

Programmierübungen 2

Sommersemester 2008

4. Übungsblatt

29. April 2008

Abgabe bis Montag, 19. Mai 14:00 Uhr

Auf 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.

Bitte bearbeiten Sie die Übungsaufgaben in Kleingruppen mit bis zu 3 Teilnehmern. Alle Teilnehmer einer Kleingruppe müssen in der selben Übungsgruppe eingetragen sein. Ihre Abgabe muss eine Datei `contributions.txt` enthalten, in der aufgelistet ist welche Anteile der Bearbeitung von jedem einzelnen Teilnehmer erstellt wurden. Jeder Teilnehmer muss die Funktionsweise aller Teile der Bearbeitung erläutern können. 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/SS2008/inf-prokurs>

Aufgabe 4.1: Bit-Vektoren

(8 Punkte)

Aus der Vorlesung Einführung in die Informatik 2 kennen Sie Bitvektoren. Mit Hilfe von Bitvektoren lassen sich Teilmengen einer bestimmten Grundmenge (Universum) effizient darstellen. Mögliche Anwendungsfälle sind z. B.:

- Darstellen der Teilmenge aller Studierenden, die an einer bestimmten Lehrveranstaltung teilnehmen
- Auswahl einer bestimmten Menge von Ausstattungsmerkmalen eines neuen Computers aus der Menge aller erhältlichen Komponenten

In Ada könnte ein solcher Bitvektor offensichtlich durch einen Array repräsentiert werden:

```
type Bit_Vector is array (Index_Range range <>) of Boolean;
```

Der Einsatz dieser Technik wirft in der Praxis interessante Fragen auf, die auf diesem Aufgabenblatt untersucht werden sollen. Im Besonderen wird selten bereits vor Beginn der Ausführung eines Programms bekannt sein, wie viele Elemente die Grundmenge tatsächlich enthält. Ferner muss häufig für ein bestimmtes Element der Grundmenge der zugehörige Index in dem Bit-Vektor einer Teilmenge bestimmt werden.

Als Grundmenge soll für diese Aufgabe eine Textdatei mit Namen dienen. Jede Zeile der Datei soll genau einen Namen enthalten und kein Name soll mehrfach vorkommen. Entwerfen Sie ein Paket, das diese Funktionalität anbietet:

- Lesen der Grundmenge aus einer Textdatei
- Erzeugen von mehreren verschiedenen Teilmengen, repräsentiert durch Bit-Vektoren
- Hinzufügen eines bestimmten Elements zu einer bestimmten Teilmenge

- Löschen eines bestimmten Elements aus einer bestimmten Teilmenge
- Aufzählen aller Elemente einer Teilmenge (z. B. durch einen Cursor oder durch Rückgabe eines entsprechenden Arrays)
- Bilden der Vereinigung, des Schnitts sowie der Differenz zweier bestimmter Teilmengen

Hinweis Um den benötigten Speicherplatz für Objekte des Typs `Bit_Vector` zu minimieren kann nach der Deklaration des Typs ein `pragma Pack (Bit_Vector)`; angegeben werden. Eine Komponente (vom Typ `Boolean`) benötigt dann nur 1 Bit Speicherplatz.

Aufgabe 4.2: Benutzungs-Schnittstelle (7 Punkte)

Implementieren Sie ein Programm, das das Paket aus der vorigen Aufgabe verwendet, um eine Textdatei mit Namen einzulesen und interaktiv mehrere Teilmengen zu erzeugen. Die Textdatei soll durch Benutzereingabe oder per Kommandozeilen-Parameter angegeben werden können. Analog zu den Fähigkeiten des Pakets soll der Benutzer die Möglichkeit haben, leere Teilmengen zu erzeugen, eine Teilmenge zu wählen und einen Namen hinzuzufügen/zu löschen, zwei Teilmengen zu vereinigen/zu scheiden/ihre Differenz zu bilden oder den Inhalt einer Teilmenge anzeigen zu lassen. Vereinigung, Schnitt sowie Differenz zweier Mengen soll jeweils eine neue Menge erzeugen. Teilmengen sollen durch Zahlen identifiziert werden.

Bei Eingabe nicht existierender Teilmengen oder Namen sollen entsprechende Fehlermeldungen ausgegeben werden.

Beispiel (Textdatei `set.txt`):

```
A
B
C
D
E
```

Beispiel (möglicher Programmablauf):

```
$ ./setoperations set.txt
Grundmenge enthält 5 Elemente.
```

Keine Teilmengen.

Mögliche Aktionen:

```
(E)rzeugen einer Teilmenge,
(B)earbeiten einer Teilmenge,
(M)engenoperation
Beende (N)
```

Auswahl? E

Teilmenge 1 erzeugt.

Teilmengen:

```
1: {}
```

Mögliche Aktionen:

(E)rzeugen einer Teilmenge,
(B)earbeiten einer Teilmenge,
(M)engenoperation
Auswahl? B
Mögliche Aktionen:
(H)inzufügen eines Namens,
(L)öschen eines Namens,
Auswahl? H
Welchen Namen? B

Teil Mengen:
1: { B }
Mögliche Aktionen:
(E)rzeugen einer Teilmenge,
(B)earbeiten einer Teilmenge,
(M)engenoperation
Beende (N)
Auswahl? B
Mögliche Aktionen:
(H)inzufügen eines Namens,
(L)öschen eines Namens,
Auswahl? H
Welchen Namen? E

Teil Mengen:
1: { B, E }
Mögliche Aktionen:
(E)rzeugen einer Teilmenge,
(B)earbeiten einer Teilmenge,
(M)engenoperation
Beende (N)
Auswahl? E
Teilmenge 2 erzeugt.

Teil Mengen:
1: { B, E }
2: {}
Mögliche Aktionen:
(E)rzeugen einer Teilmenge,
(B)earbeiten einer Teilmenge,
(M)engenoperation
Beende (N)
Auswahl? M
Mögliche Aktionen:
(V)ereinigung
(S)chnitt
(D)ifferenz
Auswahl? V

linke Menge: 2
rechte Menge: 1
Teilmenge 3 erzeugt.

Teilmengen:

1: { B, E }
2: {}
3: { B, E }

Mögliche Aktionen:

(E)rzeugen einer Teilmenge,
(B)earbeiten einer Teilmenge,
(M)engenoperation

Beende (N)

Auswahl? N

Aufgabe 4.3: Ausgabe

(5 Punkte)

Fügen Sie dem Programm Funktionalität hinzu, um die Teilmengen ausgeben zu lassen. Die Ausgabe soll im dot-Format erfolgen und soll in eine Datei gespeichert werden. Im Grundstudiumspool unter Linux ist das Werkzeug „doty“ installiert, mit dessen Hilfe diese Ausgabe gelesen werden kann, um eine Visualisierung als Graph anzeigen zu lassen, siehe Abbildung 1. Jedes Element der Grundmenge soll als ein Knoten im Graph dargestellt werden und jede Teilmenge soll alle möglichen Kanten zwischen ihren Elementen erzeugen. Die Kanten sollen mit der Nummer der Teilmenge beschriftet sein und in einer einheitlichen Farbe eingefärbt sein.

Mit dem Kommando `man dot` können Sie Dokumentation zu dem dot-Format abrufen. Die wichtigsten Eigenschaften können Sie auch dem folgenden Beispiel entnehmen.

Beispiel (Teilmengen, Grundmenge sei { A, B, C, D, E, F }):

```
1: { B, E }  
2: {}  
3: { B, E }  
4: { C, D, E, F }
```

Beispiel (Ausgabe im dot-Format `output.dot`):

```
graph "Sets" {  
  
  A;  
  B;  
  C;  
  D;  
  E;  
  F;  
  
  B -- E [color="red", label="1"];  
  
  B -- E [color="green", label="3"];
```

```

C -- D [color="blue", label="4"];
C -- E [color="blue", label="4"];
C -- F [color="blue", label="4"];
D -- E [color="blue", label="4"];
D -- F [color="blue", label="4"];
E -- F [color="blue", label="4"];
}

```

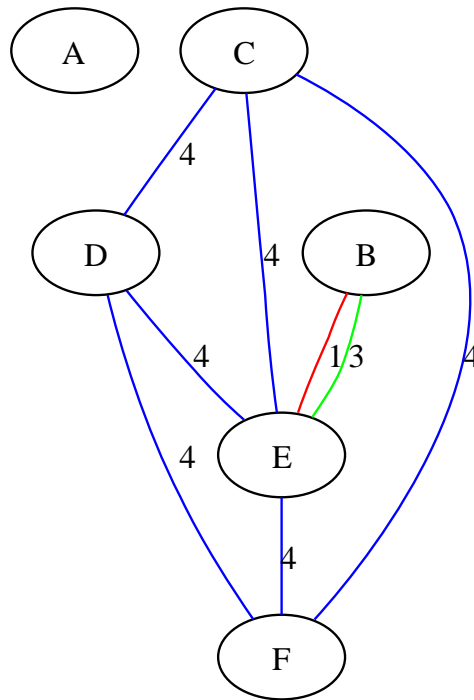


Abbildung 1: Graphische Darstellung von `output.dot`

Hinweis₁ Da mehrere Kanten zwischen einem Paar zweier Knoten verlaufen können spricht man von einem Multigraph.

Hinweis₂ Mögliche Werte für Farben von Kanten sind: `white`, `black`, `red`, `green`, `blue`, `yellow`, `magenta`, `cyan`, `burlywood`. Sollten diese Farben nicht ausreichen, so können verschiedene Teilmengen die gleiche Farbe erhalten.