Apêndice B

Reconhecedor Sintáctico SLR

O Reconhecedor ficheiro "reconhecedorSLR.c"

```
/*******************************************************/
/* Reconhecedor SLR para a linguagem definida pela     */
/* seguinte gramática:                                 */
/*    G=({a,b},{I,A},I,{I->Aa|b;A->Aa|ϵ})              */
/*                                                     */
/* Pedro Quaresma, 20/11/2002                          */
/*******************************************************/

/* Inclusão das Estruturas e Funções referentes às Pilhas */
#include "pilhas.h"

#include <stdlib.h>

#define ERRO -1
#define n_estados 5
#define n_simb 6
#define n_simb_terminais 3

/* Aos diferentes símbolos da linguagem vai-se fazer */
/* corresponder números inteiros, para os caracteres */
/* isolados esse número será a sua codificação ASCII */
/* para os outros é necessário fazer uma codificação */
/* explícita.                                        */

/* Definição dos não-terminais */
#define Il 128
#define I  129
#define A  130
```

[1](Versão 1.7)

```
#define EstadoInicial 0

/* Tabela de Transições (elementos são os índices dos estados) */
int tt[n_estados][n_simb] = {
  {ERRO,2,ERRO,ERRO,1,3},
  {ERRO,ERRO,ERRO,ERRO,ERRO,ERRO},
  {ERRO,ERRO,ERRO,ERRO,ERRO,ERRO},
  {4,ERRO,ERRO,ERRO,ERRO,ERRO},
  {ERRO,ERRO,ERRO,ERRO,ERRO,ERRO}
};

/* Estrutura acções (accção,inteiro), e.g. r5=('r',5) */
struct par {
  char accao;
  int  numero;
};
typedef struct par Par;

/* Tabela de Acções, os elementos são as acções */
/* representadas sobre a forma de pares, e.g.   */
/* r5 = ('r',5), d2 = ('d',2),                  */
/* aceita = ('a',0), erro = ('e',-1)            */
Par ta[n_estados][n_simb_terminais] = {
  {{'r',5}, {'d',2}, {'e',-1}},
  {{'e',-1},{'e',-1},{'a',0}},
  {{'e',-1},{'e',-1},{'r',3}},
  {{'d',4}, {'e',-1},{'e',-1}},
  {{'r',4}, {'e',-1},{'r',2}}
};

/* Produções, representadas sobre a forma de pares           */
/* (elemento da esquerda, comprimento da sequência da direita */
/* Por exemplo, I->Aa é representado por (I,2), sendo que o   */
/* o símbolo não terminal é objecto de uma definição "define" */
struct prod {
  int esquerdo;
  int comprimento;
};
typedef struct prod Producao;

Producao producoes[] = {{I1,2},{I,2},{I,1},{A,2},{A,0}};

/* vector que estabelece as posições relativas dos   */
```

```
/* diferentes símbolos da gramática nas tabelas TA e TT. */
int simbolos2indices[] ='a','b','$',Il,I,A;

/* A rotina principal */
main(){
 int i,j;
 int reconhecedorSLR();

 if (reconhecedorSLR())
   printf("\n\t%s\n","Aceita");
 else
   printf("\n\t%s\n","Erro");
}

/* O Reconhecedor SLR */
int reconhecedorSLR(){
  Elem_pilha simb,estado;
  Pilha *p;
  Par ac;
  Producao prod;
  int i,j;

  /* Inicialização da Leitura e da Pilha de Reconhecimento */
  simb=yylex();
  p=vazia(p);
  p=push(EstadoInicial,p);
  do {
    /* Verificar qual a acção a tomar */
    ac=ta[top(p)][simb2ind(simb)];
    if (ac.accao=='d') {
      /* Caso em que se vai proceder a uma deslocação */
      estado=tt[top(p)][simb2ind(simb)];
      p=push(simb,p);
      p=push(estado,p);
      simb=yylex();
    };
    if (ac.accao=='r') {
      /* Caso em que se vai proceder a uma redução */
      prod=producoes[ac.numero-1];
      for (i=0;i<2*prod.comprimento;i++)
        p=pop(p);
      estado=tt[top(p)][simb2ind(prod.esquerdo)];
      p=push(prod.esquerdo,p);
      p=push(estado,p);
```

```
    };
  }
  while (ac.accao=='d' || ac.accao=='r');
  return (ac.accao=='a');
}

/* Rotina auxiliar para fazer a correspondência entre   */
/* símbolos e as suas posições nas tabelas TA e TT      */
int simb2ind(Elem_pilha simb){
  int i=0;

  while (simb!=simbolos2indices[i]) i++;
  return i;
}

/* O Reconhecedor Léxico */
int yylex(){
  int c;

  c=getchar();
  if (c=='a' || c=='b' || c=='$')
    return c;
  else {
    printf("Erro léxico\n");
    return c;
  }
}
```

Pilhas ficheiro "pilhas.c".

```
/*****************************************************/
/*  Implementação de Pilhas de Inteiros (int)        */
/*                                                   */
/*  Módulo auxiliar à construção do Reconhecedor     */
/*  Sintáctico do tipo SLR                           */
/*                                                   */
/*  Pedro Quaresma, 18/11/2002                       */
/*****************************************************/
/* Para se puder usar a constante NULL */
#include <stdlib.h>

/* Para se ter a informação acerca das Estruturas de Dados */
#include "pilhas.h"
```

```
Pilha *pilhaalloc(void){
  return (Pilha *) malloc(sizeof(Pilha));
}

Pilha *vazia(Pilha *p) {
  return NULL;
}

Pilha *push(Elem_pilha elem, Pilha *p) {
  Pilha *novo;
  Pilha *pilhaalloc(void);

  novo=pilhaalloc();
  (*novo).elems=elem;
  (*novo).prox=p;
  p=novo;
  return p;
}

Pilha *pop (Pilha *p) {
  Pilha *aux;

  if (p==NULL) {
    printf("\n%s\n","ERRO pop, Pilha vazia");
  }
  else {
    aux=p;
    p=(*p).prox;
    free(aux);
  }
  return p;
}

Elem_pilha top(Pilha *p) {
  if (p==NULL) {
    printf("\n%s\n","ERRO top, Pilha vazia");
    return ((Elem_pilha) 0);
  }
  else {
    return (*p).elems;
  }
}
```

Pilhas (declarações) ficheiro "pilhas.h".

```
/*****************************************************/
/*  Implementação de Pilhas de Inteiros (int)        */
/*                                                   */
/*  Cabeçalhos                                       */
/*                                                   */
/*  Pedro Quaresma, 18/11/2002                       */
/*****************************************************/

/* Definição do Tipo "Elem_pilha" */
typedef int Elem_pilha;

/* Definição do tipo Pilha  */
struct pilha {
  Elem_pilha elems;
  struct pilha *prox;
};

typedef struct pilha Pilha;

Pilha *pilhaalloc(void);

Pilha *vazia(Pilha *);

Pilha *push(Elem_pilha, Pilha *);

Pilha *pop (Pilha *);

Elem_pilha top(Pilha *);
```

Compilação Para criar o reconhecedor basta compilar os ficheiros referentes ao próprio reconhecedor e às rotinas para a manipulação da pilha.

```
> gcc reconhecedorSLR.c pilhas.c -o reconhecedorSLR
```