Universität Regensburg
Fakultät Physik IT-Ausbildung

Aufgaben zum C/C++ Kurs

Kapitel 1 — Erste Schritte

Aufgabe 1

Schreiben Sie ein Programm, das den Namen des Benutzers einliest und ihn dann mit seinem Namen begrüßt. Verwenden Sie sowohl `printf` bzw. `scanf` als auch `cout` und `cin`. Erweitern Sie dazu das in Kapitel 1 abgedruckte ‚Hello world'-Programm.

Aufgabe 2

Lesen Sie mit `scanf` und `cin` ein Wort, ein Zeichen und eine Fließkommazahl ein und geben Sie die Inhalte mit `printf` und `cout` wieder aus.

Aufgabe 3

Lesen Sie zwei ganze Zahlen ein und geben Sie deren Summe, Produkt, Differenz und Quotienten aus. Achten Sie beim Teilen auf die Richtigkeit des Ergebnisses!

Aufgabe 4

Geben Sie mithilfe des Modulo-Operators % die letzte Ziffer einer vom Benutzer eingegebenen Ganzzahl (`int`) aus.

Aufgabe 5

Ein Zeichen soll eingelesen und dann wieder ausgegeben werden. Dabei soll die Ausgabe sowohl als Zeichen als auch als Dezimal- bzw. Hexadezimalwert erfolgen (z.B.: `z` → `z 122 0x7A`). Verwenden Sie sowohl `scanf` als auch `cin` für die Eingabe, bei der Ausgabe probieren Sie `cout` und `printf` aus.

↪

Kapitel 2/3 — Variablen und Kontrollstrukturen

Aufgabe 6

Erstellen Sie ein Programm, das mithilfe einer `for`-Schleife alle durch 7 teilbaren Zahlen zwischen zwei zuvor eingegebenen Grenzen ausgibt.

Aufgabe 7

Schreiben Sie ein Programm, das zu einem gegebenen Anfangskapital bei einem vorgegebenem jährlichen Zinssatz berechnet, wie viele Jahre benötigt werden, damit das Kapital eine bestimmte Zielsumme überschreitet.

Aufgabe 8

Erweitern Sie das in Kapitel 3.3 abgedruckte Programm so, dass ein Schachbrettmuster auf dem Bildschirm erscheint.

Aufgabe 9

Schreiben Sie ein Programm, das das kleine Einmaleins berechnet und in Tabellenform auf dem Bildschirm ausgibt.

Aufgabe 10

Geben Sie alle Primzahlen zwischen 2 und einer vom Benutzer einzugebenen Obergrenze aus. Bestimmen Sie die Primzahlen mittels Ganzzahldivision.

Aufgabe 11

Geben Sie einen ‚Christbaum' durch Ausdruck entsprechend vieler Leerzeichen und Sterne auf dem Bildschirm aus.

Aufgabe 12

In unserem Kalender sind zum Ausgleich der astronomischen und kalendarischen Jahreslänge in regelmäßigen Abständen Schaltjahre eingebaut. Zur exakten Festlegung der Schaltjahre dienen die folgenden Regeln:

- Ist die Jahreszahl durch 4 teilbar, so ist das Jahr ein Schaltjahr. Diese Regel hat allerdings eine Ausnahme:
- Ist die Jahreszahl durch 100 teilbar, so ist das Jahr kein Schaltjahr. Diese Ausnahme hat wiederum eine Ausnahme:
- Ist die Jahreszahl durch 400 teilbar, so ist das Jahr doch ein Schaltjahr.

Erstellen Sie ein Programm, das berechnet, ob eine vom Benutzer eingegebene Jahreszahl ein Schaltjahr bezeichnet oder nicht.

↪

Aufgabe 13

Programmieren Sie das Spiel ‚Zahlenraten': Der Computer generiert eine Zufallszahl zwischen 1 und 100, die der Spieler erraten muß. Es wird bei jedem Durchgang mitgeteilt, ob die eingegebene Zahl zu groß oder zu klein war.

Tipp: Zufallszahlen werden wie folgt generiert: Ein *einmaliger* Aufruf `srand(time(NULL))` initialisiert den Zufallszahlengenerator. Anschließend liefert *jeder* Aufruf von `rand()%100` eine Zufallszahl zwischen 0 und 99. Die Funktionen sind in `stdlib.h` bzw. `time.h` deklariert.

Aufgabe 14 (fakultativ)

Lösen Sie die quadratische Gleichung $ax^2 + bx + c = 0$ für vom Benutzer eingegebene Parameter a, b und c. Falls Ihnen komplexe Zahlen geläufig sind, sollen auch komplexe Lösungen ausgegeben werden.

Tipp: Die Lösungsformel für die quadratische Gleichung lautet: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Die Funktion `float sqrt(float)` zur Berechnung der Wurzel einer positiven Zahl ist in `math.h` deklariert. Zum Testen des Programms können Sie folgende Parametersätze verwenden:
4, -36, 81 → 4.5; 2, 2, -12 → 2, 3; 1, -2, 2 → $1 + -i$.

Aufgabe 15

Programmieren Sie mithilfe der `switch-case` Anweisung einen Mini-Taschenrechner, der zwei Zahlen und einen Operator (`+, -, *, /`) einliest und das Ergebnis ausgibt. Eine eventuelle Division durch 0 soll abgefangen werden.

Aufgabe 16

Primzahlberechnung nach dem Sieb-Verfahren: Schreiben Sie alle Zahlen (z.B. von 2 bis 100) in ein Array. Beginnend mit der kleinsten Zahl wird die Zahl als Primzahl auf dem Bildschirm ausgegeben und gleichzeitig alle Vielfachen dieser Zahl im Array auf 0 gesetzt d.h. aus der Liste gestrichen. Anschliessend wird die nächste Zahl $\neq 0$ im Array bearbeitet.

Aufgabe 17

Lesen Sie eine Integerzahl ein und geben Sie deren Bitmuster aus (z.B.: 23 → `00010111`).

Aufgabe 18

Bestimmen Sie die Primfaktorzerlegung einer eingegebenen Zahl (z.B.: `200 = 2*2*2*5*5`).

Aufgabe 19

Sortieren Sie ein mit Zufallszahlen belegtes Array beliebiger Größe mit dem Bubble-Sort-Verfahren: jeweils zwei benachbarte Feldelemente werden vertauscht, wenn sie in der falschen Reihenfolge sind.

↪

Aufgabe 20

Schreiben Sie ein Sortierprogramm, das nach dem Selection-Sort Algorithmus arbeitet: 1. suche das kleinste/größte Element des Arrays, 2. vertausche dieses mit dem ersten Element des Arrays, 3. gehe wieder zu Schritt 1, jetzt aber mit dem verkürzten Array. Wieso ist dieses Verfahren effizienter als Bubble-Sort?

Aufgabe 21 (fakultativ)

Familie Müller ist zu einer Geburtstagsfeier eingeladen. Leider können sich die Familienmitglieder (Anton, Berta, Claus und Doris) nicht einigen, wer hingeht und wer nicht. In einer gemeinsamen Diskussion kann man sich jedoch auf die folgenden Grundsätze verständigen:

- Mindestens ein Familienmitglied geht zu der Feier.
- Anton geht auf keinen Fall zusammen mit Doris.
- Wenn Berta geht, dann geht Claus mit.
- Wenn Anton und Claus gehen, dann bleibt Berta zu Hause.
- Wenn Anton zu Hause bleibt, dann geht entweder(!) Doris oder Claus.

Helfen Sie Familie Müller, indem Sie ein Programm erstellen, das alle Gruppierungen ermittelt, in denen Familie Müller zur Feier gehen könnte.

Tipp: Erstellen Sie alle denkbaren Kombinationen und prüfen Sie *jede* darauf, ob sie *alle* obigen Grundsätze erfüllt.

Aufgabe 22 (fakultativ)

Man berechne die Nullstellen einer Funktion mithilfe des sogenannten Bisektionsverfahrens: eine linke und rechte Intervallgrenze wird eingelesen, die Funktionswerte hier und in der Mitte des Intervalls werden bestimmt. Dann wird das Intervall immer wieder halbiert, bis genügend Genauigkeit erreicht ist. Verwenden Sie als Beispiel die Funktion $f(x) = x^3 - x^2 - x + 0.04$. Diese hat Nullstellen in den Intervallen [0,1], [-1,0] und [1,2].

Kapitel 4 — Funktionen

Aufgabe 23

Freunden Sie sich zunächst einmal mit Unterprogrammen an. Im Skript sind folgende Programmfragmente zu finden:

- Berechnung von n^m: Funktion `mypow`
- Näherung für $\exp(x)$: Funktion `myexp`
- Einfache Rekursion: Funktion `fak`
- Türme von Hanoi: Funktion `hanoi`

Machen Sie daraus jeweils ein lauffähiges Programm.

↪

Aufgabe 24

Schreiben Sie das Programm zur Primzahlzerlegung (Aufgabe 18) so um, dass das Einlesen der Zahl von einer Funktion `eingabe` erledigt wird und die Berechnung und Ausgabe der Primfaktoren von einer anderen Funktion `zerlegung`. Überlegen Sie zuerst, welche Variablentypen die Funktionen erhalten und zurückliefern sollen!

Aufgabe 25

Erstellen Sie ein Programm zur numerischen Integration einer Funktion mit der Rechtecks-Methode. Das Programm soll eine Funktion `double f(double x)` enthalten, die zu jedem `x` einen Funktionswert liefert, sowie eine Funktion `double integral(double a, double b, int n)`. Darin soll das Intervall $[a, b]$ in n kleine Intervalle aufgeteilt werden, die Fläche der Rechtecke unter dem Funktionsgraphen berechnet werden und die Gesamtfläche zurückgeliefert werden.

Tipp: Zum Testen Ihres Programms können Sie $\int_0^1 \exp(-x^2)\mathrm{d}x \approx 0.746824$ benutzen.

Aufgabe 26

Schreiben Sie eine Funktion `int palin(char *c, int erstes, int letztes)`, die rekursiv prüfen soll, ob ein gegebenes Wort oder ein Satz ein Palindrom[i] ist. Ihr wird das Wort oder der Satz in einem `char`-Array `c` sowie die Stelle des ersten und letzten Zeichens als `int` übergeben. Stimmen beide Zeichen überein, so kann man mit dem Prüfen des verkürzten Wortes oder Satzes fortfahren.

Tipp: Die Funktion `strlen(text)` liefert die Länge des Textes `char text[]`. Sie ist in `string.h` deklariert. Überlegen Sie sich auch, wie Sie Leerzeichen behandeln (siehe Bsp. in der Fussnote). Beachten Sie bitte die Schreibweise `...palin(char *c...` zur Übergabe des `char`-Arrays.

Aufgabe 27

Das Treppenproblem: Sie können bei einer Treppe entweder genau eine Stufe pro Schritt nehmen, oder, falls sie sportlich und durchtrainiert sind, 2 Stufen auf einmal. Damit haben sie mehrere Möglichkeiten eine mehrstufige Treppe zu überwinden (z.B.: 1 Stufe $\rightarrow$ 1 Möglichkeit, 2 Stufen $\rightarrow$ 2 Möglichkeiten, 3 Stufen $\rightarrow$ 3 Möglichkeiten, 4 Stufen $\rightarrow$ 5 Möglichkeiten, ...). Schreiben Sie ein Programm, welches Ihnen rekursiv die Anzahl der Möglichkeiten berechnet, eine Treppe T mit n Stufen zu erklimmen:

$$T_n = \begin{cases} n & \text{für} \quad n < 3 \\ T_{(n-1)} + T_{(n-2)} & \text{für} \quad n \geq 3 \end{cases}$$

Berechnen Sie dann die Anzahl der Möglichkeiten für n=25. Wieso und wie müssen Sie ihr Programm modifizieren, damit dieses auch die Anzahl der Möglichkeiten für n=768 (Ulmer Münster) und n=1860 (Empire State Building) berechnen kann?

Tipp: Bei der Treppe zur Physik-Bibliothek (n=25) haben Sie 121.393 Möglichkeiten, beim Ulmer Münster etwa 23 x 10^{159} Möglichkeiten und beim Empire State Building immerhin etwa 37 x 10^{387} Möglichkeiten.

↪

[i] vorwärts ⇜⇝ rückwärts z.B.: ‚anna', ‚otto', ‚ein *schwarzer* mit gazelle zagt im regen nie'. Danke an dieser Stelle an Hr. De Muirier von der DLR, der mich darauf hingewiesen hat, dass das im Original-Palindrom verwendete Wort heutzutage „rassistisch und stark pejorativ ist". (Zitat)

Aufgabe 28 (fakultativ)

Zur Berechnung der Wurzel einer Zahl gibt es ein rekursives Verfahren. Die Rekursionsformel lautet:

$$\mathrm{w}(n,x) = \begin{cases} \frac{1}{2}\left[\mathrm{w}(n-1,x) + \frac{x}{\mathrm{w}(n-1,x)}\right] & \text{für} \quad n \geq 1 \\ 1 & \text{für} \quad n = 0 \end{cases}$$

Schreiben Sie eine Funktion `double wurzel(int n, double x)`, die rekursiv die Wurzel einer eingegebenen Zahl `x` berechnet. Die Zahl `n` gibt die Rekursionstiefe an. Vergleichen Sie das Ergebnis mit dem exakten Wert.

Aufgabe 29

Man berechne für kleine ganze Zahlen m, n $(m \leq 3, n \leq 8)$ die Ackermann-Funktion $a(m,n)$:

$$a(m,n) := \begin{cases} n+1 & \text{falls} \quad m = 0 \\ a(m-1,1) & \text{falls} \quad n = 0 \\ a(m-1, a(m,n-1)) & \text{sonst} \end{cases}$$

Tipp: Zum Testen Ihres Programms können Sie $a(1,1) = 3, \quad a(3,3) = 61, \quad a(3,8) = 2045$ benutzen.

Kapitel 5 — Abgeleitete Datentypen

Aufgabe 30

Erstellen Sie ein Programm mit einer Struktur namens `person`, die einige Daten (Name, Vorname, Alter, Schuhgröße,...) einer Person speichert. Eine Funktion `eingabe` soll einen Datensatz einlesen und zurückliefern. Das Programm soll dann mithilfe von `eingabe` die Daten zweier Personen aufnehmen und den Namen des Älteren ausgeben.

Kapitel 7 — Dateibearbeitung

Aufgabe 31

Im Sript finden Sie ein Beispielprogramm zur Dateibearbeitung. Versuchen Sie, es zu verstehen, und experimentieren Sie damit.

Aufgabe 32

Erstellen Sie ein Programm, das einen Text aus einer Datei einliest und auf dem Bildschirm ausgibt, wobei Groß- in Kleinbuchstaben verwandelt werden sollen und umgekehrt.

↪

Aufgabe 33

Schreiben Sie ein Programm, das einen Text aus einer Datei und einen Buchstaben einliest und ermittelt, wie oft der Buchstabe in dem Text vorkommt.

Aufgabe 34

Schreiben Sie ein Programm, das aus einem Text in einer Datei alle mehrfach vorkommenden Leerzeichen entfernt und den Text auf dem Bildschirm ausgibt.

Kapitel 8 — Pointer

Aufgabe 35

Es ist `feld` ein `int`-Array und `p` ein `int`-Zeiger. Welche der folgenden Zuweisungen sind zulässig, welche nicht? (Ggf. ausprobieren!)

- `p = feld;`
- `feld = p;`
- `p = &feld[3];`
- `feld[2] = p[5];`

Aufgabe 36

Es seien `p1` und `p2` zwei `int`-Zeiger und `i` eine `int`-Variable. Welche Zuweisungen wird der Compiler akzeptieren, welche nicht? (Ggf. ausprobieren!)

- `p1 = p2 + i;`
- `p1 = i + p2;`
- `i = p1 * p2;`
- `i = p1 - p2;`
- `i = p1 + p2;`

Aufgabe 37

Schreiben Sie ein bestehendes Programm (z.B. Lösung der quadratischen Gleichung) so um, dass die Benutzereingabe von einer Funktion `eingabe` erledigt wird, der man die Variablen übergeben muss (Pointer!).

Aufgabe 38

Schreiben Sie eine Funktion `stringlength`, an die ein `char`-Array übergeben wird, und die die Länge dieser Zeichenkette zurückliefert.

Tipp: Das Ende einer Zeichenkette ist durch `'\0'` (ASCII: 0) gekennzeichnet.

↪

Aufgabe 39

Der Benutzer soll ein Wort und ein Zeichen eingeben. Es soll das Wort, das um dieses Zeichen verlängert worden ist, ausgegeben werden (z.B.: `Hall` + o → `Hallo`). Verwenden Sie dazu die vorher (Aufgabe 38) definierte Funktion `stringlength`, um die Stelle des letzten Zeichens zu ermitteln.

Aufgabe 40

Schreiben Sie das Bubble-Sort-Programm (Aufgabe 19) so um, dass das Sortieren in einer Funktion geschieht, der man den Zeiger auf das Array übergibt.

Aufgabe 41

Schreiben Sie ein Programm, das eine Funktion `prosumo` enthält. Dieser Funktion wird ein Vektor übergeben; sie liefert die Summe und das Produkt der Vektorelemente zurück.

Aufgabe 42

Schreiben Sie das Integrations-Programm (Aufgabe 25) so um, dass die zu integrierende Funktion `f(x)` der Funktion `integral` als Pointer übergeben wird.

Aufgabe 43

Erzeugen Sie dynamisch Speicherplatz für n `double`-Zahlen, lesen Sie sie ein und bestimmen Sie das Maximum und die Summe. Wie sieht die Speicheralloziierung in reinem C und in C++ aus?

Aufgabe 44

Erstellen Sie eine Funktion zur Sortierung eines Arrays von `int`-Zahlen. Übergeben Sie der Sortierfunktion als Parameter eine Funktion zum Vergleich zweier Zahlen. Erzeugen Sie unterschiedliche Sortierungen (aufsteigend, absteigend, nach letzter Ziffer, nach der Quersumme,...), indem Sie unterschiedliche Vergleichsfunktionen an die Sortierfunktion übergeben.

↪

Kapitel 9 — Fortgeschrittene Programmiertechniken

Aufgabe 45

Schreiben Sie ein Programm zur Verwaltung einer verketteten Liste. Als ‚Nutzinhalt'der Liste wählen Sie eine `char`-Variable, in der Sie genau ein Zeichen speichern können. Füllen Sie die Kette in einer Funktion `fangan` mit den Kleinbuchstaben des Alphabets. Mit einer Funktion `gibaus` können Sie jeweils den Inhalt der gesamten Liste ausgeben. Experimentieren sie dann, indem sie ihr Programm mit einer Funktion `fuegein` erweitern, die an beliebiger Stelle n-mal den Buchstaben 'X' einfügt. Vergessen Sie nicht eine Funktion `loesche` zu implementieren, mit der Teile der Liste gelöscht werden können. Eine weitere Ausbaumöglichkeit ist eine Funktion `ersetze`, die nach Wunsch beliebige Kleinbuchstaben durch Grossbuchstaben ersetzt.

Tipp: Lassen Sie sich durch verkettete Listen nicht abschrecken! Setzen Sie sich in aller Ruhe mit Bleistift, Papier und einer guten Tasse Tee bzw. Capuccino[ii] hin und dröseln Sie Element für Element und Pointer für Pointer auf. Vergessen Sie dabei nicht, dass auch Pointervariablen ganz einfache, lokale Variablen sein können...

[ii] oder Espresso, je nach Tageszeit