Realizar as alterações na pasta parte 3 [**RECOMENDADA**] *obs: fazer backup antes de interagir*
Alterações: funcao, tipo, nome da funcao, tipo dos argumentos, declaracao de variaveis e a mesma

utils.c - ampliar tabela de símbolos, criar funcões

- Na tabela de símbolos: além de endereço (trocar descrição por deslocamento) e tipo

```
#define MAX_PAR 20

// Sugestao:
// ……..
// Desenvolver uma rotina para ajustar o endereco dos parametros
// na tabela de simbolos e o vetor de parametros da funcao
// depois que for cadastrado o ultimo parametro

// Modificar a rotina mostraTabela para apresentar os outros
// campos (esc, rot, cat,...) da tabela.

char id[100];          // identificador
int end;               // endereco (global) ou deslocamento (local)
int tip;               // tipo
char cat;              // categoria: 'f'=FUN, 'p'=PAR, 'v'=VAR
char esc;              // escopo: 'g'=GLOBAL, 'l'=LOCAL
int rot;               // rotulo (especifico para funcao)
int npa;               // numero de parametros (para funcao)
int par[MAX_PAR];      // tipos dos parametros (para funcao)
// int *par;           // tipos dos parametros (para funcao)
```

obs: a implementação será feita com vetor fixo (20 espaços) e não alocação dinâmica

Funções de Busca e Inserção continuarão da mesma forma

Trocar o tipo da pilha

```
struct {
        int valor;
        char tipo; // 'r' = rotulo, 'n' = nvars, 't'=tipo, 'p' = posicao
} pilha[TAM_PIL]

void empilha(int valor, char tipo)
```

```
void desempilha(char tipo)
{
        if(pilha[topo].tipo != tipo)
                yyerror("Desempilhamento ERRADO!");
        return pilha[topo–].valor;
}
```

lexico.l - aumentar a palavra chave

- ➔ func = contexto local
- ➔ retorne = retorna token (a se renomear)
- ➔ func e fim func = retorna token
- ➔ os tokens devem ser nomeados tambem depois

```
// acrescentar a palavra chave retorne



retorne         return T_RETORNE
func            return T_FUNC
fimfunc         return T_FIMFUNC
```

sintaxico.y -

[comentário abaixo de %start programa]
%expect 1

```
// acrescentar os tokens para as palavras chave retorne, func, fimfunc
%token T_RETORNE
%token T_FUNC
%token T_FIMFUNC
```

programa (depois de 'variaveis', após prinf AMEM, antes de `funcoes`):

```
// acrescentar as funcoes
```

lista_variaveis (após as duas partes de tipo):

```
// elemTab.esc= escopo;
```

lista de comandos:

comando

```
        : entrada_saida
        | repeticao
        | selecao
        | atribuicao
        | retorno
        ;

retorno
        T_RETORNE expressao
        // deve gerar (depois da trad. da expressao)
        // ARZL (valor de retorno), DMEM (se tiver variavel local)
        // RTSP n
        ;

// regra para as funcoes

funcoes
        : /* vazio */
        | funcao funcoes
        ;

funcao
        : T_FUNC tipo T_IDENTIF T_ABRE parametros T_FECHA
        variaveis T_INICIO lista_comandos T_FIMFUNC
        ;

parametros
        : /* vazio */
        | parametro parametros
        ;

parametro
        : tipo T_IDENTIF
        ;
```

obs: parâmetros não vão ter mecanismo

```
lista_comandos
        : /* vazio */
        | comando lista_comandos
        ;

identificador
        : T_IDENTIF
        ;

// A funcao eh chamada como um termo numa expressao
chamada
```

```
        : // sem parenteses he uma variavel
        | T_ABRE lista_argumentos T_FECHA
        ;

lista_argumentos
        : /* vazio */
        | expressao lista_argumentos
        ;

termo
        : identificador chamada
        …
        obs: a parte seguinte desta função será colocada em outro local (deve ser
comentada)


Alteração da Pilha:

????
empilha(contaVar, ‘n’);

escrita
desempilha(‘t’)

repeticao
empilha(rotulo, ‘r’)
empilha(rotulo, ‘r’)
desempilha(‘r’)
desempilha(‘r’)

selecao
desempilha(‘t’);
empilha(rotulo, ‘r’)
empilha(rotulo, ‘r’)
```

MAIS DICAS

```
função
        : T_FUNC tipo T_IDENTIF     [atomo]
        { inserir nome, tipo, cat, rotulo da funcao, tabela de simbolos

T_ABRE parametros T_FECHA
        {ajustar_parametros()}

strcpy(elemTab.id, atomo);
elemTab.tip = ‘tipo’;
```

```
elemTab.cat = ‘f’;
elemTab.rot = ++rotulo;
fprintf(yyour, “L%d\t ENSP\n”, rotulo);

parametro
        : tipo T_IDENTIF
        {cadastrar o parametro}

func inteiro fi (inteiro a inteiro b)

fi -5
a -4
b -3
```

TAREFAS

1- Tabela de Símbolo (definição e chamada, durante a declaração)
2- Chamada de Função

```
expressão:
        …
        | termo
        ;

termo: identificador chamada
        | T_NUMERO
        …
        ;

identificador
        : T_IDENTIF
                {...}
        ;
chamada
        : [simbolo lambda]
        | T_ABRE
        argumentos
        | T_FECHA

funcoes
        | [simbolo lambda]
        | funcao funcoes
        ;

funcao
        : T_FUNC T_TIPO T_IDENTIF
        T_ABRE lista_parametros T_FECHA {ajustar_parametros();}
```

variaveis
T_INICIO
lista_comandos
T_FIMFUNC{...}

ajustar_parametros(); = [atribui o endereço aos parâmetros da função]

3- Chamada Retorne
4- Verificação de Tipos

func inteiro f1(inteiro A logico B)

tab.simbolos

#	id	esc	rot	tipo	cat	npar	par
x	f1	G	-5	INT	FUN	2	[INT, LOG]
x+1	A	L	-4	INT	PAR	-	-
x+2	B	L	-3	LOG	PAR	-	-

comando retorne
…
comando
: leitura_escrita
| atribuicao
| selecao
…
| retorne
;

retorne
: T_RETORNE expressao
{verificar se está no escopo local}
;

obs: retorne só pode ser usado em escopo local, em contexto global daria erro. Pode-se fazer isso com um indicador local/global

…
funcao
:T_FUNC tipo T_IDENTIF
T_ABRE lista_parametros T_FECHA
variaveis
T_INICIO

lista_comandos
T_FIMFUNC

ARZL *
DMEM *
RTSP *

Alternativa para desempilhamento de símbolos (facilita depuração)
- Modificação (utils)