technische universität dortmund

fakultät für informatik

Amer Krivošija
Denis Kurz
Eugen Rusakov

Dortmund, den 17. November 2016

Übung zur Vorlesung EidP (WS 2016/17)

Blatt 4

Block gelb

Es können 4 Punkte und 3 Bonuspunkte erreicht werden.

Abgabedatum: 24. November 2016, 23:59 Uhr

Hinweise

- Bitte beachten Sie die aktuellen Hinweise unter

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep1617uebung/`

- Für die Abgabe sind die Dateien `Aufgabe_04_0.txt`, `Aufgabe_04_1.cpp` und bei Bedarf `Aufgabe_04_2.cpp` zu erstellen.
- In der optionalen Datei `Anmerkungen.txt` können Sie allgemeine Anmerkungen bezüglich Ihrer Lösungen notieren.
- Stellen Sie sicher, dass alle von Ihnen abgegebene Dateien reine Textdateien im UTF-8-Format sind.
- Dies ist das letzte Blatt im gelben Block! Das bedeutet, dass Sie spätestens mit der Abgabe Ihrer Lösungen zum vorliegenden Blatt die zum Bestehen des gelben Blocks nötigen 8 Punkte erreichen müssen. Wenn Sie die nötigen Punkte noch nicht erreicht haben, haben Sie mit Aufgabe 2 auf diesem Blatt die Möglichkeit, Bonuspunkte zu erhalten.
- Am 20.12.2016 findet von 12:15 bis 13:45 Uhr eine Übungsklausur im HG II, HS 3 (Vorlesungsraum) statt. Weitere Hinweise werden auf der Veranstaltungswebsite bekannt gegeben.

Aufgaben

Aufgabe 0: Grundlagen (1 Punkt)

Speichern Sie Ihre Ergebnisse in der Ergebnisdatei `Aufgabe_04_0.txt`.

a) An welcher Stelle der Deklaration `int *p;` fehlt ein `const`-Qualifizierer, damit auch nach der Initialisierung des Zeigers `p`, die Zuweisung `p = nullptr;` möglich ist, nicht aber die Zuweisung `*p = 4;`? (0.1 Punkte)

b) Nennen Sie alle Operatoren, die der Erzeugung oder dem Löschen von Datenstrukturen im dynamischen Speicher dienen. (0.3 Punkte)

c) Wie definiert man eine mehrfache Auswahl mit `switch`? Bitte geben Sie ein Beispiel an. (0.2 Punkte)

d) Welche Bedeutung hat der `default`-Zweig einer `switch-case-default`-Anweisung? (0.2 Punkte)

e) Was passiert, wenn bei einer `switch`-Anweisung zwischen zwei `case`-Zweigen keine `break`-Anweisung steht? (0.2 Punkte)

Aufgabe 1: Mehrfache Auswahl (3 Punkte)

Ergänzen Sie in der Datei `Aufgabe_04_1.cpp` das unten angegebene Programmfragment so, dass für ein eingegebenes Datum der darauffolgende Tag berechnet wird. Dabei soll zuvor geprüft werden, ob das Datum gültig ist. Wird ein Datum vor dem 1.1.1600 oder nach dem 31.12.2600 eingegeben, soll `99.99.9999` ausgegeben werden. Dasselbe soll ausgegeben werden, wenn ein ungültiges Datum eingegeben wird. Verwenden Sie zur Bestimmung der Anzahl an Tagen des eingegebenen Monats eine Mehrfachauswahl. Achten Sie dabei insbesondere auf Schaltjahre. Alle Jahre, die durch 400 teilbar sind, sind Schaltjahre. Alle anderen Jahre sind genau dann Schaltjahre, wenn sie durch 4, aber nicht durch 25 teilbar sind.
Kompilieren Sie das Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse für die Eingaben `0.11.2016`, `11.11.1111`, `30.11.2016`, `28.02.2016` und `28.02.2017` als Block-Kommentar an das Ende der Datei `Aufgabe_04_1.cpp`.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      unsigned int tag, monat, jahr;
6      cout << "Geben Sie Tag, Monat und Jahr ein" << endl;
7      cin >> tag;
8      cin >> monat;
9      cin >> jahr;
10     //************************************
11     // Hier beliebig viele Zeilen ergaenzen
12     //************************************
13     cout << "Naechster Tag ist "
14          << tag << "." << monat << "." << jahr << endl;
15     return 0;
16 }
```

Bonusaufgabe 2: Dynamische Speicherallokation (3 Bonuspunkte)

Das Pascalsche Dreieck besteht aus versetzt angeordneten Zeilen von Zahlen, wobei jede Zeile genau eine Zahl mehr enthält als die vorherige. Jede Zeile beginnt und endet mit einer Eins. Alle anderen Werte ergeben sich jeweils aus der Summe der beiden Werte schräg links und rechts darüber.

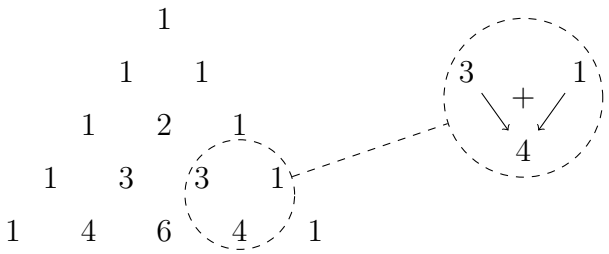

Abbildung 1: Pascalsches Dreieck mit Seitenlänge 5

Ergänzen Sie in der Datei `Aufgabe_04_2.cpp` das folgende Programmfragment so, dass das Pascalsche Dreieck mit der Seitenlänge `size` berechnet und ausgegeben wird. Allokieren Sie für die Variable `triangle` genau so viel dynamischen Speicher zur Laufzeit, wie für die Darstellung des Dreiecks benötigt wird, und geben Sie diesen Speicher geeignet wieder frei. Kompilieren Sie das Programm und führen Sie es aus. Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei `Aufgabe_04_2.cpp`.

Hinweis: Das Programm soll auch bei Veränderung des Wertes von `size` die korrekten Ausgaben berechnen, ohne dass weitere Anpassungen nötig sind.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      unsigned int size = 10;
6      //************************************
7      // Hier beliebig viele Zeilen ergaenzen
8      //************************************
9      for (unsigned int n = 0; n < size; ++n) {
10         for (unsigned int indent = 0; indent < size - n; ++indent)
11             cout << "  ";
12         for (unsigned int k = 0; k < n + 1; ++k) {
13             if (triangle[n][k] < 100)
14                 cout << ' ';
15             cout << triangle[n][k] << ' ';
16             if (triangle[n][k] < 10)
17                 cout << ' ';
18         }
19         cout << endl;
20     }
21     //************************************
22     // Hier beliebig viele Zeilen ergaenzen
23     //************************************
24     return 0;
25 }
```