

```
// Add 2 Integer typed in by the user via keyboard
```

```
#include <iostream>  
using namespace std;
```

```
int main()
```

```
{
```

```
    int a,b,sum;  
    bool teilbar = true ;
```

```
    cout << "Enter integers to be added:" << endl;
```

```
    cin >> a >> b ;
```

```
    sum = a + b ;
```

```
    if ( sum%4 > 0 ) teilbar = false ;
```

```
    if (teilbar) {  
        cout << "The sum is " << sum << " and is divisible by 4" << endl ;  
    }  
    else {  
        cout << "The sum is " << sum << " and ist not divisible by 4 " << endl ;  
    }  
}
```

```
    return 0 ;
```

```
}
```

```
if ( Logischer Operator ) {  
    Anweisungen  
}  
else {  
    Anweisungen  
}
```

Diagram illustrating the execution of an if-else statement. The condition (Logischer Operator) is evaluated. If true, the first block of instructions (Anweisungen) is executed. If false, the second block of instructions (Anweisungen) is executed.

# Bedingte Anweisungen

Operatoren verknüpfen Variable zu neuen Ausdrücken, wir unterscheiden

- Arithmetische Operatoren  
Berechnung von Werten
- Bit Operatoren  
Manipulation einzelner Bits
- Vergleichsoperatoren  
Überprüfung von Aussagen
- Logische Operatoren  
Verknüpfung von Aussagen

## Operatoren

## Bedeutung

==

ist gleich

Nicht mit = verwechseln

!=

ungleich

>

groesser als

<

kleiner als

>=

groesser gleich

<=

kleiner gleich

Es kann innerhalb der if Anweisung eine bool Variable verwendet werden

## • Syntax

```
bool IFeelGood = true ;  
if ( IFeelGood ) cout << "Yeah" << endl ;
```

# Bedingte Anweisungen

Operatoren verknüpfen Variable zu neuen Ausdrücken, wir unterscheiden

- Arithmetische Operatoren  
Berechnung von Werten
- Bit Operatoren  
Manipulation einzelner Bits
- Vergleichsoperatoren  
Überprüfung von Aussagen
- Logische Operatoren  
Verknüpfung von Aussagen

Operatoren	Bedeutung
&&	und
	oder
!	not

Innerhalb der `if` Anweisung können `bool` Variablen und Vergleichs-Operatoren verknüpft werden.

- **Syntax**

```
bool IFeelGood = false ;
bool RainyDay = true , Cloudy = true ;
double SunShineHours = 8.2 ;
if ( !RainyDay && (SunShineHours > 8.0 || !Cloudy) )
    IFeelGood = true ;
```

# Bedingte Anweisungen

Operatoren verknüpfen Variable zu neuen Ausdrücken, wir unterscheiden

- Arithmetische Operatoren  
Berechnung von Werten
- Bit Operatoren  
Manipulation einzelner Bits
- Vergleichsoperatoren  
Überprüfung von Aussagen
- Logische Operatoren  
Verknüpfung von Aussagen

## • Beispiele

### Operatoren

a	b
true	true
false	false
false	true
true	false

### Möglichkeiten

!(a)	a && b	a    b
false	true	true
true	false	false
true	false	true
false	false	true

- Logische Ausdrücke werden von links nach rechts geprüft
- Bei Kombination wird geprüft bis das Resultat fest steht  
A && B && C : Wenn A falsch ist, wird B und C nicht mehr ausgewertet

# Bedingte Anweisungen

## Implementieren von Fallunterscheidungen

- Wenn ... dann .. andernfalls ...

```
if( Bedingung ) { Anweisung }  
else { Anweisung }
```

- Wenn ... dann .. andernfalls ... Wenn ... dann

```
if ( Bedingung ) { Anweisung }  
else if ( Bedingung ) { Anweisung }  
else { Anweisung }
```

In den Anweisungsblöcken können weitere bedingte Anweisungen verwendet werden.

```
if ( Bedingung ) {  
    Anweisung  
    if (Bedingung 2) { Anweisung }  
}  
else { Anweisung }
```

# Bedingte Anweisungen

Alternative Schreibweise, um abhängig vom Wert eines Ausdrucks eine Zuweisung vorzunehmen

- Auswahl Operator

- **Syntax**

(Zuweisung mit Bedingung) `?` Zuweisung wenn `true` `:` Zuweisung wenn `false`

**Beispiel:** Minimum von 2 Werten soll zugewiesen werden

```
min = a < b ? a : b
```

Ausführliche Schreibweise

```
if( a < b ) {  
    min = a;  
}  
else {  
    min = b;  
}
```

# Bedingte Anweisungen

Alternativ kann die switch Anweisung verwendet werden um Fallunterscheidungen zu implementieren.

- **Syntax**

Anweisungsblock wird erreicht, wenn die Bedingung erfüllt ist.

```
switch ( int Variable ) {  
    case integerA : { Anweisungsblock ; break }  
    ....  
    case integerB : { Anweisungsblock ; break }  
    default: { Anweisungsblock }  
}
```

wird ausgeführt, wenn nichts zutrifft.

Sprung zum Ende von switch.

Ohne break werden durch eine einmal erfüllte Bedingung alle folgenden case statements als true durchlaufen!

Häufig sieht man statt des Typs int im Kopf von switch auch char.

```
switch ( myChar ) {  
    case 'b' : { cout << "Position B reached" << endl;  
                break ;}  
    ....  
    default: { cout << "No match found" << endl; }  
}
```

```
// Add 2 Integer typed in by the user via keyboard
```

```
#include <iostream>  
using namespace std;
```

```
int main()
```

```
{
```

```
    int a,b,sum;  
    bool teilbar = true ;
```

```
    cout << "Enter integers to be added:" << endl;
```

```
    cin >> a >> b ;
```

```
    sum = a + b ;
```

```
    if ( sum%4 > 0 ) teilbar = false ;
```

```
    if (teilbar) {  
        cout << "The sum is " << sum << endl ;
```

```
    }  
    else {  
        cout << "The sum is " << sum << endl ;
```

```
    }  
  
    return 0 ;
```

```
}
```

conditional\_0.cc

### Arbeitsvorschläge:

- modifizieren Sie das Programm so, das die Division durch eine weitere Zahl getestet wird.
- passen Sie die Ausgabe an.
- Sie wollen 5 Zahlen testen, die eingegeben werden. Welches Sprach-Element in C++ brauchen wir ?

conditional.cc



# Schleifen

Wichtiges Element einer Programmiersprache

→ Wiederholung von Anweisungen

- Bearbeiten einer festgelegten Anzahl von Schleifenfolgen

```
for ( Anzahl der Schleifen ) { C++ Anweisungen }
```

Schleifenkopf: Hier wird die Anzahl der Schleifenfolgen kontrolliert und „eigenständig“ gezählt.

Anweisungsblock: Hier steht eine Iterationsvariable zur Verfügung

- Bearbeiten einer Anzahl von Schleifenfolgen mit Abbruchbedingung

```
while ( Abbruchbedingung ) { C++ Anweisungen }
```

Die Abbruchbedingung wird vorher getestet.

Anweisungsblock: Hier wird die Abbruchbedingung iteriert.

```
do { C++ Anweisungen } while (Abbruchbedingung)
```

Anweisungsblock: Hier wird die Abbruchbedingung iteriert.

Die Abbruchbedingung wird am Ende getestet.

Schleife wird hier mindestens einmal durchlaufen!

# Schleifen

Wichtiges Element einer Programmiersprache

→ Wiederholung von Anweisungen

- Bearbeiten einer festgelegten Anzahl von Schleifenfolgen

```
int j ; const int jmax = 100 ;  
double a[jmax] ;  
for (j=0 ; j < jmax ; j++ ) {  
    a[j] = static_cast<double> (j * j) ;  
}
```

- Bearbeiten einer Anzahl von Schleifenfolgen mit Abbruchbedingung

```
int j = 1 , jmax = 10 ;  
while (j <= jmax ) {  
    j++;  
    if ( j%7 ) jmax = 13;  
}
```

```
int j = 1 , jmax = 0 ;  
do {  
    j++;  
} while ( j <= jmax) ;
```

# Arrays

Arrays sind Listen von Elementen eines Datentyps. Einzelne Elemente können durch Array Indices angesprochen werden.

- **Syntax**

```
Datentyp Name [Anzahl];
```

```
Datentyp Name [N] = {a0, ... , aN-1} ;
```

Überschreitungen des Indexbereiches werden nicht notwendigerweise als Fehler gemeldet. Zufällige Speicherbereiche werden überschrieben!

```
const int maxStunden = 24 ;
```

Angabe der Arraylänge

```
double Temperatur[maxStunden] = {0.} ;
```

Definition und Initialisierung

```
Temperatur [12] = 22.0 ;
```

Zuweisung eines Elements

```
Temperatur [24] = 15.0 ;
```

Fehler: Array Grenze überschritten

Arrays können auch in mehreren Dimensionen definiert werden

- **Syntax**

```
Datentyp Name [Anzahl] [Anzahl] ;
```

# Arrays

Arrays sind Listen von Elementen eines Datentyps. Einzelne Elemente können durch Array Indices angesprochen werden.

- **Syntax**

```
Datentyp Name [Anzahl] ;
```

```
Datentyp Name [N] = {a0, ... , aN-1} ;
```

Überschreitungen des Indexbereiches werden nicht notwendigerweise als Fehler gemeldet. Zufällige Speicherbereiche werden überschrieben!

```
const int maxStunden = 24 ;
```

Angabe der Arraylänge

## Arbeitsvorschlag:

- modifizieren Sie das Programm so, das die Summe der eingegebenen Zahlen auf die Dividierbarkeit durch eine Liste von 3 Zahlen getestet wird.

Die 3 Zahlen sollen ebenfalls eingelesen werden.

Hinweise:

- Lesen Sie zunächst die Zahlen ein. Welches Sprachelement wird gebraucht?

Schreiben Sie erst diesen Programmteil und stellen sie sicher, dass das funktioniert.

- Welches Sprachelement wird fuer den 3 maligen Test gebraucht?

erung  
ments

hritten

```
// Add 2 Integer typed in by the user via keyboard
// and test division by a list of integers also specified
// via input
```

```
#include <iostream>
using namespace std;
```

## ConditionalExtended.cc

```
int main()
```

```
{
    int a,b,sum;
    bool teilbar = true ;
    const int range = 3;
    int d[range] ;
```

```
    cout << "Enter integers to be added:" << endl;
    cin >> a >> b ;
    sum = a + b ;
    cout << "Enter 3 integers to be used for division test:" << endl;
    cin >> d[0] >> d[1] >> d[2] ;
```

```
    for (int k=0; k < range ; k++) {
        teilbar = true ;
        if ( sum%d[k] > 0 ) teilbar = false ;

        if (teilbar) {
            cout << "The sum is "<<sum<<" and is divisible by "<<d[k]<<endl ;
        }
        else {
            cout << "The sum is "<<sum<<" and ist not divisible by "<<d[k]<<endl ;
        }
    }
    return 0 ;
```

```
}
```