Programmierübungen

Wintersemester 2006/2007

3. Übungsblatt
10. November 2006

Abgabe bis Samstag, 18. November 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 Vortrags-Folien 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`

Aufgabe 3.1: Programm-Verständnis (4 Punkte)

Einer Ihrer Kollegen hat gestern an einem Programm gearbeitet, das Texte invertiert (das Spiegelwort berechnet). Sein Programm soll heute ausgeliefert werden, doch Ihr Kollege ist bereits zu einer Weltreise aufgebrochen und nicht mehr erreichbar. Sie wissen aber, dass gestern die Kaffeemaschine defekt war und haben deshalb kein Vertrauen in seine Arbeitsergebnisse.

Suchen Sie Fehler und korrigieren Sie den Quelltext (die Datei mirror.adb kann auch von der Webseite herunter geladen werden):

```
with Ada.Text_IO;
procedure Mirror is Buffer:String(1..1_001); use
Ada.Text_IO; Last:Natural;
function Invert (T: in String) return String
is begin return T(t'Last) &Invert(T(T'First..T'Last-1));end;
begin Put("zu invertierender Text: ");Get_Line(Buffer,Last);
Put_Line(invert(Buffer(1..Last)));
end Mirror;
```

Es ist bekannt, dass die Texte, die mit diesem Programm invertiert werden sollen, nie länger als 1000 Zeichen sind. Sollte das Programm eine Beschränkung auf Texte dieser Länge enthalten, so wäre das kein Fehler.

Formatieren Sie den Quelltext um, so dass er der Programmierrichtlinie entspricht. Korrigieren Sie den Quelltext, löschen Sie jedoch so wenig wie möglich heraus und fügen Sie so wenig wie möglich hinzu.

Ausführungsbeispiel (so soll es nach Ihren Änderungen funktionieren):

```
zu invertierender Text: abcd
dcba
```

Hinweis: Vollziehen Sie den Ablauf des Programms auf einem Blatt Papier nach, um es zu verstehen.

Aufgabe 3.2: Freitag der 13. (4 Punkte)

Schreiben Sie ein Programm, das alle Freitage eines Jahres auflistet, die auf den 13. eines Monats fallen. Als Eingabe können Sie vom Benutzer höchstens drei Daten erfragen:

- die Jahreszahl,
- den Wochentag des 1.1. in diesem Jahr und
- die Information, ob das Jahr ein Schaltjahr ist.

Als Ausgabe soll Ihr Programm eine Liste von Daten ausgeben, z. B. für das Jahr 2006:

```
13. 1.2006
13.10.2006
```

Überlegen Sie selbst wie die Interaktion mit dem Benutzer ablaufen soll.

Aufgabe 3.3: Goldbachsche Vermutung (12 Punkte)

In Wikipedia wird die Goldbachsche Vermutung wie folgt definiert: „Jede gerade Zahl größer als 2 kann als Summe zweier Primzahlen geschrieben werden.“ Diese Vermutung ist bislang noch nicht bewiesen worden.

In dieser Aufgabe soll ein Programm entwickelt werden, um zu überprüfen, ob die Vermutung für Zahlen in einem bestimmten Bereich zutrifft oder nicht.

Das Programm soll den folgenden Ablauf haben:

1. Das Programm meldet sich mit:

   ```
   Goldbachsche Vermutung beweisen
   von:
   ```

2. Die Benutzerin gibt eine natürliche Zahl m ein.

3. Das Programm gibt den Text aus:

   ```
   bis:
   ```

4. Die Benutzerin gibt eine natürliche Zahl n ein.

5. Das Programm listet für jede gerade Zahl i mit $m \leq i \leq n$ in nach i sortierter Reihenfolge folgende Ausgabe auf:

   a) Falls die Goldbachsche Vermutung für die Zahl i zutrifft:
   i = p_1 + p_2, wobei p_1 und p_2 Primzahlen sind.

   b) Falls die Vermutung nicht zutrifft:
   i: G.V. trifft nicht zu.

Aufgabenstellung:
Erstellen Sie das folgende Paket, fügen Sie auch die üblichen Kommentare hinzu:

```
package Goldbach is

   subtype Prime_Number is Integer range 2 .. Positive'Last;

   type Prime_Field is array (Positive range <>) of Boolean;

   procedure Eratosthenes
     (Sieve : in out Prime_Field);

   procedure Prove_Conjecture
     (Number   : in     Positive;
      Is_Prime : in     Prime_Field;
      First    :    out Prime_Number;
      Second   :    out Prime_Number;
      Found    :    out Boolean);

end Goldbach;
```

Erstellen Sie den Body zu diesem Paket. Der Body darf natürlich beliebige weitere Deklarationen enthalten. Schreiben Sie ein Haupt-Programm „GC_Prover“ und implementieren Sie es nach folgendem Schema:

1. Die Benutzer-Interaktion wird durchgeführt (siehe oben).

2. Es wird ein Feld des Typs Prime_Field in geeigneter Größe erzeugt.
   Hinweis: Sie können ein Block-Statement mit Deklarationsteil verwenden.

3. Es wird die Prozedur Eratosthenes aufgerufen, um die Komponenten dieses Felds sinnvoll zu belegen. Die Komponenten sollen an jeder Index-Position, die eine Primzahl ist, den Wert True enthalten, an allen anderen den Wert False.

   Verwenden Sie für die Prozedur Eratosthenes den Algorithmus „Sieb des Eratosthenes“ (nach dem gleichnamigen griechischen Mathematiker): es werden zunächst alle Komponenten des Felds auf True gesetzt. Dann werden die Zahlen 2, 3, 4, ... aufgezählt und alle Komponenten, deren Index ein Vielfaches dieser Zahlen ist, auf False gesetzt.

4. Es wird für alle relevanten geraden Zahlen jeweils Prove_Conjecture aufgerufen. Diese Prozedur nimmt als Eingabe Number eine gerade Zahl und das vorberechnete Feld Is_Prime. Die Prozedur setzt den Ausgabe-Parameter Found auf True, falls eine Zerlegung der geraden Zahl in die Summe zweier Primzahlen existiert, andernfalls auf False. Falls die zwei Primzahlen existieren, so speichert die Prozedur diese in den Ausgabe-Parametern First und Second. Existieren mehrere solche Primzahlpaare, so kann ein beliebiges gewählt werden.