第A章 Haskell のレイアウトルール

これまで Haskell のレイアウトルールについては明確に述べていなかった。Haskell は字下げ（インデント）の仕方により、字句解析・構文解析が影響を 受ける言語である。（ブレースやセミコロンを明示的に使って影響を受けない ようにすることも可能である。）

例えば、

```
f x = let a = 1; b = 2
          g y = exp2
      in exp1
```

という式は、

```
f x = let {a = 1; b = 2
          ;g y = exp2
      } in exp1
```

と解釈される。レイアウトルールはこのようなブレースやセミコロンがどのような場合に挿入されるかを定める。

以下の字下げ部分は Haskell の仕様書である“Haskell 2010 Language Report”の§2.7 からの抜粋である。（なお、同書の § 10.3 に、より詳細な仕様が掲載されている。）

> Haskell permits the omission of the braces and semicolons used in several grammar productions, by using layout to convey the same information.
> Haskellは、 同等の情報を持つレイアウトを使うことによって、いくつかの文法規則で使われているブレースとセミコロンを省略することを許している。
>
> （中略）
>
> The effect of layout on the meaning of a Haskell program can be completely specified by adding braces and semicolons in places determined by the layout. The meaning of this augmented program is now layout insensitive.
> レイアウトの Haskell プログラムに対する効果は、レイアウトによって決定される場所にブレースとセミコロンを追加することにより完全に記述することができる。すると、その追加の結果のプログラムの意味はレイアウトに依存しなくなる。
>
> Informally stated, the braces and semicolons are inserted as follows.
> ブレースとセミコロンはおおむね次のように挿入される。
>
> The layout (or “off-side”) rule takes effect whenever the open brace is omitted after the keyword **`where`**, **`let`**, **`do`**, or **`of`**.

> レイアウト（あるいは"オフサイド"）ルールは **where**, **let**, **do**, **of** というキーワードの後で開ブレースが省略されたときに、有効になる。

```
let
  f x y =
    case x of
       0 -> foo x 2
       1 -> bar 1 x 999999
              3 4 5
       _ -> baz 6 x
     +
    case y of 2 -> qux 9 y 1
              _ -> quux y  in f 0 1
```

> When this happens, the indentation of the next lexeme (whether or not on a new line) is remembered and the omitted open brace is inserted (the whitespace preceding the lexeme may include comments).
> このときは、（新しい行にあるかどうかに係わらず）次の字句の字下げ数が記憶され、省略された開ブレースが挿入される（タブ文字は次の **8** の倍数文字目までの空白文字と解釈される。 Windows のエディタはタブ文字が次の **4** の倍数文字目までの空白文字と解釈されるものが多いので注意すること）。（字句の前の空白はコメントの場合もある（漢字・かなどの全角文字も Haskell の処理系にとっては英数字と同じ一文字なので、コメントがレイアウトに関係する場合は注意すること。）。）

```
-- 開ブレース挿入後
let
 {f x y =
    case x of
      {0 -> foo x 2
       1 -> bar 1 x 999999
              3 4 5
       _ -> baz 6 x
     +
    case y of{2 -> qux 9 y 1
              _ -> quux y  in f 0 1
```

> For each subsequent line, if it contains only whitespace or is indented more, then the previous item is continued (nothing is inserted); if it is indented the same amount, then a new item begins (a semicolon is inserted); and if it is indented less, then the layout list ends (a close brace is inserted).
> それに続く各行について、もし、それが空白のみを含むか、より多くの字下げがされているならば、直前の項目が継続される（つまりなにも挿入されない）。もし、同じだけ字下げされているなら、新しい項目が始まるとみなす（セミコロンが挿入される）。もし字下げ数が少ないならばレイアウトリストは終わりとみなされる（閉ブレースが挿入される）。

```
-- セミコロン挿入後
let
 {f x y =
```

```
    case x of
      {0 -> foo x 2
      ;1 -> bar 1 x 999999
              3 4 5 -- この行は挿入なし
      ;_ -> baz 6 x
    }+
    case y of{2 -> qux 9 y 1
             ;_ -> quux y  in f 0 1
```

> A close brace is also inserted whenever the syntactic category containing the layout list ends; that is, if an illegal lexeme is encountered at a point where a close brace would be legal, a close brace is inserted.
> 閉ブレースはレイアウトリストを含む構文カテゴリが終了するときにも挿入される。つまり、閉ブレースが現れても良いときに不正な字句が現れたとき、閉ブレースが挿入される。

```
-- 閉ブレース挿入後
let
 {f x y =
    case x of
      {0 -> foo x 2
      ;1 -> bar 1 x 999999
              3 4 5 -- この行は挿入なし
      ;_ -> baz 6 x
    }+
    case y of{2 -> qux 9 y 1
             ;_ -> quux y}}in f 0 1
```

> The layout rule matches only those open braces that it has inserted; an explicit open brace must be matched by an explicit close brace.
> レイアウト規則は開ブレースが挿入されたときにのみ発動される。明示的な開ブレースがあったときは、明示的な閉ブレースで終らなければならない。
>
> Within these explicit open braces, *no* layout processing is performed for constructs outside the braces, even if a line is indented to the left of an earlier implicit open brace.
> このような明示的な開ブレースのなかでは、そのブレースの外側の構成要素に対しては、たとえある行が以前にあった暗黙の開ブレースよりも左から始まったとしても、レイアウトルールは適用されない。

```
let{foo = let {
  x = 0; y = 1;
  z = 2 }
    in x * y
   ;bar = 99}in foo * bar
```
