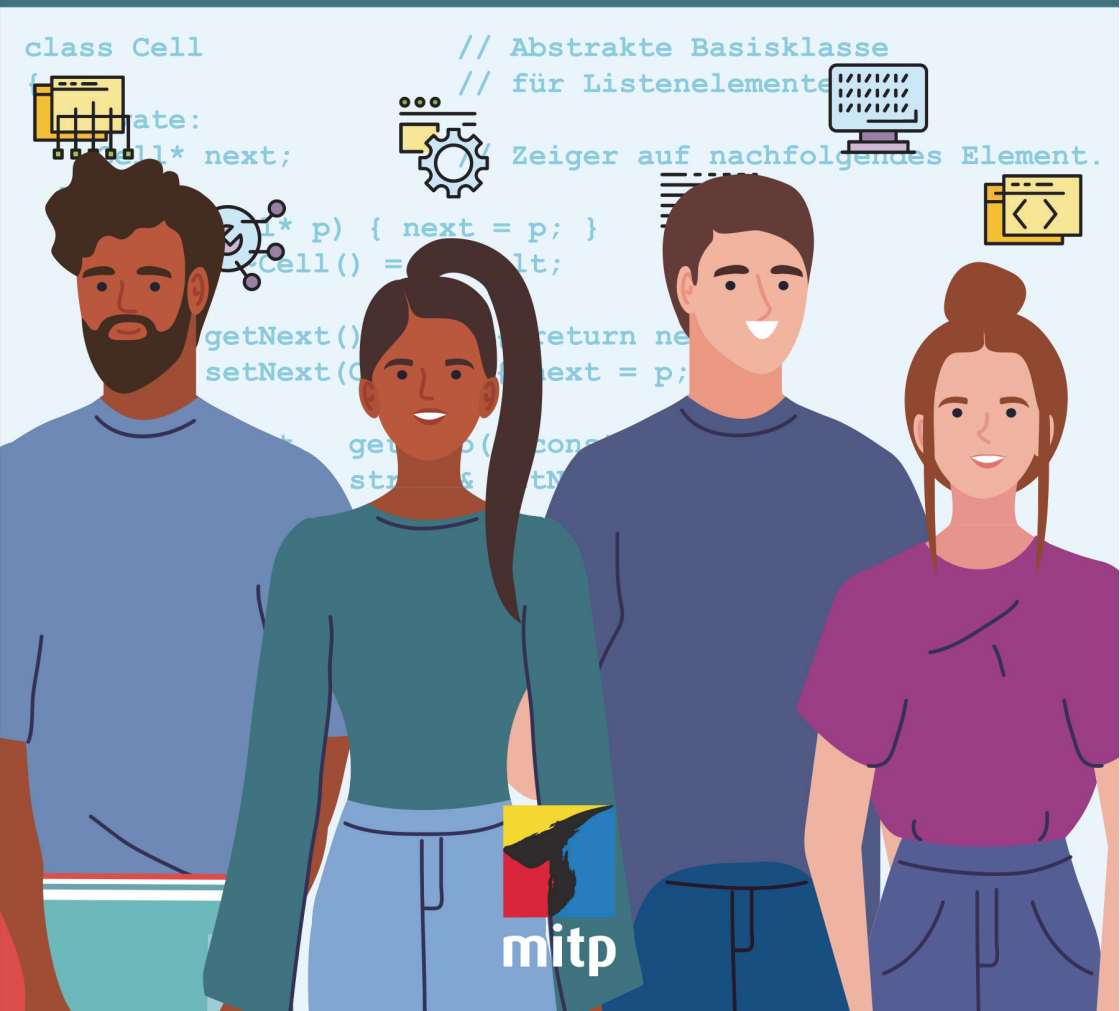
Ulla Kirch | Peter Prinz

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;
}
```

2.4 **a)** Ungültig, da in C++ `const` ein Schlüsselwort ist.

b) Gültig, da C++ Klein- und Großbuchstaben unterscheidet. CHAR ist deshalb kein Schlüsselwort.

c) Gültig

d) Ungültig, da der Name mit einer Zahl beginnt.

e) Gültig

f) Ungültig, da das Zeichen @ in Variablennamen nicht enthalten sein darf.

g) Ungültig, da ein Bindestrich in Variablennamen nicht enthalten sein darf.

h) Gültig

i) Ungültig, da das Zeichen ! in Variablennamen nicht enthalten sein darf.

2.5 **a)**

```
cout << " Ungueltiger Dateiname: " << endl
     << "C:\\temp\\Grocer\'s.cpp"  << endl;
```

b)

```
cout << "Ihre Eingabe: \a" << endl;
```

c)

```
cout << "\"Left to themselves,\"\n"
     << "\t\"things tend to go\n"
     << "\t\t\"from bad to worse.\" (Murphy)"
     << endl;
```

2.6

```
double side(1.0), circumference, area;
```
(oder: `double side = 1.0, circumference, area;`)

2.7 **a)** Die Typangabe vor dem `i` ist falsch. Korrekt wäre: `int n, i;`

b) Zwei Variablen desselben Typs werden durch ein Komma getrennt (oder: Ein Variablenname darf kein Blank enthalten).

c) `Short` ist kein Typname.

d) In der Definition der Variablen `result` fehlt ein abschließendes Semikolon.

e) Die Verwendung von eckigen Klammern [ ] bei der Initialisierung einer `char`-Variablen ist falsch. Korrekt wäre: `char c('a');`

f) Eine Variable vom Typ `double` kann nicht mit einer String-Konstanten initialisiert werden.

2.8

```
// ---------------------------------------------------------
// ex02_08.cpp
// Gibt die Zeichen mit den ASCII-Codes 66 und 98 aus.
// ---------------------------------------------------------
#include <iostream>
using namespace std;

int main()
{
   char c1 = 66, c2 = 98;
   cout << "Das Zeichen mit dem ASCII-Code 66 ist: "
        << c1 << endl;
   cout << "Das Zeichen mit dem ASCII-Code 98 ist: "
        << c2 << endl;
   return 0;
}
```

2.9 **a)** Die Variable `x` wurde nicht definiert.

b) Die Variable `count` wurde vor ihrer Verwendung nicht initialisiert.

c) Eine Variable vom Typ `char` kann nicht zwei Zeichen speichern.

2.10

```
Der Wert von x in test() ist 10
Der Wert von x in main() ist 20
```

Kapitel 3

Verwenden von Funktionen und Klassen

In diesem Kapitel werden vordefinierte Funktionen und Klassen der C++-Standardbibliothek eingesetzt. Dazu werden Sie

- Funktionen deklarieren
  Vor ihrer Verwendung muss jede Funktion deklariert werden. Dabei wird dem Compiler die Schnittstelle der Funktion bekannt gegeben, d.h. der Typ jedes Parameters und der Typ des Return-Wertes (also des Wertes, den die Funktion zurückgibt). Man nennt dies auch den Prototyp der Funktion. Eine Funktion, die keinen Wert zurückgibt, ist vom Typ `void`. Die Prototypen von Standardfunktionen sind bereits in Standard-Header-Dateien enthalten.
- Funktionen aufrufen
  Beim Aufruf einer Funktion wird für jeden Parameter ein entsprechendes Argument übergeben. Wenn im Prototyp der Funktion kein Parameter deklariert ist, erhält die Funktion kein Argument. Beim Funktionsaufruf selbst handelt es sich um einen Ausdruck, der den Typ und den Wert des Return-Wertes der Funktion aufweist. Der Compiler überprüft den Funktionsaufruf anhand des Prototyps und gibt bei einem falschen Aufruf eine Fehlermeldung aus.
- Header-Dateien verwenden
  Header-Dateien sind Textdateien, die typischerweise Prototypen von Funktionen und Definitionen von Klassen enthalten. Die Header-Dateien, die mit der Programmiersprache C standardisiert wurden, gehören auch zum C++-Standard. Ihre Namen beginnen mit `c` (z.B. `cmath` statt `math.h`). Wenn in der `#include`-Direktive der Name der Header-Datei in spitzen Klammern <...> angegeben ist, wird die Datei nur in den Verzeichnissen mit den Standard-Header-Dateien gesucht. Ist der Name in Hochkommas "..." angegeben, wird zusätzlich zuerst im aktuellen Verzeichnis gesucht.
- Standardklassen einsetzen
  Die C++-Standardbibliothek definiert zahlreiche Klassen, wie z.B. die Streamklassen zur Ein-/Ausgabe und die Klasse `string` zur Darstellung von Zeichenketten. Bei einer Klasse handelt es sich um einen Datentyp mit Datenelementen und Methoden (Funktionen, die zur Klasse gehören). Ein Objekt ist eine Variable vom Typ einer Klasse. Für ein Objekt können die `public`-Methoden der Klasse aufgerufen werden. Dabei wird der Name des Objekts vom Namen der Methode durch einen Punkt getrennt.

Verständnisfragen

3.1 In einem C++-Programm muss jeder Name, bei dem es sich nicht um ein Schlüsselwort handelt, vor seiner Verwendung _____ werden.

3.2 Ein Funktionsaufruf ist ein Ausdruck, dessen Typ bestimmt ist durch

a) die an die Funktion übergebenen Argumente.

b) die im Funktionskopf deklarierten Parameter.

c) den Return-Wert der Funktion.

3.3 Der Prototyp einer Funktion stellt dem Compiler Informationen über

a) den Return-Typ der Funktion bereit.

b) die Namen der Parameter bereit.

c) den Typ jedes Parameters bereit.

3.4 Bei dem Argument, das einer Funktion übergeben wird, darf es sich

a) nur um eine Konstante

b) nur um eine Variable

c) einen beliebigen Ausdruck

vom Typ des entsprechenden Parameters handeln.

3.5 Gemäß dem Prototyp

```
double calc(int n, float x);
```

hat der Ausdruck

```
calc(7, 12.9)
```

den Typ _____.

3.6 Beim Aufruf einer Standardfunktion benötigt der Compiler

a) keine weiteren Informationen.

b) den Prototyp der Funktion.

c) die Definition der Funktion.

3.7 Ein Compiler erkennt eine falsche Anzahl von Argumenten nicht.

☐ Richtig ☐ Falsch

3.8 Man kann eine Funktion schreiben, die Anweisungen enthält, aber keinen Return-Wert liefert.

☐ Richtig ☐ Falsch

3.9 Eine Folge von Zufallszahlen kann durch wiederholte Aufrufe der Standardfunktion _____ generiert werden.

3.10 In C++ ist eine Standard-Header-Datei

a) eine Objektdatei.

b) eine ausführbare Datei.

c) eine Textdatei.

3.11 Wenn der Name einer Header-Datei in doppelten Hochkommas angegeben ist, sucht der Compiler die Datei

a) nur in den Verzeichnissen des Compilers.

b) nur im aktuellen Verzeichnis.

c) zuerst im aktuellen Verzeichnis.

3.12 Zur Ausführung der Anweisung

```
cout << "Hi, friends!";
```

genügt es, die Header-Datei `iostream` in Ihrem Programm zu inkludieren.

☐ Richtig ☐ Falsch

3.13 Die in den C-Header-Dateien (Kennung `.h`) enthaltenen Bezeichner sind

a) lokal deklariert.

b) global deklariert.

c) im Namensbereich `std` deklariert.

3.14 In C++ liegt für jede C-Header-Datei eine entsprechende C++-Header-Datei vor, die die gleichen Bezeichner im Namensbereich `std` deklariert. Beispielsweise entspricht der C-Header-Datei `math.h` in C++ die Header-Datei _____.

3.15 Ein Objekt vom Typ einer Klasse wird auch als Instanz der Klasse bezeichnet.

☐ Richtig ☐ Falsch

3.16 Beim Erzeugen eines Objekts wird

a) Speicher für die Datenelemente reserviert.

b) jedes Datenelement mit einem passenden Wert initialisiert.

c) jede Methode der Klasse aufgerufen.

3.17 Um für ein Objekt eine Methode aufzurufen, wird der Name des Objekts durch einen _____ vom Namen der Methode getrennt angegeben.

3.18 Mit welcher globalen Funktion kann eine ganze Textzeile in ein Objekt vom Typ `string` eingelesen werden?

3.19 Die Länge eines Strings entspricht

a) der Anzahl Bytes, die das Objekt im Speicher belegt.

b) der Anzahl Zeichen im String ohne nachfolgende Blanks.

c) der Anzahl Zeichen im String.

3.20 Um zwei Strings (also zwei Objekte vom Typ `string`) aneinander zu reihen, kann folgender Operator verwendet werden:

a) -

b) +

c) *

Aufgaben

3.1 Geben Sie die Prototypen folgender Funktionen an:

a) Die Funktion `sum()` liefert die Summe von drei `double`-Werten, die als Argumente übergeben werden.

b) Die Funktion `cubes()` besitzt einen Parameter `n` vom Typ `int`. Sie summiert die ersten `n` positiven Zahlen »hoch drei« auf, berechnet also den Wert $1^3 + 2^3 + ... + n^3$, und liefert das Ergebnis als Return-Wert zurück.

c) Die Funktion `ggt()` bestimmt den größten gemeinsamen Teiler von zwei als Argument übergebenen ganzen Zahlen.

d) Die Funktion `wordCount()` erhält einen String als Argument und liefert die Anzahl der im String enthaltenen Worte zurück.

e) Die Funktion `isLeapYear()` erhält eine Jahreszahl als Argument und gibt `true` zurück, falls das Jahr ein Schaltjahr ist, andernfalls `false`.

f) Die Funktion `displayStatus()` gibt den Status des Programms auf dem Bildschirm aus. Die Funktion hat keinen Parameter und keinen Return-Wert.

3.2 Bestimmen Sie die Fehler in folgenden Prototypen:

a)

```
double calculate double x, double y;
```

b)

```
void myFunc(int n, m);
```

c)

```
int your-Func();
```

d)

```
Bool test(void);
```

3.3 Was gibt das folgende Programm auf dem Bildschirm aus?

```
// ------------------------------------------------
// ex03_03.cpp
// ------------------------------------------------
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
   double x = 9.0;
   cout << x << "   "
        << sqrt(x) << "   "
        << pow(x,2.0) << endl;
   return 0;
}
```

3.4 Schreiben Sie ein C++-Programm, das eine Gleitpunktzahl im Dialog einliest. Anschließend wird der Sinus und der Cosinus dieser Zahl ausgegeben. Verwenden Sie die in der Header-Datei `cmath` deklarierten Funktionen `sin()` und `cos()`.

Bemerkung: Für eine Variable `x` vom Typ `double` können Sie mit der Anweisung

```
cin  >> x;
```

eine Gleitpunktzahl im Dialog einlesen.

Beispielausgabe:

```
Geben Sie eine Gleitpunktzahl ein: 2.5
Der Sinus von   2.5 ist: 0.598472
Der Cosinus von 2.5 ist: -0.801144
```

3.5 Welche der folgenden Funktionsaufrufe sind korrekt?

a)

```
int max( int, int, int);
int result = max( 7, 12);
```

b)

```
long pow10(int), result = pow10(2);
```

c)

```
void put( char c);
char c = put('A');
```

d)

```
double square( double), x = 2.1;
cout << square(x);
```

e)

```
int random(void);
random(1);
```

3.6 Die Standardfunktion

```
void srand( unsigned seed);
```

initialisiert den Zufallszahlengenerator mit dem Keim `seed`. Mit verschiedenen Keimen können unterschiedliche Folgen von Zufallszahlen erzeugt werden.

Die Standardfunktion

```
int rand( void );
```

liefert eine Zufallszahl zwischen 0 und der Konstanten `RANDMAX`, die mindestens den Wert 32767 hat.

Die Prototypen beider Funktionen befinden sich in der Header-Datei `cstdlib`.

Schreiben Sie ein C++-Programm, das eine Ganzzahl im Dialog einliest, um den Zufallszahlengenerator zu initialisieren. Anschließend werden zwei Zufallszahlen erzeugt und zusammen mit ihrer Differenz ausgegeben.

3.7 Die Standardfunktion

```
double ceil(double x);
```

liefert die kleinste Ganzzahl, die größer oder gleich `x` ist. Die Funktion ist in der Header-Datei `cmath` deklariert.

Schreiben Sie ein C++-Programm, das die Funktion `ceil()` einmal mit einer positiven und dann mit einer negativen Gleitpunktzahl aufruft. Das Argument und der Return-Wert werden jedes Mal ausgegeben.

Beispielausgabe:

```
ceil(1.42) = 2
ceil(-1.65) = -1
```

3.8 Welche der nachfolgenden Definitionen sind zulässig, welche nicht?

a)

```
string s('Da bin ich!');
```

b)

```
string stars_and_stripes("*** ---");
```

c)

```
string shorts = "O";
```

d)

```
string stars(80, *);
```

3.9 Ist in folgenden Quellcodes etwas falsch?

a)

```
#include <string.h>
using namespace std;
string s("Test");
```

b)

```
string s;
cout << "Geben Sie Ihren Vornamen ein: ";
getline(s, cin);
```

c)

```
string s("wunderbarer ");
cout << "Was fuer ein " << s + "Morgen!";
```

3.10 Was gibt folgendes Programm auf dem Bildschirm aus?

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

int main()
{
   string s1 = "Nichts ist ", s2 = "so ";

   cout << s1 + s2 + "einfach,";
   s2 = "wie es aussieht.";
   cout << s2 + " (Murphy's Gesetz)" << endl;
   return 0;
}
```

Lösungen zu den Verständnisfragen

3.1 deklariert

3.2 **c)**

3.3 **a)** und **c)**

3.4 **c)**

3.5 `double`

3.6 **b)**

3.7 Falsch

3.8 Richtig

3.9 `rand()`

3.10 **c)**

3.11 **c)**

3.12 Falsch

3.13 **b)**

3.14 `cmath`

3.15 Richtig

3.16 **a)** und **b)**

3.17 Punkt

3.18 `getline()`

3.19 **c)**

3.20 **b)**

Lösungen zu den Aufgaben

3.1 **a)**

```
double sum( double, double, double);
```

b)

```
int cubes( int n);      // oder: long cubes( int n);
```

c)

```
int ggt( int n, int m);
```

d)

```
int wordCount(string s);
```

e)

```
bool isLeapYear( int n);
```

f)

```
void displayStatus();   // oder: void displayStatus(void);
```

3.2 **a)** Die Parameter sind in runden Klammern einzuschließen.

b) Jeder Parameter und sein Typ muss separat deklariert werden.

c) Ein Funktionsname darf keinen Bindestrich enthalten.

d) C++ unterscheidet Groß- und Kleinbuchstaben. `Bool` ist deshalb kein Typname.

3.3 9 3 81

3.4

```
// ----------------------------------------------------
// ex03_04.cpp
// Liest eine Gleitpunktzahl im Dialog ein und
// gibt den Sinus und Cosinus dieser Zahl aus.
// ----------------------------------------------------
#include <iostream>
#include <cmath>
using namespace std;
```

```
int main()
{
   double x;

   cout << "Geben Sie eine Gleitpunktzahl ein: ";
   cin >> x;

   cout << "Der Sinus von   " << x << " ist: "
        << sin(x) << endl;
   cout << "Der Cosinus von " << x << " ist: "
        << cos(x) << endl;

   return 0;
}
```

3.5 **a)** Die Anzahl Parameter im Prototyp der Funktion stimmt nicht mit der Anzahl Argumente beim Aufruf der Funktion überein.

b) Richtig

c) Die Funktion `put()` hat keinen Return-Wert. Deshalb ist die Zuweisung unzulässig.

d) Korrekt

e) Unzulässig. Die Funktion `random()` erhält kein Argument.

3.6

```
// ------------------------------------------------------
// ex03_06.cpp
// Liest eine Ganzzahl im Dialog ein, um den Zufalls-
// zahlengenerator zu initialisieren. Zwei Zufallszahlen
// und ihre Differenz werden ausgegeben.
// ------------------------------------------------------
#include <iostream>
#include <cstdlib>    // Prototypen von srand() und rand()
using namespace std;

int main()
{
  unsigned int seed;

  cout << "Geben Sie eine ganze Zahl ein: ";
  cin >> seed;
  srand(seed);    // Zufallszahlengenerator initialisieren

  int rn1 = rand(),
      rn2 = rand();
```

```
  cout << "\nZwei Zufallszahlen: " << rn1 << "    " << rn2
       << "\nund ihre Differenz: " << rn1 - rn2
       << endl;
  return 0;
}
```

3.7

```
// ----------------------------------------------------
// ex03_07.cpp
// Ruft die Funktion ceil() mit einer positiven und
// einer negativen Gleitpunktzahl auf.
// ----------------------------------------------------
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
   double x = 1.42;
   cout << "ceil(" << x << ") = " << ceil(x) << endl;

   x = -1.65;
   cout << "ceil(" << x << ") = " << ceil(x) << endl;

   return 0;
}
```

3.8 **a)** Unzulässig. Eine Stringkonstante ist in doppelten Hochkommas anzugeben.

b) Korrekt

c) Korrekt

d) Unzulässig. Das Zeichen * muss in einfachen Hochkommas angegeben werden.

3.9 **a)** Der Name der zu inkludierenden Header-Datei ist `string` und nicht `string.h`.

b) Die Reihenfolge, in der die Argumente an `getline()` übergeben werden, ist falsch. Ein zulässiger Aufruf ist: `getline(cin, s);`.

c) Korrekt (vorausgesetzt, die Header-Datei `string` wurde inkludiert).

3.10

```
Nichts ist so einfach, wie es aussieht. (Murphy's Gesetz)
```

Ein- und Ausgaben mit Streams

Bei den Streams `cin` und `cout` handelt es sich um Objekte der Klassen `istream` und `ostream`. Sie werden zum Lesen von der Standardeingabe bzw. Schreiben auf die Standardausgabe verwendet und beim Programmstart automatisch angelegt.

In den Übungen dieses Kapitels werden Sie

- Zahlen formatiert ausgeben
  Bei der Ausgabe mit dem Operator << wird die Formatierung durch Flags der Basisklasse `ios` gesteuert. Diese können auf einfache Weise mithilfe von Manipulatoren verändert werden. Bei Ganzzahlen kann das Zahlensystem festgelegt werden (`dec`, `oct` und `hex`). Positive Dezimalzahlen lassen sich mit oder ohne Vorzeichen anzeigen (`showpos, noshowpos`) und Hexadezimalzahlen können mit Klein- oder Großbuchstaben (`uppercase, nouppercase`) dargestellt werden. Gleitpunktzahlen werden standardmäßig mit einer Genauigkeit von sechs Ziffern und ohne abschließende Nullen nach dem Dezimalpunkt angezeigt. Eine Gleitpunktzahl ist auch als Festpunktzahl (`fixed`) oder in exponentieller Notation (`scientific`) mit einer anderen Genauigkeit (`setprecision()`) darstellbar.
- Zahlen formatiert einlesen
  Beim Einlesen mit dem Operator >> kann bei Ganzzahlen ebenfalls die Basis des Zahlensystems festgelegt werden (`dec`, `oct`, `hex`). Gleitpunktzahlen werden stets dezimal als Festpunktzahl oder in exponentieller Form eingelesen.
- Feldbreiten verwenden
  Mit dem Operator << werden mindestens so viele Zeichen ausgegeben, wie durch die aktuelle Feldbreite festlegt ist (`setw()`). Statt des Blanks kann ein anderes Füllzeichen ausgewählt (`setfill()`) und die Ausgabe im Feld rechts- oder linksbündig ausgerichtet (`left`, `right`) werden. Beim Lesen mit dem Operator >> werden höchsten so viele Zeichen gelesen, wie durch eine aktuell gesetzte Feldbreite > 0 vorgegeben ist. Dabei werden führende Zwischenraumzeichen nicht eingerechnet.
- Zeichen und Textzeilen unformatiert einlesen und ausgeben
  Die unformatierte Ein-/Ausgabe verwendet keine Formatierungsflags und keine Felder. Einzelne Zeichen, auch Zwischenraumzeichen, werden mit den Methoden `get()` und `put()` gelesen bzw. geschrieben. Die globale Funktion `getline(cin, s)` liest eine ganze Textzeile in den String `s`. Dabei wird das abschließende Newline-Zeichen gelesen, aber nicht im String gespeichert.

Verständnisfragen

4.1 Der Operator << ist definiert für die folgende Klasse:

a) `ios`

b) `istream`

c) `ostream`

4.2 Ein Objekt der Klasse `ostream` für die ungepufferte Fehlerausgabe ist der Standardstream _____.

4.3 Manipulatoren können verwendet werden, um

a) Streams zu erzeugen.

b) Fehlermeldungen auszugeben.

c) Formatierungen für nachfolgende Ein- und Ausgaben festzulegen.

4.4 Die in der Klasse `ios` definierten Formatierungsflags bestimmen, wie Zeichen mit den Operatoren >> und << eingelesen oder ausgegeben werden.

☐ Richtig ☐ Falsch

4.5 Ganze Zahlen werden standardmäßig ausgegeben als

a) Dezimalzahlen.

b) Oktalzahlen.

c) Hexadezimalzahlen.

4.6 Um positive Zahlen mit dem Vorzeichen + auszugeben, kann der Manipulator _____ verwendet werden.

4.7 Bei der Ausgabe von Ganzzahlen im oktalen oder hexadezimalen Format werden diese stets ohne Vorzeichen interpretiert.

☐ Richtig ☐ Falsch

4.8 Zum Aufruf eines Standardmanipulators mit einem oder mehreren Argumenten muss die Header-Datei _____ inkludiert sein.

4.9 Die Anweisung

```
cout << 70.0;
```

gibt standardmäßig _____ auf dem Bildschirm aus.

4.10 Zur Ausgabe einer Gleitpunktzahl in Festpunktdarstellung können Sie den Manipulator _____ verwenden.

4.11 Bei der Ausgabe in Felder berücksichtigt der Operator `<<`

a) eine vorgegebene Feldbreite.

b) die Ausrichtung im Feld.

c) ein vorgegebenes Füllzeichen.

4.12. Wenn die auszugebende Zeichenfolge länger als die vorgegebene Feldbreite ist, wird die Ausgabe abgeschnitten.

☐ Richtig ☐ Falsch

4.13 Ist die Feldbreite größer als die auszugebende Zeichenfolge, werden restliche Stellen im Feld standardmäßig mit folgendem Zeichen aufgefüllt:

a) Blank.

b) Stern.

c) Punkt.

4.14 Beim Einlesen mit dem Operator `>>` werden führende Zwischenraumzeichen ignoriert.

☐ Richtig ☐ Falsch

4.15 Um die Fehlerflags eines Streams zu löschen, kann die Methode _____ aufgerufen werden.

4.16 Damit der Operator >> eine eingegebene Zeichenfolge als Hexadezimalzahl interpretiert, kann der Manipulator _____ verwendet werden.

4.17 Gegeben seien folgende Anweisungen:

```
float x;   cin >> x;
```

Angenommen, das erste eingelesene Zeichen ist der Buchstabe A. In diesem Fall

a) wird ein Pseudo-Wert in die Variable `x` geschrieben.

b) kein Wert in die Variable `x` geschrieben.

c) ein internes Fehlerflag gesetzt.

4.18 Bei der unformatierten Ein- und Ausgabe werden

a) keine Felder verwendet.

b) intern gesetzte Formatierungsflags ignoriert.

c) Zwischenraumzeichen nicht überlesen.

4.19 Wird die Methode `get()` ohne Argument aufgerufen, liefert sie den Code des eingelesenen Zeichens vom Typ

a) `char`.

b) `int`.

c) `unsigned int`.

4.20 Wie lautet die Anweisung, um mit der globalen Funktion `getline()` von der Standardeingabe einen Text in ein Objekt `str` vom Typ `string` einzulesen, bis das Begrenzungszeichen `!` auftritt?

Aufgaben

4.1 Schreiben Sie ein C++-Programm, das die Zahl 255 ausgibt, und zwar

- mit positivem Vorzeichen,
- als Hexadezimalzahl mit Großbuchstaben,
- als Hexadezimalzahl mit Kleinbuchstaben,
- als Dezimalzahl ohne Vorzeichen.

Ausgabe: `+255  FF  ff  255`

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

```
// ------------------------------------------
// ex04_02.cpp
// Was gibt das folgende C++ Programm aus?
// ------------------------------------------
#include <iostream>
using namespace std;
int main()
{
   float x = 1.23f;
   cout << showpoint   << x << endl;
   cout << noshowpoint << x << endl;
   cout << fixed       << x << endl;
   return 0;
}
```

4.3 Schreiben Sie ein C++-Programm, das die Zahl 9.876 wie folgt anzeigt:

```
9.88
9.87600
10
```

Verwenden Sie die Methode `precision()` oder den Manipulator `setprecision()`.

4.4 Schreiben Sie ein C++-Programm, das eine Gleitpunktzahl im Dialog einliest und die Zahl in Gleitpunktdarstellung und exponentieller Darstellung mit zwei Ziffern hinter dem Dezimalpunkt ausgibt.

Beispielausgabe:

```
Geben Sie eine Gleitpunktzahl ein: 1234.5678
1234.57
1.23e+003
```

4.5 Schreiben Sie ein C++-Programm, das

- die größte darstellbare Zahl vom Typ `unsigned int` und
- die Zahl `-1` in dezimaler, oktaler und hexadezimaler Darstellung ausgibt.

Platzieren Sie jede Ausgabe linksbündig in ein Feld der Breite 15. Verwenden Sie die in der Header-Datei `climits` definierte Konstante `UINT_MAX`, die die größte darstellbare Zahl vom Typ `unsigned int` darstellt.

Beispielausgabe:

```
Dezimal         Oktal           Hexadezimal
4294967295      37777777777     FFFFFFFF
-1              37777777777     FFFFFFFF
```

4.6 Was gibt das folgende C++-Programm auf dem Bildschirm aus?

```
#include <iostream>
using namespace std;
int main()
{
   cout.fill('*');

   cout.width(4);  cout << 9    << endl;
   cout.width(4);  cout << 99   << endl;
   cout.width(4);  cout << "+++++" << endl
                        << 108  << endl;
   return 0;
}
```

4.7 Schreiben Sie ein C++-Programm, das im Dialog

- ein Zeichen
- ein einzelnes Wort

- eine Oktalzahl
- eine Hexadezimalzahl

einliest und auf dem Bildschirm anzeigt.

Beispielausgabe:

```
Geben Sie ein Zeichen ein:          $
Geben Sie ein Wort ein:             Hi!
Geben Sie eine Oktalzahl ein:       4567
Geben Sie eine Hexadezimalzahl ein: 9Ab

Ihre Eingabe:
Das Zeichen:         $
Das Wort:            Hi!
Die Oktalzahl:       4567
Die Hexadezimalzahl: 9ab
```

4.8 Was gibt das Programm aus der Aufgabe 4.7 aus, falls Sie versuchen, folgendes einzugeben?

`+` (als Zeichen),
`Warum?` (als Wort),
`787` (als Oktalzahl)
`EF` (als Hexadezimalzahl)

4.9 Lokalisieren und korrigieren Sie die Fehler in den folgenden Anweisungen:

a)

```
char c;
get( cin, c);
```

b)

```
string s;
cin.getline(s);
```

c)

```
cout << put('A');
```

d)

```
string question;
getline( question, '?', cin);
```

4.10 Schreiben Sie die erforderlichen Anweisungen, um Folgendes einzulesen:

a) das nächste Zeichen (inklusive Zwischenraumzeichen) in eine Variable vom Typ `char`

b) eine Textzeile in ein Objekt vom Typ `string`

c) eine Zeichenfolge bis zum ersten Zeichen `'*'` in ein Objekt vom Typ `string`

Lösungen zu den Verständnisfragen

4.1 **c)**

4.2 `cerr`

4.3 **c)**

4.4 Richtig

4.5 **a)**

4.6 `showpos`

4.7 Richtig

4.8 `iomanip`

4.9 `70`

4.10 `fixed`

4.11 **a)**, **b)** und **c)**

4.12 Falsch

4.13 **a)**

4.14 Richtig

4.15 `clear()`

4.16 `hex`

4.17 **b)** und **c)**

4.18 **a)**, **b)** und **c)**

4.19 **b)**

4.20
```
getline(cin, str, '!');
```

Lösungen zu den Aufgaben

4.1

```
// ------------------------------------------------------
// ex04_01.cpp
// Das Programm gibt die Zahl 255 aus, und zwar
//     mit positivem Vorzeichen,
//     als Hexadezimalzahl mit Großbuchstaben,
//     als Hexadezimalzahl mit Kleinbuchstaben,
//     als Dezimalzahl ohne Vorzeichen.
// ------------------------------------------------------
#include <iostream>
using namespace std;
int main()
{
   cout << showpos           << 255 << "    "
        << uppercase  << hex << 255 << "    "
        << nouppercase       << 255 << "    "
        << noshowpos  << dec << 255 << endl;
   return 0;
}
```

4.2

```
1.23000
1.23
1.230000
```

4.3

```
// ------------------------------------------------
// ex04_03.cpp
// Das Programm zeigt die Zahl 9.876 wie folgt an:
// 9.88
// 9.87600
// 10
// ------------------------------------------------
#include <iostream>
#include <iomanip>  // Falls setprecision() verwendet wird.
using namespace std;
int main()
{
   double x = 9.876;

   cout.precision(3);
   cout << x << endl;
   // oder:  cout << setprecision(3) << x << endl;
```

```
   cout.precision(5);
   cout << showpoint << x << endl;
   // oder:  cout << setprecision(5) << showpoint << x << endl;

   cout.precision(0);
   cout << fixed << noshowpoint << x << endl;
   // oder:  cout << setprecision(0) << fixed << noshowpoint
   //             << x << endl;
   return 0;
}
```

4.4

```
// ----------------------------------------------------------
// ex04_04.cpp
// Das Programm liest eine Gleitpunktzahl im Dialog ein
// und gibt die Zahl in Gleitpunktdarstellung und in
// exponentieller Notation mit zwei Ziffern hinter dem
// Dezimalpunkt aus.
// ----------------------------------------------------------
#include <iostream>
using namespace std;
int main()
{
   double x;
   cout << " Geben Sie eine Gleitpunktzahl ein: ";
   cin >> x;
   cout.precision(2);
   cout << fixed << x << endl;
   cout << scientific << x << endl;
   return 0;
}
```

4.5

```
// ----------------------------------------------------------
// ex04_05.cpp
// Das Programm gibt
//   die größte darstellbare Zahl vom Typ unsigned int und
//   die Zahl -1 in dezimaler, oktaler und hexadezimaler
//   Darstellung linksbündig in ein Feld der Breite 15 aus.
// ----------------------------------------------------------
#include <iostream>
#include <climits>
#include <iomanip>       // Falls setw() verwendet wird.
using namespace std;
```

```
int main()
{
    cout << left;                      // Ausgabe linksbündig

    // Mit dem Manipulator setw():
    cout << setw(15) << "Dezimal"
         << setw(15) << "Oktal  "
         << setw(15) << "Hexadezimal" << endl;

    cout << uppercase                  // Für Hex-Ziffern
         << setw(15) << UINT_MAX
         << setw(15) << oct  << UINT_MAX
         << setw(15) << hex  << UINT_MAX << endl;

    cout << setw(15) << dec  << -1
         << setw(15) << oct  << -1
         << setw(15) << hex  << -1 << endl;
/*
    // Oder mit der Methode width():
    cout.width(15);  cout << "Dezimal";
    cout.width(15);  cout << "Oktal  ";
    cout.width(15);  cout << "Hexadezimal" << endl;
    // etc.
*/
    return 0;
}
```

4.6

```
***9
**99
+++++
108
```

4.7

```
// ------------------------------------------------------
// ex04_07.cpp
// Das Programm liest im Dialog
//    ein Zeichen
//    ein Wort
//    eine Oktalzahl
//    eine Hexadezimalzahl
// ein und gibt sie auf dem Bildschirm aus.
// ------------------------------------------------------
#include <iostream>
#include <string>
```

```
using namespace std;

int main()
{
    char c;
    int n1, n2;
    string s;

    cout << "Geben Sie ein Zeichen ein:          ";
    cin >> c;
    cout << "Geben Sie ein Wort ein:             ";
    cin >> s;
    cout << "Geben Sie eine Oktalzahl ein:       ";
    cin >> oct >> n1;
    cout << "Geben Sie eine Hexadezimalzahl ein: ";
    cin >> hex >> n2;

    cout <<  endl << "Ihre Eingabe: " << endl;
    cout << "Das Zeichen:           " << c   << endl;
    cout << "Das Wort:              " << s   << endl;
    cout << "Die Oktalzahl:         " << oct << n1 << endl;
    cout << "Die Hexadezimalzahl:   " << hex << n2 << endl;
   return 0;
}
```

4.8 Ausgabe:

```
Geben Sie ein Zeichen ein:          +
Geben Sie ein Wort ein:             Warum?
Geben Sie eine Oktalzahl ein:       787
Geben Sie eine Hexadezimalzahl ein:
Ihre Eingabe:
Das Zeichen:         +
Das Wort:            Warum?
Die Oktalzahl:       7
Die Hexadezimalzahl: 87
```

4.9 **a)** Bei `get()` handelt es sich nicht um eine globale Funktion, sondern um eine Methode der Klasse `istream`. Ein korrekter Aufruf lautet wie folgt:

```
cin.get(c);
```

b) Die globale Funktion `getline()` liest eine Textzeile in ein Objekt vom Typ `string`. Ein korrekter Aufruf lautet wie folgt:

```
getline( cin, s);
```

c) Bei `put()` handelt es sich um eine Methode der Klasse `ostream`. Ein gültiger Aufruf lautet wie folgt:

```
cout.put('A');
```

d) Die Reihenfolge der Argumente ist nicht korrekt. Ein gültiger Aufruf lautet wie folgt:

```
getline( cin, question, '?');
```

4.10 a)

```
char c;
cin.get(c);
```

b)

```
string s;
getline(cin, s);
```

c)

```
string s;
getline(cin, s, '*');
```

Operatoren für elementare Datentypen

Ein Operator verknüpft Operanden (z.B. Konstanten und Variablen) zu einem Ausdruck. Dieser hat einen Typ und einen Wert, nämlich das Ergebnis der Operation, und kann daher wieder als Operand eines Operators eingesetzt werden.

Wenn ein Ausdruck mehrere Operatoren enthält, bestimmt der Vorrang (die Priorität) die Zuordnung der Operanden zu den Operatoren. Haben Operatoren den gleichen Vorrang, so wird gewöhnlich »von links« zusammengefasst, bei einigen Operatoren, z.B. der Zuweisung, erfolgt die Zusammenfassung »von rechts« (vgl. Vorrangtabelle). Unäre Operatoren, also Operatoren mit einem Operanden, haben generell einen höheren Vorrang als binäre Operatoren. Die Zuordnung von Operanden zu Operatoren kann durch das Setzen von Klammern selbst festgelegt werden.

In diesen Übungen sind die folgenden vier Gruppen von Operatoren relevant. Sie sind gemäß ihrem Vorrang in absteigender Reihenfolge aufgelistet:

- Arithmetische Operatoren
  Für Berechnungen gibt es die unären Operatoren +, – (positives, negatives Vorzeichen), ++, -- (um 1 inkrementieren, dekrementieren) und die binären Operatoren + (Summe), - (Differenz), * (Multiplikation), / (Division), % (Modulodivision). Es gelten die »üblichen Rechenregeln«, d.h. die Operatoren *, / und % haben einen höheren Vorrang als die binären Operatoren + und –.
- Vergleichsoperatoren
  Die klassischen Vergleichsoperatoren sind < (kleiner), > (größer), <= (kleiner oder gleich), >= (größer oder gleich), == (gleich) und != (ungleich). Sie liefern den Wert `true` oder `false`. Dagegen liefert der mit C++20 neu eingeführte *Drei-Wege-Vergleichsoperator* <=> ein Objekt, das mit `0` vergleichbar ist. Die Flexibilität dieses Operators kommt vor allem bei der Operatorüberladung zum tragen.
- Logische Operatoren
  Die logischen Operatoren sind && (und), || (oder) und ! (nicht). Ein logischer Ausdruck, z.B `(i>0 && i<10)`, liefert `true` oder `false`.
- Zuweisungsoperatoren
  Die *einfache Zuweisung* ordnet einer Variablen mit dem Operator = einen Wert zu (z.B. `y = 2*x`). Der Wert des Ausdrucks ist der zugewiesene Wert. Deshalb sind auch *Mehrfachzuweisungen* möglich (z.B. `z = y = 3.4`). Mit jedem binären arithmetischen Operator kann ein *zusammengesetzter Zuweisungsoperator* gebildet werden (z.B. ist `i*=3` äquivalent zu `i = i*3`).

Verständnisfragen

5.1 Der Wert des Ausdrucks `9/4` ist ___ .

5.2 Die Modulodivision kann nur für ganzzahlige Operanden ausgeführt werden.

☐ Richtig ☐ Falsch

5.3 Welcher der folgenden Ausdrücke hat den Wert 1 ?

a) `-3 + 4 * 5 - 6`

b) `-1 + 4 % 5 - 2`

c) `-2 + 4 % 5 + 2`

5.4 In Ausdrücken kann die Reihenfolge, in der Operanden und Operatoren zusammengefasst werden, durch _____ geändert werden.

5.5 Der Dekrementoperator `--` kann auch auf Konstanten angewendet werden.

☐ Richtig ☐ Falsch

5.6 Die Inkrement- und Dekrementoperatoren können nur für ganzzahlige Operanden eingesetzt werden.

☐ Richtig ☐ Falsch

5.7 Angenommen, die Variablen `x` und `y` haben einen arithmetischen Typ, dann werden die Operanden im Ausdruck

```
++x + 2 * y--
```

bei der Auswertung wie folgt zusammengefasst:

a) `(++x) + ((2*y)--)`

b) `((++x) + 2) * (y--)`

c) `(++x) + (2 * (y--))`

5.8 Der Ausdruck

```
4 % 3 * 2
```

hat den Wert _____.

5.9 Eine Zuweisung hat einen Wert und einen Typ.

☐ Richtig ☐ Falsch

5.10 Mit der Definition

```
int x = 2;
```

weist die Variable `x` nach der zusammengesetzten Zuweisung

```
x *= 3 + 4;
```

den Wert ____ auf.

5.11 Im Anschluss an die Definition

```
int i = 2, j;
```

bewirkt die Anweisung

```
i *= j = 4;
```

a) eine Fehlermeldung des Compilers.

b) die Zuweisung von `4` an die Variablen `i` und `j`.

c) die Zuweisung von `8` an `i` und `4` an `j`.

5.12 Gegeben sind die `double`-Variablen `x` und `y`. Dann liefert der Ausdruck `(x <=> y) < 0` genau dann den Wert `true`, wenn `x` kleiner als `y` ist.

☐ Richtig ☐ Falsch

5.13 Der Vorrang von Vergleichsoperatoren ist _____ als der Vorrang von Zuweisungsoperatoren.

5.14 Vergleichsoperatoren haben einen _____ Vorrang als arithmetische Operatoren.

5.15 Nach den Definitionen

```
bool flag;   int x = 3, y = 2;
```

gibt die Anweisung

```
cout << (flag = x == y);
```

folgendes auf dem Bildschirm aus:

a) `0` (oder `false`)

b) `1` (oder `true`)

c) 2

5.16 Die Booleschen Operatoren in C++ sind folgende:

a) `AND`, `OR` und `NOT`

b) `&`, `|` und `!=`

c) `&&`, `||` and `!`

5.17 Angenommen, die `int`-Variable `x` speichert die Zahl 9. Dann hat der logische Ausdruck

```
x-- == 9 && x == 8
```

den Wert

a) `true`.

b) `false`.

5.18 Ist der Operand eines logischen Operators eine Zahl, so wird diese als `true` interpretiert, wenn ihr Wert _____ ist.

5.19 Der Operator `&&` hat eine höhere Priorität als der Operator `||`.

☐ Richtig ☐ Falsch

5.20 Angenommen, die `int`-Variable `count` speichert die Zahl 10. Dann hat der logische Ausdruck

```
!(count == 0 || count >= 10)
```

den Wert _____ .

Aufgaben

5.1 Schreiben Sie ein C++-Programm, das zwei ganze Zahlen im Dialog einliest und ihr Produkt, den Quotienten und den Divisionsrest ausgibt.

5.2 Erstellen Sie ein C++-Programm, das drei Gleitpunktzahlen im Dialog einliest und deren Summe und Durchschnitt ausgibt.

Beispielausgabe:

```
Geben Sie drei Gleitpunktzahlen ein: 2.7 8.9 5.3
Die Summe:        16.9
Der Durchschnitt: 5.63333
```

5.3 Schreiben Sie ein C++-Programm, das

a) eine Fahrenheit-Temperatur im Dialog einliest und in Celsius umrechnet,

b) eine Celsius-Temperatur einliest und in Fahrenheit umrechnet

und die Ergebnisse mit zwei Stellen hinter dem Dezimalpunkt ausgibt.

Hinweis: Verwenden Sie die Formel: 5*(Fahrenheit - 32) = 9 * Celsius

Beispielausgabe:

```
Geben Sie eine Temperatur in Fahrenheit ein: 100
100.00 Fahrenheit entsprechen 37.78 Grad Celsius.

Geben Sie eine Temperatur in Celsius ein: 28
28.00 Celsius entsprechen 82.40 Grad Fahrenheit.
```

5.4 Was gibt folgendes C++-Programm auf dem Bildschirm aus?

```
#include <iostream>
using namespace std;
int main()
{
   int n(15);

   n += 25;   cout << n << endl;
   n %= 9;    cout << n << endl;
   n = 5;     cout << n++ << endl;
   n *= n;    cout << n << endl;
   return 0;
}
```

5.5 Erstellen Sie ein C++-Programm, das eine Anzahl von Sekunden im Dialog einliest und die entsprechende Anzahl Stunden, Minuten und Sekunden ausgibt.

Beispielausgabe:

```
Geben Sie eine Anzahl Sekunden ein: 7885
Stunden:  2
Minuten:  11
Sekunden: 25
```

5.6 Schreiben Sie ein C++-Programm zur Berechnung eines Kredits, den Ihnen eine Bank gewährt. Die monatliche Ratenzahlung, die Anzahl von Monaten, in denen die Rate gezahlt wird, und der monatliche Zinssatz werden dabei vorgegeben.

Hinweis: Verwenden Sie die folgende Formel:

$$Kredit = Rate * (1 - q^n) / (q^n - q^{n+1})$$