

1 Verwendung des OCaml-Compilers

Der erste Schritt besteht darin eine Datei `test.ml` mit folgendem Inhalt zu erstellen:

```
(** Der Aufruf fact n liefert die Fakultät von n zurück *)  
let rec fact n =  
  if n = 0 then  
    1  
  else  
    n * (fact (n - 1))  
  
let _ = print_string "?_n=_"  
let n = read_int ()  
  
let o = (string_of_int n) ^ "!_=" ^ (string_of_int (fact n)) ^ "\n"  
let _ = print_string o
```

Kompiliert wird diese Datei mit dem Aufruf

```
ocamlc -o test test.ml bzw. ocamlc -o test.exe test.ml
```

der eine ausführbare Datei Namens `test` bzw. `test.exe` erstellt. Ausgeführt wird das erstellte Programm dann mit dem Aufruf:

```
./test bzw. test.exe
```

Das Programm liest vom Benutzer eine Zahl n ein und liefert den die Fakultät $n!$ zurück. Es erscheint folgendes:

```
? n = 5  
5! = 120
```

2 Tricks

1. Der Aufruf `ocamlc -i test.ml` liefert:

```
val fact : int -> int  
val n : int  
val o : string
```

2. Dem Interaktiven OCaml-Interpreter kann man beim Start eine `.cmo`-Datei mitgeben. Das in dieser Datei definierte Modul steht dann zur Verfügung. In unserem Beispiel bewerkstelligt dies der Aufruf `ocaml test.cmo`. Im Interpreter muss man dann nur noch das Modul `Test` mit dem Aufruf `open Test;;` öffnen.

3 Vordefinierte Funktionen

3.1 Standard-Funktionen

- `val print_string : string -> unit`
`print_string s` gibt den String `s` aus.
- `val print_int : int -> unit`
`print_int n` gibt die Zahl `n` aus.
- `val read_line : unit -> string`
`read_line ()` liest eine Zeile ein und liefert sie zurück.
- `val read_int : unit -> int`
`read_int ()` liest eine Zahl ein und liefert sie zurück.
- `val string_of_int : int -> string`
`string_of_int n` liefert die String-Darstellung der Zahl `n`.
- `val int_of_string : string -> int`
`int_of_string s` liefert die durch den String `s` repräsentierte Zahl zurück.