

In der ersten Aufgabe soll der Umgang mit der Sprache C sowie dem Werkzeug Make geübt werden. Inhaltlich wird es u. a. um die Auswertung von Kommandozeilenargumenten sowie um Operationen zur Ein- und Ausgabe gehen.



- ## 1 Erkennung von Bezeichnern

Parameter

- `moco`
Aufruf ohne Parameter: lese von der Standardeingabe und schreibe auf die Standardausgabe
- `moco infile`
Aufruf mit einem Parameter: lese von der Datei `infile` (sofern diese lesbar ist) und schreibe auf die Standardausgabe

- **moco infile outfile**

Aufruf mit einem Parameter: lese von der Datei **infile** (sofern diese lesbar ist) und schreibe in die Datei **outfile** (sofern diese schreibbar ist)

Beispiel

Für eine Eingabedatei **hello.m** mit dem Inhalt

```
MODULE HelloWorld;

TYPE
    myInt = INTEGER;

CONST
    pi = 3.1415926;

VAR
    x : REAL;

BEGIN
    f := 2 + pi;
END HelloWorld.
```

würde das Problem wie folgt arbeiten:

```
$ ./moco hello.m
MODULE
HelloWorld
TYPE
myInt
INTEGER
CONST
pi
VAR
x
REAL
BEGIN
f
pi
END
HelloWorld

$ ./moco hello.m output

$ cat output
MODULE
HelloWorld
TYPE
myInt
INTEGER
CONST
pi
VAR
x
REAL
BEGIN
f
pi
END
HelloWorld
```

Fehlermeldungen

Falls es ein Problem mit den Eingabe- oder Ausgabedateien gibt oder der Nutzer zu viele Parameter angibt, soll das Programm mit einer Fehlermeldung und dem Exit-Code 1 abbrechen.

```
$ ./moco unlesbaredatei.m
error: cannot open file for reading
```

```
$ ./moco hello.m nichtschreibbaredatei.m
error: cannot open file for writing

$ ./moco hello.m output ueberfluessigerparameter
error: wrong number of parameters
```

Makefile

Auch wenn es für diese einfache Aufgabe noch nicht notwendig ist, soll ein Makefile geschrieben werden, das den Übersetzungsprozess steuert. Beim Aufruf von **make** soll in zwei Schritten zunächst aus der Quelldatei (**main.c**) eine Objektdatei (**moco.o**) und im nächsten Schritt das ausführbare Programm (**moco**) erzeugt werden. Der Aufruf von **make clean** löscht alle generierten Dateien.

Vorgaben

- Erstellen Sie genau zwei Dateien (**main.c** und **Makefile**).
- Mittels **make** muss sich das lauffähige Programm **moco** erstellen lassen.
- Implementieren und nutzen Sie für die Fehlermeldungen eine Funktion **void yyerror(char *msg)**, die die Fehlermeldung **msg** ausgibt und das Programm mit Exit-Code 1 beendet.
- Nutzen Sie eine globale Variable **char *yytext**, in der stets der aktuelle Bezeichner steht.

Hinweise

- Modula-2 macht keine Annahmen über die Länge von Bezeichnern.
- Vermeiden Sie es, die Eingabe zunächst komplett in einen Puffer zu laden. Compiler sind On-the-fly-Werkzeuge – sowohl aus Zeit- als auch aus Speichergründen.

2 Erkennung von Schlüsselworten (Zusatzaufgabe)

Sofern es sich bei einem erkannten Bezeichner um ein Token wie z.B. **OR** oder ein reserviertes Schlüsselwort von Modula-2 handelt, soll dieser bei der Ausgabe nicht berücksichtigt werden.

Für das Beispiel oben wäre die Ausgabe also:

```
$ ./moco hello.m
HelloWorld
myInt
pi
x
f
pi
HelloWorld
```