ソフトウェア技法: 補足 1 (let と二重セミコロンの使い方)

亀山幸義 (筑波大学情報科学類)

この資料では、let とセミコロンの使い方について補足する。

1 let

OCaml の let には、2 通りの使いかたがあり、(1) トップレベルで定義で使う let (in を伴っても伴わなくてもよい) と、(2) 普通の式としての let 式 (必ず in を伴う) の 2 つがある。

(1) は、これまで何度も使ってきたものである。

```
let n = 10 ;;
let foo x = x + 1;;
let _ = foo n ;;
let goo x = x + 1 in
  goo 10 ;;
(* goo はこの後使えない *)
```

トップレベルの let は in を伴う必要はなく、in を伴わない let (上記では変数 n の定義や関数 foo の定義) で定義された変数や関数は、そのあとずっと使える。

一方、トップレベルの let は、in を伴ってもよく、その場合、let で定義された変数や関数は、「その場限り」である。上記のプログラムでは、goo は、こちらのやりかたで定義されているので、そのあとは使えなくなる。

一方 (2) については、これまではほとんど言ってこなかったが、普通の式の一種となり、トップレベルでなくても使えるものである。こちらは、必ず in を伴う必要がある。

```
(* 以下のものは エラー*)
let foo x =
  1 + 2 * 3 + (let y = x * 2) * 4 + 5 ;;

(* 以下のものは OK*)
let foo x =
  1 + 2 * 3 + (let y = x * 2 in y + y) * 4 + 5

(* let ... in ... は普通の式なので、式を書けるところで自由につかってよい*)
let foo x =
  let n = 10 in
  if n <= 2 then 1
  else
    let y = 10 in
      y * y + (n - let z = x * y + n in z) + 4
```

(2) の用法 (式としての let) は、関数本体の中で、一時的に計算結果を保存しておく (そして、あとで、何度か利用する) ために便利である。

2 二重セミコロン

OCaml では、二重セミコロン (;;) も不思議である。

プログラムを直接 OCaml の対話モード (1 つの式を入力するとすぐに実行して答えが返ってくるモード) で実行するときは、式の終わりに必ず二重セミコロン (;;) が必要である。

```
# let n = 10 ;;
# #use "abac.ml" ;;
# #quit;;
```

一方、プログラムをファイルに書いた場合は、式の終わりに二重セミコロンを書いてもいいが、OCaml は賢く「式の区切り」を見分けてくれるので、二重セミコロンを書かなくてもよいことが多い。

```
(* ファイルの中身 *)
let foo x = x + 1
let rec goo x =
  if x = 0 then 1
  else  x * (goo (x - 1))
let _ = goo 13
```

このように、ファイルに書きこんだときは、トップレベルの let の区切りとして二重セミコロンを書かなくても、OCaml は区切りを理解してくれる。

そこで、多くの OCaml プログラムのファイルは、二重セミコロンを書いていない。そのようなプログラムをコピーして、対話モードで使うときは、自分で二重セミコロンを補わないといけないので注意されたい。

この授業のファイルでは、できるだけ、このような煩わしさがなくなるよう、二重セミコロンをなるべく書くことにしているが、ときどき「いつもの癖」で二重セミコロンを忘れてしまっていることがある。(それでもファイルの中身である間は動くので。) そこで、皆さんの方でも気をつけて、二重セミコロンで区切られていない let を見つけたら、適宜区切って考えてほしい。