Abschlußtest Programmieren

13. Juni 2025

Name: Punkte von 32:

Gruppe: ☐ Uni-Haase ☐ TU-Rosenberger

1. **(6 = 1+1+1+1+2 P)** Schreiben Sie die folgenden mathematischen Ausdrücke als **korrekte** C++-Anweisungen (Variablendeklarationen, Speicherallokierungen und Fileinkludierungen sind nicht nötig):

   (a) $\sqrt{\frac{e^x+e^{-x}}{2}}$

   (b) $c = \max(x, y)$ ohne Verwendung der Funktion `max`

   (c) $\frac{0.5}{xe^x} + \cos^2(x)\mathrm{e}^{\ln(x)}$ (Kein Test auf $x > 0$ nötig)

   (d) Test, ob y ungleich Null ist und gleichzeitig $\frac{x}{y^2}$ negativ ist

   (e) $b_k = \sum_{j=0}^{k} \frac{e^{-jx}}{(j+1)^2} \quad \forall k = 0, \ldots, 10$, wobei **b** als leerer (STL-)Vektor deklariert wurde. Verwenden Sie die Methode `push_back`.

Name: ☐ Uni-Haase ☐ TU-Rosenberger

2. **(4 P)** Welche Ausgaben liefert das folgende Programm in den drei `cout` Anweisungen? Begründen Sie Ihre Antwort jeweils stichwortartig!

```
 1 #include <algorithm>  // swap(a,b)    :  a <--> b
 2 #include <iostream>
 3 #include <vector>
 4 using namespace std;
 5
 6 vector<double> myfun(vector<double>& y, double a, double b) {
 7     vector<double> tmp;
 8     if (b<a) { swap(a,b); }          // exchange  a <--> b
 9
10     for(size_t k=0; k<y.size(); ++k) {
11         if (a<=y[k] && y[k]<=b)
12         { tmp.push_back(y[k]); }
13         else
14         { y[k] = a; }
15     }
16     return tmp;
17 }
18
19 int main() {
20     vector<double> x{-2.1, -5.0, 6.2, 4.0, 1.1};
21     double a(4.5), b(-2.5);
22     auto z = myfun(x, a, b);
23
24     cout << "b: " << b << endl;
25     cout << "x: " << x.size() << "  " << x[2] << endl;
26     cout << "z: " << z.size() << "  " << z[2] << endl;
27     return 0;
28 }
```

Name: ☐ Uni-Haase ☐ TU-Rosenberger

3. **(4 P = 3+1)**
Ein Pythagoreisches Tripel wird von drei **natürlichen** Zahlen $a, b, c \in \mathbb{N}$ gebildet, für die

$$a^2 + b^2 = c^2$$

gilt.

- Schreiben Sie ein Programm, welches für eine einzugebende obere Schranke $u \in \mathbb{N}$ alle Pythagoreischen Tripel (a, b, c) aus $[1, u]^3$ bestimmt und im Terminal ausgibt. Dabei gelten z.B. die Tripel $3^2 + 4^2 = 5^2$ und $4^2 + 3^2 = 5^2$ als gleich und es darf nur eines davon ausgegeben werden.
- Geben Sie zum Schluss die Anzahl der gefundenen Pythagoreischen Tripel aus.

Es dürfen keine `float`-oder `double`-Variablen und keine Funktionen aus *cmath* (z.B., `sqrt`, `pow`, `log`, `exp`) für die Lösung dieser Aufgabe verwendet werden.

Hierbei sind keine `break`, `continue`, `goto`, `return` erlaubt.

Name: ☐ Uni-Haase ☐ TU-Rosenberger

Name: ☐ Uni-Haase ☐ TU-Rosenberger

4. **(10 P = 1.5 + 4.5 + 1 + 2 + 1)**
Jede natürliche Zahl $n \geq 2$ besitzt eine eindeutige Zerlegung in Primfaktoren. Das heißt, es gibt ein $r \in \mathbb{N}$ und nicht notwendigerweise verschiedene Primzahlen $p_1, p_2, \cdots, p_r \in \mathbb{N}$ sodass gilt:

$$n = p_1 \cdot p_2 \cdot \cdots \cdot p_r \ .$$

Zum Beispiel gilt $220 = 2 \cdot 2 \cdot 5 \cdot 11$, es gilt aber auch $11 = 11$ da 11 eine Primzahl ist. Folgender C++-Code ist aufzuschreiben:

(a) Schreiben Sie eine Funktion *is_divisor* mit dem Rückgabetypen *bool*, welche für zwei natürliche Zahlen k, n als Elemente der Parameterliste nur dann *true* zurückgibt, wenn $k \mid n$ (k teilt n). Der Modulo–Operator wird hier nützlich sein.

(b) Schreiben Sie eine Funktion *prime_factorization*, die für eine per Parameterliste übergegebene natürliche Zahl n alle (auch mehrfach vorkommende) Primfaktoren berechnet und diese Faktoren in einem Container v (z.B., `vector` oder `list`) an das aufrufende Programm zurückgibt. Nutzen Sie dabei Ihre Funktion *is_divisor*.
Hinweise:
In obigem Beispiel ist 2 der erste gefundene Teiler von 220 und anschließend wird $220/2 =: 110$ auf weitere Teiler untersucht (Wie lange muß man weitersuchen?).
Ein Liste der Teiler wird nicht benötigt, da durch das Finden der kleineren Primzahlteiler die zusammengesetzten Teiler wie $4 = 2 \cdot 2$ und $10 = 2 \cdot 5$ bereits enhalten sind (also aufsteigend auf alle Zahlen im Bereich testen).

(c) Im dazugehörigen Hauptprogramm lesen Sie so lange eine natürliche Zahl n ein, bis $n \geq 2$ gilt und rufen im Anschluss die Funktion *prime_factorization* auf.

(d) Anschließend im Hauptprogramm: Berechnung der Summe aller Primfaktoren, sowie das Produkt aller Primfaktoren für n.
Bauen Sie dabei einen Sicherheitstest ein, der am Bildschirm "Alarm" schlägt, wenn das Produkt aller Primfaktoren von n ungleich der Zahl n ist!

(e) Die einzelnen Primfaktoren, sowie deren Summe geben Sie anschließend über `cout` aus.

Die Teile (c), (d) und (e) können auch ohne vorhandenen Code zu (a) oder (b) angegeben werden.

Hierbei sind keine `break`, `continue`, `goto` erlaubt.
Es darf in der Funktion nur **genau ein** `return`-Statement auftreten, dieses darf sich nicht innerhalb eines Zyklus oder einer Alternative befinden.
Die Verwendung von Pointern ist nicht erlaubt.

Name: ☐ Uni-Haase ☐ TU-Rosenberger

Name: ☐ Uni-Haase ☐ TU-Rosenberger

5. **(10 P)** Deklarieren und definieren Sie eine Klasse `hotel` sodaß das unten gegebene Hauptprogramm ab Zeile 6 **unverändert** funktioniert. Vom Compiler automatisch erzeugte Konstruktoren/Operatoren müssen nicht implementiert werden.

   Ein Hotel besitzt jeweils eine fixe Anzahl von 1-und 2-Bettzimmern (Kapazität).
   Mit Beginn der Urlaubssaison werden diese Zimmer gebucht und (später) wieder freigegeben. Dabei wird stets ein Zweibettzimmer von 2 Personen gebucht und freigeben, es gibt bei uns keine Einzelbelegung eines Doppelzimmers.
   Das Hauptprogramm simuliert eine solche Saison.

   In folgenden Codezeilen werden Methoden Ihrer Klasse bzw. weitere Funktionen benötigt:

   - Zeile 8: Parameterkonstruktor um die Zimmerkapazitäten (für 1- und 2-Bettzimmer) festzulegen. Zu Anfang sind alle Zimmer unbelegt/ungebucht.
   - Zeile 9: Ausgabe der vorhanden freien (also ungebuchten) Zimmer.
   - Zeile 14: War die Reservierung eines 1- oder 2-Bettzimmers erfolgreich (*true*)? Nichterfolgreich (*false*) bedeutet, daß in der gewünschten Kategorie kein Zimmer mehr frei ist.
   - Zeile 17: Ausgabe der verbliebenen freien Gesamtbettenanzahl.
   - Zeile 22: Freigabe (*true*) eines 1-, 2-Bettzimmers, falls ein solches noch belegt war.
   - Zeile 25: Wiederhole den Zyklus bis alle Zimmer belegt sind.

```
1 #include <cstdlib>              // rand(), srand()
2 #include <iostream>
3 using namespace std;
4 // Ihre Klassendeklaration und -definition
5     ...
6 int main()
7 {//       1-Bett, 2-Bett max. Kapazitaet des Hotels
8     hotel jufa(5,8);
9     cout << jufa << endl;
10
11    do
12    {
13        int iPers = 1+(rand()%2);    // aus [1,2]: 1- oder 2-Bettzimmer
14        if (jufa.bookRoom(iPers))    // Buchung erfolgreich?
15          { cout << jufa << "     nach " << iPers << "-bed reservation." << endl; }
16        else                         // kein passendes Zimmer verfuegbar
17          { cout <<"not booked: "<<jufa.actualVacancyBeds()<<" beds available.\n";}
18
19        int rPers = 1+(rand()%10);  // aus [1,10]
20        if ( 2>=rPers )        // Soll ein 1- oder 2-Bettzimmer freigegeben werden?
21        {       // War ein solches Zimmer belegt und wurde es freigegeben?
22            if (jufa.releaseRoom(rPers))
23               cout << "Room released" << endl;
24        }
25    } while (!jufa.occupied());
26
27    return 0;
28 }
```

Name: ☐ Uni-Haase ☐ TU-Rosenberger

Name: ☐ Uni-Haase ☐ TU-Rosenberger

Name: ☐ Uni-Haase ☐ TU-Rosenberger