Informatique : Rappels syntaxe C et OCaml

1 Opérateurs booléens

– Et : `&&` dans les deux langages
– Ou : `||` dans les deux langages
– Négation : `not` en OCaml, et `!` en C
– Test d'égalité : `=` en OCaml, et `==` en C. En OCaml, `==` teste l'égalité des adresses mémoires (à proscrire)
– Différence : `<>` en OCaml, et `!=` en C. En OCaml, `!=` teste la différence des adresses mémoire.

2 Déclarations en OCaml

– Déclaration locale :

```
let nom = valeur in expression
(*Surtout pas : *)
let nom = valeur;
expression
```

– Déclaration globale :

```
let nom = valeur
(*ou*)
let nom = valeur;;
```

3 Expressions conditionnelles

```
if condition then expression1 else expression2
```

avec `expression1` et `expression2` de **même type**.

```
x + (if b then 42 else 0)
```

Attention :

```
if b then
    Printf.printf "Message d'erreur\n";
    0
else
    expression
```

fait un `Syntax error` car le point virgule fait croire à OCaml que le `if` s'arrête à cet endroit.

```
while b1 do
    if b2 then
        0
    else
        incr x
done;
!x
```

Plusieurs problèmes : pas le même type dans le *if*. De plus, on ne peut pas arrêter la boucle `while` ainsi.

On peut faire :

```
let b3 = ref true in
while b1 && !b3 do
    if b2 then begin
        b3 := false;
        x := 0
    end
    else
        incr x
done;
!x
```

ou :

```
exception Break

try
    while b1 do
        if b2 then
            raise Break
        else
            incr x
    done;
    !x
with
| Break -> 0
```

```
let l = [] in
for i = 0 to 42 do
    i::l
done;
l
```

Plusieurs erreurs. Version corrigée :

```
let l = ref [] in
for i = 0 to 43 do
    l := i::!l
done;
!l
```

Les seuls objets mutables sont :

– Les cases des tableaux ;
– Les références ;
– Les champs d'enregistrement déclarés mutables.