Einführung in die Programmierung

Pakete

1. Zweck

Funktionen oder Prozeduren solle einmal geschrieben und mehrmals verwendet werden (Verwendbarkeit). Deswegen ist es unpraktisch, wenn sie immer in der selben Datei stehen, wo das Hauptprogramm. Funktionen oder Prozeduren werden in extra Dateien platziert, sogenannte Pakete (Units). Die Pakete sollten die Codes enthalten, die einem Thema gewidmet sind, z.B. Verarbeitung der Zeichenketten, Datenkonvertierung, Ein- und Ausgabe, u.s.w. Der Compiler erstellt die Objekt-Dateien (.o), der Linker verbindet alle Objekt-Dateien zu einer ausführbaren Datei (.exe) und löst die Modul-Verweise in diese Datei auf. Wichtig ist nur, dass man die richtigen Pfade für eigene Pakete schreibt.

Folgende zwei Situationen werden weiter betrachtet:

- Hauptprogramm und Pakete befinden sich im gleichen (aktuellen) Verzeichnis — einfache Situation.
- Hauptprogramm und Pakete befinden sich in unterschiedlichen Verzeichnisse — bisschen kompliziertere Situation.

2. Alles im aktuellen Verzeichnis

Die Pakete in Pascal haben die Erweiterung .pas, sie haben aber einen anderen Aufbau als das Hauptprogramm.

```
// ------- Inhalt der Datei mit dem Hauptprogramm ----------

PROGRAM Mein1Programm;  // Das ist Hauptprogramm
{$codepage utf8}        // Name der Datei ist unerheblich, aber ".pas"

USES libcourse;         // Das ist das Paket "libcourse" (Bibliothek)
BEGIN                   // Source-Code befindet sich in "libcourse.pas"
  Ausgabe('Version ');  // Das ist eine Prozedur aus "libcourse"
END.

// ------- Inhalt der Datei "libcourse.pas" – achten Sie auf den Aufbau ---------

UNIT libcourse;  // Name des Pakets stimmt mit dem Dateinamen überein
{$codepage utf8}

INTERFACE  // Weiter stehen nur die Köpfe (Deklarationen) von Funktionen/Prozeduren
PROCEDURE Ausgabe(s: String);  // Wird im Hauptprogramm aufgerufen

IMPLEMENTATION  // Weiter stehen vollständige Beschreibungen (Definitionen)
USES crt;       // von Funktionen/Prozeduren
PROCEDURE Ausgabe(s: String);
VAR n: INTEGER = 77;
BEGIN
  Writeln(s, n, ' von meinem Paket');
END;

END.  // Ende des Pakets
```

Hauptprogramm wird compiliert und gestartet. Der Compiler sucht die Datei "libcourse.o", die durch Anweisung

"USES libcourse;" angekündigt wurde. Beim ersten Start findet er sie selbstverständlich nicht, und sucht deswegen nach der Datei "libcourse.pas". Sie liegt auch im aktuellen Verzeichnis, sie wird compiliert, es entstehen im aktuellen Verzeichnis binäre Dateien "libcourse.o" und "libcourse.ppu". Die Dateien kann man in Übungen ruhig löschen, da der Compiler sie in dem Fall neu erstellen wird.

In der Sprache Go wird vorausgesetzt, dass ein Paket in einem extra dafür bestimmten Verzeichnis liegt, da laut der Definition "Ein Paket ist eine Sammlung von Quelldateien im selben Verzeichnis". Dementsprechend müssen alle Dateien die gleiche package-Anweisung (z.B. "package main") am Anfang haben. Dann besteht ein Paket aus mehreren Dateien.

```
// ------- Inhalt einer Datei ---------

package main  // Name der Datei ist unerheblich, z.B. g1.go

func main() {
  Ausgabe("... on Titanic")          //  ausgabe(...) funktioniert auch
}

// ------- Inhalt der anderen Datei ---------

package main  // Name der Datei ist unerheblich, z.B. g2.go

import "fmt"

func Ausgabe(s string) {            //  ausgabe(...) funktioniert auch
  fmt.Println("Do not panic!", s)
}

// Dateien müssen zusammen compiliert und gestartet werden
// go run g1.go g2.go
// oder dasselbe:
// go run g2.go g1.go
```

3. Hauptprogramm und Paket in unterschiedlichen Verzeichnissen

Erstellen Sie im aktuellen Verzeichnis für Pascal ein Unterverzeichnis namens "myUnits". Platzieren Sie Source-Code des Pakets "libcourse.pas" in diesem Unterverzeichnis.

```
PROGRAM Mein1Programm;  // Das ist Hauptprogramm
{$codepage utf8}        // Name der Datei ist unerheblich, aber ".pas"
{$UNITPATH ./myUnits}   // Relativer Pfad zum Paket, KEINE ÄNDERUNGEN SONST
// {$UNITPATH /Source/fpc/myUnits}  // Oder absoluter Pfad zum Paket

USES libcourse;         // Das ist das Paket "libcourse" (Bibliothek)
BEGIN                   // Source-Code befindet sich in "libcourse.pas"
  Ausgabe('Version ');  // Das ist eine Prozedur aus "libcourse"
END.

// ------- Inhalt der Datei "libcourse.pas" – achten Sie auf den Aufbau ---------
UNIT libcourse;  // Name des Pakets
{$codepage utf8}

INTERFACE  // Weiter stehen nur die Köpfe (Deklarationen) von Funktionen/Prozeduren
PROCEDURE Ausgabe(s: String);  // Wird im Hauptprogramm aufgerufen

IMPLEMENTATION  // Weiter stehen vollständige Beschreibungen (Definitionen)
USES crt;       // von Funktionen/Prozeduren
PROCEDURE Ausgabe(s: String);
VAR n: INTEGER = 77;
BEGIN
  Writeln(s, n, ' von meinem Paket');
END;

END.  // Ende des Pakets
```

Hauptprogramm wird wieder gestartet. Der Compiler erstellt beim ersten Start die beiden Dateien "libcourse.o" und "libcourse.ppu" im Unterverzeichnis "myUnits".

Erstellen Sie im aktuellen Verzeichnis für Go ein Unterverzeichnis namens "myUnits". Platzieren Sie die Datei "libcourse.go" in diesem Unterverzeichnis. Alle Dateien in diesem Unterverzeichnis müssen als erste Zeile die Anweisung "package libcourse" haben. Dadurch gehören alle Dateien, unabhängig von ihren Namen, zum Paket "libcourse".

```
// ------- Inhalt der Datei mit Hauptprogramm – Name der Datei ist unerheblich ------

package main  // Name der Datei ist unerheblich
import "./myUnits" // Achtung - groß/klein-Schreibung !

func main() {
  libcourse.Ausgabe("Mary Ann")
}

// ------- Inhalt der Datei "libcourse.go" – Name der Datei ist unerheblich ---------

package libcourse  //  Der Name ist wichtig
import "fmt"

func Ausgabe(s string) {
  fmt.Println("Do not panic!", s)
}
```

```
// Verzeichnis- und Paketnamen müssen nicht übereinstimmen
// Dateien eines Paketen müssen in EINEM Verzeichnis liegen
// Zwei unterschiedliche Pakete in EINEM Verzeichnis dürfen nicht liegen
// Zwei Dateien von dem selben Paket können in EINEM Verzeichnis liegen
// Zwei Funktionen mit gleichem Namen dürfen in EINEM Paket nicht liegen
```

Aufgabe 1. Schreiben Sie folgendes Programm in Pascal und in Go. Zwei Funktionen Min(a,b) und Max(a,b) müssen in einem Paket zusammengefasst werden. Die Funktionen haben zwei ganzzahlige Parameter und geben entsprechend die kleinste und die größte Zahl zurück. Im Hauptprogramm werden diese Funktionen getestet. Das Hauptprogramm und Paket liegen im selben Verzeichnis.

Aufgabe 2. Erweitern Sie die vorige Aufgabe, indem das Hauptprogramm und Paket in unterschiedlichen Verzeichnissen liegen.