Programmiertechnik II

Sommersemester 2020 – Aufgabenblatt 7

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Anmerkungen zu Aufgabenblatt 7

Der zu verwendende Programmrahmen steht als Zip-Archiv `uebung7.zip` im Moodle unter der Adresse `https://moodle.hpi3d.de/course/view.php?id=137` zum Download zur Verfügung.

Aufgabe 7.1: Baumdarstellung als Vektorgrafik (8 Punkte 1+2+5)

Sie haben in der Vorlesung viel über Bäume gehört und möchten nun ein Tool entwickeln, mit dessen Hilfe Sie schnell eine Vorstellung von gegebenen Binärbaumen erhalten können.

a) Erweitern Sie die den Programmrahmen `svg.cpp` um eine Tiefensuche in der Methode `depthFirstTraversal` und geben Sie die Knoten des Baumes in der entsprechenden Reihenfolge auf der Kommandozeile aus.

b) Implementieren Sie die Methode `breadthFirstTraversal`, indem Sie eine Breitensuche im gegebenen Baum durchführen und die Knoten ebenfalls auf der Kommandozeile ausgeben.

c) Sie möchten nun eine bildliche Darstellung des Baumes erzeugen. Dazu eignet sich das SVG-Format, in welchem textbasiert Vektorgrafiken definiert werden können. Implementieren Sie `writeSVG` so, dass der Baum als Vektorgrafik im SVG-Format ausgegeben wird. Die Ausgabe soll in etwa so aussehen, wie in Abbildung 1 dargestellt.

- Machen Sie sich dazu mit dem Dateiformat SVG[1] vertraut.
- Achten Sie darauf, gültiges XML mit einem korrekten Header in die Ausgabedatei zu schreiben und verwenden Sie als Version SVG 1.1.
- Geben Sie explizit `width`, `height` und `viewBox` in Ihrer erzeugten Datei an.
- Die Knoten und Kanten sollen durch die Methoden `writeSVGNode` und `writeSVGEdge` erzeugt werden, die neben dem Ausgabestream jeweils die benötigten Koordinaten und die ID des Knotens erhalten. Die Koordinaten müssen im Programm berechnet und dürfen nicht statisch vorgegeben werden.
- Ergänzen Sie den Programmrahmen um zusätzliche Methoden, wenn das erforderlich ist. Vermeiden Sie Dopplungen und achten Sie allgemein auf sauberen Code. Fügen Sie Kommentare ein, wenn die Funktionsweise nicht direkt ersichtlich ist oder Sie komplexere Berechnungen durchführen.
- Die Ausgabe muss auch für andere Bäume funktionieren. Sie können zur Vereinfachung vier Level als größtmögliche Ausdehnung annehmen.

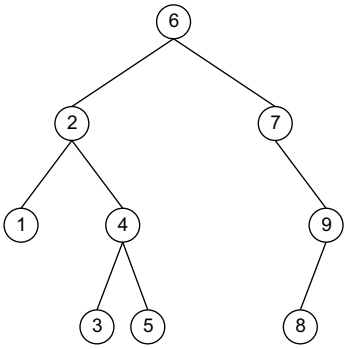

Abbildung 1: Beispielhafte Ausgabe für den ersten Baum aus dem Programmrahmen.

[1] `https://www.w3.org/TR/SVG11/`

Aufgabe 7.2: Textsuche mit Rolling-Hash-Funktion (6 Punkte $^{1+1+2+2}$)

Vervollständigen Sie die in der Datei `hashsearch.txt` gegebene Implementierung einer Textsuche mit Hilfe einer Rolling-Hash-Funktion. Der Programmrahmen lädt eine Textdatei mit mehreren Zeilen und ruft für einen gegeben Suchbegriff (`pattern`) die Suchmethode auf. Anschließend werden alle gefundenen Vorkommen mit Zeilennummer und Position innerhalb der Zeile ausgegeben. Sie können die Ausgabe der Anzahl der gefundenen Vorkommen mit dem im Programmrahmen angegebenen Hinweis auf die erwarteten Vorkommen kontrollieren.

a) Berechnen Sie in der Methode `search` die über die gesamte Suche konstant bleibende Variable `p_pow`, die $p^{patternLength}$ entspricht, wobei – wie in der Vorlesung behandelt – bei jeder Multiplikation der Wertebereich sichergestellt wird.

b) Implementieren Sie ebenfalls in `search` die Berechnung des Hashwertes des Patterns (`patternHash`) sowie den Hashwert des initialen Substrings (`substringHash`) am Anfang des Textes.

c) Implementieren Sie die Methode `nextHash`, die auf Basis des vorherigen Hashwertes, `p_pow` und den Randbuchstaben des sich verschiebenden Suchfensters einen neuen Hashwert bestimmt.

d) Vervollständigen Sie nun die Methode `search` um den schrittweisen Abgleich des Textes mit dem Pattern und stellen Sie bei Hash-Gleichheit sicher, dass Textausschnitt und Pattern tatsächlich übereinstimmen. Berechnen Sie zudem den Hash des folgenden Suchfensters, wenn das erforderlich ist.

Aufgabe 7.3: C++-Programmiersprachtest (10 Punkte $^{2+2+2+2+2}$)

Sehen Sie sich die nachfolgenden Quellcodes an und lösen Sie die zugehörigen Aufgaben theoretisch. Bei den Multiple-Choice-Aufgaben sind keine, eine, mehrere oder alle Aussagen wahr. Erstellen Sie zur Abgabe **genau eine Datei „`7_3.txt`“ mit fünf Zeilen**, in der Sie pro Zeile ohne Trennzeichen die Antworten der jeweiligen Aufgabe angeben. Geben Sie in Reihenfolge der Antworten jeweils an, ob diese wahr (`W`) oder falsch (`F`) ist[2].

a) Variablendeklaration (Programmcode 1):

1. Die Zeilen 2 und 4 führen zu Laufzeitfehlern.
2. In Zeile 3 gibt es einen Compilerfehler.
3. Zeile 5 erzeugt einen string mit dem Wert „Test“.
4. Das Programm wird ohne Fehler kompiliert.

```
1  int main(int argc, char** argv) {
2      int var1 = 32.4;
3      double var2 = 10.0f;
4      char var3 = "Test";
5      char* var4 = { 'T', 'e', 's', 't' };
6
7      return 0;
8  }
```

b) Const-Schlüsselwort (Programmcode 2):

1. Das Programm wird ohne Fehler kompiliert.
2. In den Zeilen 2 und 12 gibt es eine Compilerwarnung.
3. In Zeile 7 gibt es einen Compilerfehler.
4. Der Versuch, die Variable `test2` in Zeile 22 zu ändern, führt zu einem Compilerfehler.

```
1  const int foo1(float test) {
2      int quadrat = test*test;
3      return quadrat;
4  }
5
6  const float foo2(const float test) {
7      test = test / 2.0;
8      return test;
9  }
10
11 const float foo3(const float test) {
12     return test*2.0;
13 }
14
15 int main(int argc, char** argv) {
16     float test1 = 2.0;
17     test1 = foo1(test1);
18
19     test1 = foo2(test1);
20
21     auto test2 = foo3(test1);
22     test2 += 2.0;
23
24     return 0;
25 }
```

[2] Wenn für eine Aufgabe mit sechs möglichen Antworten die Antworten abwechselnd wahr und falsch wären, müsste die Zeile beispielsweise so aussehen: `WFWFWF`

c) Sortierung von Vektoren (Programmcode 3):

1. Das Programm wird ohne Fehler kompiliert.
2. Die Funktion `selectionSort()` verursacht einen Laufzeitfehler und terminiert nicht.
3. Die Funktion `selectionSort()` terminiert und das Ergebnis ist der Eingabevektor mit aufsteigend sortierten Werten.
4. Die Funktion `selectionSort()` terminiert und das Ergebnis ist der Eingabevektor mit absteigend sortierten Werten.

```cpp
#include <iostream>
#include <vector>

using namespace std;

void selectionSort(vector<int>& A) {
    const int N = A.size();
    for (int j = N; j >= 0; --j) {
        // search maximum in the range [0, j]
        int m = j;
        for (int i = 0; i<j; ++i) {
            if (A[i]<A[m]) m = i;
        }
        // swap the maximal element to the j-th position
        swap(A[j], A[m]);
    }
}

int main(int argc, char** argv) {
    vector<int> test = { 10, 20, 1, 7, 13, 6, 18 };

    for (auto &t : test)
    {
        cout << t << " ";
    }

    cout << endl;

    selectionSort(test);

    for (auto &t : test)
    {
        cout << t << " ";
    }

    return 0;
}
```

d) Klassen (Programmcode 4):

1. Das Programm wird ohne Fehler kompiliert.
2. Die Zeilen 20 und 21 verursachen einen Laufzeitfehler.
3. Die Zeilen 23 und 24 sind korrekt.
4. Die Klasse Test ist kompilierbar.

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  class Test{
7  public:
8      const string getLabel() const {
9          return label_;
10     }
11
12 private:
13     Test() {};
14     Test(string label) : label_(label) {};
15     ~Test() {};
16     string label_;
17 };
18
19 int main(int argc, char** argv) {
20     Test* t1 = new Test("Test1");
21     Test t2("Test2");
22
23     cout << t1.getLabel() << endl;
24     cout << t2->getLabel() << endl;
25
26     return 0;
27 }
```

e) Remove Funktion (Programmcode 5):

1. Das Programm wird ohne Fehler kompiliert.
2. Die Ausgabe des Programms ist `17 9 14 9`.
3. Die Ausgabe des Programms ist `17 89 14 89`.
4. Die Ausgabe des Programms ist `17 89 14`.

```
1  #include <iostream>
2  #include <list>
3
4  using namespace std;
5
6  int main()
7  {
8      int myints[] = { 17, 89, 7, 14, 89 };
9      list<int> mylist(myints, myints + 5);
10
11     mylist.remove(8);
12     mylist.remove(7);
13
14     for (auto it = mylist.begin(); it != mylist.end(); ++it)
15         cout << ' ' << *it;
16
17     return 0;
18 }
```

Allgemeine Hinweise zur Bearbeitung und Abgabe

- Die Aufgaben können allein oder zu zweit bearbeitet werden. Je Gruppe ist nur eine Abgabe notwendig.
- Bitte reichen Sie Ihre Lösungen bis spätestens **Donnerstag, den 30. Juli um 12:00** ein.
- Die Implementierung kann auf einer üblichen Plattform (Windows, Linux, OS X) erfolgen, darf aber keine plattformspezifischen Elemente enthalten, d. h. die Implementierung soll plattformunabhängig entwickelt werden.
- Bestehen weitere Fragen und Probleme, kontaktieren Sie den Übungsleiter oder nutzen Sie das Forum im Moodle.
- Archivieren Sie zur Abgabe Ihren bearbeiteten Programmrahmen und die Lösungen der theoretischen Aufgaben als Zip-Archiv und ergänzen Sie Ihre Namen im Bezeichner des Zip-Archivs im folgenden Format: **`uebung7_vorname1_nachname1_vorname2_nachname2.zip`**. Beachten Sie, dass dabei nur die vollständigen Lösungen sowie eventuelle Zusatzdaten gepackt werden (alle Dateien, die im gegebenen Programmrahmen vorhanden waren). Laden Sie keine Kompilate und temporären Dateien (*.obj, *.pdb, *.ilk, *.ncb, *.exe, etc.) hoch. Testen Sie vor dem Hochladen, ob die Abgabe fehlerfrei kompiliert und ausgeführt werden kann.
- Reichen Sie Ihr Zip-Archiv im Moodle ein:
`https://moodle.hpi3d.de/course/view.php?id=137`.