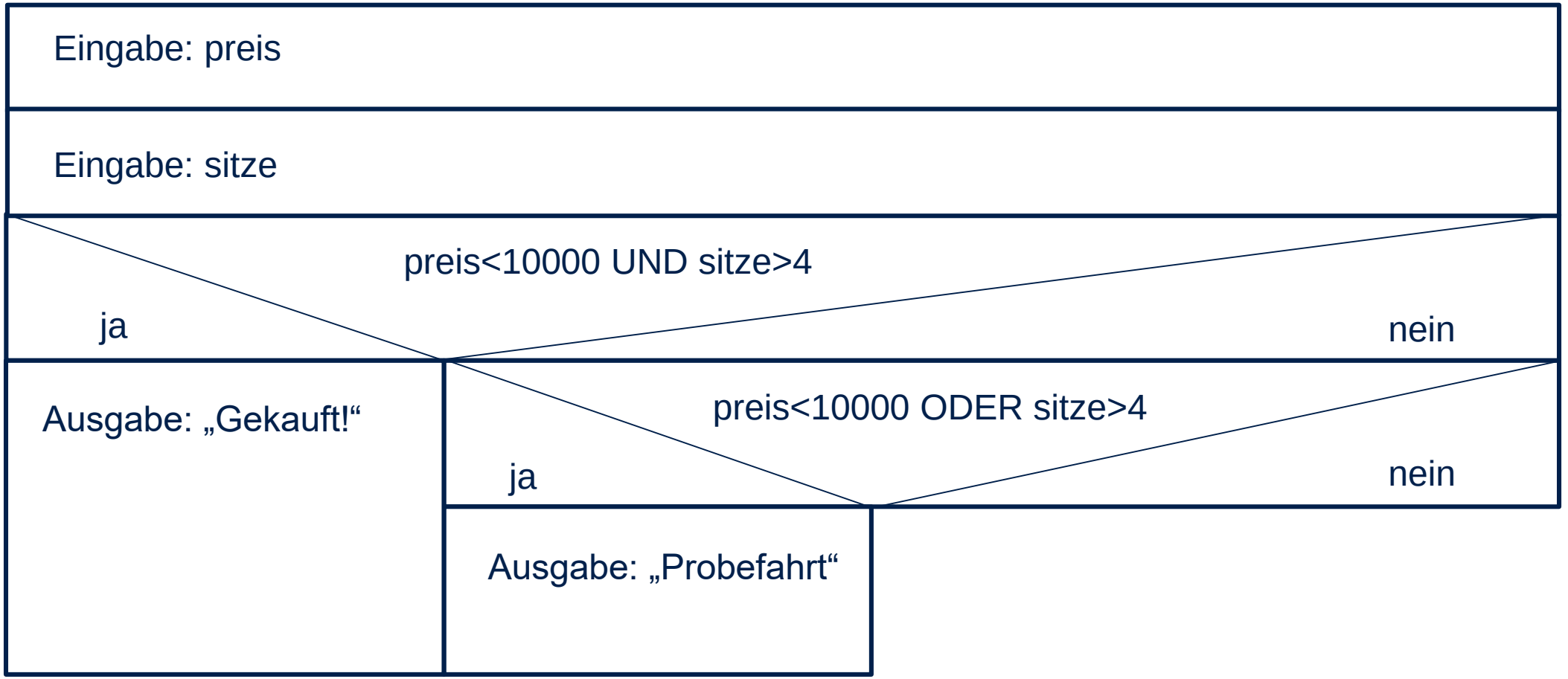
Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**



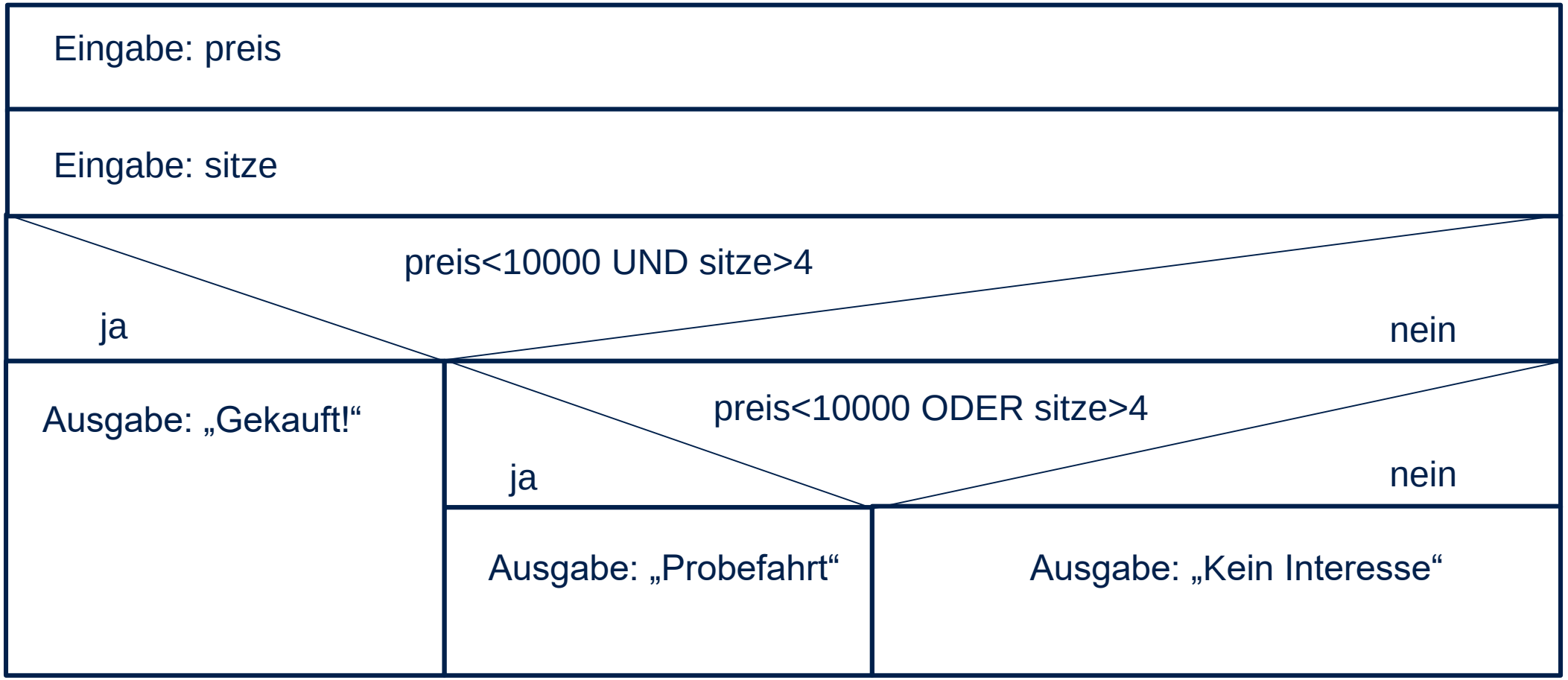
Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**



Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**



Komplexe Bedingung – **Beispielaufgabe** – **Struktogramm**



Komplexe Bedingung – **Beispielaufgabe** – **Pseudocode**

```
Programm „Komplexe Bedingung - Beispiel“
{
        Eingabe: preis
        Eingabe: sitze
        Wenn(preis<10000 UND sitze>4)
        {
                Ausgabe: „Gekauft!“
        }
        Sonst
        {
                Wenn(preis<10000 ODER sitze>4)
                {
                        Ausgabe: „Probefahrt“
                }
                Sonst
                {
                        Ausgabe: „Kein Interesse“
                }
        }
}
```

Komplexe Bedingung – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
		int preis,sitze;

		printf(“Geben Sie bitte den Einkaufspreis des Autos ein: “);
		scanf(“%d“,&preis);

		printf(“Geben Sie bitte die Anzahl der Sitze ein: “);
		fflush(stdin);
		scanf(“%d“,&sitze);


		if(preis<10000 && sitze>4)
		{
				printf(“Gekauft!“);
		}
		else
		{
				if(preis<10000 || sitze>4)
				{
						printf(“Probefahrt“);
				}
				else
				{
						 printf(“Kein Interesse“);
				}
		}
}
```

Komplexe Bedingung – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
		int preis,sitze;

		printf(“Geben Sie bitte den Einkaufspreis des Autos ein: “);
		scanf(“%d“,&preis);

		printf(“Geben Sie bitte die Anzahl der Sitze ein: “);
		fflush(stdin);
		scanf(“%d“,&sitze);


		if(preis<10000 && sitze>4)
		{
				printf(“Gekauft!“);
		}
		else
		{
				if(preis<10000 || sitze>4)
				{
						printf(“Probefahrt“);
				}
				else
				{
						 printf(“Kein Interesse“);
				}
		}
}
```

Komplexe Bedingung – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
		int preis,sitze;

		printf(“Geben Sie bitte den Einkaufspreis des Autos ein: “);
		scanf(“%d“,&preis);

		printf(“Geben Sie bitte die Anzahl der Sitze ein: “);
		fflush(stdin);
		scanf(“%d“,&sitze);


		if(preis<10000 && sitze>4)
		{
				printf(“Gekauft!“);
		}
		else
		{
				if(preis<10000 || sitze>4)
				{
						printf(“Probefahrt“);
				}
				else
				{
						 printf(“Kein Interesse“);
				}
		}
}
```

Komplexe Bedingung – **Beispielaufgabe** – **Quellcode**

```
#include<stdio.h>

main()
{
        int preis,sitze;

        printf("Geben Sie bitte den Einkaufspreis des Autos ein: ");
        scanf("%d",&preis);

        printf("Geben Sie bitte die Anzahl der Sitze ein: ");
        fflush(stdin);
        scanf("%d",&sitze);


        if(preis<10000 && sitze>4)
        {
                printf("Gekauft!");
        }
        else
        {
                if(preis<10000 || sitze>4)
                {
                        printf("Probefahrt");
                }
                else
                {
                        printf("Kein Interesse");
                }
        }
}
```

Hinweis:

Auch für den Operator „NICHT“ gibt es in ANSI C ein eigenes Symbol, nämlich das Ausrufungszeichen: **!**

Wir kennen dieses Symbol bereits von der Schreibweise für das Ungleichheitszeichen: **!=**

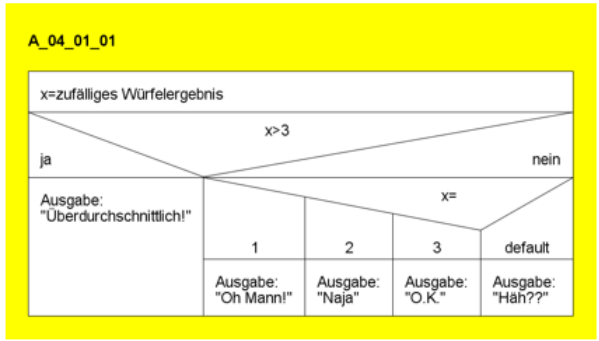
Daher ist **!(A==B)** identisch mit **A!=B**

Verschachtelte Verzweigung + Komplexe Bedingung – Gemeinsame Übung A_04_01_01

WBS TRAINING

Aufgabe_04_01_01

Gegeben sei das folgende Struktogramm:



Aufgabenstellung:
Bitte erstellen Sie dazu einen geeigneten **Quellcode** in ANSI C.

WBS TRAINING

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT!