Friedrich-Schiller-Universität Jena **Fakultät für Mathematik und Informatik** **Institut für Informatik** **Lehrstuhl für Programmiersprachen und Compiler**	**Höhere Programmierung** **SS 2001**	**Praktikumsblatt 4** **Ausgabe: 14.05.2001** **Termin: 17.05.2001**

Aufgabe 1: File Ein-/Ausgabe

In dieser Aufgabe geht es darum, ein Textfile zu kopieren. Das Textfile soll ein vom Benutzer bestimmbares File sein und der Name des Ausgabefiles soll "kopie.cpy" sein.
Für diese Aufgabe muss die Ein- und Ausgabe erweitert werden. Bisher wurde die Ein- und Ausgabe für die Standardein-/-ausgabe benutzt. Als erstes benötigen wir den im Paket `Ada.Text_Io` definierten Typ `File_Type`, mit diesem Typ wird die Dateivariable definiert.

```
<Dateivariable>: Ada.Text_Io.File_type;
```

Um aus einer existierenden Datei lesen zu können, muss erst zwischen der <Dateivariable> und der vom Betriebssystem verwalteten Datei eine Verbindung hergestellt werden. Dies geschieht mit dem Prozeduraufruf

```
Ada.Text_Io.Open(File => <Dateivariable>,
                 Mode => <Dateimodus>,
                 Name => <Dateiname>)
```

Es gibt noch zwei weitere Parameter, die erklärt werden müssen. Der erste ist <Dateimodus>, er gibt an, in welchem Modus die Datei geöffnet wird. Der <Dateimodus> ist vom Typ `Ada.text_io.File_Mode`. Seine Definition sieht wie folgt aus:

```
type File_Mode is (In_File, Out_File, Append_File);
```

Die Bedeutung der drei verschiedene Modi ist:

1. In_File – nur lesen von Anfang bis Ende,
2. Out_File – nur schreiben vom Anfang der Datei an, dabei wird beim ersten Schreibzugriff der gesamte Inhalt des Datei gelöscht, und
3. Append_File – nur schreiben ab Ende.

Um den Modus direkt übergeben zu können muss der Paketname vorangestellt werden. Zum Beispiel: `Ada.Text_Io.In_File`.
Der zweite unbekannte Parameter ist der <Dateiname>, er ist vom Typ String und gibt den Namen der externen Datei an, z.B. "program1.ada".
Die Datei mit dem Namen "kopie.cpy" soll neu erstellt werden. Um eine neue externe Datei im Betriebssystem zu erzeugen benötigen wir den folgenden Prozeduraufruf:

```
Ada.text_Io.Create(File => <Dateivariable>, Name => <Dateiname>)
```

Mit diesem Aufruf wird die neu erzeugte Datei zum Schreiben, d.h. im Modus `Ada.Text_Io.Out_File`, geöffnet. Existierte bereits eine Datei mit dem Namen <Dateiname>, so wird diese gelöscht.
Mit dem Prozeduraufruf

```
Ada.Text_Io.Close(<Dateivariable>)
```

wird eine Datei geschlossen, das heißt die Verbindung zwischen <Dateivariable> und der externen Datei wird gelöst.

Eine externe Datei, die nicht mehr benötigt wird, kann auch gelöscht werden. Dafür gibt es die Prozedur `Ada.Text_Io.Delete` mit folgendem Aufruf:

```
Ada.Text_Io.Delete(<Dateivariable>)
```

Wie können nun die Daten aus dem einen File gelesen und in das andere geschrieben werden? Dafür gibt es Prozeduren in dem Paket Ada.text_Io, die den Prozeduren für die Standardeingabe/-ausgabe ähnlich sind. Der einzige Unterschied ist ein weiterer Parameter, der das zu verwendende File angibt. Im folgenden nun ein paar Beispiele.

```
Ada.Text_Io.Get_Line(<Dateivariable>, <Stringvariable>, <Stringlaenge>)
Ada.Text_Io.Get(<Dateivariable>, <Charactervariable>)

Ada.Text_Io.Put_Line(<Dateivariable>, <Stringvariable>)
Ada.Text_io.Put(<Dateivariable>, <Charactervariable>)
```

`Get_Line` liest eine Zeile aus dem File <Dateivariable> und speichert sie in die <Stringvariable>. Im Parameter <Stringlaenge> wird die Länge des gelesenen String gespeichert, er hat den Typ Natural. Durch `Get_Line` werden höchstens so viele Zeichen gelesen, wie in die <Stringvariable> passen. `Put_Line` schreibt den Inhalt der <Stringvariable> in die <Dateivariable> an die aktuelle Position.
Es kommt zu der Ausnahmen `Ada.Io_Exceptions.Mode_Error`, wenn in ein zum Lesen geöffnetes File geschrieben oder aus einem zum Schreiben geöffnetes File gelesen wird.
Wir können also jetzt ein File zum Lesen/Schreiben öffnen oder zum Schreiben neu erzeugen. Mit den Prozeduren `Get_Line`, `Get` oder `Put_Line`, `Put` können jeweils Daten gelesen oder geschrieben werden. Bleibt noch zu klären, woran das Ende eines Files erkannt wird. Wird aus einem File gelesen, das keine Daten mehr enthält – sprich das Ende des Files wurde erreicht –, dann wird die Ausnahme `Ada.Io_Exceptions.End_Error` ausgelöst. Um dieser Ausnahme vorzubeugen, gibt es den Funktionsaufruf

```
End_Of_File(<Filevariable>)
```

der einen booleschen Wert zurückgibt. Ist der Wert True, so ist das Ende des Files erreicht, sonst ist der zurückgegebene Wert False.
Mit dieser Einführung sollte es möglich sein, die Aufgabe zu lösen. Testen sie ihr Programm anhand der Quelldatei des in dieser Aufgabe erstellten Programms.

Aufgabe 2: Aufsummieren mit der grafischen Benutzeroberfläche

Im ersten Praktikumsblatt wurde das erste Mal mit der grafischen Benutzungsoberfläche gearbeitet. Auch auf diesem Praktikumsblatt soll wieder mit dem GUI Builder gearbeitet werden. Dieses Mal ist die Aufgabe, ein Programm zu schreiben, das einzelne Zahlen einliest und aufsummiert. Erstellen Sie ein Fenster mit den folgenden Komponenten:

1. Ein Beschriftungsobjekt,
2. Ein Textfeld, wo die Zahlen eingegeben werden,
3. Einem Button 'Addiere', der die eingegebene Zahl zur Summe dazuaddiert und
4. einem Button 'Reset', der die Summe auf 0 zurücksetzt.

Das Textfeld enthält einen String. Der Funktionsaufruf

```
text(<Windowname>.<Textfeldname>);
```

gibt den Inhalt des Textfeldes als String zurück. <Windowname> ist der Name des Fensters und ist am Anfang `Window1`. Mit dem Attribut `Integer'Value(text(<Windowname>.<Textfeldname>))` erhält man den zugehörigen Integerwert.

- Leeren Sie nach jeder Addition das Textfeld.
- Vermeiden Sie durch Fallunterscheidungen mögliche Überläufe.
- Die Summe soll nach jeder Addition in einer Messagebox angezeigt werden. Die Prozedur dafür ist

  ```
  MsgBox(<Anzuzeigende Nachricht>);
  ```

  <Anzuzeigende Nachricht> ist ein String der beispielsweise wie folgt aussehen kann:

  ```
  "Summe = " & Integer'Image(Summe)
  ```

- Fehler sind auch auf diesem Wege anzuzeigen. Die Fehlermeldung sollte aussagekräftig gewählt werden.