

06. Exemplos finais flex

Exemplo final 1: Calculadora de inteiros, pós-fixa

Este exemplo flex pretende ser uma calculadora de inteiros com suporte às 4 operações básicas, em modo pós-fixa (*postfix*, também chamada de calculadora RPN – *reverse polish notation*, ou notação polaca inversa), onde o operador segue os dois operandos (e.g.: “2 5 +” seria 7).

```
%option main
#include <stdio.h>
#include <stdlib.h>
int a = 0;
int b = 0;

%%
int r;

[+-]?[0-9]+    a = b;  b = atoi(yytext);
[+]            r=a+b; printf("%d+%d=%d\n", a,b,r); b=r;
[-]            r=a-b; printf("%d-%d=%d\n", a,b,r); b=r;
[*]            r=a*b; printf("%d*%d=%d\n", a,b,r); b=r;
[/]            r=a/b; printf("%d/%d=%d\n", a,b,r); b=r;
.\|r|\n
```

Exemplo final 2: Contador de identificadores

Este exemplo flex pretende ser um contador de identificadores e soma do valor de todos os inteiros, num código fonte em qualquer linguagem. Um identificador é definido como uma sequência de um ou mais caracteres alfanuméricos ou *underscore* (sublinhar). O identificador não pode começar com um caracter numérico.

```
%option noyywrap
#include <stdio.h>
#include <stdlib.h>
enum { TOKEN_IDENT, TOKEN_INT, TOKEN_MISC, TOKEN_EOF };
int este_int;

%%
```

```

[a-zA-Z_][a-zA-Z0-9_]*      return TOKEN_IDENT;
[+-]?[0-9]+                { este_int = atoi(yytext);
                             return TOKEN_INT; }
.|\\r|\\n                  return TOKEN_MISC;
<<EOF>>                    return TOKEN_EOF;

%%

int main()
{
    int num_idents = 0;
    long soma_total = 0L;
    int token;

    do
    {
        switch( token=yylex() )
        {
            case TOKEN_IDENT:    ++num_idents;
                                break;
            case TOKEN_INT:      soma_total += este_int;
                                break;
            /* outro TOKEN_*: ignorar */
        }
    }
    while( token != TOKEN_EOF );

    printf( "Identificadores: %d\\n",    num_idents );
    printf( "Soma dos inteiros: %ld\\n", soma_total );
    return 0;
}

```

Repare-se que aqui as acções fazem `return`, saindo assim da função `yylex()`. Esta é uma forma comum de permitir que a função `yylex()` seja apenas auxiliar em detectar *tokens*, e não tenha o trabalho central de processamento dos mesmos. Esta será a forma de usar o `flex` quando avançarmos para o `bison`.

Exercício final flex

Combina os dois exemplos anteriores, criando uma calculadora de números reais, infixa, mas onde a leitura de cada *token* (elemento) da calculadora é feita pela função `main()`, chamando o `yylex()` para ler cada *token*, como o exemplo 2, acima.