Answers to the quiz on Compilers

Christian Rinderknecht

29 November 2005

Question. Consider the following Lex regular expression and propose a transition diagram which allows the recognition of the same lexemes as num.

```
num  (\.[0-9]+|[+\-]?[0-9]+(\.[0-9]+)?)(E[+\-]?[0-9]+)?
```

Answer.

Question. Consider the following Lex regular expression and propose a transition diagram which allows the recognition of the same lexemes as `id`.

```
id   [A-Za-z]([_]*[A-Za-z0-9])*
```

Answer.

Question. Assume the following input buffer where ␣ represents a blank character.

```
return␣␣x+␣.5E2+y_0
```

Complete the following Lex skeleton

```
%{
#include<string.h>
char* keyword[] = {"else", "if", "return", "then"};
%}
num   (\.[0-9]+|[+\-]?[0-9]+(\.[0-9]+)?)(E[+\-]?[0-9]+)?
id    [A-Za-z]([_]*[A-Za-z0-9])*



%%
{num}  {


       }
{id}   {









       }


%%
```

such that the array `keyword` is used[1] and that the output is

```
kwd<return>
id<x>
plus<>
num<.5E2>
plus<>
id<y_0>
```

Answer.

```
%{
#include<string.h>
char* keyword[] = {"else", "if", "return", "then"};
```

[1] You may use the C function `int strcmp(const char *s1, const char *s2)` which returns 0 if the strings `s1` and `s2` are equal.

```
%}
num  (\.[0-9]+|[+\-]?[0-9]+(\.[0-9]+)?)(E[+\-]?[0-9]+)?
id   [A-Za-z]([_]*[A-Za-z0-9])*
ws   [ \n\t]+
%%
{num}  { printf ("num<%s>\n",yytext); }
{id}   { int index = 0;
         while (index <= 3 && strcmp(keyword[index],yytext))
           index++;
         if (index == 4) printf("id<%s>\n",yytext);
         else printf("kwd<%s>\n",yytext);
       }
{ws}   {}
"+"    { printf ("plus<>\n"); }
%%
```

Question. Define the meaning of the pointers $\uparrow$, $\restriction$ and $\Uparrow$ presented in class and show how the input is analysed using the transition diagrams of the previous questions.

Answer. See handouts and previous answers.