Prof. Dr. F. Sadlo, L. Büch *Heidelberg, 20. Dezember 2015*

Übungsblatt 10

Anmerkung: Beachten Sie die Abgabefrist (bis zum 07.01.2016) und dass die **Abgabe für alle Aufgabenteile obligatorisch elektronisch per Email** erfolgen muss. Alle Punkte dieses "Weihnachtsblatts" 10 werden als **Bonuspunkte** gewertet. Bitte beachten Sie, dass sich diese Modalitäten vom am 22.12.2015 ausgegebenen Übungsblatt 11 unterscheiden.

Zusatzaufgabe II Weihnachtsbäume [10 Punkte]

Schreiben Sie ein Programm, das einen Weihnachtsbaum auf dem Bildschirm ausgibt. Ihrer Kreativität sind dabei keine Grenzen gesetzt. So können Sie Ihren Baum zum Beispiel mit zufällig verteiltem Schmuck oder brennenden Kerzen verzieren.

Es werden alle Abgaben akzepiert, ob sie nun einfach aus kleinen Sternchen die Silhouette eines Baumes erzeugen oder durch geschickte Platzierung von Steuerzeichen des Terminals einen in hellen Farben erstrahlenden Baum ausgeben.

Punkte gibt es u.a. für:

- Ein funktionierendes Programm (d. h. die Ausgabe lässt sich zumindest mit Wohlwollen als Baum interpretieren) [*2 Punkte*]
- Ein gut dokumentiertes und vor allem gut durchdachtes Programm (d. h. die einzelnen Einheiten des Programms sind sinnvoll in Klassen gekapselt, und die Gedankengänge, die zu dieser Kapselung führten, sind aus den Kommentaren ersichtlich) [*2 Punkte*]
- Kreativität und Funktionsumfang (z. B. randomisierte Platzierung des Schmucks, Sicherstellen, dass genügend freier Platz zwischen den Kugeln bleibt, farbige Ausgabe, variable Größe oder Breite des Baumes, oder was Ihnen eben sonst so einfällt) [*6 Punkte*]

Schreiben Sie zu Beginn Ihres Programmes einen großen Kommentar, in dem Sie auflisten, welche Besonderheiten Ihres Programmes Ihrer Meinung nach für die Bewertung bzgl. des dritten Punktes von Belang sind.

Anmerkung: Farbige Ausgabe wird unter Unix durch Escape-Sequenzen[1] gesteuert. Diese haben die Form

`\033[x;ym` oder `\033[ym` (hier ist $x = 0$) oder auch `\033[m` (hier sind $x = y = 0$),

wobei x und y für Codes stehen, die die Eigenschaften der Schrift beschreiben. So steht $x = 1$ für Fettdruck und $y = 34$ für Blau, so dass ein Text, der zwischen

[1] `http://de.wikipedia.org/wiki/Escape-Sequenz`

`\033[1;34m` und `\033[m`

steht, in blauem Fettdruck gesetzt wird. `\033[m` setzt die Schrifteigenschaften wieder auf das Default. Die möglichen Farbkombinationen sind an vielen Stellen im Netz aufgelistet, z. B. unter `http://www.pixelbeat.org/docs/terminal_colours/`.

Um ein lesbares Programm zu erhalten, bietet es sich an, eine Klasse zu schreiben, die in der Lage ist, übergebene Strings "farbig" zurück zu geben, indem sie sie mit den passenden Steuersequenzen umgibt.

Zusatzaufgabe III Aufwand und Laufzeiten [✍ 5 Punkte]

Das Tupel $A = (a_1, ..., a_n)$ enthalte n reelle Zahlen. Gesucht seien weitere n Zahlen $M = (m_1, ..., m_n)$, die folgendermaßen definiert sind:

$$m_i := \frac{1}{i} \sum_{k=1}^{i} a_k,$$

d. h. m_i ist der Mittelwert der ersten i Zahlen aus dem Tupel A.

Wir wollen zwei Funktionen für diese Aufgabe schreiben und deren Laufzeitverhalten bei wachsender Anzahl n anhand einer kleinen Testreihe aus n_{Level} Versuchen beobachten. In der Datei `mittelwerte.cc` finden Sie das Hauptprogramm, welches über die Kommandozeile drei Zahlen einliest: n_{Start}, n_{Level} und $n_{\text{Verbosity}}$. n_{Start} wird nach jedem Versuch verdoppelt, insgesamt $n_{\text{Level}} - 1$ mal. $n_{\text{Verbosity}}$ schaltet die Ausgabe der Daten ein ($\neq 0$) oder aus ($= 0$).

Die Funktion `repeatTest` soll für ein festes n die obige Aufgabe durchführen. Es legt ein dynamisches Feld `a` der Größe n an.

1. Schreiben Sie die fehlenden Funktionen
   (a) `initialize`, die das Feld `a` mit n möglichst verschiedenen reellen Zahlen auffüllt, die nicht zu schnell wachsen, z. B.:

   $$a_k = (-1)^{k+1} k \cos(k) \; .$$

   Mathematische Funktionen in C++ sind z. B. auf der Seite `http://en.cppreference.com/w/cpp/numeric/math` mit vielen Beispielen dokumentiert. Sie werden über `#include <cmath>` verfügbar gemacht. Wie hoch ist die Komplexität in Abhängigkeit von n? [*1 Punkte*]
   (b) `printArray`, die den Inhalt von `a` in einer einzigen Zeile ausgibt (mit zwei Leerzeichen Abstand zwischen jeder Zahl). Zeilenumbruch bitte erst einfügen, nachdem die n Zahlen ausgegeben sind. Wenn $n > 5$ ist, bitte nur die letzten 5 Zahlen ausgeben! [*1 Punkte*]
   (c) `meanValues`, die nacheinander die Zahlen m_i nach obiger Formel berechnet und in das Feld `meanArray1` steckt. Wie hoch ist die Komplexität in Abhängigkeit von n? [*1 Punkte*]
2. Wie lässt sich der Aufwand um eine Potenz reduzieren? (Hinweis: Bei der Berechnung der m_i dürfen bereits berechnete Vorgänger benutzt werden!) Schreiben Sie dafür die fehlende Funktion `meanValuesFast`. [*2 Punkte*]

Vergleichen Sie die Entwicklung der Laufzeiten der beiden Funktionen `meanValues` und `meanValueFast`, z. B. für $n_{\text{Start}} = 2000$ und $n_{\text{Level}} = 6$.

Zusatzaufgabe IV Komplexität von Algorithmen [✉ 5 Punkte]

a) Gauss-Elimination:

```
for ( k=0; k<n-1; k=k+1 )
  for ( i=k+1; i<n; i=i+1 )
    for ( j=k+1; j<n; j=j+1 )
      a[i][j] = a[i][j] - a[i][k] * a[k][j] / a[k][k];
```

Die Berechnung innerhalb der innersten Schleife ist von den Werten der Schleifenzähler unabhängig, d. h. benötigt eine konstante Zeit t_g. Wie ist die algorithmische Komplexität in Abhängigkeit von `n`? [*2 Punkte*]

b) Rekursiver Funktionsaufruf:

```
int f( int a, int b )
{
  if ( a >= b )
    return g( a );
  else
    return f( a, b-1 ) + f( a+1, b );
}
```

Überlegen Sie zuerst, welche Größe als Komplexitätsparameter sinnvoll ist! Die Zeit, um `g( a )` auszurechnen, soll vom Wert von `a` unabhängig sein. Wie ist die algorithmische Komplexität? [*3 Punkte*]

Abgabe: 7. Januar 2016, 14:15 Uhr,
ausschließlich per E-Mail (✉) an Ihren Tutor