054: strlen.hs で renPat: LitExp (LitString "" (1,8))

↑ up

- issued: 2020-05-03
- 分類： A サンプルコードが fail
- status: Closed (2020-05-04)

概要

パターン中における String Literal に未対応のため、strlen.hs がエラーする。
strlen.hs:

```
strlen ""     = 0
strlen (c:cs) = 1 + strlen cs

main = print $ strlen "Haskell"
```

現象：

```
$ ./tcheck testcases/strlen.hs
# 1. test-compile
source file: testcases/strlen.hs
dst dir: /strlen
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc.exe: renPat: LitExp (LitString "" (1,8))
CallStack (from HasCallStack):
  error, called at src\Rename.hs:491:12 in main:Rename
```

注意： 将来 OverloadedStrings 相当に対応するときのことも多少考慮した実装にしたい。

2020-05-04 (Mon)

Rename.renPat を対処し、String リテラル "abc" を 'a':'b':'c':[] に変換して処理するようにした。
それによって、strlen.hs は別の問題にぶつかるようになった：

```
$ ./tcheck testcases/strlen.hs
```

```
# 1. test-compile
source file: testcases/strlen.hs
dst dir: /strlen
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: Error: dictionary not found: Prelude.print, ("Prelude.Show",Tyvar "a1" Star,[]
CallStack (from HasCallStack):
  error, called at src/DictPass.hs:249:33 in main:DictPass
```

これは、やっぱり、defaulting がいいかげんなせいだな。数値リテラルに由来した曖昧性なのだが、main 中ではそれはわからないので、数値リテラルにいきあたったら Integer にっていう現状のやっつけ仕事では解決しないやつ。

パターンにおける "" は使えるようになったので、lib/Prelude.hs でそれを避けていた個所は修正：

```
--- a/compiler/lib/Prelude.hs
+++ b/compiler/lib/Prelude.hs
@@ -161,8 +161,7 @@ instance Show Char where
   -- todo: escape
   show c = ['\'', c, '\'']
   showList cs = (:) '"' . showl cs
-    where -- showl "" = (:) '"'
-          showl [] = (:) '"'
+    where showl "" = (:) '"'
           -- showl ('"':cs) = (++) "\\\"" . showl cs
           showl (c:cs) = showLitChar c . showl cs
```

また、strlen.hs に型宣言を追加して現状でも通るようにしたものを、sample170 として追加してく。

その２

Defaulting を少し改良することで、strlen.hs は通るようになった。その内容については、045 に記載することとし、本件はクローズとする。

strlen.hs は sample171.hs に。