# Grundelemente von C++

- Zeichensatz
- Kommentare
- Token
- Bezeichner
- Schlüsselwörter
- Einfache Typen
- Literale
    - Ganze Zahlen
    - Reelle Zahlen
    - Wahrheitswerte
    - Zeichen
- Variablen und Variablendeklarationen
- Benannte Konstanten
- Referenzen

# Grundelemente von C++

## Zeichensatz

bekannte Zeichensätze:

- **EBCDIC** (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode)
  8-Bit-Code,
  Verwendung im Großrechnerbereich

- **ASCII** (**A**merican **S**tandard **C**ode for **I**nfomation **I**nterchange)
  7-Bit-Code, d. h. 128 Zeichen

- **ISO-8859-1 Latin-1**
  Erweiterung des ASCII-Zeichensatzes auf
  8 Bit, d. h. 256 Zeichen

- **Unicode**
  16-Bit-Zeichensatz,
  d. h. 65536 verschiedene Zeichen darstellbar

C++ verwendet **ASCII** für die interne Darstellung von Zeichen.

Mit dem Datentyp `char` (8 Bits) sind 256 Zeichen darstellbar. Die 128 zusätzlichen Zeichen sind nicht genormt, sondern unterscheiden sich für verschiedene Rechner- und Betriebssystemtypen.

# Grundelemente von C++

## Kommentare

zwei Arten von Kommentaren:

```
// Dies ist ein sog. Zeilenende-Kommentar.
```

Alle Zeichen von `//` an bis zum Zeilenende werden ignoriert.

```
/* Dies ist ein Kommentar wie in C. */
```

Alle Zeichen zwischen `/*` und dem nächsten `*/` werden ignoriert.

Die Schachtelung von Kommentaren ist nicht möglich.

# Grundelemente von C++

## Token

Die **Token** einer Sprache, auch

- **Symbole** oder
- **lexikalische Einheiten**

genannt, sind die Wörter, auf denen die Sprache basiert.

Beispiele für Token:

- Bezeichner
- Schlüsselwörter
- Operatoren

**Wortzwischenräume** wie

- Leerzeichen
- Tabulatoren
- Zeilenvorschub-Zeichen
- Seitenvorschub-Zeichen

haben keine Bedeutung, außer daß sie die Token voneinander trennen.

Es sei denn, sie sind in Zeichen- oder Zeichenketten-Literalen enthalten.

# Grundelemente von C++

## Bezeichner

unterliegen den folgenden Regeln:

- Ein Bezeichner ist eine Folge von Buchstaben, Ziffern und Unterstrich (_).
- Ein Bezeichner beginnt stets mit einem Buchstaben oder Unterstrich.
- Groß-/Kleinschreibung wird unterschieden.
- Bezeichner dürfen beliebig lang sein.

gültige Bezeichner:

```
i
this_is_a_most_unusually_long_name
_99
_read
PI
pi
Pi
```

ungültige Bezeichner:

```
split Word
monthlyIncomeIn$
5prozent
fuenf%
```

# Grundelemente von C++

## Schlüsselwörter

Schlüsselwörter sind reservierte Begriffe und dürfen nicht als Bezeichner verwendet werden. Sie bestehen ausschließlich aus Kleinbuchstaben.

### Verwendet für Typennamen

```
bool
char
short    int      long
float    double
void
signed   unsigned
wchar_t
```

### Verwendet für benutzerdefinierte Datentypen

```
enum     struct   union    typedef
```

### Verwendet für Ausdrücke

```
true     false
new      delete   sizeof   this
```

# Grundelemente von C++

## Schlüsselwörter

### Verwendet in Anweisungen

*Auswahlanweisungen*

```
if       else
switch   case     break    default
```

*Wiederholungsanweisungen*

```
for      continue
do       while
```

*Anweisungen zur Übertragungssteuerung*

```
goto     return   throw
```

*Überwachungssysteme (exceptions)*

```
try      catch
```

### Verwendet zur Modifikation von Deklarationen

```
signed      unsigned
const       mutable     volatile
private     protected   public
```

# Grundelemente von C++

## Schlüsselwörter

### Verwendet als Speicherklassenbezeichner

```
auto  extern  register  static
```

### Verwendet für Klassen oder Methoden

```
class       friend
inline      virtual     operator
template    typename    export
```

### Verwendet für Namensräume

```
namespace  using
```

### Verwendet für explizite Typanpassungen

```
const_cast           dynamic_cast
reinterpret_cast     static_cast
explicit
```

### Verwendet zur Typerkennung zur Laufzeit

```
typeid
```

### Verwendet zum Einfügen von Assembler-Anweisungen

```
asm
```

# Grundelemente von C++

## Einfache Typen

| Typ | Inhalt | typische Größe |
|---|---|---|
| `bool` | `true` oder `false` | 8 Bit |
| `char` | ASCII-Zeichen | 8 Bit |
| `wchar_t` | „lange“ Zeichen (z.B. Unicode) | 16 Bit |
| `short` | Integer mit Vorzeichen | 16 Bit |
| `int` | Integer mit Vorzeichen | 32 Bit (oder 16 Bit) |
| `long` | Integer mit Vorzeichen | 64 Bit (oder 32 Bit) |
| `float` | IEEE 754 Fließpunktzahl | 32 Bit |
| `double` | IEEE 754 Fließpunktzahl | 64 Bit |
| `long double` | IEEE 754 Fließpunktzahl | 80 Bit |

Die tatsächlich verwendete Anzahl von Bits variiert je nach Rechnersystem; sie ist implementierungsabhängig.

Der C++-Standard legt fest:

- Bits(`short`) <= Bits(`int`) <= Bits(`long`)
- Genauigkeit(`float`) <= Genauigkeit(`double`) <= Genauigkeit(`long double`)

Durch das Schlüsselwort `unsigned` werden ganze Zahlen *ohne* Vorzeichen definiert, z.B. `unsigned long`.

# Grundelemente von C++

## Literale

In C++ besitzt jeder Typ Literale, die für die Konstanten des Typs stehen.

**Ganze Zahlen** (`short`, `int`, `long`)

Ganzzahl-Literale werden durch Zeichenketten beschrieben, die aus oktalen, dezimalen oder hexadezimalen Ziffern bestehen.

Welche Basis eine Zahl hat, wird durch die vorderen Zeichen bestimmt:

- führende `0` (Null)
  => Oktalzahl (Basis 8)
- `0x` oder `0X` am Anfang
  => Hexadezimalzahl (Basis 16)
- alle anderen Ziffern
  => Dezimalzahl (Basis 10)

Die folgenden Zahlen haben alle den gleichen Wert:

```
29      035      0x1D      0X1d
```

Die Endungen `l` bzw. `L` bedeuten `long` und die Endungen u bzw. `U` bedeuten `unsigned`.

# Grundelemente von C++

**Literale Ganze Zahlen** (`short`, `int`, `long`)

Der Datentyp eines ganzzahligen Literals hängt von dessen Form, Wert und Endung ab:

- Falls das Literal dezimal ist und keine Endung hat, hat es den ersten der Datentypen `int`, `long int` oder `unsigned long int`, der diesen Wert darstellen kann.
- Falls das Literal oktal oder hexadezimal ist und keine Endung hat, hat es den ersten der Datentypen `int`, `unsigned int`, `long int` oder `unsigned long int`, der diesen Wert darstellen kann.
- Falls das Literal die Endung `u` oder `U` besitzt, hat es den ersten der Datentypen `unsigned int` oder `unsigned long int`, der diesen Wert darstellen kann.
- Falls das Literal die Endung `l` oder `L` besitzt, hat es den ersten der Datentypen `long int` oder `unsigned long int`, der diesen Wert darstellen kann.
- Falls das Literal eine der Endungen `ul`, `lu`, `Ul`, `lU`, `uL`, `Lu`, `UL` oder `LU` hat, ist der Datentyp `unsigned long int`.

Bsp.:

| Literal | Typ auf Maschinen mit 32-Bit-`int` | Typ auf Maschinen mit 16-Bit-`int` und 32-Bit-`long` |
|---|---|---|
| `100000` | `int` | `long int` |
| `0xA000` | `int` | `unsigned int` |
| `100000L` | `long int` | `long int` |
| `0xA000U` | `unsigned int` | `unsigned int` |

# Grundelemente von C++

## Literale

### Reelle Zahlen (`float`, `double`)

Gleitkommazahlen werden durch Dezimalzahlen mit optionalem Dezimalpunkt dargestellt, denen ein Exponent folgen kann. Es muss mindestens eine Ziffer vorhanden sein.

Die folgenden Literale beschreiben alle die gleiche reelle Zahl:

```
18.     1.8e1     .18E2
```

Gleitkomma-Literale ohne Suffix sind vom Typ `double`.

Ein angehängtes kleines oder großes ‚F' spezifiziert eine Konstante vom Typ `float`.

Ein angehängtes kleines oder großes ‚L' spezifiziert eine Konstante vom Typ `long double`.

### Wahrheitswerte (`bool`)

```
true und false
```

werden von Vergleichen wie z. B. `<` zurückgeliefert

# Grundelemente von C++

## Literale

### Zeichen (`char, signed char, unsigned char`)

`char` bedeutet systemabhängig entweder `signed char` oder `unsigned char`.

Zeichen-Literale werden zwischen zwei Apostrophen (auch Hochkomma genannt) geschrieben; z. B. `'H'`.

Einige Sonderzeichen können durch spezielle Ersatzdarstellungen (Escape-Sequenzen) ausgedrückt werden.

| Ersatzdarstellung | Zeichen | Erläuterung |
|---|---|---|
| `\a` | BEL | Signalton |
| `\b` | BS | backspace (Zeichen zurück) |
| `\f` | FF | Seitenvorschub (engl. *form feed*) |
| `\n` | LF | Zeilenvorschub (engl. *newline* oder *line feed*) |
| `\r` | CR | Wagenrücklauf (engl. *carriage return*) |
| `\t` | HT | (horizontaler) Tabulator |
| `\v` | VT | Zeilensprung (engl. *vertical tabulator*) |
| `\"` | `"` | Anführungszeichen oben (engl. *double quote*) |
| `\'` | `'` | Apostroph (engl. *single quote*) |
| `\\` | `\` | inverser Schrägstrich (engl. *backslash*) |
| `\ddd` | | Zeichen mit dem oktalen ASCII-Wert ddd (0000 <= ddd <= 0377) |
| `\xdddd` `\Xdddd` | | Zeichen mit der hexadezimalen Codierung dddd (0x0000 <= dddd <= 0xffff) |

# Grundelemente von C++

## Literale

### Zeichen (`wchar_t`)

Der Datentyp `wchar_t` ist eine Erweiterung des Datentyps `char`.

„Wide characters“ sind für Zeichensätze gedacht, bei denen ein Byte nicht zur Darstellung eines Zeichens ausreicht, z.B. den Unicode-Zeichensatz.

`sizeof(wchar_t)` ergibt typischerweise 2.
Damit sind 65536 verschiedene Zeichen darstellbar.

Ein Zeichen-Literal vom Typ `wchar_t` beginnt mit einem `L`, z. B. `L'??'`.

# Grundelemente von C++

## Variablen und Variablendeklarationen

Eine **Variable** benennt einen bestimmten Speicherbereich. Eine Variable besteht aus den folgenden drei Angaben:

- Name `radius`
- Datentyp `float`
- Wert `17.5e3F`

Der **Name** ist ein Bezeichner und identifiziert die Speicheradresse.

Der **Datentyp** legt fest, in welchem Wertebereich der Wert der Variablen liegen kann.

Der **Wert** der Variablen ist der momentane Inhalt des Speicherbereichs.

# Grundelemente von C++

## Variablen und Variablendeklarationen

Variablen müssen vor ihrer Verwendung deklariert werden. Sie können bei der Deklaration initialisiert werden.

Syntax der Variablendeklaration (vereinfacht):

***Modifizierer-Liste Typname Bezeichner-Liste ;***

Bezeichner-Liste ist eine Folge von Bezeichnern, die durch Komma getrennt sind. Für jeden Bezeichner kann ein Initialisierungsausdruck angegeben werden.

Beispiele:

```
int   daysInMonth, daysInYear = 365;
short smallInt;
long  bigInt;
int   one = 1, two = 2, three;
```

# Grundelemente von C++

## Benannte Konstanten

Häufig verwendete Konstanten kann man benennen und dann im Programm diesen Bezeichner verwenden.

Dies erhöht die Lesbarkeit und Flexibilität des Programms.

Zur Definition von benannten Konstanten wird der Modifizierer `const` benutzt.

Die Konstante PI kann folgendermaßen definiert werden:

```
const double pi = 3.141592654;
```

Der Wert von `pi` kann im Programm nicht verändert werden. Die Anweisung

```
pi = 0.05;
```

führt zu einer Fehlermeldung.

# Grundelemente von C++

## Referenzen

Eine **Referenz** ist ein Datentyp, der einen *Verweis* auf ein Objekt liefert.

Eine Referenz bildet einen *Aliasnamen* für ein Objekt, über den es ansprechbar ist. Das Objekt hat damit zwei Namen.

Beispiel:

```
int intValue;
int &ref = intValue;
```

Die folgenden beiden Anweisungen haben dann den gleichen Effekt:

```
intValue = 100;   // aendert intValue
ref = 100;        // aendert intValue
```

Referenzen müssen bei der Deklaration initialisiert werden:

```
int &ref;
```

ist nicht erlaubt.

Eine Referenz auf eine Referenz und die Neuinitialisierung einer Referenz sind nicht möglich.