

Programmierübungen

Wintersemester 2006/2007

12. Übungsblatt

26. Januar 2007

Abgabe bis Samstag, 3. Februar 23:59 Uhr.

Die Abgabe Ihrer Bearbeitung können Sie im eClaus-System durchführen. Erarbeiten Sie Lösungsideen zu den Aufgaben möglichst in Kleingruppen. Es wird jedoch von Ihnen erwartet, dass jeder Teilnehmer eine eigene Lösung abgibt. Sollten kopierte Quelltexte abgegeben werden, so werden grundsätzlich alle Kopien mit 0 Punkten bewertet. In den Vortragsfolien der Programmierübungen oder im Skript zur Einführung in die Informatik abgedruckte Quelltexte können verwendet werden, müssen aber der Programmierrichtlinie entsprechend formatiert und kommentiert werden.

Beachten Sie die Programmierrichtlinie und kommentieren Sie Ihren Quelltext. Dokumentieren Sie unbedingt Ihre Lösungsidee in den Quelltext-Kommentaren.

<http://www.iste.uni-stuttgart.de/ps/Lehre/WS0607/inf-prokurs>

Am Montag 5.2.2007, 8:00 Uhr in V38.01 wird das Aufgabenblatt kurz vorgestellt und Sie haben Gelegenheit Fragen zu den Aufgaben zu stellen.

Bitte beachten Sie, dass zum Scheinerwerb unter anderem auf den letzten 4 Aufgabenblättern 50 % der Punkte erreicht werden müssen. Das letzte Aufgabenblatt wird Nummer 13 sein.

Aufgabe 12.1: Generische Pakete

(8 Punkte)

Im Kapitel 3.7.6 des Skripts zur „Einführung in die Informatik 1“ werden Suchbäume definiert. Erstellen Sie ein generisches Paket, das Suchbäume durch einen abstrakten Datentyp `Tree` anbietet. Als generische Parameter sollen ein Element-Typ `Element_Type`, eine Vergleichsfunktion `"<"` und eine Prozedur `Visit` verwendet werden. Die Deklaration des Pakets sieht also wie folgt aus (die Deklarationen innerhalb des Pakets sind hier nicht angegeben):

```
generic
  type Element_Type is private;

  with function "<"
    (Left  : in Element_Type;
     Right : in Element_Type)
    return Boolean;

  with procedure Visit
    (Element : in Element_Type);

package Search_Trees is

  -- ...

end Search_Trees;
```

Fügen Sie Unterprogramme für folgende Operationen ein:

- Einfügen eines neuen Elements in einen Suchbaum; dieses Unterprogramm soll zusätzlich berechnen, wie viele Vergleiche zum Einfügen benötigt wurden,
- Abfrage, ob ein bestimmtes Element in einem Suchbaum enthalten ist; dieses Unterprogramm soll zusätzlich berechnen, wie viele Vergleiche zur Suche benötigt wurden,
- Berechnen der Höhe eines Suchbaums,
- Durchlauf durch einen Suchbaum in Inorder-Reihenfolge, wobei die Prozedur Visit mit jedem Element des Suchbaums als aktueller Parameter aufgerufen wird.

In den zu berechnenden Anzahlen von Vergleichen zählt jeder Aufruf des generischen Parameter-Operators "<" sowie jede Verwendung des Operators "=" für den Typ Element_Type. Sonstige Vergleiche, z. B. Tests auf null-Zeiger werden nicht mitgezählt.

Aufgabe 12.2: Generische Instanz

(6 Punkte)

Erstellen Sie ein Programm Date_Manager. Das Programm verwendet eine Instanz des generischen Pakets Search_Trees aus der vorigen Aufgabe. Als Element_Type soll der Typ Ada.Calendar.Time verwendet werden. Als Visit soll eine Prozedur übergeben werden, die ein Datum gefolgt von zwei Leerzeichen ausgibt.

Das Programm Date_Manager soll die Datei dates.txt einlesen und die darin enthaltenen Daten der Reihe nach in einen Suchbaum einfügen. Nach jedem Einfügen soll die Höhe des Baums und der gesamte Inhalt sortiert ausgegeben werden. Schließlich soll das Programm den Benutzer zur Eingabe eines weiteren Datums auffordern. Das Programm prüft, ob dieses Datum in dem Baum enthalten ist (ohne es einzufügen), gibt das Resultat aus und teilt die Anzahl der benötigten Vergleiche mit.

Format der Datei dates.txt: In jeder Zeile steht genau ein Datum im üblichen Format:

```
1.1.2007
2.5.2004
7.3.2008
1.12.1993
17.2.1970
2.5.2004
5.6.2005
```

Die Ausgabe des Programms könnte so aussehen (Hinweis: die Anzahl der Vergleiche kann gegenüber Ihrer Implementierung variieren, die Höhe des resultierenden Baums jedoch nicht):

```
Eingefügt: 01.01.2007, Anzahl Vergleiche: 0
Höhe des Baums: 0, Inhalt: 01.01.2007
```

```
Eingefügt: 02.05.2004, Anzahl Vergleiche: 1
Höhe des Baums: 1, Inhalt: 02.05.2004 01.01.2007
```

```
Eingefügt: 07.03.2008, Anzahl Vergleiche: 1
```

Höhe des Baums: 1, Inhalt: 02.05.2004 01.01.2007 07.03.2008

Eingefügt: 01.12.1993, Anzahl Vergleiche: 2

Höhe des Baums: 2, Inhalt: 01.12.1993 02.05.2004 01.01.2007
07.03.2008

Eingefügt: 17.02.1970, Anzahl Vergleiche: 3

Höhe des Baums: 3, Inhalt: 17.02.1970 01.12.1993 02.05.2004
01.01.2007 07.03.2008

Eingefügt: 02.05.2004, Anzahl Vergleiche: 2

Höhe des Baums: 3, Inhalt: 17.02.1970 01.12.1993 02.05.2004
02.05.2004 01.01.2007 07.03.2008

Eingefügt: 05.06.2005, Anzahl Vergleiche: 3

Höhe des Baums: 3, Inhalt: 17.02.1970 01.12.1993 02.05.2004
02.05.2004 05.06.2005 01.01.2007 07.03.2008

Suchdatum: 1.12.1993

Datum ist im Baum enthalten, Anzahl Vergleiche: 4

Suchdatum: 30.12.2006

Datum nicht enthalten, Anzahl Vergleiche: 7

Suchdatum: 6.3.2008

Datum nicht enthalten, Anzahl Vergleiche: 3

Hinweis: Um das Einlesen und Ausgeben eines Datums zu realisieren bietet es sich an, Quelltext aus dem Paket Dates des letzten Aufgabenblatts anzupassen und wiederzuverwenden.

Aufgabe 12.3: Erreichbarkeit in Graphen

(6 Punkte)

Auf dem vergangenen Übungsblatt haben Sie Funktionalität implementiert, um einen gerichteten Graphen aus einer Datei zu laden und als Adjazenzlisten zu verwalten. Sollten Sie Übungsblatt 11 nicht bearbeitet haben, so können Sie Ada-Quelltext mit dieser Funktionalität ab Dienstag, 30.01.2007 von der Webseite herunterladen.

Schreiben Sie ein Programm Reachability, das zunächst einen Graph aus der Datei graph.txt einliest und danach den Benutzer zur Eingabe einer Knoten-Id auffordert. Diese Knoten-Id identifiziert einen Start-Knoten. Verwenden Sie den Algorithmus zum Graphendurchlauf um von diesem Start-Knoten aus alle Knoten T_i aufzulisten, die von S aus erreichbar sind.

Beispiel für die Datei graph.txt:

- A (1) C
- A (2) E
- B (2) E
- C (3) E
- E (5) D

Beispiel für einen Programmablauf:

Bitte Startknoten eingeben: A

Erreichbare Knoten: C, E, D

Aufgabe 12.4: Optimale Wege

(bis zu 2 Zusatzpunkte)

Für diese Aufgabe können zwei Zusatzpunkte vergeben werden, jedoch kann die Punktzahl für die Bearbeitung des gesamten Übungsblatts 20 nicht übersteigen.

Erweitern Sie Ihr Programm aus der vorigen Aufgabe, so dass zu jedem erreichbaren Knoten T_i zusätzlich das Gewicht des optimalen Pfads $((v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n))$ mit $v_1 = S$ und $v_n = T_i$ angegeben wird.

Das Gewicht eines Pfads ist die Summe der Gewichte aller Kanten, die auf dem Pfad liegen. Ein Pfad zwischen zwei Knoten S und T_i ist optimal, falls er unter allen Pfaden zwischen S und T_i minimales Gewicht hat.

Beispiel für einen Programmablauf:

Bitte Startknoten eingeben: A

Erreichbare Knoten: C: 1, E: 2, D: 7