4. Verzweigungen

Prof. Dr. Markus Gross

Informatik I für D-ITET (WS 03/04)

- Ausdruck und Anweisungen
- Verkürzte Operatoren, Vergleichsoperatoren
- Die `if` Anweisung
- Die `if else` Anweisung
- Logische Operatoren `&&`, `||`, und `!`
- Der `? :` Operator
- Die `switch` Anweisung

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Ausdruck und Anweisung

- Jede bewertbare Kombination aus Variablen, Konstanten und Operatoren ist ein *Ausdruck*
- Somit hat jeder Ausdruck einen Wert
  - `24 + 25`
  - `x = 20`
- Ebenso komplexere Ausdrücke
  - `maids = (cooks = 4) + 3;`
  - `x = y = z;`
- Zuweisung ist links nach rechts assoziativ
- Vergleichsoperatoren werden `boolean` evaluiert
- *Anweisung* durch abschliessendes `;`

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Verkürzte Opertoren

- Besonderheit in C/C++ sind *verkürzte Operatoren* `++`, `--`, `+=`, `*=`, ...
- Kommen als *prefix*-Version und als *postfix*-Version vor
  - `x++; // postfix`
  - `++x; // prefix`
- Unterschied im *Zeitpunkt* der Anwendung
- Bei postfix-Version erfolgt Anwendung NACH Auswertung des Ausdrucks
  - `int x = y++; // int x = y; y = y + 1;`
- Bei prefix-Version erfolgt Anwendung VOR Auswertung des Ausdrucks
  - `int x = ++y; // y = y + 1; int x = y; !`

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

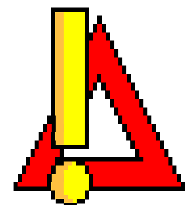
Verkürzte Operatoren

- *Verkürzte Zuweisungs-Operatoren* ebenso möglich
  - `x += 2; // x = x + 2;`
  - `x -= 2; // x = x - 2;`
  - `x *= 2; // x = x * 2;`
  - `x /= 2; // x = x / 2;`
  - `x %= 2; // x = x % 2;`
- Sehr oft verwendet

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Vergleichsoperatoren

- *Vergleichsoperatoren* (*relational operators*) dienen zum Vergleich zweier Grössen
- Evaluieren zum Typ `bool` (`true, false`)
- Typen von Vergleichsoperatoren
  - `<        // less`
  - `<=       // less than or equal to`
  - `==       // equal to`
  - `>        // greater`
  - `>=       // greater than or equal to`
  - `!=       // not equal to`
- Priorität geringer als bei arithmetischen Opertoren
  - `x + 3 > y - 2      // (x + 3) > (y - 2)`
- Zuweisung `=` und Vergleich `==` oft verwechselt



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Kontrollanweisungen

- Eine *Verzweigung* ermöglicht einen Sprung im Programmfluss (Branching)
  - Unbedingte Verzweigungen (goto, longjmp)
  - Bedingte Verzweigungen (Conditional branches)
- In C++ gibt es verschiedene Verzweigungen
  - `if`
  - `if else`
  - `switch`
- `switch` ersetzt Ketten von `if else` Konstruktionen
- Werden in Verbindung mit logischen Operatoren verwendet

Copyright: M. Gross, ETHZ, 2003

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

`if`-Anweisung

- Kommt in zwei verschiedenen Formen:
  - `if`
  - `if else`
- Syntax recht einfach:

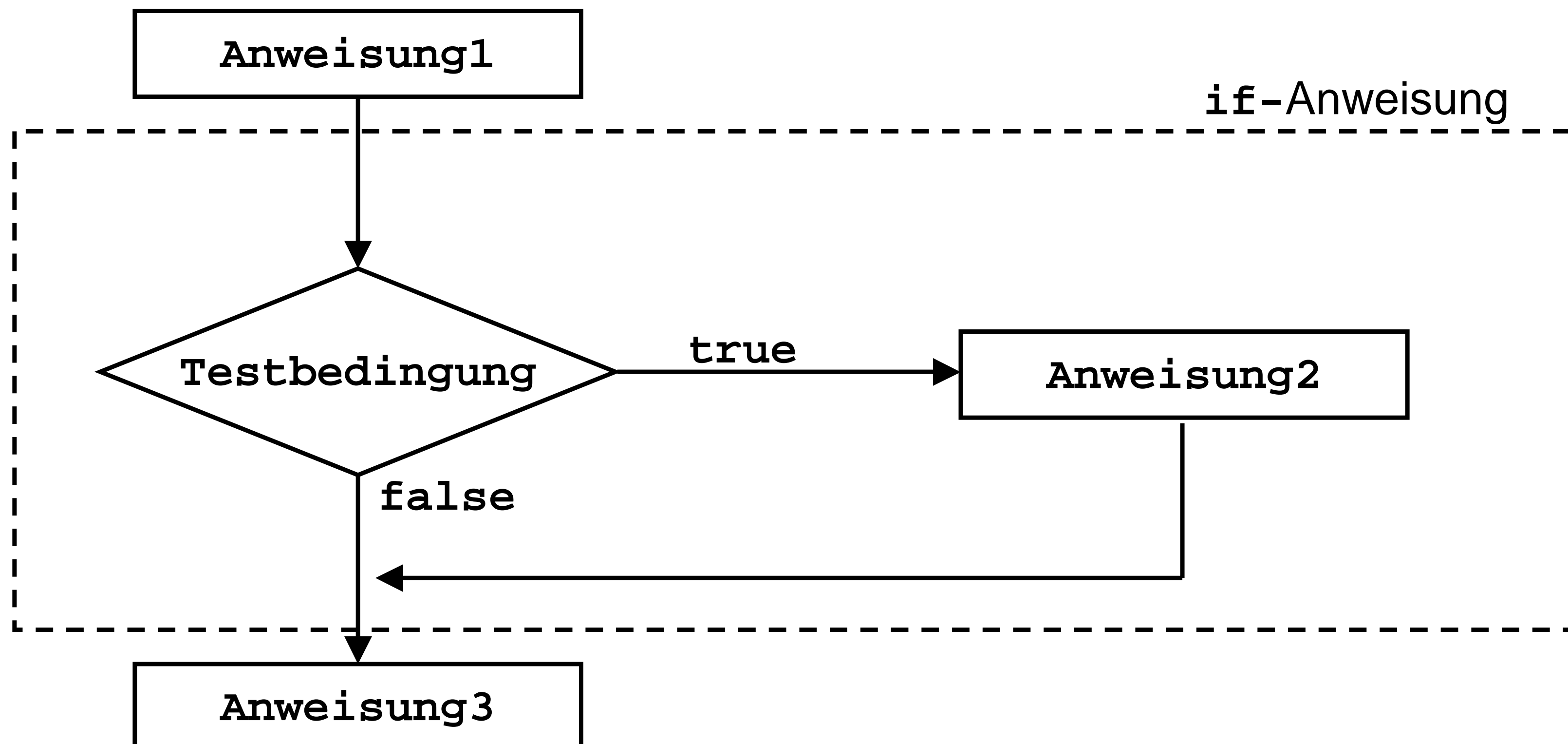
```
if (Testbedingung)
      Anweisung;
```

- Wenn *Testbedingung wahr* ist, wird die Anweisung ausgeführt
- Wenn *Testbedingung falsch* ist, wird die Anweisung übersprungen
- Die Konstruktion zählt als EINE Anweisung

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Bild `if`-Anweisung

```
Anweisung1;
if(Testbedingung)
         Anweisung2;
Anweisung3;
```



Beispiel_1: if-Anweisung

```
// if.cpp -- using the if statement

#include <iostream>
using namespace std;

int main()
{
    char ch;
    int spaces = 0;
    int total = 0;
    cin.get(ch);
    if (ch == ' ')  // check if ch is a space
        spaces++;
    total++;        // done every time

    // Repeat the same 4 times.
    cin.get(ch);
    if (ch == ' ') spaces++;
    total++;

    cin.get(ch);
    if (ch == ' ') spaces++;
    total++;

    cin.get(ch);
    if (ch == ' ') spaces++;
    total++;

    cin.get(ch);
    if (ch == ' ') spaces++;
    total++;

    cout << spaces << " spaces, " << total;
    cout << " characters total in sentence\n";
    return 0;
}
```

If-Anweisung

wird geg. übersprungen

wird immer ausgeführt

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

if else-Anweisung

- if-Anweisung kontrolliert die Ausführung einer einzelnen Anweisung oder eines Blocks
- if else-Anweisung ermöglicht die Ausführung einer von zwei Anweisungen
- Syntax ebenfalls recht einfach:

```
if (Testbedingung)
    Anweisung 1;
else
    Anweisung 2;
```

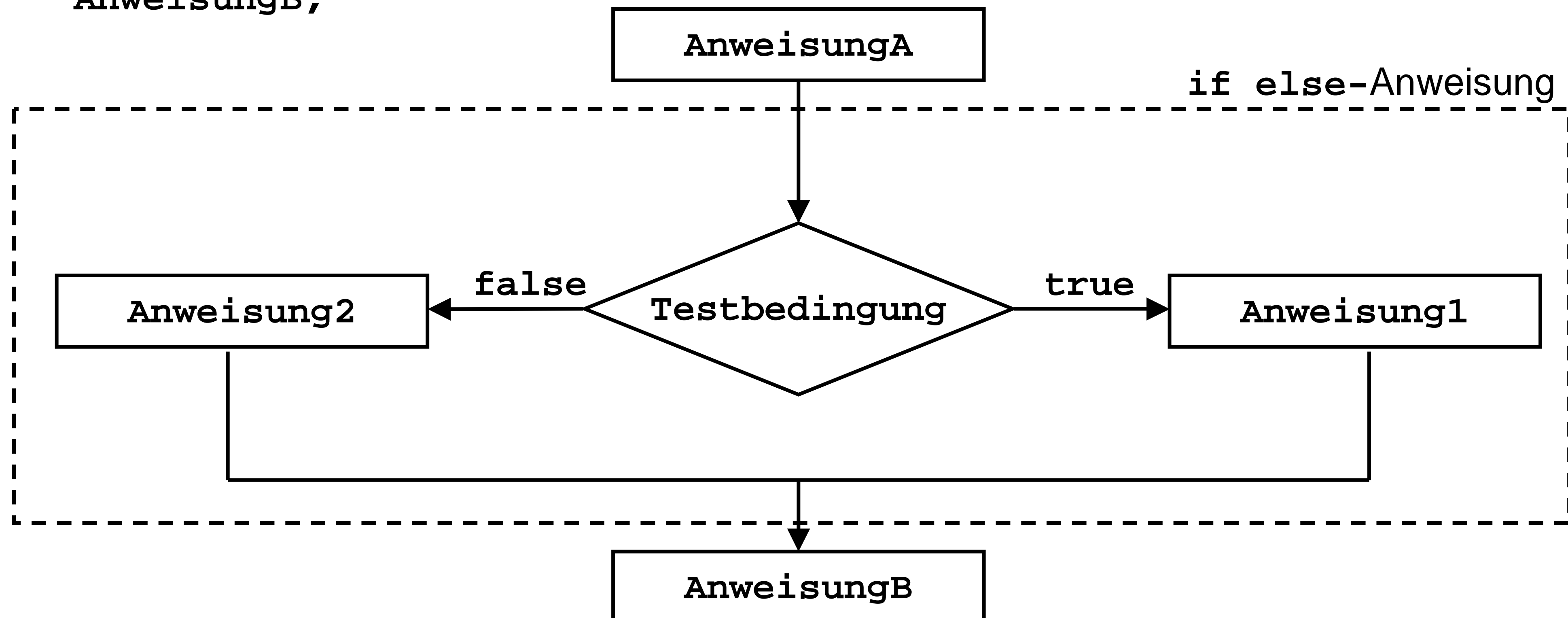
- Wenn *Testbedingung wahr* ist, wird Anweisung 1 ausgeführt
- Wenn *Testbedingung falsch* ist, wird Anweisung 2 ausgeführt

Copyright: M. Gross, ETHZ, 2003

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Bild if else-Anweisung

```
AnweisungA;
if(Testbedingung)
        Anweisung1;
else
        Anweisung2;
AnweisungB;
```



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Beispiel_2: if else Anweisung

```
// ifelse.cpp -- using the if else statement

#include <iostream>
using namespace std;

int main()
{
    char ch;

    cout << "Type, and I shall repeat.\n";
    cin.get(ch);
    if (ch == '\n')
            cout << ch;      // done if newline
    else
            cout << ++ch;    // done otherwise

    // Repeat 4 times
    cin.get(ch);
    if (ch == '\n') cout << ch;
    else cout << ++ch;

    cin.get(ch);
    if (ch == '\n') cout << ch;
    else cout << ++ch;

      cin.get(ch);
    if (ch == '\n') cout << ch;
    else cout << ++ch;

    cin.get(ch);
    if (ch == '\n') cout << ch;
    else cout << ++ch;

    // try ch + 1 instead of ++ch for interesting effect
    cout << "\nPlease excuse the slight confusion.\n";
    return 0;
}
```

Anweisung 1

Anweisung 2

wird immer ausgeführt

Copyright: M. Gross, ETHZ, 2003

Formatierung

- Block nach `if` und `else` muss immer in geschweiften Klammern folgen
- Einrückungen (tabs) sind sehr sinnvoll

```
if (a == b)
{
   i++;
   k=j;
}
else
   l=j;
e=f;
```

- Bei mehreren Anweisungen zwischen `if` und `else` ohne `{}` → Syntax Error

Copyright: M. Gross, ETHZ, 2003

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

`if else if else` -Sequenzen

- `if else`-Anweisungen können beliebig ineinander verschachtelt werden
- Beispiel:

```
if (ch == 'A')
    a_grade++;      // Alternative 1
else if (ch == 'B')
    b_grade++;      // Alternative 2
else
    soso++;         // Alternative 3
```

- Gesamte Konstruktion zählt als eine einzige Anweisung
- Alternative: `switch`-Anweisung

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Logische Ausdrücke

- *Logische* Operatoren erlauben die Verknüpfung von Ausdrücken
- In C++ gibt es drei elementare logische Operatoren:
  - `&& // logisch AND`
  - `|| // logisch OR`
  - `!  // logisch NOT`
- Mit Methoden der Boolschen Algebra kann ein beliebig komplexer logischer Ausdruck in eine Kombination dieser drei Elementaroperatoren zerlegt werden

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Logische Ausdrücke

- *OR* Operator `||` folgt der Wahrheitstabelle von ODER
- Bei der Verwendung muss die Priorität berücksichtigt werden
- Auswertungen auf linker Seite erfolgen VOR Anwendung des Operators
  - `5 > 8 || 5 < 10 // true`
  - `i++ < 6 || i == j // zuerst Inkrementieren`
- *AND* Operator `&&` folgt der Wahrheitstabelle von UND
- Wird oft in Eingabeschleifen verwendet
  - `if (17 < age < 35) // Vorsicht, immer true`

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Beispiel_3: Zahlenbereiche

```
// more_and.cpp -- use logical AND operator
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "Enter your age in years: ";
    cin >> age;
    int minimum = 0;
    int maximum = 0;

    if (age > 17 && age < 35) {
        minimum = 18;
        maximum = 34;
    }
    else if (age >= 35 && age < 50) {
        minimum = 35;
        maximum = 49;
    }
    else if (age >= 50 && age < 65) {
        minimum = 50;
        maximum = 64;
    }

    if (minimum != 0)
       cout << "Your age is in between " << minimum
            << " and " << maximum << " years!\n";

    return 0;
}
```

Initialisierung

Testsequenzen

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Logische Ausdrücke

- *NOT* Operator `!` negiert einen logischen Ausdruck
- Hat höhere Priorität als Vergleichsoperatoren
  - `if !(x < 5) // x >= 5 ist besser`
  - `if (!x < 5) // zuerst NOT x, dann Vergleich`
- AND hat höhere Priorität als OR
  - `age > 35 && age < 50 || weight > 300`
- Zuerst links ausgewertet

Copyright: M. Gross, ETHZ, 2003

Beispiel_4:Fehlerbehandlung

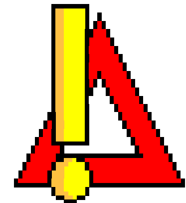
```
// not.cpp -- using the not operator
#include <iostream>
#include <climits>
using namespace std;
bool is_int(double);

int main()
{
    double num;

    cout << "Yo, dude! Enter an integer value: ";
    cin >> num;
    if (!is_int(num))    // check if num is not intable
    {
        cout << "Out of range\n";
    }
    else
    {
        int val = num;
        cout << "You've entered the integer " << val << "\n";
    }
    return 0;
}

bool is_int(double x)
{
    if (x <= INT_MAX && x >= INT_MIN)   // use climits values
        return true;
    else
        return false;
}
```

Prüfung auf Fehler mittels Negation

Der `? :` -Operator

- Kann anstelle einer `if else` Anweisung verwendet werden

  `Ausdruck 1 ? Ausdruck 2 : Ausdruck 3`



- Produziert einen Ausdruck, welcher beliebig weiterverwendet werden kann

  ```
  int c = a > b ? a : b;
  ```

- Syntax ist kompakt, aber *gewöhnungsbedürftig*
- Wird Anfängern NICHT empfohlen

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

switch-Anweisung

- switch-Anweisung ist eine Alternative zu if else-Sequenzen
- Ermöglicht elegant eine komplexe Flusskontrolle
- Syntax wie folgt:

```
switch (Integer-Ausdruck)
{
   case label_1 : Anweisung(en);
   case label_2 : Anweisung(en);
   case label_3 : Anweisung(en);
   default      : Anweisung(en);
}
```

- Das Programm springt zur Zeile, die ein entsprechendes Integer-*Label* enthält

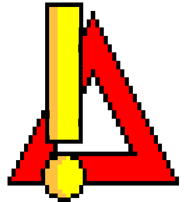
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Beispiel_5: Eingabemenü

```
// switch.cpp -- use the switch statement
// Programm-Fragment
#include <iostream>
using namespace std;
void showmenu();    // function prototypes
void report();
void comfort();
int main()
{
    showmenu();
    int choice;
    cin >> choice;

    switch(choice)
    {
    case 1  :   cout << "\a\n"; break;
    case 2  :   report(); break;
    case 3  :   cout << "The boss was in all day.\n";
                break;
    case 4  :   comfort(); break;
    default :   cout << "That's not a choice.\n";
    }

    return 0;
}
```

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

switch-Anweisung



- Nach Sprung wird Programm in nächster Zeile weitergeführt
- *Beim nächsten Label wird NICHT aus dem* **`switch`** *gesprungen*
- Dazu muss eine explizite **`break`**-Anweisung erfolgen
  - **`break`** springt zur Anweisung nach dem **`switch`**
- Wenn keiner der Fälle eintritt, springt **`switch`** zum Label **`default`**
  - Ist kein **`default`**-Label vorhanden, wird zur ersten Anweisung nach dem **`switch`** gesprungen
- Auch **`char`** Labels sind möglich
  - Characters werden intern als Integers verwaltet
- **`switch`** kann keine Wertebereiche behandeln