| | | |
|---|---|---|
| **Friedrich-Schiller-Universität Jena**<br>**Fakultät für Mathematik und Informatik**<br>**Institut für Informatik**<br>**Lehrstuhl für Programmiersprachen und Compiler** | **Höhere Programmierung**<br>**SS 2001** | **Praktikumsblatt Nr. 2**<br><br>**Ausgabe: 17.04.2001**<br>**Termin: 19.04.2001** |

## Aufgabe1: Ausnahmebehandlung

In dieser Aufgabe wird das Verhalten eines Programms mit und ohne Ausnahmebehandlung untersucht. Für diesen Fall soll ein Programm erstellt werden, das zwei Operanden vom Typ Integer einliest und dann nacheinander die vier Grundrechenarten $+_{bin}$, $-_{bin}$, / und * auf die Operanden anwendet und die Ergebnisse am Bildschirm ausgibt.
Überprüfen Sie anhand des Programms ohne Ausnahmebehandlung, welche Ausnahmen bei fehlerhafter Eingabe und welche während der Berechnung der Operationen auftreten können. Prüfen Sie anhand unterschiedlichster Eingabemöglichkeiten. Schreiben Sie dann die Ausnahmebehandlung und versuchen Sie die verschiedenen erkannten Ausnahmen getrennt zu behandeln.
Diese Einführung in die Ausnahmebehandlung soll nicht vortäuschen, dass Ausnahmen der Standard der Programmierung sei. Es gibt Bereiche der Programmierung, wo Ausnahmen mindestens genauso gefährlich sind wie auftretende Laufzeitfehler. Besser ist es, gleich Ausnahmen vollständig zu vermeiden. Einzig die Benutzereingabe ist nicht vor Fehlern gefeit. Hier ist es sinnvoll die Ausnahmebehandlung zu verwenden. Kombiniert mit einer Schleife, die Thema einer der folgenden Praktikumblätter sein wird, kann mit Hilfe von Ausnahmen eine wiederholte Eingabe gefordert werden.
In der Vorlesung wurde die Prozedurdefinition bereits mit der Ausnahmebehandlung eingeführt.

```
PROCEDURE <Bezeichner> IS

    <Definitionen>

BEGIN

    <Anweisungsfolge>

EXCEPTION

    WHEN <Bezeichner> => <Anweisungsfolge>;
    {WHEN <Bezeichner> => <Anweisungsfolge>;}

END <Bezeichner>;
```

Dabei können beliebig viele Ausnahmen abgefangen und bearbeitet werden. Ist eine Ausnahme aufgetreten wird die Programmausführung nach dem Ende der Prozedur oder des Blocks fortgesetzt, in dem die Ausnahme abgehandelt wurde. In den einfachen Fällen bisher bedeutet das, dass das Programm beendet wird.
Manche Ausnahmen müssen aus vordefinierten Paketen importiert werden, wo die Ausnahme definiert wird. Soll ein solcher Fehler abgefangen werden muss, wie bei den Ein-/Ausgabefunktionen, das entsprechende Paket eingebunden werden. Wird das Programm ohne Ausnahmebehandlung ausgeführt und der Anwender gibt als erstes Zeichen keine Ziffer ein, so wird der in Abbildung 1 gezeigte Laufzeitfehler ausgegeben.
Die für uns wichtige Information ist die zweite Zeile.

```
Exception raised : Ada.IO_Exceptions.Data_Error
```

Die ausgelöste Ausnahme heißt also `Ada.IO_Exceptions.Data_Error`. Diese Ausnahme tritt

bei syntaktischen Eingabefehlern (z.B. Buchstabe ist das erste Zeichen für eine Integereingabe) oder Wertebereichsverletzung bei der Eingabe auf. Wenn versucht wird, diesen Fehler so abzufangen, wird der Compiler beim Compilieren einen Fehler melden, dass die Ausnahme `Ada.IO_Exceptions.Data_Error` nicht vorher definiert wurde. Da Data_Error in dem Paket Ada_IO_Exceptions definiert is muß auch dieses importiert werden:

```
WITH Ada.Io_Exceptions;
```

Nun kann die Ausnahme wie folgt abgefangen werden.

```
EXCEPTIONS

    WHEN Ada.Io_Exceptions.Data_Error => <Anweisungsfolge>;
```

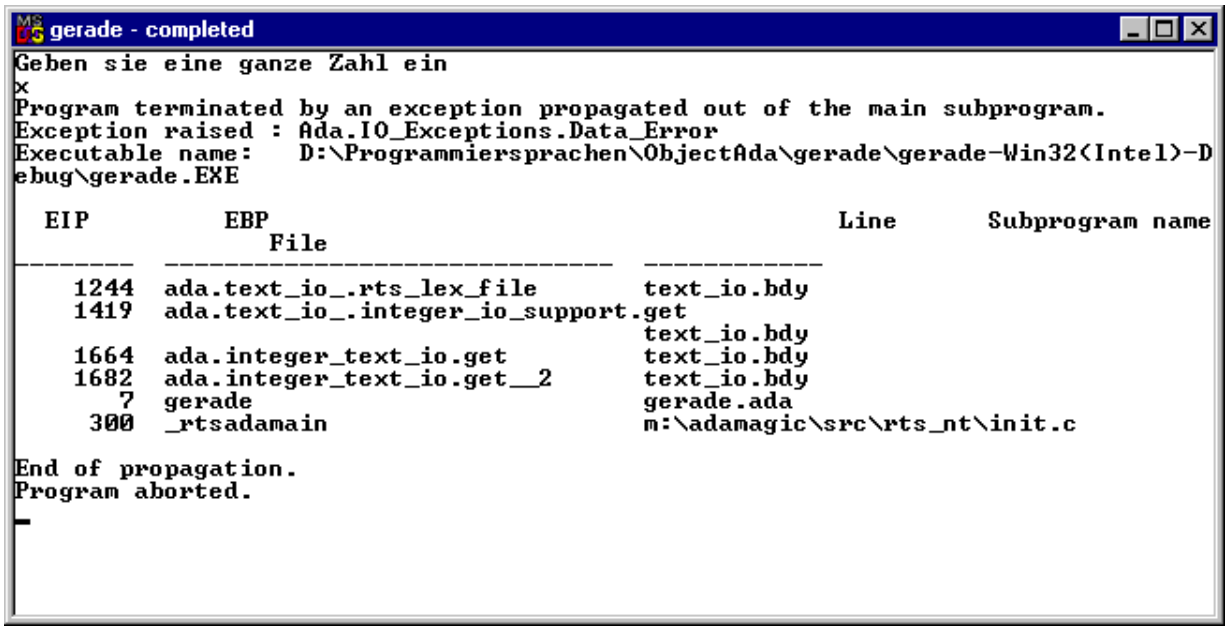


**Abbildung 1: Laufzeitfehler bei einer möglichen Falscheingabe durch den Benutzer. (ObjectAda)**

**Aufgabe2: Berechnungsreihenfolge**

In der Mathematik gilt für die Addition das Assoziativgesetz, das heißt das Ergebnis D:=A+B+C ist nicht abhängig von der Reihenfolge der einzelnen Berechnungen. Mit dieser Aufgabe soll überprüft werden, ob das auch für die Programmiersprache Ada gilt.
Wir wollen diese Überprüfung anhand des Datentyps Integer durchführen. Schreiben Sie zwei Programme, die jeweils die folgende Berechnung enthalten:

1. D:=(A+B)+C und
2. D:=A+(B+C)

Überprüfen sie das Programmverhalten für folgende Werte von A, B und C:

- A=2_000_000_000, B=2_000_000_000, C=-2_000_000_000
- A=-2_000_000_000, B=2_000_000_000, C=2_000_000_000
- A=-2_000_000_000, B=-2_000_000_000, C=2_000_000_000
- A=2_000_000_000, B=-2_000_000_000, C=-2_000_000_000

Was ist der Grund für das Verhalten?