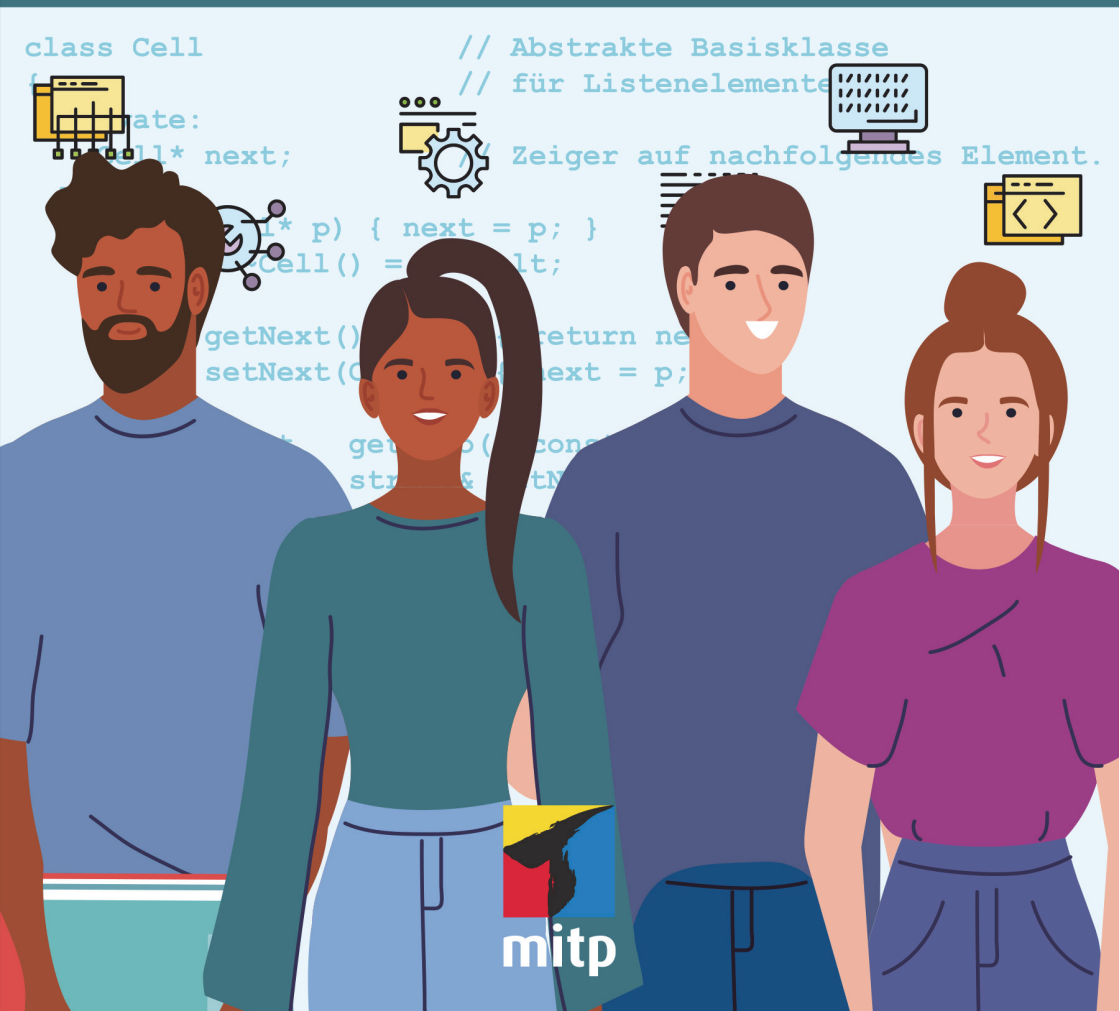
Ulla Kirch | Peter Prinz

C++

DAS ÜBUNGSBUCH

Testfragen und Aufgaben mit Lösungen

6. AUFLAGE



Ulla Kirch,
Peter Prinz

C++
Das Übungsbuch
Testfragen und Aufgaben mit Lösungen

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN 978-3-7475-0638-7
6., überarbeitete Auflage 2023

www.mitp.de

E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Sabine Schulz
Sprachkorrektorat: Petra Heubach-Erdmann
Coverbild: Gstudio / stock.adobe.com
Satz: III-satz, Husby, www.drei-satz.de

Inhaltsverzeichnis

Einleitung

Dieses Buch wendet sich an Leser, die ihre C++-Kenntnisse durch »Learning by Doing« vertiefen möchten. Es ist ideal, um sich im Stil eines Workshops auf Prüfungen oder auf die Mitarbeit in einem C++-Projekt vorzubereiten.

Die Gliederung des Stoffs entspricht der des Buches »C++ Lernen und professionell anwenden«, das ab der 9. Auflage den neuesten C++-Standard von 2020 (kurz C++20) berücksichtigt. Neue Sprachelemente sind in aktuellen Compilern noch nicht voll integriert und werden deshalb in diesem Buch mit (*) gekennzeichnet.

Aber es ist nicht wesentlich, wie Sie C++ gelernt haben. Jedes Kapitel beginnt mit einer Zusammenfassung des Stoffs, zu dem anschließend Fragen und Aufgaben gestellt werden. Beispielsweise ist das Thema des 9. Kapitels »Die Standardklasse `string`«. Wenn Ihnen dann der Inhalt der Zusammenfassung zur `string`-Klasse vertraut ist, sollten Sie auch ohne größere Probleme die anschließenden Fragen und Aufgaben lösen können.

Jedes Kapitel besteht neben der einführenden Beschreibung des Themas aus drei weiteren Teilen: Verständnisfragen, Programmieraufgaben und den Musterlösungen zu allen Fragen und Aufgaben. Mit jeweils 20 Verständnisfragen können Sie testen, wie gut Sie sich in dem jeweiligen Themenbereich auskennen. Die Art der Fragen sind entweder Ja-Nein-Fragen, Multiple-Choice-Fragen oder es muss eine Aussage vervollständigt werden.

Im Aufgabenteil können Sie dann Ihr Wissen praktisch umsetzen. In jedem Kapitel gibt es mindestens zehn Aufgaben mit steigendem Schwierigkeitsgrad. Die Bearbeitung einfacher Aufgaben ist oft in wenigen Minuten erledigt. Dagegen kann die Lösung umfänglicher Aufgaben auch Tage in Anspruch nehmen. Dies gilt insbesondere bei Aufgaben zu den Themen »Dynamische Elemente«, »Vererbung« und »Polymorphie«. Umfangreichere Problemstellungen sind dabei oft auf mehrere Aufgaben verteilt.

Bei der Auswahl der Problemstellungen für Aufgaben wurde stets darauf geachtet, dass sie typisch und praxisnah sind. Auf diese Weise lernen Sie viele interessante Algorithmen und Datenstrukturen kennen. Auch durch die eigenständige Implementierung von »Iteratoren« und »Intelligenten Zeigern« vertiefen Sie Ihr Verständnis für die Konzepte der Standardbibliothek. In jedem Fall verfügen Sie nach der Durcharbeitung des Buches über fundierte Programmiererfahrungen und einen umfangreichen Fundus von Beispiel-Code.

Trotz ausfühlicher Aufgabenstellungen und vieler Hinweise kann es immer mal vorkommen, dass man nicht zum Ziel kommt. Dann hilft ein Blick in die kommentierten Musterlösungen. Außerdem ist es sicher immer interessant, die eigene Lösung mit der im Buch zu vergleichen. Die Musterlösungen finden Sie auch im Internet unter `http://www.mitp.de/0637`.

Dem Leser wünschen wir viele Erfolgserlebnisse beim Lösen der Übungen.

Ulla Kirch
kirch@hm.edu

Peter Prinz
prinz_peter@t-online.de

Grundlagen

Dieses Kapitel umfasst grundlegende Fragen und Aufgaben zur Erstellung von C++-Programmen. Hierzu zählen auch das

- Inkludieren von Header-Dateien
  Eine Header-Datei beinhaltet Informationen, die von einem C++-Programm verwendet werden. In der Header-Datei `iostream` beispielsweise sind Informationen enthalten, die zur Ein-/Ausgabe von Daten erforderlich sind. Eine Header-Datei wird mit der `#include`-Direktive in ein Programm kopiert.
- Verwenden der `using`-Direktive
  Vordefinierte Namen, wie z.B. `cout`, gehören zum Namensbereich `std`. Die Direktive `using namespace std;` ermöglicht es, diese Namen ohne den Vorsatz `std::` direkt zu verwenden.
- Formulieren von Anweisungen
  Eine Anweisung legt fest, was das Programm tun soll, und wird stets mit einem Semikolon abgeschlossen. Zur Ausgabe von Daten auf den Bildschirm wird in C++ der Stream `cout` verwendet, z.B. `cout << "Hallo";`
- Definieren einer `main`-Funktion
  Die erste Funktion, die in einem C++-Programm ausgeführt wird, ist stets die `main`-Funktion. Die auszuführenden Anweisungen stehen im Funktionsblock, d.h. innerhalb der Klammern { }. Bei Erreichen der `return`-Anweisung wird die Funktion verlassen.
- Kommentieren von Quelldateien
  Kommentare dienen zur Dokumentation in einem Programm. Sie verbessern die Lesbarkeit und können bei der Fehlersuche nützlich sein. Jede Zeichenfolge, die in `/* ... */` eingeschlossen ist oder mit `//` beginnt, ist ein Kommentar. Der Compiler ignoriert Kommentare.

Verständnisfragen

1.1 C++ ist eine rein objektorientierte Sprache.

☐ Richtig ☐ Falsch

1.2 Die umfangreiche in C entwickelte Software kann auch in C++-Programmen verwendet werden.

☐ Richtig ☐ Falsch

1.3 Eine Quelldatei wird zur Übersetzung an den _____ übergeben.

1.4 Der _____ bindet eine Objektdatei mit anderen Modulen zu einer ausführbaren Datei.

1.5 Die gebräuchlichsten Endungen im Namen von Quelldateien sind

a) .c **b)** .cpp **c)** .cc

1.6 Standardisierte Funktionen und Klassen sind in der _____ enthalten.

1.7 Bei der Suche nach Fehlern in einem C++-Programm beginnen Sie immer mit

a) dem letzten vom Compiler angezeigten Fehler.

b) irgendeinem angezeigten Fehler.

c) dem ersten angezeigten Fehler.

1.8 Eine Warnung kann einen

a) Syntaxfehler anzeigen.

b) logischen Fehler anzeigen.

c) Laufzeitfehler anzeigen.

1.9 Jedes C++-Programm enthält die Funktion _____.

1.10 In einem C++ Programm bedeutet das Doppelkreuz # am Anfang einer Zeile, dass diese Zeile für

a) den Compiler bestimmt ist.

b) den Präprozessor bestimmt ist.

c) die Header-Datei bestimmt ist.

1.11 Vordefinierte Namen der C++-Standardbibliothek befinden sich im Namensbereich _____.

1.12 Die Programmausführung beginnt (abgesehen von der Initialisierung globaler Objekte) mit

a) der ersten #include-Direktive.

b) der ersten Anweisung in der Funktion `main()`.

c) der zuerst definierten Funktion.

1.13 Der Name `cout` bezeichnet ein Objekt, das zuständig ist für

a) Eingaben.

b) den Programmstart.

c) Ausgaben.

1.14 In der Funktion `main()` bewirkt die Anweisung

```
return 0;
```

a) das Verlassen von `main()`.

b) die Beendigung des Programms.

c) die Rückgabe des Exitcode `0` an das aufrufende Programm.

1.15 Die kürzeste Anweisung besteht aus _____.

1.16 C++-Funktionen müssen in einer bestimmten Reihenfolge definiert werden.

☐ Richtig ☐ Falsch

1.17 Die erste Funktion, die in einer Quelldatei definiert wird, ist stets die Funktion `main()`.

☐ Richtig ☐ Falsch

1.18 Die Anweisungen, die in der Funktion `main()` ausgeführt werden, stehen im _____.

1.19 Zeichenfolgen werden als Kommentare interpretiert, wenn sie

a) mit `/*` beginnen.

b) in `/*` `*/` eingeschlossen sind.

c) mit `//` beginnen.

1.20 In einer Zeile können mehrere Präprozessor-Direktiven angeführt werden.

☐ Richtig ☐ Falsch

Aufgaben

1.1 Was gibt das folgende Programm auf dem Bildschirm aus?

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

int main()
{
   cout << "Hi Leute, ";
   cout << endl;
   cout << "was habt Ihr heute noch vor";
   cout << "?" << endl;
   return 0;
}
```

1.2 Formulieren Sie die entsprechenden Anweisungen, um

```
Mir geht's gut!
```

a) beginnend bei der aktuellen Cursorposition auszugeben.

b) am Anfang der nächsten Zeile auszugeben.

1.3 Jedes der folgenden Programme enthält einen Fehler. Bestimmen und korrigieren Sie jeden Fehler.

a)

```
#include <iostream>
int main()
{  // Und jetzt kommt der berühmteste Spruch
   // aus der Welt der Programmiersprachen:
   cout << "Hello, World!" << endl;
   return 0;
}
```

b)

```
#include <iostream>
using namespace std;
int main()
{
   cout >> "Hello, World!" >> endl;
}
```

c)

```
#include <iostream>
using namespace std;
int main()
{
   / Wer zum Teufel hat das gesagt? /
   cout << "Hello, World!" << endl;
   return 0;
}
```

d)

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hallo, Universum! ";
        << endl;
   return 0;
}
```

1.4 Schreiben Sie ein C++-Programm, das Ihren Namen, Ihre Adresse, Telefonnummer und E-Mail-Adresse in je einer Zeile auf dem Bildschirm ausgibt.

1.5 Fügen Sie Kommentare in die Lösung zur Aufgabe 1.4 ein, und zwar einen Programmnamen, den Namen des Programmierers sowie eine Beschreibung, was das Programm macht.

1.6 Schreiben Sie ein C++-Programm, das folgendes Menü ausgibt:

```
********  Telefonverzeichnis  ********

    E  =  Neuen Eintrag einfuegen
    L  =  Eintrag loeschen
    S  =  Telefonnummer suchen
    A  =  Alle Eintraege anzeigen
    B  =  Programm beenden

Ihre Wahl:
```

1.7 Sind die folgenden C++-Programme vollständig und fehlerfrei?

a)

```
int main()
{
  return 0;
}
```

b)

```
include <iostream>
using namespace std;
int main()
{
   cout << "Hey, los!" << return 0;
}
```

c)

```
#include <iostream>
using namespace std;
int main(
){
cout <<
"Das wär's für heute!" << endl; return 0
;}
```

1.8 Angenommen, die folgenden Anweisungen befinden sich in einer `main`-Funktion. Was ist falsch?

a) `cout >> "Weiter mit <return>" >> endl;`

b) `return "Alles klar!";`

c) `cout "<< Geben Sie eine Zahl ein: <<" endl;`

1.9 Verfolgen Sie den Ablauf des folgenden C++-Programms und beschreiben Sie, was auf dem Bildschirm ausgegeben wird.

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

void star1(), star2(), star3();

int main()
{
```

```
   star1();
   star2();
   star3();
   star2();
   star1();
   return 0;
}

void star1() {  cout << "****" << endl; }

void star2() {  cout << "********" << endl; }

void star3() {  cout << "************" << endl; }
```

1.10 Ändern Sie die `main`-Funktion aus der letzten Aufgabe so, dass folgende Grafik ausgegeben wird:

```
************
********
****
********
************
```

Fügen Sie außerdem Kommentare in den Quellcode ein und erklären Sie, was das Programm macht.

Lösungen zu den Verständnisfragen

1.1 Falsch (C++ ist eine Erweiterung der prozeduralen Programmiersprache C.)

1.2 Richtig

1.3 Compiler

1.4 Linker

1.5 **b)** und **c)**

1.6 C++-Standardbibliothek

1.7 **c)**

1.8 **b)**

1.9 `main()`

1.10 **b)**

1.11 `std`

1.12 **b)**

1.13 **c)**

1.14 **a)**, **b)** und **c)**

1.15 einem Semikolon

1.16 Falsch

1.17 Falsch

1.18 Funktionsblock von `main()`

1.19 **b)** und **c)**

1.20 Falsch

Lösungen zu den Aufgaben

1.1

```
Hi Leute,
was habt Ihr heute noch vor?
```

1.2

```
cout << "Mir geht's gut!";
cout << endl << "Mir geht's gut!";
(oder:  cout << "\nMir geht's gut!"; )
```

1.3 **a)** Hinter der Direktive `#include <iostream>` fehlt in einer neuen Zeile:

```
using namespace std;
```

Alternativ kann auch `std::cout` und `std::endl` verwendet werden.

b) Statt >> ist der Operator << zu verwenden. Die abschließende Anweisung `return 0;` darf fehlen. Sie wird dann vom Compiler eingefügt.

c) Innerhalb der `main()`-Funktion ist der Kommentar syntaktisch nicht korrekt. Richtig wäre beispielsweise:

```
// Wer zum Teufel hat das gesagt?
/* Wer zum Teufel hat das gesagt? */
```

d) In der ersten Zeile im Rumpf der `main()`-Funktion muss das Semikolon entfernt werden.

1.4

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Sarah Miller"       << endl
        << "Karenstr. 123  "    << endl
        << "80123 Muenchen"     << endl
        << "Tel. (089) 6543210" << endl
        << "sarah.m@yahoo.com"  << endl;
   return 0;
}
```

1.5

```
// ------------------------------------------------
// Programmname: ex01_05.cpp
// Autor: Sarah Miller
// Das Programm gibt einen Namen, eine Adresse, eine
// Tel.-Nr. und eine E-Mail-Adresse auf dem Bildschirm aus.
// ------------------------------------------------
#include <iostream>
using namespace std;
int main()
{
   // Wie in der Lösung zur Aufgabe 1.4.
}
```

1.6

```
// ------------------------------------------------
// ex01_06.cpp
// Gibt ein Menü für ein Telefonverzeichnis aus.
// ------------------------------------------------
#include <iostream>
using namespace std;

int main()
{
   cout << "********  Telefonverzeichnis  ********"
        << endl << endl;
   cout << "    E  =  Neuen Eintrag einfuegen" << endl;
   cout << "    L  =  Eintrag loeschen"          << endl;
   cout << "    S  =  Telefonnummer suchen"      << endl;
   cout << "    A  =  Alle Eintraege anzeigen"   << endl;
   cout << "    B  =  Programm beenden"          << endl
        << endl;
```

```
   cout << "Ihre Wahl: ";

   cout << endl;
   return 0;
}
```

1.7 **a)** Das Programm tut zwar nichts, der Quellcode ist aber fehlerfrei und vollständig.

b) Im Quellcode liegen zwei Fehler vor:

1. Das Zeichen `#` fehlt vor `include`.
2. `return 0;` muss als separate Anweisung angeführt werden, d.h. nicht als Teil der Anweisung `cout << ....;`.

c) Der Quellcode ist fehlerfrei und vollständig, aber schlecht lesbar.

1.8 **a)** Anstelle von `>>` ist das Symbol `<<` zu verwenden, um den Text in den Ausgabestrom einzufügen.

b) Bei dem Return-Wert der `main`-Funktion muss es sich um eine Ganzzahl handeln.

c) Die Symbole `<<` müssen sich außerhalb des Strings `"Geben Sie eine Zahl ein: "` befinden.

1.9

```
****
********
************
********
****
```

1.10

```
// ----------------------------------------------------
// ex01_10.cpp
// Modifizierung des Programms aus Aufgabe 1.9.
// ----------------------------------------------------
int main()
{
   star3();    // Gibt 3*4 = 12 Sterne aus.
   star2();    // Gibt 2*4 =  8 Sterne aus.
   star1();    // Gibt 4  Sterne aus.
   star2();    // Gibt 8  Sterne aus.
   star3();    // Gibt 12 Sterne aus.
   return 0;
}
```

Elementare Datentypen, Konstanten und Variablen

In diesem Kapitel arbeiten Sie mit

- ganzzahligen Typen
  Für die Darstellung von Zeichen stehen in C++ die Typen `char`, `char8_t`(*), `char16_t`, `char32_t` und `wchar_t` zur Verfügung. Sie unterscheiden sich durch die darstellbaren Zeichensätze. Für Ganzzahlen gibt es die Typen `short`, `int`, `long` und `long long`, die sich durch ihre Wertebereiche unterscheiden und die standardmäßig mit Vorzeichen interpretiert werden. Durch Voranstellen des Schlüsselwortes `signed` oder `unsigned` wird explizit festgelegt, ob der Typ mit oder ohne Vorzeichen interpretiert wird.
- Gleitpunkttypen
  Zur Darstellung von Gleitpunktzahlen stehen die Typen `float`, `double` sowie `long double` zur Verfügung. Sie unterscheiden sich durch ihren Wertebereich und die Genauigkeit. Die Genauigkeit `n` bedeutet, dass zwei Gleitpunktzahlen, die sich in den ersten `n` Dezimalziffern unterscheiden, intern verschieden gespeichert werden.
- Literalen
  Bei einem Literal handelt es sich um eines der Schlüsselwörter `true` oder `false` oder um eine Zeichenfolge, die eine numerische Konstante, eine Zeichenkonstante oder eine String-Konstante darstellt. Ganzzahlige Konstanten können dezimal, oktal (mit führender `0`) oder hexadezimal (mit führendem `0x` oder `0X`) dargestellt werden. Gleitpunktkonstanten werden auch in exponentieller Schreibweise (z.B. `1.8E-2`) verwendet. Zeichenkonstanten bestehen aus einem Zeichen eingeschlossen in einfachen Hochkommas (z.B. `'A'`). String-Konstanten werden in doppelte Hochkommas eingeschlossen (z.B. `"ok?"`). Sonderzeichen können als Escape-Sequenzen angegeben werden (z.B. `\n`).
- Variablen
  Daten können in Variablen gespeichert werden. Bevor eine Variable im Programm verwendet wird, muss sie definiert werden. Dabei werden Typ und Name der Variablen festgelegt und die Variable ggfs. initialisiert. Bei einer Definition mit Initialisierung ist eine *automatische Typableitung* möglich: Statt eines konkreten Typs wird `auto` angegeben und die Variable erhält den Typ des Initialisierers.

  Namen bestehen aus einer Folge von Buchstaben (ohne Umlaute und ß), Ziffern oder Unterstrichen. Das erste Zeichen darf keine Ziffer sein. Groß- und Kleinschreibung wird unterschieden. Schlüsselwörter (wie z.B. `namespace`) dürfen nicht als Name verwendet werden.

Verständnisfragen

2.1 Ein Datentyp bestimmt

a) die Art der Darstellung der Daten auf dem Bildschirm.

b) die Art der internen Darstellung der Daten.

c) die Größe des benötigten Speicherplatzes.

2.2 Der Wert `false` wird intern dargestellt durch _____.

2.3 Datentypen zur Darstellung von Gleitpunktzahlen in C++ sind folgende:

a) `float`

b) `long`

c) `long double`

2.4 Die ganzzahligen Typen `short`, `int`, `long` und `long long` werden

a) ohne Vorzeichen interpretiert.

b) mit Vorzeichen interpretiert.

2.5 Der Ausdruck

```
sizeof(double)
```

liefert die Größe eines Objekts vom Typ `double` in Anzahl

a) Bits.

b) Bytes.

c) Millimeter.

2.6 Bei einer Genauigkeit von 6 Dezimalziffern ist garantiert, dass die Zahlen 5.12345 und 5.123456 unterschieden werden.

☐ Richtig ☐ Falsch

2.7 Eine oktale Konstante beginnt mit einer führenden `0` und eine hexadezimale Konstante beginnt mit den beiden Zeichen `0x` oder `0X`.

☐ Richtig ☐ Falsch

2.8 Welche der folgenden Konstanten hat einen Gleitpunkttyp?

a) 12

b) 12.

c) 1f2

2.9 Welche der folgenden Konstanten hat den Typ `char`?

a) `0`

b) `'0'`

c) `"0"`

2.10 Das Stringendzeichen `'\0'` entspricht dem Zeichen `'0'`.

☐ Richtig ☐ Falsch

2.11 Der String `"Oh!"` belegt ____ Bytes.

2.12 Escape-Sequenzen werden zur Darstellung nicht druckbarer Zeichen eingesetzt.

☐ Richtig ☐ Falsch

2.13 Stringkonstanten, die nur durch Zwischenraumzeichen (Blanks, Tabs und Newline-Zeichen) getrennt sind, werden zu einem String zusammengezogen.

☐ Richtig ☐ Falsch

2.14 Bei welcher der Zeichenfolgen handelt es sich um einen zulässigen Namen?

a) f_x

b) C++

c) 5_f

2.15 Wird eine Variable ohne Initialisierung definiert, so wird

a) der Typ und Name der Variablen festgelegt.

b) der Variablen automatisch ein Anfangswert zugewiesen.

c) der entsprechende Speicherplatz für die Variable reserviert.

2.16 Eine außerhalb jeder Funktion definierte Variable heißt auch _____.

2.17 Jede globale Variable, die nicht explizit initialisiert wird, wird mit _____ vorbelegt.

2.18 Bei der Ausgabe mit `cout` werden Ganzzahlen standardmäßig

a) oktal dargestellt.

b) hexadezimal dargestellt.

c) dezimal dargestellt.

2.19 Zur Definition einer Variablen, die einmal initialisiert wird und später nicht mehr verändert werden kann, wird das Schlüsselwort _____ verwendet.

2.20 Gegeben ist die folgende Variablendefinition:

```
auto var = 'X';
```

Dann hat die Variable `var` den Typ _____ .

Aufgaben

2.1 Bestimmen Sie den Typ der folgenden Konstanten:

a) `2L`	**b)** `1.23456f`	**c)** `0302`
d) `'\0101'`	**e)** `100ULL`	**f)** `.1e-5`
g) `0x10`	**h)** `1.2345678`	**i)** `0xFL`

2.2 Schreiben Sie ein C++-Programm, das den Wertebereich der Datentypen `char` und `wchar_t` ausgibt. Verwenden Sie die Konstanten `CHAR_MIN`, `CHAR_MAX`, `WCHAR_MIN` und `WCHAR_MAX`, die den kleinsten und größten möglichen Wert des jeweiligen Typs darstellen. Diese Konstanten sind in der Header-Datei `climits` definiert.

2.3 Schreiben Sie ein C++-Programm, das die Größe des Speicherplatzes, den größten Wert, den kleinsten positiven Wert und die Genauigkeit des Datentyps `double` ausgibt. Die Anzahl Bytes, die ein Objekt eines bestimmten Typs *typ* im Speicher benötig, liefert der Operator `sizeof(`*typ*`)`. Das Ergebnis hat den Typ `size_t`, der als `unsigned long` oder `unsigned long long` definiert ist.

Verwenden Sie die Konstanten DBL_MAX, DBL_MIN und DBL_DIG, die den größten Wert, den kleinsten positiven Wert und die Genauigkeit darstellen. Die Konstanten sind in der Header-Datei `cfloat` definiert.

2.4 Bestimmen Sie, welche der folgenden Variablennamen in C++ gültig sind:

a) `const`	**b)** `CHAR`	**c)** `size1`
d) `2_length`	**e)** `myFile`	**f)** `my@address`
g) `go-on`	**h)** `slow_down`	**i)** `okay!`

2.5 Formulieren Sie die entsprechenden Anweisungen zur Ausgabe von:

a)

```
Ungueltiger Dateiname:
C:\temp\Grocer's.cpp
```

b)

```
Ihre Eingabe:
```

und einem Ton, der die Aufmerksamkeit des Benutzers weckt.

c)

```
"Left to themselves,"
       "things tend to go"
              "from bad to worse." (Murphy)
```

Verwenden Sie Escape-Sequenzen, um doppelte Anführungsstriche, horizontale Tabs und Zeilenvorschübe auszugeben.

2.6 In C++-Programmen sind die Variablen `side`, `circumference` und `area` erforderlich, um den Umfang und den Flächeninhalt eines Quadrats zu berechnen. Definieren Sie die Variablen und initialisieren Sie die Variable `side` mit dem Wert `1.0`.

2.7 Welche Fehler liegen in den folgenden Variablendefinitionen vor?

a) `int n, int i;`
b) `double side length;`
c) `Short min(0);`
d) `int n;  double result(.5)`
e) `char c['a'];`
f) `double slow_down = ".1E-4";`

2.8 Schreiben Sie ein C++-Programm, das die Zeichen mit den ASCII-Codes 66 und 98 ausgibt.

2.9 Welche Fehler liegen in den folgenden `main`-Funktionen vor?

a)

```
#include <iostream>
using namespace std;
int main()
{
   cout << x << endl;
   return 0;
}
```

b)

```
#include <iostream>
using namespace std;
int main()
```

```
{
   int count;
   cout << count << endl;
   return 0;
}
```

c)

```
#include <iostream>
using namespace std;
int main()
{
   char c = 'A!';
   cout << c << endl;
   return 0;
}
```

2.10 Was gibt das folgende C++-Programm aus?

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

void test(void);
int x = 10;

int main()
{
   test();
   cout << "Der Wert von x in main() ist "
        << x << endl;
   return 0;
}
void test()
{
   cout << "Der Wert von x in test() ist "
        << x << endl;
   x = x + 10;
}
```

Lösungen zu den Verständnisfragen

2.1 **b)** und **c)**

2.2 0

2.3 **a)** und **c)**

2.4 **b)**

2.5 **b)**

2.6 Falsch

2.7 Richtig

2.8 **b)**

2.9 **b)**

2.10 Falsch

2.11 4

2.12 Richtig

2.13 Richtig

2.14 **a)**

2.15 **a)** und **c)**

2.16 global

2.17 0

2.18 **c)**

2.19 `const`

2.20 `char`

Lösungen zu den Aufgaben

2.1 **a)** `long`

b) `float`

c) `int` (oktale Konstante)

d) `char`

e) `unsigned long long`

f) `double`

g) `int` (hexadezimale Konstante)

h) `double`

i) `long` (hexadezimale Konstante)

2.2

```
// ----------------------------------------------------------
// ex02_02.cpp
// Gibt die kleinsten und größten Werte
// für die Datentypen char and wchar_t aus.
// ----------------------------------------------------------
#include <iostream>
#include <climits>
using namespace std;

int main()
{
   cout << "Wertebereich der Typen char und wchar_t "
        << endl << endl;
   cout << "Typ        Minimum      Maximum" << endl;
   cout << "-------------------------------" << endl;
   cout << "char        " <<  CHAR_MIN << "         "
                         <<  CHAR_MAX << endl;
   cout << "wchar_t     " << WCHAR_MIN << "            "
                         << WCHAR_MAX << endl;
   return 0;
}
```

2.3

```
// ----------------------------------------------------------
// ex02_03.cpp
// Speicherbedarf, kleinster positiver und größter
// Wert sowie die Genauigkeit des Datentyps double.
// ----------------------------------------------------------
#include <iostream>
#include <cfloat>
using namespace std;

int main()
{
  cout << "Speicherbedarf, Wertebereich und "
       << "Genauigkeit des Datentyps double\n " << endl;
  cout << "Speicherbedarf:           " << sizeof(double)
       << endl;
  cout << "Groesster Wert:           " << DBL_MAX << endl;
  cout << "Kleinster positiver Wert: " << DBL_MIN << endl;
  cout << "Genauigkeit:              " << DBL_DIG << endl;
  return 0;
}
```