# LISP-ALGOL

AUTOR: ANTONIO CARLOS BADELUCCI

ORIENTADOR: PROF. DR. VALDEMAR SETZER

DISSERTACAO APRESENTADA A COMISSAO DE POS-GRADUACAO DO INSTITUTO DE MATEMATICA, ESTATISTICA E CIENCIA DA COMPUTACAO DA UNIVERSIDADE ESTADUAL DE CAMPINAS PARA CUMPRIMENTO DE REQUISITOS PARCIAIS PARA OBTENCAO DO TITULO DE MESTRE EM CIENCIA DA COMPUTACAO.

LISP ALGOL

DEDICO ESTE TRABALHO A MEU IRMAO, HOMENAGEM POSTUMA A SUA SERIEDADE PROFISSIONAL, HONESTIDADE CIENTIFICA E DEDICACAO SOCIAL.

AGRADECO A MEUS PAIS PELO PERMANENTE APOIO E CONFIANCA E A DIVINA PROVIDENCIA QUE MUITO CONTRIBUIU PARA A CONCRETIZACAO DESTE TRABALHO.

# PREFACIO

ESTE TRABALHO VISA SER UM SUBSIDIO AQUELES QUE SE INTERESSAM, NA CIENCIA DA COMPUTACAO, PELA RESOLUCAO DE PROBLEMAS REFERENTES A MANIPULACAO SIMBOLICA ESTRUTURADA.

O SISTEMA APRESENTADO NAO PRETENDE SER COMPLETO OU FINAL; SUA CONCEPCAO PREVE E ESPERA COMPLEMENTOS.

ESPERO COM ELE ABRIR PERSPECTIVAS DE DESENVOLVIMENTO NA AREA DAS LINGUAGENS E NA RESOLUCAO DE ALGUNS PROBLEMAS DE LINGUISTICA COMPUTACIONAL E INTELIGENCIA ARTIFICIAL.

O SISTEMA APRESENTADO VEM PRECEDIDO DE UMA CRITICA DE OUTROS SISTEMAS EM ASPECTOS EM QUE ESTES NOS PARECERAM INEFICIENTES OU MAL CONCEBIDOS; PREOCUPAMO-NOS EM FORNECER AO USUARIO A MAXIMA FLEXIBILIDADE NA UTILIZACAO DOS RECURSOS DO SISTEMA, COM UM MINIMO DESGASTE NA INTRODUCAO DE NOVOS CONCEITOS  E TENDO SEMPRE EM CONTA UMA FACILITACAO DO PROCESSO DE APRENDIZAGEM.

ESSA PREOCUPACAO VEM ACOMPANHADA DE OUTRA: NO DESENVOLVIMENTO DO SISTEMA TIVEMOS SEMPRE PRESENTE UMA OTIMA UTILIZACAO DE MEMORIA E A MINIMIZACAO DOS TEMPOS DE EXECUCAO.

A CONCEPCAO GERAL E ACOMPANHADA DE LISTAGENS E EXPLICACOES SOBRE CADA PECA PARTICULAR DO SISTEMA.

SUMARIO

# I. UMA ABORDAGEM SOB O PONTO DE VISTA DAS LINGUAGENS DE PROGRAMACAO.

## I. 1- CONSIDERACOES GERAIS.

CONSIDERACOES GERAIS

AS LINGUAGENS DE COMPUTACAO SAO PROJETADAS PARA DESCREVER A SOLUCAO E PERMITIR SUA EXECUCAO EM MAQUINA.

AS LINGUAGENS DE ALTO NIVEL CONSEGUIRAM SUA MEIA DE SUCESSO PRINCIPALMENTE NESSE ESPIRITO. CADA UMA DETERMINADA CLASSE DE PROBLEMAS. O FORTRAN E O ALGOL FORAM CRIADOS PARA RESOLVER PROBLEMAS NUMERICOS. O COBOL FOI ORIENTADO PARA O PROCESSAMENTO DE ARQUIVOS. O LISP FOI CRIADO PARA PERMITIR O PROCESSAMENTO DE LISTAS, ETC.

APOS UM CERTO TEMPO SE VERIFICOU QUE A EXISTENCIA DE UM GRANDE NUMERO DE LINGUAGENS LEVAVA A UM DESGASTE ENORME DE ESFORCO HUMANO E A IMPOSSIBILIDADE DE UM NUMERO SEMPRE CRESCENTE DE OPERACOES SOBRE CADA UMA DELAS. UMA DAS ORIENTACOES QUE SE SEGUIU A ESSA CONSTATACAO FOI A DE PROJETAR UMA LINGUAGEM QUE TIVESSE VARIOS FINS E, ASSIM, QUE PUDESSE SERVIR A RESOLUCAO DE PROBLEMAS NUMERICOS, PROCESSAMENTO DE LISTAS, MANIPULACAO DE BITS E C. TAL ORIENTACAO FOI SEGUIDA PELO GRUPO DE TRABALHO DA IFIP E PELOS EUROPEUS NO ALGOL-68. FALHAS NA CONCEPCAO DESSAS LINGUAGENS NAO SE DEVEM AO FATO DE TENTAR RESOLVER DIFERENTES SITUACOES MAS NO EXCESSO DE POSSIBILIDADES, IMPLICANDO EM DIFICULDADES DE CONSTRUCAO E LENTIDAO DE EXECUCAO DOS RESPECTIVOS COMPILADORES.

CONCOMITANTEMENTE A ESSA IDEIA SURGIU OUTRA QUE PARECE ESTAR TENDO MAIOR SUCESSO E CONSISTE EM EXPANDIR UMA LINGUAGEM JA EXISTENTE, DOTANDO-A DE INSTRUMENTOS QUE PERMITAM EXECUTAR OPERACOES QUE O DESENVOLVIMENTO DA COMPUTACAO TORNA NECESSARIAS.

NO ALGOL DO PDP-10 SEGUIU-SE ESTA LINHA. ALEM DE SIMPLIFICACOES EM ALGUNS PONTOS DAS DEFINICOES SINTATICAS DO ALGOL-60 INTRODUZI-

RAM-SE, ATRAVES DA PORTA ABERTA DA FUNCOES DE BIBLIOTECA OPERACOES COM STRINGS E BITS.

DESDE HA MUITO TEMPO TEM-SE CLARO QUE DETERMINADOS TIPOS DE PROBLEMAS IMPLICAM UMA ESTRUTURACAO DOS DADOS; UMA ORGANIZACAO HIERARQUICA OU SEQUENCIAL E COMUM A VARIAS CLASSES DE DADOS E ALGORITMOS DA COMPUTACAO.

ESSES PROBLEMAS SE ENTRELACAM COM OUTRA CLASSE DE PROBLEMAS DE ATUALIDADE E IMPORTANCIA NA AREA DOS COMPUTADORES, QUE E A MANIPULACAO SIMBOLICA.

ESSA PREOCUPACAO (MANIPULACAO DE DADOS SIMBOLICOS ESTRUTURADOS) TEM GERADO SIMPOSIOS E CONGRESSOS ONDE SE TEM PROCURADO CLARIFICAR E ESTABELECER ALGUMAS IDEIAS BASICAS SOBRE OS TEMAS. AS DISCUSSOES EM GERAL PAUTAM-SE POR DIVIDIR AS QUESTOES EM 2 GRANDES AREAS: UMA, A QUE DEFINE A CAPACIDADE DE EXPRESSAO DA LINGUAGEM; A OUTRA QUE SE ENCARREGA DOS PROBLEMAS DE IMPLEMENTACAO.

UMA COMPILACAO DOS PROBLEMAS MAIS SIGNIFICATIVOS NA CONCEPCAO E IMPLEMENTACAO DE UMA LINGUAGEM PARA MANIPULACAO SIMBOLICA DE DADOS ESTRUTURADOS FOI FEITA POR FOSTER ( VIDE BIBLIOGRAFIA).

FOSTER ASSEVERA EM SEU LIVRO SOBRE PROCESSAMENTO DE LISTAS QUE UMA LINGUAGEM QUE SE PROPONHA A MANIPULAR LISTAS DEVE RESPONDER AS SEGUINTES PERGUNTAS:

1. WHAT SORT OF STRUCTURE DOES THE LANGUAGE ITSELF HAVE?
2. HOW LARGE A LIBRARY OF ROUTINES IS AVAILABLE?
3. ARE THE PRIMITIVE LIST OPERATIONS AVAILABLE TO THE USER?
4. IS THE REPRESENTATION OF LIST FIXED BY THE SYSTEM, OR HAS THE USER THE ABILITY TO MAKE UP NEW REPRESENTATIONS?
5. WHAT METHOD OF GARBAGE COLLECTION IS USED AND HOW MUCH

RESPONSABILITY DOES IT GIVE THE USER?

6. IS RECURSION EASY?

7. HOW EASY IS IT TO DO ARITHMETIC? IS THERE A LARGE LIBRARY OF ORDINARY NUMERICAL ROUTINES?

8. WHAT ARE THE STANDARD NOTATIONS FOR LISTS USED BY THE PROGRAMMER? ARE THERE ROUTINES FOR READING AND PRINTING LISTS IN A CLEAR WAY?

9. ARE THERE STANDARD METHODS FOR REPRESENTING LISTS ON BACKING STORE?

10. HOW FAST IS THE PROGRAM PRODUCED?

11. IS AN INTERPRETER OR A COMPILER USED?

12. HOW ARE PROGRAMS STORED IN THE COMPUTER?

13. ARE COMMON SUB-LISTS ALLOWED?

14. ARE CIRCULAR LISTS ALLOWED?

VARIAS LINGUAGENS EXISTEM QUE DAO RESPOSTAS A ESSAS PERGUNTAS; AS MAIS IMPORTANTES SAO AS DA LINHAGEM IPL, O SLIP E O LISP.

AS LINGUAGENS DA LINHAGEM IPL SAO DE BAIXO NIVEL, COM DA ORDEM DE 100 A 150 INSTRUCOES PARA A REALIZACAO DE OPERACOES ELEMENTARES EM LISTAS. E UMA LINGUAGEM BEM DOCUMENTADA E AMPLAMENTE IMPLEMENTADA; NO ENTANTO E ENCARADA COMO SUPERADA VISTO SER DE BAIXO NIVEL, NAO RECUR_ SIVA E REQUERER INTERPRETADOR.

O SLIP CONSISTE DE UM CONJUNTO DE ROTINAS EM FORTRAN. DESTA FORMA TODO O PODER DE FAZER ARITMETICA DO FORTRAN E DISPONIVEL E PODE SER APROVEITADO. O PROCESSAMENTO DAS LISTAS NAO E MUITO RAPIDO POIS EMBORA O SISTEMA SEJA COMPILADOR E NAO INTERPRETADOR, AS OPE_ RACOES SAO EXECUTADAS ATRAVES DA CHAMADA DE SUBROTINAS E NAO POR COMPILACAO DIRETA.

OS PROGRAMAS E OS DADOS SAO ESTRUTURAS RADICALMENTE DIFERENTES NO SLIP E ASSIM HA GRANDE DIFICULDADES EM FAZER UM PROGRAMA AUTOMODIFICAVEL.

UM GRANDE NUMERO DE ROTINAS E DISPONIVEL NO SISTEMA, CHEGANDO A 100 EM ALGUMAS INTERPRETACOES. AS CELULAS CONTEM TRES ITENS. DOIS DELES SAO SEMPRE APONTADORES E INDICAM A PROXIMA CELULA ABAIXO NA LISTA E A PROXIMA SUPERIOR A ELA.

O TERCEIRO ITEM PODE SER UM NUMERO OU UM OUTRO APONTADOR, DESTA VEZ UM PARA O ELEMENTO DA LISTA.

TODA LISTA TEM UMA CELULA CABECA DE LISTA, PELA QUAL A LISTA E CONHECIDA. TODOS OS PONTEIROS PARA A LISTA SAO REALMENTE APONTADORES PARA O CABECA DE LISTA. ELE CONTEM TRES ITENS DE INFORMACAO, 2 APONTADORES, UM PARA O TOPO DA LISTA E O OUTRO PARA O FIM DELA, E UM NUMERO QUE CONTA QUANTAS VEZES AQUELA LISTA E USADA COMO SUB-LISTA( (PARA FINS DE UTILIZACAO NO GARBAGE COLLECTOR).

AS OPERACOES PRIMITIVAS SAO DISPONIVEIS PARA O USUARIO. AS MAIS ELEMENTARES SAO AS SEGUINTES:

ID(CELL) TEM COMO VALOR O CAMPO IDENTIFICADOR DA CELULA, QUE E UM CAMPO DE 2 BITS INDICANDO A NATUREZA DO ITEM.

LNKL(CELL) TEM COMO VALOR O PONTEIRO ESQUERDO DA CELULA.

LNKR(CELL) TEM COMO VALOR O PONTEIRO DIREITO DA CELULA.

SETDIR(I, L, R, CELL)
COLOCA I NO CAMPO IDENTIFICADOR, L NO APONTADOR ESQUERDO, R NO APONTADOR DIREITO DA CELULA.

SETDIR(DATUM, CELL)
COLOCA DATUM NA CELULA CELL.

CONT(A) TEM COMO VALOR A INFORMACAO NA CELULA CUJO ENDERECO DE MAQUINA E A.

INHALT(A) TEM COMO VALOR A INFORMACAO DA PALAVRA CUJO ENDERECO DE MAQUINA E A.

A LISTA LIVRE E MANIPULADA POR UM GARBAGE COLLECTOR BASTANTE INTERESSANTE QUE FAZ USO DOS CONTADORES DAS CABECAS DE LISTAS E DEVOLVE AS CELULAS COM UM MINIMO DE TRABALHO.

A CONSTRUCAO DE UMA PILHA PARA PERMITIR RECURSIVIDADE NAO OFERECE MAIORES PROBLEMAS.

NOSSO DESINTERESSE PELA SOLUCAO SLIP, ALEM DE CONSIDERACOES DE ESPACO UTILIZADO (TRABALHA COM 3 CAMPOS DE APONTADORES )VEM TAMBEM DA CONSIDERACAO DE QUE O FORTRAN COMO CONCEPCAO DE LINGUAGEM DEIXA MUITO A DESEJAR VISTO SER NAO RECURSIVO, NAO ESTRUTURADO E DESPROVIDO DE UMA DEFINICAO SINTATICA FORMAL.

NAO OBSTANTE, CONSIDERAMOS O SLIP UMA SOLUCAO VALIDA PARA O PROBLEMA DE PROCESSAMENTO DE LISTAS VISTO O AMPLO USO DO FORTRAN NOS MEIOS TECNICOS E CIENTIFICOS.

ALEM DO MAIS ACHAMOS INTERESSANTE A IDEIA DE EXTENDER UMA LINGUAGEM EXISTENTE AO INVES DE CRIAR UMA NOVA COM TODO O PROBLEMA DE NOVAS DEFINICOES SINTATICAS. NOSSAS OPCOES, NO ENTANTO, ACOMPANHANDO ESSE PONTO DE VISTA, SE ENCAMINHARAM PARA O ALGOL.

COM ESSE INTUITO, PASSAREMOS A FALAR SOBRE O LISP QUE NOS PARECE TER SIDO O PADRAO A PARTIR DO QUAL SE PAUTOU O DESENVOLVIMENTO DAS LINGUAGENS VOLTADAS PARA O TRATAMENTO ESTRUTURADO DE SIMBOLOS.

AS PROPOSTAS INICIAIS QUE LEVARAM A CRIACAO DA LINGUAGEM LISP (LIST PROCESSOR) SE TRADUZIAM BASICAMENTE NO DESEJO DE TER UMA LIN-

GUAGEM DE PROGRAMACAO QUE PERMITISSE A MANIPULACAO DE INFORMACAO NAO_ NUMERICA RELACIONADA COM A MATEMATICA. EXPLICITAMENTE, MCCARTHY, SEU CRIADOR, PROPOS UM SISTEMA QUE, ENTRE OUTRAS COISAS, SERVISSE PARA:

" WRITING A PROGRAM TO CHECK PROOFS IN A CLASS OF FORMAL LOGI_ CAL SYSTEMS.

WRITING PROGRAMS FOR FORMAL DIFFERENTIATION AND INTEGRATION.

WRITING PROGRAMS TO REALIZE VARIOUS ALGORITHMS FOR GENERATING PROOFS IN PREDICATE CALCULUS. "

ESSE SISTEMA FOI IMPLEMENTADO NO MIT PELO SEU PROPRIO IDEALI_ ZADOR E EM VARIAS OUTRAS PARTES DO MUNDO POR VOLTA DE 1961 E ANOS SUBSEQUENTES. CUMPRE RESSALTAR QUE A TOTALIDADE DOS OBJETIVOS A QUE A LINGUAGEM SE PROPUNHA FOI ATINGIDA.

OUTROSSIM, NUMA REVISAO HISTORICA RECENTE DAS LINGUAGENS DE PROGRAMACAO POR 2 DOS MAIS REPUTADOS ESPECIALISTAS DA TEORIA E PRO_ JETO DE LINGUAGENS DE PROGRAMACAO, UM DELES NEM CITA O LISP ENQUANTO QUE O OUTRO (SAMMET) REFERE-SE BREVEMENTE A ELA COMO APENAS USADA PELOS PESQUISADORES EM INTELIGENCIA ARTIFICIAL.

ESSA DECADENCIA, QUE E SIMULTANEA COM O NOVO ENFOQUE DE CEN_ TRALIZACAO DAS CARACTERISTICAS ESPECIFICAS DE VARIAS LINGUAGENS EM LINGUAGENS MAIS COMPLETAS NAO DEVE LEVAR A UM MENOSPREZO DO LISP.

O LISP REPRESENTOU E AINDA REPRESENTA UM PASSO ADIANTE NA TEO_ RIA DAS LINGUAGENS E DEVE SER ESTUDADO A FUNDO POR QUEM QUER QUE SE INTERESSE PELO SETOR

UMA PARTICULAR IDEIA QUE NOS AGRADA BASTANTE NA LINGUAGEM E A ELIMINACAO DO COMANDOS DE DESVIO INCONDICIONAL, QUE PERMITE UMA ME_ LHORIA CONSIDERAVEL NA INTELECCAO DE PROGRAMAS. VEJAMOS NAS PAGINAS

QUE SE SEGUEM UM RESUMO DA LINGUAGEM LISP E UMA POSTERIOR CRITICA A ALGUNS DOS CONCEITOS POR ELA INTRODUZIDOS.

## I.2- RESUMO SUCINTO DA LINGUAGEM LISP.

## RESUMO SUCINTO DA LINGUAGEM LISP

OS PROGRAMAS NA LINGUAGEM LISP SAO FUNCOES QUE UTILIZAM FUNCOES DE BASE PREVIAMENTE DEFINIDAS E EMBUTIDAS NO SISTEMA; SEUS DADOS SAO LISTAS QUE CONTEM OS SIMBOLOS A SEREM MANIPULADOS.

EM GERAL OS PROGRAMAS SAO EXECUTADOS DE MODO INTERPRETATIVO. O INTERPRETADOR LISP ARMAZENA A INFORMACAO SIMBOLICA SOBA FORMA DE FOLHAS DE ARVORES BINARIAS E EXECUTA OPERACOES SOBRE ESSAS ARVORES SEGUNDO O PROGRAMA.

CARACTERISTICA IMPORTANTE DA LINGUAGEM E A RECURSIVIDADE.

NESTE RESUMO DA LINGUAGEM LISP PROCURAREMOS SEMPRE QUE POSSIVEL USAR A NOTACAO DE BACKUS; FAREMOS UM APANHADO APENAS DA M-LINGUAGEM, VISTO QUE A S-LINGUAGEM VISA APENAS ATENDER A ASPECTOS PRATICOS DE IMPLEMENTACAO QUE NAO INTERESSAM A UM ESTUDO DO PONTO DE VISTA DAS LINGUAGENS.

DEFINIREMOS:

1. ATOMO

SEGUNDO MCCARTHY, E UMA SEQUENCIA DE CARACTERES, COM POSSIVEIS BRANCOS INDIVIDUAIS INTERCALADOS.

NA NOTACAO DE BACKUS:

<ATOMO> ::= <SIMBOLO NAO BRANCO> < NUCLEO DE ATOMO>
<SIMBOLO NAO BRANCO>

<NUCLEO DE ATOMO> ::= <BRANCO> <ATOMO>
<SIMBOLO NAO BRANCO> <NUCLEO DE ATOMO>
<SIMBOLO NAO BRANCO>

EXEMPLO: ATOMO SIMBOLICO DO LISP

E IMPORTANTE NOTAR QUE NA IMPLEMENTACAO DO MIT, EM OPO-

SICAO AO TEXTO ORIGINAL ONDE DEFINIU O LISP, O BRANCO SEPARADOR FOI ELIMINADO. WEISSMAN DIVIDE OS ATOMOS EM ATOMOS NAO NUMERICOS(SINTAXE ANALOGA A DE UM IDENTIFICADOR EM ALGOL) E ATOMOS NUMERICOS(SINTAXE ANALOGA A DOS NUMEROS EM FORTRAN). EM NENHUM CASO E PERMITIDA A INSERCAO DE BRANCOS.

2. S-EXPRESSAO

E UM ATOMO OU PAR PUNTUADO DE S-EXPRESSOES. NA NOTACAO DE BACKUS:

<S-EXPRESSAO> ::= <ATOMO>

( <S-EXPRESSAO> . <S-EXPRESSAO> )

EXEMPLOS: SIMBOLO

( ESQUERDA . DIREITA )

3. EQUIVALENCIA ENTRE NOTACAO DE PONTO E NOTACAO DE LISTA

A S-EXPRESSAO ( ELEMENTO . NIL ) E EQUIVALENTE A LISTA ( ELEMENTO ).

A S-EXPRESSAO ( ELEMENTO1 . ( ELEMENTO2 . ( ...

... ( ELEMENTO . NIL ) ... ))) E EQUIVALENTE A LISTA (ELEMENTO1 ELEMENTO2 ... ELEMENTON ).

NESTE PONTO CUMPRE LEMBRAR A EXISTENCIA DA LISTA VAZIA, CONCEITUALMENTE IGUAL AO ATOMO NIL.

4. PREDICADOS

SAO FUNCOES CUJOS ARGUMENTOS SAO S-EXPRESSOES ( OU PODEM SER REDUZIDOS A S-EXPRESSOES) COM VALORES NO ESPACO T OU F.

OS PREDICADOS ELEMENTARES EMBUTIDOS NO SISTEMA SAO:

ATOM[X] : T SE X FOR ATOMO

F CASO CONTRARIO

EXEMPLO: ATOM[SIMBOLO]=T

ATOM[(A. B)]=F

EQ[X;Y]: T SE X IGUAL A Y

(X,Y ATOMOS)

F CASO CONTRARIO

EXEMPLO: EQ[A;A]=T

EQ[A;B]=F

ALEM DESSES PREDICADOS FUNDAMENTAIS ESTAO DEFINID

SEGUINTES PREDICADOS NO LISP:

NUMBERP[X]: T SE X FOR NUMERICO

X ATOMICO

F CASO CONTRARIO

NULL[X]: T SE X IGUAL A NIL

F CASO CONTRARIO

OS SEGUINTES PREDICADOS SAO TODOS NUMERICOS:

FIXP[N]: T SE N FOR INTEIRO

F CASO CONTRARIO

FLOAT[N]: T SE N E NUMERO DE PONTO FLUTUANTE

F CASO CONTRARIO

ZEROP[N]: T SE N E ZERO

F CASO CONTRARIO

MINUSP[N]: T SE N E NEGATIVO

F CASO CONTRARIO

GREATERP[N1;N2]: T SE N1 E MAIOR QUE N2

F CASO CONTRARIO

LESSP[N1;N2]: T SE N1 MENOR QUE N2

F CASO CONTRARIO

5. FUNCOES

O SENTIDO E ANALOGO AO MATEMATICO; OS ARGUMENTOS SAO S_EXPRESSOES OU FUNCOES QUE LEVAM A S-EXPRES_SOES.

AS FUNCOES ELEMENTARES EMBUTIDAS NO SISTEMA SAO AS SEGUINTES:

CAR[X] = E1 SE X = (E1 . E2) OU X = ( E1 E2 ... EN )

CDR[X] = E2 SE X = (E1. E2)

( E2 ... EN ) SE X = ( E1 E2 ... EN )

CONS[X;Y] = (E1 . E2 ) SE X = E1 E X = E2

EXEMPLOS: CAR[(A. B)] = A

CAR[(A B C D )] = A

CDR[(A. B)] = B

CDR[( A B C D )]= ( B C D )

CONS[A;B] = ( A . B )

ALEM DESSAS FUNCOES , PRIMORDIAIS NOS SISTEMAS LISP SAO COSTUMEIRAS AS SEGUINTES:

EF[X] = TEM COMO VALOR O PRIMEIRO SIMBOLO ATO_MICO DA S-EXPRESSAO DESIGNADA POR X.

EXEMPLO: EF[(A B C)] = A

ASSOC[X;Y] = TEM COMO VALOR A LISTA ASSOCIADA DAS

CUJO I-ESIMO ELEMENTO E O PAR PUNTUADO FORMADO PELOS ELE
MENTOS DE ORDEM I DE SEUS ARGUMRNTOS.

EXEMPLO: PAIR[ ( X Y Z ); ( A B C ) ]=

( ( X . A ) ( Y . B ) ( Z . C ) )

ASSOC[X;A] = NESTA FUNCAO O ARGUMENTO A E UMA LIS_
TA DO TIPO ( ( U1 V1 ) (U2 V2 ) ... ( UN VN ) ) E X E
UM DOS UI; O VALOR DA FUNCAO E O VI CORRESPONDENTE.

EXEMPLO: ASSOC[Y; ( ( X A ) ( Y ( C D ) )= ( C D )

SUBLIS[A;Y] = O ARGUMENTO A E UMA LISTA DO TIPO:
( ( U1 . V1 ) ( U2 . V2 ) ... ( UN . VN ) ) ON_
DE OS UI SAO ATOMICOS. Y E UMA S-EXPRESSAO QUALQUER.
A FUNCAO SUBLIS TRATA OS UI COMO VARIAVEIS, QUANDO
ELES OCORREM EM Y E SUBSTITUE_OS PELOS CORRESPONDENTES
VI DA LISTA.

EXEMPLO: SUBLIS[ ( ( X . CAMOES ) ( Y . ( OS LUSIA_
DAS ) ) ) ; ( X ESCREVEU Y ) ] = ( CAMOES ESCREVEU
( OS LUSIADAS ) ).

SUBST[ X ; Y ; Z] = ESTA FUNCAO DA O RESULTADO DA
SUBSTITUICAO DA S-EXPRESSAO X PARA TODAS AS OCORREN_
CIAS DO SIMBOLO ATOMICO Y NA S-EXPRESSAO Z.

EXEMPLO: SUBST[ ( PEDRO . PAULO ) ; X ;
( X COLOU NO EXAME E X FOI PEGO EM FLAGRANTE) ] =
( ( PEDRO . PAULO ) COLOU NO EXAME E ( PEDRO . PAULO )
FOI PEGO EM FLAGRANTE ) .

ALEM DESSAS FUNCOES , CLASSICAS E TRATANDO DE OPE_
RACOES EM LISTAS, EXISTE UM NUMERO ENORME DE FUNCOES
PARA A EXECUCAO DE OPERACOES ARITMETICAS:

PLUS[X1;X2; ...XN]: E UMA FORMA ESPECIAL COM UM NUMERO INDEFINIDO DE ARGUMENTOS, O VALOR DA QUAL E A SOMA ALGEBRICA DOS VALORES DOS ARGUMENTOS.

DIFFERENCE[X;Y] :POSSUE DOIS ARGUMENTOS E E DEFINIDA COMO A DIFERENCA DOS VALORES DE SEUS ARGUMENTOS.

MINUS[X]: TEM UM SO ARGUMENTO E O RESULTADO E O VALOR X COM SINAL TROCADO.

TIMES[ X1; X2; ... XN ]: TEM UM NUMERO INDEFINIDO DE ARGUMENTOS E SEU VALOR E O PRODUTO DOS VALORES DE SEUS ARGUMENTOS.

ADD1[X]: TEM UM SO ARGUMENTO E COMO VALOR O VALOR DO ARGUMENTO ACRESCIDO DE UMA UNIDADE.

SUB1[X]: TEM COMO ARGUMENTOU UMA SO VARIAVEL E COMO VALOR O VALOR DO ARGUMENTO DIMINUIDO DE UMA UNIDADE.

MAX[ X1; X2; ... XN ] : ADMITE UM NUMERO INDEFINIDO DE ARGUMENTOS E TEM COMO VALOR O MAIOR VALOR ENTRE SEUS ARGUMENTOS.

MIN[ X1; X2 ; ... XN ] : ADMITE NUMERO QUALQUER DE ARGUMENTOS E TEM COMO VALOR O MENOR VALOR DOS SEUS ARGUMENTOS.

QUOTIENT[X;Y] : FORNECE O QUOCIENTE ENTRE OS DOIS VALORES DE SEUS ARGUMENTOS.

REMAINDER[X;Y]: FUNCAO DE 2 ARGUMENTOS, FORNECE COMO VALOR O RESTO DA DIVISAO INTEIRA DE X POR Y.

DIVIDE[X;Y]: FORNECE COMO RESULTADO UMA LISTA CUJO PRIMEIRO ELEMENTO ATOMICO E NUMERICO E QUOTIENT[X;Y] E CUJO SEGUNDO ELEMENTO, TAMBEM ATOMO NUMERICO E

REMAINDER[X ; Y].

EXPT[X ; Y]: TEM 2 ARGUMENTOS O PRIMEIRO NAO NEGATI
VO SE Y NAO FOR INTEIRO E FORNECE COMO VALOR X ELE_
VADO A Y.

SQRT[X ] : TEM COMO VALOR A RAIZ QUADRADA
DO VALOR ABSOLUTO DO ARGUMENTO.

RECIP[X] : DEVOLVE COMO VALOR O RECIPROCO DO SEU AR_
GUMENTO.

ABSVAL[ X ] : O VALOR DA FUNCAO ABSVAL E O MODULO DO
VALOR DO SEU ARGUMENTO.

FLOAT[X]: FAZ A FLUTUACAO DO VALOR DO ARGUMENTO X, OU
SEJA , TEM COMO VALOR O EQUIVALENTE EM PONTO FLUTU_
ANTE DO VALOR DO ARGUMENTO.

CONVEM RESSALTAR QUE AS FUNCOES FICAM INDEFINIDAS PARA
ATOMOS NAO NUMERICOS. OS ARGUMENTOS PODEM TER VA_
LORES INTEIROS OU EM PONTO FLUTUANTEOU EM AMBOS
MISTURADAMENTE. SE TODOS OS VALORES DOS ARGUMENTOS
SAO INTEIROS ENTAOO VALOR DA FUNCAO ARITMETICA SERA IN_
TEIRO; SE PELO MENOS UM ARGUMENTO TEM VALOR NUMERI_
CO EM PONTO FLUTUANTE ENTAO O VALOR DA FUNCAO SERA
EM PONTO FLUTUANTE.

AS FUNCOES DEFINIDAS ATRAS, BEM COMO OS PREDICADOS
SAO AQUELES QUE SE ENCONTRAM COM MAIS FREQUENCIA NAS
IMPLEMENTACOES; ALGUNS PODEM TER SIDO OMITIDOS OU
TALVEZ OUTROSINSERIDOS SEJAM DE POUCA IMPORTANCIA.
NOSSA MAIOR PREOCUPACAO E DAR UMA VISAO GERALE
GENERICA DO LISP; NAO PRETENDEMOS SER EXAUSTIVOS.

6. FORMAS

SAO MANEIRAS DE APRESENTAR E DAR NOME A FUNCOES DE VARIAS VARIAVEIS. SEJA POR EXEMPLO A FUNCAO

F = Y^2+X

NA NOTACAO LAMBDA, DEVIDA A CHURCH, NOS A ESCREVERIAMOS DA SEGUINTE FORMA:

F = LAMBDA ( (X,Y) , Y^2+X )

ESSA MANEIRA DE COLOCAR AS COISAS EVITA AMBIGUIDADES NA ATRIBUICAO DE VALORES PARTICULARES A FUNCAO.

POR EXEMPLO, COM A PRIMEIRA NOTACAO, O CALCULO DE F( 3,4 ) PODERIA NOS LEVAR ALTERNATIVAMENTE AOS RESULTADOS 13 OU 19 DEPENDENDO DA FORMA COMO FIZESSEMOS A ASSOCIACAO DOS PARAMETROS. COM A NOTACAO LAMBDA TERIAMOS O VALOR UNIVOCO E NAO SUJEITO A DUVIDAS 19.

A NOTACAO LABEL PERMITE DAR NOME AS FUNCOES.

POR EXEMPLO:

LABEL ( FUNCAO , LAMBDA ( (X ; Y ) Y^2+X ) )

COM ESSA FORMA DE APRESENTAR AS COISAS TORNAMOS INAMBIGUA NOSSA FUNCAO FUNCAO E ELA PASSA A TER O NOME FUNCAO.

7. EXPRESSOES CONDICIONAIS

SAO EXPRESSOES DA FORMA:

[ P1 -- E1 ; P2 -- E2 ; ... PN -- EN ]

ONDE:

PI: INDICA PREDICADO;

EI: INDICA FUNCAO, FORMA OU EXPRESSAO CONDICIONAL.

NA EXPRESSAO CONDICIONAL, SE O VALOR DO PREDICADO PI FOR T, PASSAREMOS A CALCULAR EI; CASO CONTRARIO CALCULAREMOS O PREDICADO PI+1. O PROCESSO VAI CONTINUA ATE ENCONTRARMOS UM VALOR OU A FUNCAO FICARA INDEFINIDA.

8. PSEUDO-FUNCOES

SAO PROCEDIMENTOD QUE NAO NAO DEVOLVEM VALORES MAS SIM EXECUTAM TAREFAS PARA O SISTEMA.
AS MAIS IMPORTANTES DELAS SAO:
READ: E UMA FUNCAO SEM ARGUMENTOS QUE LE UMA LISTA, ESTRUTURA-A NA MEMORIA E GUARDA COMO VALOR O ENDE_RECO DE INICIO DA LISTA NA MEMORIA.
PRINT: TEM UM ARGUMENTO, CUJO VALOR E UMA S-EXPRESSAO. PRINT FAZ COM QUE ESSA S-EXPRESSAO SEJA IMPRESSA PELO DISPOSITIVO DE SAIDA.
DEFINE: POSSUE COMO ARGUMENTO UMA LISTA DE FUNCOES A SEREM DEFINIDAS. APOS A EXECUCAO DO DEFINE, PODEMOS UTILIZAR A FUNCAO DEFINIDA COMO SE ELA FOSSE UMA FUN_CAO EMBUTIDA. APOS A EXECUCAO DO PROGRAMA, ENTRETAN_TO ELA DEIXA DE SER DISPONIVEL.
PROG: PERMITE-NOS ESCREVER LOOPS. A FORMA PROG E A FERRAMENTA DO LISP QUE PERMITE A EXECUCAO DE ITERACOES. PARECE-NOS ENTRETANTO A NEGACAO DAS QUALI_DADES RECURSIVAS DO LISP.

UM PROGRAMA EM LISP E FORMADO POR UMA FUNCAO, FORMA OU EXPRES_SAO CONDICIONAL.. NAS IMPLEMENTACOES DO LISP A LINGUAGEM ACIMA DES_CRITA DEVE SER TRADUZIDA PARA A S-LINGUAGEM, UM MODO ESPECIAL DE

ESCREVER PROGRAMAS EM LISP, TENDO EM VISTA A INTERPRETACAO DO PRO_
GRAMA. NAO VAMOS APRESENTAR AS REGRAS PARA TAL PASSAGEM
VISTO QUE O PODER DA LINGUAGEM NAO E ALTERADO POR TAL TRANSFORMACAO.

DIRIGIREMOS AGORA NOSSAS ATENCOES DE UMA FORMA CRITICA PARA
O LISP, TENTANDO APREENDER O ESSENCIAL DA LINGUAGEM.

## I.3- CRITICA DA LINGUAGEM LISP.

## CRITICA DA LINGUAGEM LISP

PROCURAREMOS NESTA CRITICA ABORDAR OS PONTOS QUE CONSIDERAMOS FRACOS NA LINGUAGEM, MAS PROCURAREMOS TAMBEM REALCAR SUAS QUALIDADES.

UMA CRITICA APENAS NEGATIVA PODE LEVAR A UMA DESCONSIDERACAO DO REAL VALOR DO OBJETO CRITICADO; NO CASO ESPECIFICO DO LISP TAL ATITUDE SERIA INJUSTA.

A PRIMEIRA CONSIDERACAO A SER FEITA REFERE-SE AO CONCEITO DE ATOMO.

A POSSIBILIDADE DE INSERCAO DE BRANCO SEPARADOR PARECE-NOS FALHA, POIS CRIA DIFICULDADES DE LEGIBILIDADE. O PROPRIO MCCARTHY PARECE TER REVISTO SUA PROPOSICAO INICIAL POIS A INSERCAO DE BRANCO NAO E PERMITIDA NA IMPLEMENTACAO DO MIT.

WEISSMAN NAO PERMITE A INSERCAO DE UM BRANCO SEPARANDO PEDACOS DE UM MESMO ATOMO. SUA PROPOSICAO, MELHOR NESSE PONTO, CRIA DIFICULDADES SE SE DESEJA TRABALHAR COM SIMBOLOS NUMERICOS VISTO QUE O ARMAZENAMENTO DE ATOMOS NUMERICOS E DIFERENTE DO DE ATOMOS NAO NUMERICOS.

UM OUTRO PONTO, E BASTANTE SIGNIFICATIVO, QUE NOS PARECEU ESQUECIDO NO LISP FOI O INTERESSE DE SE TRABALHAR COM SIMBOLOS DOS ATOMOS. NESSE SENTIDO, OS ATOMOS DO LISP SAO BEM ATOMOS, UMA VEZ QUE NAO SAO DIVISIVEIS OU ALTERAVEIS. ESSA FORMA DE COLOCAR AS COISAS TRUNCA A RESOLUCAO DE UMA SERIE DE PROBLEMAS QUE REQUEREM ALTERACOES AO NIVEL DOS SIMBOLOS DOS ATOMOS DURANTE A EXECUCAO DE PROGRAMA.

ESSA FLEXIBILIDADE PARECE-NOS INDISPENSAVEL PARA CERTOS TIPOS DE PROBLEMAS LINGUISTICOS.

OUTRA PARCELA ALTAMENTE CONTESTAVEL DO LISP E A REFERENTE A NOTACAO DE PONTO E EM DECORRENCIA A FUNCAO CONS E O CONCEITO DE S-EXPRESSAO.

EM NOSSO PONTO DE VISTA, A NOTACAO DE PONTO E A FUNCAO CONS SAO SUPERFLUAS E INCOMODAS, UMA VEZ QUE SE PODE TRABALHAR APENAS COM A NOTACAO DE LISTA, DESDE QUE SE TROQUE A FUNCAO CONS POR UMA OUTRA FUNCAO, LIST, QUE TENHA COMO ARGUMENTOS 2 S-EXPRESSOES EM NOTACAO DE LISTA E GERE UMA LISTA EM QUE AS S-EXPRESSOES SAO ORA ELEMENTOS ORA FORNECEDORAS DE ELEMENTOS ( VER ADIANTE ).

O CONCEITO DE S-EXPRESSAO USADO ACIMA DEVE SER ENTENDIDO COMO ATOMO OU LISTA EM QUE LISTA E UM CONJUNTO DE ATOMOS OU LISTAS ENTRE PARENTESES. NUM CAPITULO SEGUINTE APRESENTAREMOS UMA DEFINICAO FORMAL PARA S-EXPRESSAO EM NOTACAO DE LISTA.

ESSAS ALTERACOES, QUE REPRESENTAM ALTERACOES FUNDAMENTAIS NOS 4 PRIMEIROS ITENS DO PARAGRAFO ANTERIOR NAO MUDAM EM NADA O PO_ DER DA LINGUAGEM, AO MESMO TEMPO EM QUE A SIMPLIFICAM BASTANTE. CREMOS QUE UMA PROVA FORMAL DAS ASSERCOES ACIMA PODE SER OBTIDA ATRAVES DE CRITERIOS DE SEMANTICA FORMAL; ISSO FOGE POREM AO AM_ BITO DESTE TRABALHO.

OS PREDICADOS FUNDAMENTAIS, BEM COMO AS FUNCOES FUNDAMENTAIS COM EXCECAO DE CONS, PARECEM-NOS BASTANTE SOLIDOS E FORMAM UM BOM NUCLEO PARA A MONTAGEM DE UMA LINGUAGEM. NOS OS MANTEREMOS NA REALIZACAO DE NOSSO TRABALHO.

AS NOCOES DE FORMA EM NOTACAO LAMBDA E NOTACAO LABEL E A S-LINGUAGEM SO TEM SENTIDO DEVIDO AO USO DE INTERPRETADORES QUE USAM AS FUNCOES APPLY, EVAL, EVALQUOTE. A IDEIA DE FAZER OS PROCE_ DIMENTOS DO LISP NUMA OUTRA LINGUAGEM DE ALTO NIVEL TORNA-AS TOTAL_

MENTE DESNECESSARIAS; ELAS ELAS SAO SUBSTITUIDAS POR OUTRAS EN_
TIDADES DESSAS LINGUAGENS. VOLTANDO NOSSOS RACIOCINIOS PARA O
ALGOL, A NOTACAO LAMBDA E LABEL E SUBSTITUIDA PELA DEFINICAO DOS
PROCEDURES, QUE FAZ A ASSOCIACAO DOS PARAMETROS ATUAIS COM OS
FICTICIOS ORDENADAMENTE, ELIMINANDO AMBIGUIDADES E PERMITINDO A C
CHAMADA DO PROCEDIMENTO PELO IDENTIFICADOR DO PROCEDURE.

A S-LINGUAGEM PERDE SEU SENTIDO POIS EM NAO HAVENDO INTER_
PRETACAO DESAPARECE AS FUNCOES QUE SOLICITAVAM PARES DE S-EXPRESSOES
COMO ARGUMENTOS.

OUTRO PONTO DISCUTIVEL NO LISP E A EXECUCAO DE
OPERACOES ARITMETICAS. O PROPRIO MCCARTHY RECONHECE EXPLICITAMENTE
QUE, APESAR DA POSSIBILIDADE DO SISTEMA OPERAR COM VALORES EM PONTO
FIXO E FLUTUANTE, A ARITMETICA NO LISP E INEFICIENTE.

O CALCULO DE FATORIAL RECURSIVAMENTE E UM OTIMO EXEMPLO DE
COMO NAO DEVEM SER USADOS OS RECURSOS DE UMA LINGUAGEM.

DEVEMOS RESSALTAR QUE, EM SE PENSANDO EM EXTENDER UMA LINGUA_
GEM PREEXISTENTE PARA ATRAVES DELA CONSEGUIR O PROCESSAMENTO DE LIS_
TAS, SOMOS LEVADOS A ABANDONAR ALGUMAS PECAS DA LINGUAGEM ORIGI_
NAL, POR EXEMPLO AS PSEUDO-FUNCOES DEFINE E PROG.

OUTRAS, NO ENTANTO, COMO READ E PRINT CONTINUAM NECESSARIAS.

E IMPORTANTE NAO DEIXAR DE DIZER TAMBEM DE UMA CARACTERIS_
TICA PARTICULARMENTE DESAGRADAVEL DO LISP: O NUMERO EXCESSIVO DE
PARENTESES DA S-LINGUAGEM. A BOA SINTAXE NO LISP REQUER ESFORCO ATE DO
PROGRAMADOR EXPERIMENTADO; E DIFICIL DISCERNIR AS S-EXPRES_
SOES, FAZER A PASSAGEM DA M-LINGUAGEM PARA A S-LINGUAGEM, ACERTAR
O NUMERO E LUGAR EXATO DOS PARENTESES. TAL FATO DESESTIMULA A MAIO-
RIA DOS PROGRAMADORES EM COMPUTACAO A TRABALHAR COM O LISP.

O ESSENCIAL NO LISP PARECE-NOS SER A POSSIBILIDADE DE OBTER INFORMACAO SIMBOLICA, ESTRUTURA-LA NA MEMORIA SOB A FORMA DE ARVORES BINARIAS E EXECUTAR OPERACOES SOBRE ESSAS ARVORES A PARTIR DAS OPERACOES FUNDAMENTAIS.

VEJAMOS COMO ISSO PODE SER FEITO ATRAVES DE EXTENSOES NO AL_ GOL, ATRAVES DA PORTA ABERTA DAS FUNCOES.

## I.4- UMA NOVA PROPOSICAO:LISP-ALGOL.

## UMA NOVA PROPOSICAO : LISP-ALGOL

APOIADO NAS CRITICAS AO LISP E TENTANDO RESPONDER AS PERGUNTAS FORMULADAS POR FOSTER VAMOS TENTAR ESTABELECER ALGUMAS PROPOSICOES TENDO EM VISTA CONSEGUIR O PROCESSAMENTO DE LISTASATRAVES DE FUNCOES DO ALGOL.

INICIAREMOS REDEFININDO ATOMO.

PARA NOSSO SISTEMA ATOMO E UM CONJUNTO DE SIMBOLOS, LETRAS DIGITOS E OUTROS CARACTERES DISPONIVEIS DISPOSTOS ORDENADAMENTE. O ARMAZENAMENTO DESSES SIMBOLOS E IGUAL EM TODOS OS CASOS E USA O CONCEITO DE STRING, CONFORME EXPLICAREMOS ADIANTE. SERAO CONSIDERADOS ATOMOS NUMERICOS AQUELES FORMADOS POR DIGITOS APENAS, COM SINTAXE IGUAL A DOS INTEIROS EM ALGOL.

EXEMPLO DE ATOMO NAO NUMERICO : QUALQUER. SEQUENCIA. DE. SIMBOLOS

EXEMPLO DE ATOMO NUMERICO : 34712

ENTENDEREMOS LISTA COMO UM CONJUNTO ORDENADO ( POSSIVELMENTE VAZIO ) DE ATOMOS OU LISTAS COLOCADOS ENTRE PARENTESES.

OS ELEMENTOS DE LISTA DEVEM TER AO MENOS UM BRANCO SEPARADOR; DA MESMA FORMA , PELO MENOS UM BRANCO DEVE SEPARAR OS PARENTESES ENTRE SI E OS PARENTESES DOS ELEMENTOS DE LISTA CONTIGUOS.

EXEMPLO DE LISTA : ( UM ELEMENTO DE LISTA PODE SER UM ATOMO OU UMA ( LISTA ) )

UMA S-EXPRESSAO SERA FORMADA PPOR UM ATOMO OU UMA LISTA.

EXEMPLO: ( LISTA VAZIA ( ) )

ATOMO

A DESCRICAO SINTATICA DE ATOMO, LISTA E S-EXPRESSAO SEGUE ABAIXO:

< BRANCO > ::= < BRANCO > _

_

O SIMBOLO _ ESTA DENOTANDO O CARATER BRANCO.

< LETRA > ::= A

B

.

.

.

Z

< DIGITO > ::= 0

1

.

.

.

9

< SIMBOLO ESPECIAL > ::=< QUALQUER SIMBOLO DISPONIVEL NO PDP-10 >

< SIMBOLO DE BASE > ::= < LETRA >

< SIMBOLO ESPECIAL >

< SIMBOLO > ::= < SIMBOLO DE BASE >

< DIGITO >

< ATOMO NUMERICO > ::= < DIGITO >

< DIGITO > < ATOMO NUMERICO >

< PARTE DE ATOMO NAO NUMERICO > ::= <PARTE DE ATOMO NAO NUMERICO >

< SIMBOLO >

E

< ATOMO NAO NUMERICO > ::= < PARTE DE ATOMO NAO NUMERICO >< SIMBOLO DE BASE > < PARTE DE ATOMO NAO NUMERICO >

< ATOMO > ::= < ATOMO NUMERICO >

< ATOMO NAO NUMERICO >

< COMECO DE LISTA > ::= ( < BRANCO >

< NUCLEO DE LISTA > ::= < ELEMENTO DE LISTA > < BRANCO >

< NUCLEO DE LISTA >

< ELEMENTO DE LISTA >

< ELEMENTO DE LISTA >::= < ATOMO >

< LISTA >

E

< FIM DE LISTA > ::= < BRANCO > )

< LISTA > ::= < COMECO DE LISTA > < NUCLEO DE LISTA > < FIM DE LISTA >

< S-EXPRESSAO > ::= < ATOMO >

< LISTA >

ANTES DE PASSARMOS AOS PREDICADOS E FUNCOES IREMOS INTRODU_ZIR 2 CONCEITOS BASICOS DE NOSSO SISTEMA, A LISTA LIVRE E O HEAP E IREMOS MOSTRAR COMO E FEITA A ESTRUTURACAO INTERNA DAS LISTAS.

A LISTA LIVRE E UM CONJUNTO DE CELULAS LOGICAMENTE CONTIGUAS E DIVIDIDAS EM 2 PARTES; A PARTE DA ESQUERDA, QUE CHA_MAREMOS CAR E A PARTE DIREITA QUE CHAMAREMOS CDR.

NA LISTA LIVRE, O CDR DE CADA CELULA CONTEM O ENDERECO DA CELULA SEGUINTE; A ULTIMA CELULA CONTEM O VALOR 0 .

O ACESSO A LISTA LIVRE E FEITO ATRAVES DE UM APONTADOR PARA O TOPO DA LISTA; A LISTA FUNCIONA COMO PILHA.

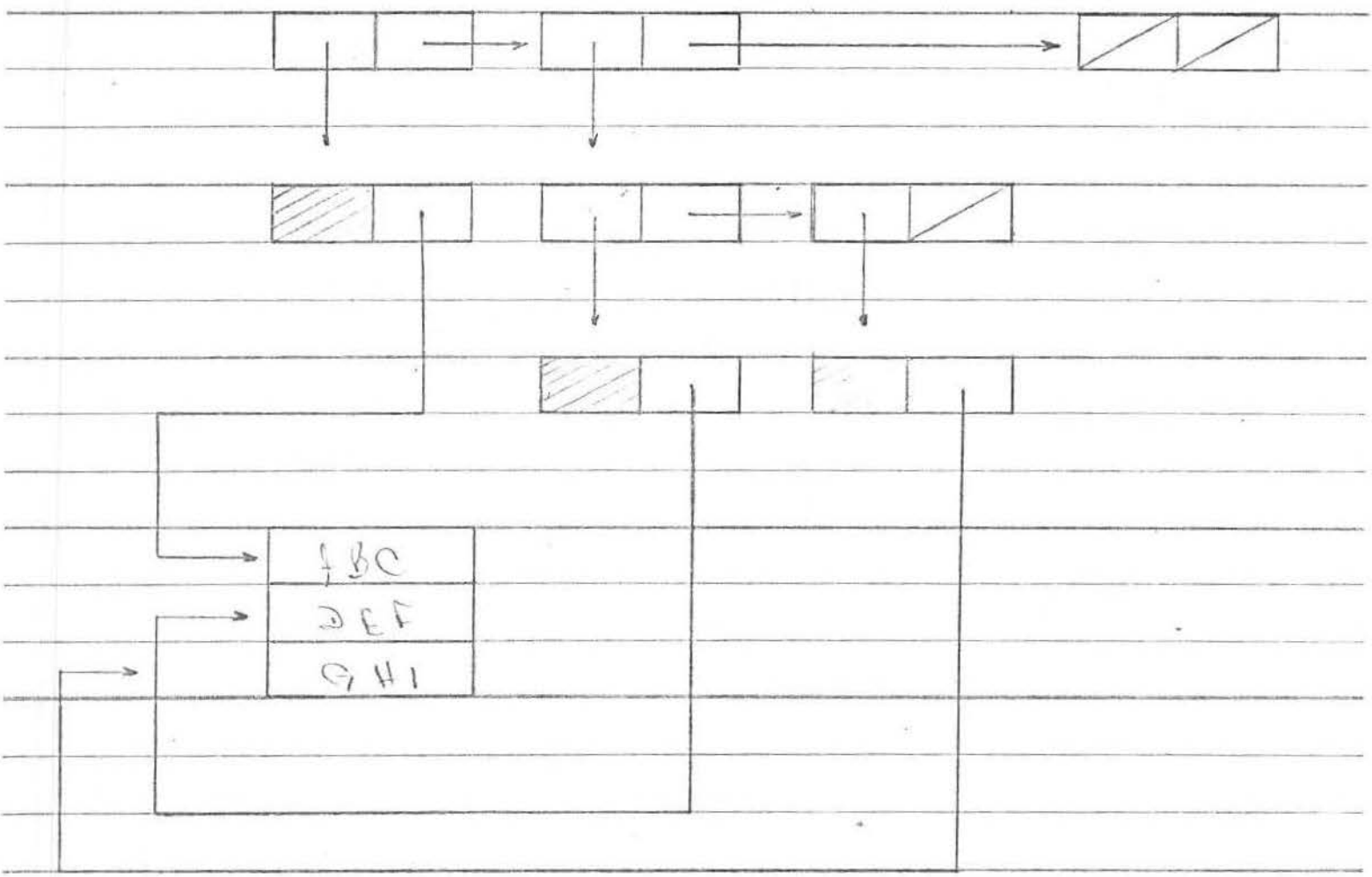
ESQUEMATICAMENTE :

O HEAP E FORMADO POR UM ARRAY UNIDIMENSIONAL DE APONTADORES PARA STRINGS, QUE ARMAZENARAO OS VALORES SIMBOLICOS DOS ATOMOS DO

PROGRAMA.

QUANDO UMA LISTA E LIDA OU UM PROCEDIMENTO QUALQUER CRIA LISTAS NA MEMORIA, CELULAS SAO REQUISITADAS DA LISTA LIVRE E DO HEAP PARA ARMAZENAR OS ENDERECOS E INDICADORES DE ATO_ MOS QUE REPRESENTAM A LISTA. ESQUEMATICAMENTE TERIAMOS O SEGUINTE QUADRO PARA A LISTA

( ABC ( DEF GHI ) ( ) )


ABC
DEF
GHI

AS CELULAS QUE PERMITEM A ESTRUTURACAO DE LISTAS GUARDAM 2 APONTADORES: O CAR APONTANDO PARA O ELEMENTO DE LISTA E O CDR APONTANDO PARA A CELULA QUE GUARDA OS APONTADORES PARA O ELEMENTO SEGUINTE. SE CHEGARMOS AO FIM DA LISTA UM VALOR ESPECIAL NO CDR INDICA I FIM DE LISTA.

SE O ELEMENTO E UM ATOMO A CELULA INDICATIVA DO ELEMENTO TEM EM SEU CAR UM BIT LIGADO INDICANDO A PRESENCA DO ATOMO E EM SEU CDR O VALOR DO INDICE DO HEAP EM QUE O ATOMO ESTA GUARDADO.

UM VALOR ESPECIAL E USADO PARA INDICAR LISTA VAZIA : NO NOSSO SISTEMA USAMOS O VALOR 1.

NA PAGINA SEGUINTE APRESENTAMOS UM MAPA DA MEMORIA COM A LISTA EXEMPLO E UM TRECHO AINDA NAO UTILIZADO DA LISTA LIVRE.

| | | | | | |
|---|---|---|---|---|---|
| 65536 | 1 | 65536 | 2 | 65536 | 3 |
| 4 | 1 | 3 | 5 | 1 | 1 |
| 6 | 7 | 2 | 8 | 0 | 11 |
| 0 | 12 | 0 | 13 | 0 | 14 |
| 0 | 15 | 0 | 16 | 0 | 17 |
| 0 | 18 | 0 | 19 | 0 | 20 |
| 0 | 21 | 0 | 22 | 0 | 23 |
| 0 | 24 | 0 | 25 | 0 | 26 |
| 0 | 27 | 0 | 28 | 0 | 29 |
| 0 | 30 | 0 | 31 | 0 | 32 |
| 0 | 33 | 0 | 34 | 0 | 35 |
| 0 | 36 | 0 | 37 | 0 | 38 |
| 0 | 39 | 0 | 40 | 0 | 0 |

ABC
DEF
GHI

END OF EXECUTION - 4K CORE

EXECUTION TIME: 0.42 SECS.

ELAPSED TIME: 1 MINS. 4.80 SECS.

NO MAPA DE MEMORIA ANTERIOR, O VALOR 63536 CORRESPONDE AO BIT 17 DE MEIA PALAVRA ( A NUMERACAO ACOMPANHA O SENTIDO CRESCENTE DO VALOR DOS BITS). CADA PAR DE IMPRESSAO CORRESPONDE A UMA CELULA; OS VALORES IMPRESSOS SAO OS VALORES INTEIROS DO CAR E CDR. O PRIMEIRO VALOR IMPRESSO CORRESPONDE A CELULA DE ENDERECO 2. A PRIMEIRA CELULA E PERDIDA PARA INDICACAO DE LISTA VAZIA E FIM DE LISTA. OS VALORES IMPRESSOS APOS A LISTA LIVRE SAO ORDENADAMENTE OS VALORES ARMAZENADOS NO HEAP A PARTIR DA POSICAO 1.

VAMOS FALAR AGORA DAS FUNCOES E PREDICADOS. ANTES DISSO, POREM, QUERIAMOS DIZER DO QUE CONSTITUE O NUCLEO DO SISTEMA.

OS PROGRAMAS DO LISP-ALGOL DEVERAO ESTAR INSERIDOS DENTRO DE UM PROGRAMA CENTRAL EM ALGOL ONDE SAO ESPECIFICADOS UMA SERIE DE PARAMETROS DO SISTEMA. NESSE PROGRAMA INICIALMENTE E FEITA A RESERVA DE MEMORIA PARA A LISTA LIVRE E O HEAP, RESERVA ESSA DE TOTAL RESPONSABILIDADE DO PROGRAMADOR. O NUMERO DE CELULAS QUE CONSTITUIRA A LISTA LIVRE E O HEAP DEVERA SER FORNECIDO AO SISTEMA QUE OSLERA E FARA A ALOCACAO DE MEMORIA. ESSES VALORES FICAM ARMAZENADOS EM DUAS VARIAVEIS GLOBAIS DO SISTEMA, RESPECTIVAMENTE TAMANHO. DA . LISTA. LIVRE E TAMANHO. DO. HEAP.

SEGUEM-SE AS DECLARACOES DAS PROCEDURES EMBUTIDAS DO SISTEMA, DECLARADAS EXTERNAL POR SEREM COMPILADAS INDEPENDENTEMENTE DO PROGRAMA PRINCIPAL. SUGERE-SE AO USUARIO DO SISTEMA QUE FACA AS SUAS DECLARACOES DE VARIAVEIS E PROCEDURES NUM BLOCO MAIS INTERNO, QUE COMECARIA LOGO APOS AS DECLARACOES DE PROCEDURES.

NA PAGINA SEGUINTE VEM UMA LISTAGEM DO ESQUEMA PROPOSTO:

```
BEGIN
  INTEGER TAMANHO.DA.LISTA.LIVRE, TAMANHO.DO.HEAP;
  READ(TAMANHO.DA.LISTA.LIVRE, TAMANHO.DO.HEAP);
  BEGIN
   INTEGER ARRAY LISTA.LIVRE[-10:TAMANHO.DA.LISTA.LIVRE];
   STRING ARRAY HEAP [-10:TAMANHO.DO.HEAP];
   EXTERNAL PROCEDURE PREPARAR.LISTA.LIVRE.E.HEAP;
   EXTERNAL INTEGER PROCEDURE LER.LISTA;
   EXTERNAL PROCEDURE IMPRIMIR;
   EXTERNAL PROCEDURE IMPRESSAO.DA.LISTA.LIVRE;
   EXTERNAL PROCEDURE LISTAGEM.DO.HEAP;
   EXTERNAL PROCEDURE RESULTADO.DO.CALCULO.DO.PREDICADO;
   EXTERNAL PROCEDURE COMPACTAR.HEAP;
   EXTERNAL PROCEDURE RECUPERAR.LISTA;
   EXTERNAL PROCEDURE ESTATISTICAS;
   EXTERNAL INTEGER PROCEDURE DECODIFICAR.ATOMO.NUMERICO;
   EXTERNAL PROCEDURE CODIFICAR.NUMERO;
   EXTERNAL BOOLEAN PROCEDURE ATOM, EQ, NUMBERP, NULL;
   EXTERNAL INTEGER PROCEDURE CAR, CDR, LIST;
    BEGIN

      PROGRAMA DO USUARIO

    END
  END
END
```

A PRIMEIRA PROCEDURE QUE ABORDAREMOS E A PROCEDURE PREPARAR.LISTA.LIVRE.E.HEAP(LISTA.LIVRE,LIMITE.INFERIOR.DA.LISTA.LIVRE, LIMITE.SUPERIOR.DA.LISTA.LIVRE, HEAP, LIMITE.SUPERIOR.DO.HEAP).

ESTA PROCEDURE CRIA A LISTA.LIVRE INICIAL, ENTRE OS VALORES LIMITE INFERIOR DA LISTA.LIVRE ( QUE DEVE SER MAIOR QUE 1, EXPLICACOES ADIANTE) E LIMITE.SUPERIOR.DA.LISTA.LIVRE ( QUE DEPENDE DA DISPONIBILIDADE DO SISTEMA E USUALMENTE E IGUALADO A VARIAVEL TAMANHO.DA.LISTA.LIVRE).

A AREA DELIMITADA PELOS VALORES [-10,1] DO ARRAY LISTA.LIVRE E RESERVADA PARA PROPOSITOS ESPECIFICOS DO SISTEMA.

A POSICAO -10 DO ARRAY GUARDA O APONTADOR PARA O TOPO DA LISTA LIVRE.

AS POSICOES -9 E -8 GUARDAM OS VALORES DE APONTADORES QUE SERAO UTILIZADOS NA LEITURA DE LISTAS DE ENTRADA. EM CADA LEITURA SERAO REINICIALIZADAS; DESSA FORMA SUA INICIALIZACAO NAO E FEITA NESTA ROTINA.

A POSICAO -7 DO ARRAY GUARDA UM CONTADOR DESTINADOA IMPRESSAO DE LISTAS; ELE TAMBEM E INCIALIZADO DENTRO DA ROTINA ESPECIFICA QUE O UTILIZA.

LISTA.LIVRE[-6] E LISTA.LIVRE[-5] GUARDAM OS VALORES DOS NUMEROS DE CELULAS UTEIS DA LISTA LIVRE E DO HEAP RESPECTIVAMENTE. ESTES SAO INICIALIZADOS NA ROTINA E ATUALIZADOS POR QUALQUER ROTINA QUE FACA USO DAS CELULAS DA LISTA LIVRE E DO HEAP.

LISTA.LIVRE[-4] GUARDA UM APONTADOR PARA A PROXIMA POSICAO DISPONIVEL DO HEAP; E INICIALIZADO COM O VALOR 1.

AS POSICOES -3,-2,-1E 0 DO ARRAY ESTAO RESERVADAS PARA UMA FUTURA AMPLIACAO DO SISTEMA.

LISTA.LIVRE[1] E RESERVADA COMO INDICADOR DE LISTA VAZIA E FIM DE LISTA.

A ROTINA INICIALIZA TAMBEM ALGUMAS POSICOES DO HEAP COM STRINGS PARTICULARES ( O BRANCO E OS PARENTESES ) QUE FORAM NECESSA_RIOS EM OUTRAS ROTINAS. AS OUTRAS POSICOES NEGATIVAS ESTAO DIS_PONIVEIS PARA ARMAZENAMENTO DE RESULTADOS INTERMEDIARIOS, MEMORI_ZACAO DOS STRINGS-LISTAS LIDOS , ETC..

A INSERCAO DE TODOS ESSES APONTADORES E VALORES DESSA FOR_MA NA LISTA LIVRE E NO HEAP DEVEU-SE AO FATO DE QUEQUIS EVITAR A SUA PASSAGEM PARA OS PROCEDURES COMO PARAMETROS FORMAIS, O QUE IRIA AUMENTAR RAZOAVELMENTE A LISTA DE PARAMETROS, VISTO QUE O ALGOL DO PDP-10 NAO ACEITA VARIAVEIS GLOBAIS EM PROCEDURES COMPILADOS IN_DEPENDENTEMENTE DO PROGRAMA PRINCIPAL. DESSA FORMA PASSA-SE "BY NAME" A LISTA LIVRE E O HEAP E COM ELES TODOS ESSES VALORES. ESSA FORMA DE COLOCAR AS COISAS EVITA TAMBEM UM NUMERO RAZOAVEL DE DECLARACOES NO NUCLEO DO PROGRAMA; SEU PONTO NEGATIVI ESTA EM SE SABER O QUE FAZ CADA POSICAO PARTICULAR DO ARRAY. A REPETICAO EXAUSTIVA DAS ATRIBUI_COES DE CADA APONTADOR E SUA RESPECTIVA POSICAO NO ARRAY SERA FEITA AO LONGO DESTAS LINHAS, DIMINUINDO , ESPERO, ESSE DEFEITO.

SEGUE UMA LISTAGEM DO PROCEDURE.

```
        PROCEDURE PREPARAR.LISTA.LIVRE.E.HEAP(LISTA.LIVRE,
LIMITE.INFERIOR.DA.LISTA.LIVRE,LIMITE.SUPERIOR.DA.LISTA.LIVRE,
HEAP,LIMITE.SUPERIOR.DO.HEAP);
        VALUE LIMITE.INFERIOR.DA.LISTA.LIVRE,LIMITE.SUPERIOR.
DA.LISTA.LIVRE,LIMITE.SUPERIOR.DO.HEAP;
        INTEGER ARRAY LISTA.LIVRE; INTEGER LIMITE.INFERIOR.
DA.LISTA.LIVRE,LIMITE.SUPERIOR.DA.LISTA.LIVRE,LIMITE.SUPERIOR.
DO.HEAP;
        STRING ARRAY HEAP;
         BEGIN
         INTEGER I;
        LISTA.LIVRE[-10]_LIMITE.INFERIOR.DA.LISTA.LIVRE;
        LISTA.LIVRE[-6]_LIMITE.SUPERIOR.DA.LISTA.LIVRE-
LIMITE.INFERIOR.DA.LISTA.LIVRE;
        LISTA.LIVRE[-5]_LIMITE.SUPERIOR.DO.HEAP;
        LISTA.LIVRE[-4]_1;
         SFIELD(LISTA.LIVRE[1],0,18,1);
         SFIELD(LISTA.LIVRE[1],18,18,1);
         FOR I_LIMITE.INFERIOR.DA.LISTA.LIVRE STEP 1 UNTIL
LIMITE.SUPERIOR.DA.LISTA.LIVRE - 1 DO
          BEGIN
           INTEGER AUXILIAR;
           AUXILIAR_I+1;
           SFIELD(LISTA.LIVRE[I],18,18,GFIELD(AUXILIAR,18,18));
          END;
         LISTA.LIVRE[LIMITE.SUPERIOR.DA.LISTA.LIVRE]_0;
        HEAP[-10]_" ";
```

```
HEAP[-9]_"(";
HEAP[-8]_")"
 END
```

A PROCEDURE LER_LISTA(LISTA_LIVRE, HEAP) E UMA PROCEDURE QUE LE UMA LISTA, ESTRUTURA-A NA MEMORIA, SOLICITANDO CELULAS A LISTA LIVRE E AO HEAP, ATUALIZA OS APONTADORES PARA A LISTA LIVRA E O HEAP E GUARDA EM SEU NOME O VALOR DO APONTADOR PARA O TOPO DA LISTA.

A EXECUCAO DESSA PROCEDURE IMPLICA NA LEITURA DE UMA LISTA QUE DEVERA SER FORNECIDA COMO STRING ( ENTRE ASPAS DUPLAS ). A LISTA DEVERA SEGUIR A SINTAXE DE LISTAS JA APRESENTADA E NAO DEVERA HAVER BRANCOS ENTRE A LISTA E AS ASPAS DUPLAS. TAIS INSTRUCOES DEVEM SER SEGUIDAS RELIGIOSAMENTE POIS A ROTINA NAO TEM MENSAGENS DE ERRO E PODER-SE-A TER RESULTADOS IMPREVISTOS E ERRADOS.

ESSA PROCEDURE E INTERESSANTE POIS E MONTADA TOTALMENTE EM TERMOS RECURSIVOS.

UMA LISTAGEM DA PROCEDURE SEGUE ADIANTE.

```
INTEGER PROCEDURE LER.LISTA(LISTA.LIVRE,HEAP);
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
BEGIN
EXTERNAL INTEGER PROCEDURE LIST;
FORWARD INTEGER PROCEDURE NUCLEO.DE.LISTA,ATOMO,
 ELEMENTO.DE.LISTA,SIMBOLOS;
FORWARD PROCEDURE SALTAR.BRANCOS;
INTEGER PROCEDURE LISTA;
BEGIN
IF HEAP[0].[LISTA.LIVRE[-9]]=HEAP[-9].[1] THEN
 BEGIN
 LISTA.LIVRE[-8]_LISTA.LIVRE[-9];
 LISTA.LIVRE[-9]_LISTA.LIVRE[-9]+1;
 SALTAR.BRANCOS;
 LISTA_NUCLEO.DE.LISTA
 END
                                          ELSE
 IF HEAP[0].[LISTA.LIVRE[-9]]=HEAP[-8].[1] THEN
 BEGIN
 LISTA.LIVRE[-8]_LISTA.LIVRE[-9];
 LISTA.LIVRE[-9]_LISTA.LIVRE[-9]+1;
 SALTAR.BRANCOS;
 LISTA_1
 END
END;
INTEGER PROCEDURE NUCLEO.DE.LISTA;
```

```
        BEGIN
        EXTERNAL INTEGER PROCEDURE LIST;
        IF HEAP[0].[LISTA.LIVRE[-9]]#HEAP[-8].[1] THEN
        NUCLEO.DE.LISTA_LIST(LISTA.LIVRE,HEAP,ELEMENTO.DE.LISTA,
         NUCLEO.DE.LISTA)
                                                    ELSE
        NUCLEO.DE.LISTA_LISTA
        END;
        INTEGER PROCEDURE ELEMENTO.DE.LISTA;
        BEGIN
        IF HEAP[0].[LISTA.LIVRE[-9]]=HEAP[-9].[1] THEN
        ELEMENTO.DE.LISTA_LISTA
                                                  ELSE
        IF HEAP[0].[LISTA.LIVRE[-9]]#HEAP[-8].[1] THEN
        ELEMENTO.DE.LISTA_ATOMO
                                                        ELSE
        ELEMENTO.DE.LISTA_LISTA
        END;
        INTEGER PROCEDURE ATOMO;
BEGIN
        ATOMO_LIST(LISTA.LIVRE,HEAP,65536,SIMBOLOS);
END;
        INTEGER PROCEDURE SIMBOLOS;
        BEGIN
        LISTA.LIVRE[-8]_LISTA.LIVRE[-9];

        WHILE HEAP[0].[LISTA.LIVRE[-9]]#HEAP[-10].[1] DO
```

```
LISTA.LIVRE[-9]_LISTA.LIVRE[-9]+1;
LISTA.LIVRE[-9]_LISTA.LIVRE[-9]-1;
HEAP[LISTA.LIVRE[-4]]_COPY(HEAP[0],LISTA.LIVRE[-8],
 LISTA.LIVRE[-9]);
LISTA.LIVRE[-5]_LISTA.LIVRE[-5]-1;
SIMBOLOS_LISTA.LIVRE[-4];
LISTA.LIVRE[-4]_LISTA.LIVRE[-4]+1;
LISTA.LIVRE[-8]_LISTA.LIVRE[-9];
LISTA.LIVRE[-9]_LISTA.LIVRE[-9]+1;
SALTAR.BRANCOS
END;
PROCEDURE SALTAR.BRANCOS;
BEGIN
INTEGER I;
IF LISTA.LIVRE[-9]<=LENGTH(HEAP[0]) THEN
WHILE HEAP[0].[LISTA.LIVRE[-9]]=HEAP[-10].[1] DO
LISTA.LIVRE[-9]_LISTA.LIVRE[-9]+1
END;
LISTA.LIVRE[-9]_1;
READ(HEAP[0]);
LER.LISTA_LISTA
END;
```

TEMOS 5 ROTINAS DE IMPRESSAO NO SISTEMA, CADA UMA VOLTADA PARA UMA TAREFA ESPECIAL . A MAIS IMPORTANTE DELAS E IMPRIMIR (LISTA.LIVRE,HEAP,I) E E A ROTINA QUE FAZ A IMPRESSAO DE LISTAS. O PARAMETRO I E O APONTADOR PARA A LISTA A SER IMPRESSA. ESSA ROTINA PERCORRE A ARVORE BINARIA ESTRUTURADA NA MEMORIA E DEPENDENDO DO TIPO DE CELULA ELA MONTA NO BUFFER DE SAIDA UM CONJUNTO DE CARACTERES DE IMPRESSAO.

A IMPRESSAO SE DA SE A LISTA FOI ESGOTADA OU SE O BUFFER ESTA CHEIO. DESSA FORMA SE OTIMIZA A SAIDA DE DADOS.

IMPRESSAO.DA.LISTA.LIVRE(LISTA.LIVRE,I,J) E UMA ROTINA QUE FAZ A IMPRESSAO DO ARRAY LISTA.LIVRE,MOSTRANDO-NOS AS LISTAS E TURADAS NA MEMORIA , OS VALORES DOS PARAMETROS DO SISTEMA (APON_ TADORES, CONTADORES, ETC.) E A LISTA DAS CELULAS DISPO_ NIVEIS. A IMPRESSAO E FEITA ATRAVES DE PARES DE VALORES, 3 EM CADA LI_ NHA. OS PARAMETROS I E J INDICAM OS LIMITES ENTRE OS QUAIS SE DE_ SEJA A IMPRESSAO.

LISTAGEM.DO.HEAP(HEAP,I,J) E A ROTINA QUE FAZ A IMPRESSAO DOS VALORES ARMAZENADOS NO HEAP. OS VALORES I E J INDICAM OS LIMITES ENTRE OS QUAIS SE QUER A IMPRESSAO.

A PROCEDURE RESULTADO DO CALCULO.DO PREDICADO(I) IMPRI_ ME O VALOR TRUE SE O PREDICADO CALCULADO FOR VERDADEIRO OU FALSE CASO CONTRARIO. O PARAMETRO I E O PREDICADO.

A PROCEDURE ESTATISTICAS(LISTA.LIVRE) NOS FORNECE OS VALO_ RES DO NUMERO DE CELULAS AINDA DISPONIVEIS NA LISTA LIVRE E NO HEAP.

AS LISTAGENS DOS PROCEDURES E UM PROGRAMA MOSTRANDO O FUNCIO_ NAMENTO DE 4 DELAS E MAIS AS 2 ANTERIORES VEM ADIANTE.

```
PROCEDURE IMPRIMIR(LISTA.LIVRE,HEAP,I);
VALUE I;
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
INTEGER I;
BEGIN
FORWARD PROCEDURE IMPRIMIR.ATOMO,IMPRIMIR.LISTA;
FORWARD PROCEDURE IMPRIMIR.ELEMENTO.DE.LISTA;
EXTERNAL BOOLEAN PROCEDURE ATOM;
EXTERNAL INTEGER PROCEDURE CAR,CDR;
PROCEDURE DECIDIR(I);
VALUE I;
INTEGER I;
 BEGIN
 IF I=1 THEN
 BEGIN
 OUTSYMBOL(HEAP[-9].[1]);
 OUTSYMBOL(HEAP[-10].[1]);
 OUTSYMBOL(HEAP[-8].[1]);
 OUTSYMBOL(HEAP[-10].[1])
 END
         ELSE

 IF ATOM(LISTA.LIVRE,I) THEN
 IMPRIMIR.ATOMO(I)
                         ELSE
 IMPRIMIR.LISTA(I);
```

```
END;
PROCEDURE IMPRIMIR.LISTA(I);
VALUE I;
INTEGER I;
BEGIN
OUTSYMBOL(HEAP[-9].[1]);
OUTSYMBOL(HEAP[-10].[1]);
LISTA.LIVRE[-7]_LISTA.LIVRE[-7]-1;
DECIDIR(CAR(LISTA.LIVRE,I));
IMPRIMIR.ELEMENTO.DE.LISTA(CDR(LISTA.LIVRE,I))
END;
PROCEDURE IMPRIMIR.ELEMENTO.DE.LISTA(I);
VALUE I;
INTEGER I;
BEGIN
IF I=1 THEN
BEGIN
OUTSYMBOL(HEAP[-8].[1]);
OUTSYMBOL(HEAP[-10].[1]);
LISTA.LIVRE[-7]_LISTA.LIVRE[-7] - 1;
IF LISTA.LIVRE[-7]=0 THEN
GO TO FIM
END
                    ELSE
BEGIN
DECIDIR(CAR(LISTA.LIVRE,I));
IMPRIMIR.ELEMENTO.DE.LISTA(CDR(LISTA.LIVRE,I))
```

```
        END;

FIM:     BREAKOUTPUT
        END;
        PROCEDURE IMPRIMIR.ATOMO(I);
        VALUE I;
        INTEGER I;
        BEGIN
        INTEGER J;
        FOR J_1 UNTIL LENGTH(HEAP[GFIELD(LISTA.LIVRE[I],18,18)])
         DO
        BEGIN
        INTEGER K; K_GFIELD(LISTA.LIVRE[I],18,18);
        K_HEAP[K].[J];
        OUTSYMBOL(GFIELD(K,29,7));
        END;
        OUTSYMBOL(HEAP[-10].[1]);
        IF LISTA.LIVRE[-7]=0 THEN
        BREAKOUTPUT
        END;
        LISTA.LIVRE[-7]_0;
        NEWLINE(3);
        DECIDIR(I)
        END;
```

```
PROCEDURE IMPRESSAO.DA.LISTA.LIVRE(LISTA.LIVRE,I,J);
VALUE I,J;
INTEGER ARRAY LISTA.LIVRE;
INTEGER I,J;
BEGIN
INTEGER K;
NEWLINE(3);
FOR K_I UNTIL J DO
BEGIN
PRINT(GFIELD(LISTA.LIVRE[K],0,18));
PRINT(GFIELD(LISTA.LIVRE[K],18,18));
END;
BREAKOUTPUT;
NEWLINE(3)
END;
```

```
PROCEDURE LISTAGEM.DO.HEAP(HEAP, I, J);
VALUE I, J;
STRING ARRAY HEAP;
INTEGER I, J;
BEGIN
INTEGER K;
NEWLINE(3);
FOR K_I UNTIL J DO
BEGIN
NEWLINE;
WRITE(HEAP[K])
END
END;
```

```
PROCEDURE RESULTADO.DO.CALCULO.DO.PREDICADO(I);
VALUE I;
BOOLEAN I;
BEGIN
NEWLINE(3);
IF I THEN
WRITE("TRUE")
                    ELSE
WRITE("FALSE")
END;
```

```
PROCEDURE ESTATISTICAS(LISTA.LIVRE);
INTEGER ARRAY LISTA.LIVRE;
BEGIN
NEWLINE(3);
WRITE("NUMERO DE CELULAS LIVRES NA LISTA LIVRE:");
PRINT(LISTA.LIVRE[-6]);
NEWLINE;
WRITE("NUMERO DE STRINGS LIVRES NO HEAP:");
PRINT(LISTA.LIVRE[-5]);
END;
```

```
 TYPE EXEMP/\/\.ALG
00010   BEGIN
00020    INTEGER TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP;
00030    READ(TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP);
00040    BEGIN
00050     INTEGER ARRAY LISTA.LIVRE[-10:TAMANHO.DA.LISTA.LIVRE];
00051     STRING ARRAY HEAP [-10:TAMANHO.DO.HEAP];
00052     EXTERNAL PROCEDURE PREPARAR.LISTA.LIVRE.E.HEAP;
00053     EXTERNAL INTEGER PROCEDURE LER.LISTA;
00054     EXTERNAL PROCEDURE IMPRIMIR;
00055     EXTERNAL PROCEDURE IMPRESSAO.DA.LISTA.LIVRE;
00056     EXTERNAL PROCEDURE LISTAGEM.DO.HEAP;
00057     EXTERNAL PROCEDURE RESULTADO.DO.CALCULO.DO.PREDICADO;
00058     EXTERNAL PROCEDURE COMPACTAR.HEAP;
00059     EXTERNAL PROCEDURE RECUPERAR.LISTA;
00060     EXTERNAL PROCEDURE ESTATISTICAS;
00061     EXTERNAL INTEGER PROCEDURE DECODIFICAR.ATOMO.NUMERICO;
00062     EXTERNAL PROCEDURE CODIFICAR.NUMERO;
00063     EXTERNAL BOOLEAN PROCEDURE ATOM, EQ, NUMBERP, NULL;
00064     EXTERNAL INTEGER PROCEDURE CAR, CDR, LIST;
00065     PREPARAR.LISTA.LIVRE.E.HEAP(LISTA.LIVRE,2,TAMANHO.DA.LISTA.LIV
RE,
00070   HEAP,TAMANHO.DO.HEAP);
00071     IMPRESSAO.DA.LISTA.LIVRE(LISTA.LIVRE,-10,TAMANHO.DA.LISTA.LIVR
E);
00072     IMPRIMIR(LISTA.LIVRE, HEAP,LER.LISTA(LISTA.LIVRE,HEAP));
00075     IMPRESSAO.DA.LISTA.LIVRE(LISTA.LIVRE,-10,TAMANHO.DA.LISTA.LIVR
E);
00080     LISTAGEM.DO.HEAP(HEAP,1,TAMANHO.DO.HEAP);
00081     ESTATISTICAS(LISTA.LIVRE)
00160    END
00170   END

.EXE
LOADING

EXEMP 2K CORE
EXECUTION
31 9


          0          2          0          0          0          0
          0          0          0         29          0          9
          0          1          0          0          0          0
          0          0          0          0          1          1
          0          3          0          4          0          5
          0          6          0          7          0          8
          0          9          0         10          0         11
          0         12          0         13          0         14
          0         15          0         16          0         17
          0         18          0         19          0         20
          0         21          0         22          0         23
          0         24          0         25          0         26
          0         27          0         28          0         29
          0         30          0         31          0          0


"<     A       CEBOLA   ( ( ( ) ) )   E UM CEREAL     )"
```

( A CEBOLA ( ( ( ) ) ) E UM CEREAL )

| | | | | | |
|---|---|---|---|---|---|
| 0 | 15 | 0 | 55 | 0 | 54 |
| 262143 | 262138 | 0 | 16 | 0 | 4 |
| 0 | 6 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 65536 | 1 | 65536 | 2 | 1 | 1 |
| 4 | 1 | 65536 | 3 | 65536 | 4 |
| 65536 | 5 | 8 | 1 | 7 | 9 |
| 6 | 10 | 5 | 11 | 3 | 12 |
| 2 | 13 | 0 | 16 | 0 | 17 |
| 0 | 18 | 0 | 19 | 0 | 20 |
| 0 | 21 | 0 | 22 | 0 | 23 |
| 0 | 24 | 0 | 25 | 0 | 26 |
| 0 | 27 | 0 | 28 | 0 | 29 |
| 0 | 30 | 0 | 31 | 0 | 0 |

A
CEBOLA
E
UM
CEREAL

NUMERO DE CELULAS LIVRES NA LISTA LIVRE: 16
NUMERO DE STRINGS LIVRES NO HEAP: 4

END OF EXECUTION - 3K CORE

EXECUTION TIME: 0.86 SECS.

ELAPSED TIME: 2 MINS. 8.58 SECS.

OS PREDICADOS BASICOS DE NOSSO SISTEMA SAO:

A) ATOM(LISTA.LIVRE,I)

ESTE PREDICADO TESTA O ELEMENTO DE LISTA APONTADO POR I E TEM O VALOR TRUE SE O ELEMENTO FOR UM ATOMO E E FALSE CASO CONTRARIO. BASICAMENTE, O PROCEDURE FAZ UM TESTE DO BIT QUE INDICA A CONDICAO DE ATOMO NO CAR DA CELULA.

B)EQ(LISTA.LIVRE,HEAP,I,J)

ESTE PREDICADO SO E DEFINIDO PARA ATOMOS. OS ATOMOS SAO APONTADOS POR I E J; SE ELES FOREM IGUAIS ENTAO O PREDICADO TEM VALOR TRUE , CASO CONTRARIO, FALSE.SE OS APONTADORES I E J APONTAREM PARA ELEMENTOS DE LISTA NAO ATOMICOS O PREDICADO FICA INDEFINIDO SENDO UMA MENSAGEM IMPRESSA .

C)NUMBERP(LISTA.LIVRE,HEAP,I)

O PREDICADO TESTA SE O ATOMO APONTADO POR I E NUMERICO OU NAO.SE O ELEMENTO APONTADO POR I NAO FOR ATOMICO OU SE O ATOMO NAO FOR NUMERICO O PREDICADO TEM VALOR FALSE. SE O PREDICADO FOR NUMERICO ( SINTAXE DE ATOMO NUMERICO APRESENTADA) ENTAO SEU VALOR E TRUE.

D)NULL(LISTA.LIVRE,I)

ESTE PREDICADO TESTA SE O ELEMENTO DE LISTA E A LISTA VAZIA OU O INDICADOR DE FIM DE ATOMO. EM CASO AFIRMATIVO, (I=1) TOMA O VALOR TRUE ; CASO CONTRARIO,FALSE. O PREDICADO E SEMPRE DEFINIDO.

NAS PAGINAS SEGUINTES APRESENTAMOS LISTAGENS DOS PROCEDIMENTOS BEM COMO UM PROGRAMA EXEMPLO MOSTRANDO SUA UTILIZACAO.

```
BOOLEAN PROCEDURE ATOM(LISTA.LIVRE,I);
VALUE I;
INTEGER ARRAY LISTA.LIVRE;
INTEGER I;
ATOM_GFIELD(LISTA.LIVRE[I],0,18)=65536;



BOOLEAN PROCEDURE EQ(LISTA.LIVRE,HEAP,I,J);
VALUE I,J;
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
INTEGER I,J;
BEGIN
INTEGER K,L;
K_GFIELD(LISTA.LIVRE[I],18,18);
L_GFIELD(LISTA.LIVRE[J],18,18);
IF GFIELD(LISTA.LIVRE[I],0,18)#65536 OR
GFIELD(LISTA.LIVRE[J],0,18)#65536 THEN
WRITE("PREDICADO INDEFINIDO")
                                ELSE
IF HEAP[K]=HEAP[L] THEN
EQ_%777777777777
                                        ELSE
EQ_%000000000000
END;
```

```
        BOOLEAN PROCEDURE NUMBERP(LISTA.LIVRE,HEAP,I);
        VALUE I;
        INTEGER ARRAY LISTA.LIVRE;
        STRING ARRAY HEAP;
        INTEGER I;
        BEGIN
        BOOLEAN PROCEDURE CODIGO.NUMERICO(I);
        VALUE I;
        INTEGER I;
        IF I>47 AND I< 58 THEN
        CODIGO.NUMERICO_%777777777777
                            ELSE
        CODIGO.NUMERICO_%000000000000;
        IF GFIELD(LISTA.LIVRE[I],0,18)#65536 THEN
        NUMBERP_%000000000000 ELSE
        BEGIN
        INTEGER K,L;
        K_GFIELD(LISTA.LIVRE[I],18,18);
        FOR L_1 UNTIL LENGTH(HEAP[K]) DO
        IF CODIGO.NUMERICO(HEAP[K].[L]) THEN
                                            ELSE
        BEGIN NUMBERP_%000000000000; GO TO FIM END;
        NUMBERP_%777777777777;
FIM:    END
        END;
```

```
BOOLEAN PROCEDURE NULL(LISTA.LIVRE,I);
VALUE I;
INTEGER ARRAY LISTA.LIVRE;
INTEGER I;
IF I=1 THEN NULL_%777777777777 ELSE NULL_%000000000000;
```

```
 TYPE EXEMPL.ALG
00010   BEGIN
00020    INTEGER TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP;
00030    READ(TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP);
00040    BEGIN
00050     INTEGER ARRAY LISTA.LIVRE[-10:TAMANHO.DA.LISTA.LIVRE];
00051     STRING ARRAY HEAP [-10:TAMANHO.DO.HEAP];
00052     EXTERNAL PROCEDURE PREPARAR.LISTA.LIVRE.E.HEAP;
00053     EXTERNAL INTEGER PROCEDURE LER.LISTA;
00054     EXTERNAL PROCEDURE IMPRIMIR;
00055     EXTERNAL PROCEDURE IMPRESSAO.DA.LISTA.LIVRE;
00056     EXTERNAL PROCEDURE LISTAGEM.DO.HEAP;
00057     EXTERNAL PROCEDURE RESULTADO.DO.CALCULO.DO.PREDICADO;
00058     EXTERNAL PROCEDURE COMPACTAR.HEAP;
00059     EXTERNAL PROCEDURE RECUPERAR.LISTA;
00060     EXTERNAL PROCEDURE ESTATISTICAS;
00061     EXTERNAL INTEGER PROCEDURE DECODIFICAR.ATOMO.NUMERICO;
00062     EXTERNAL PROCEDURE CODIFICAR.NUMERO;
00063     EXTERNAL BOOLEAN PROCEDURE ATOM, EQ, NUMBERP, NULL;
00064     EXTERNAL INTEGER PROCEDURE CAR, CDR, LIST;
00065     PREPARAR.LISTA.LIVRE.E.HEAP(LISTA.LIVRE,2,TAMANHO.DA.LISTA.LIV
RE,
00070   HEAP,TAMANHO.DO.HEAP);
00071     LISTA.LIVRE[-1]_LER.LISTA(LISTA.LIVRE,HEAP);
00072     NEWLINE(3);
00075     LISTA.LIVRE[-2]_LIST(LISTA.LIVRE,HEAP,65536,-2);
00080    HEAP[-2]_"0";
00081     WHILE NOT NULL(LISTA.LIVRE,LISTA.LIVRE[-1]) DO
00083     BEGIN
00085     NEWLINE(3);
00091     IF NOT ATOM(LISTA.LIVRE,CAR(LISTA.LIVRE,LISTA.LIVRE[-1]))THEN
00095     WRITE(" O ELEMENTO DE LISTA NAO E ATOMICO")                ELS
E
00100     IF NOT NUMBERP(LISTA.LIVRE,HEAP,CAR(LISTA.LIVRE,LISTA.LIVRE[-1
]))
00101   THEN
00105     WRITE(" O ELEMENTO DE LISTA E ATOMICO MAS NAO NUMERICO")     E
LSE
00110     IF EQ(LISTA.LIVRE,HEAP,(CAR(LISTA.LIVRE,LISTA.LIVRE[-1])),LIST
A.LIVRE
00115   [-2]) THEN
00120     WRITE(" ENCONTRAMOS FINALMENTE  O ATOMO 0")
00125           ELSE
00130     WRITE( " O ATOMO E NUMERICO MAS DIFERENTE DE 0");
00140     LISTA.LIVRE[-1]_CDR(LISTA.LIVRE,LISTA.LIVRE[-1])
00150     END;
00160     NEWLINE(3)
00165     END
00170   END

.EXE
LOADING

EXEMPL 2K CORE
EXECUTION
31 9
"( ABC ( ) 89 0 ABC )"
```

```
O ELEMENTO DE LISTA E ATOMICO MAS NAO NUMERICO

O ELEMENTO DE LISTA NAO E ATOMICO

O ATOMO E NUMERICO MAS DIFERENTE DE 0

ENCONTRAMOS FINALMENTE  O ATOMO 0

O ELEMENTO DE LISTA E ATOMICO MAS NAO NUMERICO


END OF EXECUTION - 3K CORE
EXECUTION TIME: 0.20 SECS.
ELAPSED TIME:   41.78 SECS.
```

AS FUNCOES QUE ADOTAREMOS COMO FUNDAMENTAIS SAO3. ESTA E NO ENTANTO A PORTA POR ONDE SE PODERA FAZER CRESCERE APERFEICOAR O SISTEMA. O CONJUNTO DE FUNCOES ARITMETICAS DO LISP PERDE SENTIDO NO LISP-ALGOL POIS ESSAS FUNCOES SAO SUBSTITUIDAS COM VANTAGEM PELAS FUNCOES ARITMETICAS DE BIBLIOTECA E PELOS OPERADORES DA LINGUAGEM ALGOL. DAS FUNCOES QUE MCCARTHY APRESENTA EM SEU SISTEMA NENHUMA ( A EXCECAO DAS FUNDAMENTAIS) NOS PARECEU REALMENTE SIGNIFICATIVA PARA INCLUSAO NO NOSSO SISTEMA. A CONFECCAO DE NOVAS FUNCOES FICA A CARGO DO USUARIO QUE AS FARA PARA SEUS FINS ESPECIFICOS. AS FUNCOES QUE INTRODUZIMOS SAO:

A) CAR(LISTA.LIVRE, I)

ESTA FUNCAO NOS FORNECE O VALOR DO APONTADOR ARMAZENADONA PARTE ESQUERDA DA CELULA APONTADA POR I. EM OUTRAS PALAVRAS OBTEMOS O VALOR DO APONTADOR PARA O PRIMEIRO ELEMENTO DA LISTA. ESTA FUNCAO DA MESMA FORMA QUE NO LISP ASSUME ESPECIAL IMPORTANCIA NO SISTEMA POIS E ATRAVES DELA QUE TEMOS ACESSO AOS ELEMENTOS DAS LISTAS.

B)CDR(LISTA.LIVRE, I)

ESTA FUNCAO NOS FORNECE A PARTE DIREITA DA CELULA APONTADA POR I. EM SUMA NOS DA UM APONTADOR PARA A LISTA SEM O PRIMEIRO ELEMENTO.

C)LIST(LISTA.LIVRE, HEAP, I, J)

ESTA FUNCAO E A PECA MAIS IMPORTANTE DO PROCESSAMENTO DE LISTAS. E ATRAVES DELA QUE CONSTRUIMOS NOVAS LISTAS. ESTA FUNCAO FUNCIONA DA SEGUINTE FORMA: SE A LISTA LIVRE ESTIVER ESGOTADA E FEITO UM DESVIO PARA A ROTINA QUE FAZ A RECUPERACAO DE LISTAS. CASO CONTRARIO A ROTINA APANHA UMA CELULA DA

LISTA LIVRE ( OU 2, CONFORME O CASO ) E ATUA DA SEGUINTE FORMA:

SE O VALOR DE I FOR 65536 A CELULA A SER MONTADA E UMA CE-
LULA INDICATIVA DE ATOMO. O VALOR DE I E COLOCADO NO CAR DA CELU_
LA E O DE J, INDICADOR DA POSICAO DO ATOMO NO HEAP, NO CDR. PARA COM_
PLETAR A MONTAGEM DO ATOMO E NECESSARIO UM ASSINALAMENTO DO STRING
A POSICAO J DO STRING ARRAY.

SE I FOR DIFERENTE DE 65536 ENTAO ELE PODE ESTAR APONTANDO
PARA UM ATOMO OU PARA UMA LISTA. DO MESMO MODO RACIOCINEMOS PARA J.

SE I FOR ATOMO E J FOR ATOMO A FUNCAO CRIA UMA LISTA DA QUAL
OS ATOMOS APONTADOS POR I E J SAO ELEMENTOS.

SE I FOR ATOMO E J FOR LISTA ENTAO A FUNCAO CRIA UMA LISTA
NOVA COM O ATOMO APONTADO POR I E OS ELEMENTOS DA LISTA APON_
TADA POR J.

SE I FOR LISTA E J FOR ATOMO A FUNCAO DEVOLVE UMA LISTA EM
QUE O PRIMEIRO ELEMENTO E A LISTA APONTADA POR I E O SEGUNDO E A
CELULA ATOMICA APONTADA POR J.

SE AMBOS FOREM LISTAS ENTAO A LISTA RESULTANTE E A FORMADA
PELOS ELEMENTOS DA LISTA I E DA LISTA J ORDENADAMENTE.

AS LISTAGENS DAS PROCEDURES E UM PROGRAMA MOSTRANDO O SEU
FUNCIONAMENTO SEGUEM NA OUTRA FOLHA.

```
INTEGER PROCEDURE CAR(LISTA.LIVRE,I);
VALUE I;
INTEGER ARRAY LISTA.LIVRE;
INTEGER I;
CAR_GFIELD(LISTA.LIVRE[I],0,18);



INTEGER PROCEDURE CDR(LISTA.LIVRE,I);
VALUE I;
INTEGER ARRAY LISTA.LIVRE;
INTEGER I;
CDR_GFIELD(LISTA.LIVRE[I],18,18);



INTEGER PROCEDURE LIST(LISTA.LIVRE,HEAP,I,J);
VALUE I,J;
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
INTEGER I,J;
BEGIN
 INTEGER AUXILIAR;
EXTERNAL PROCEDURE RECUPERAR.LISTA;
IF LISTA.LIVRE[LISTA.LIVRE[-10]]=0 THEN RECUPERAR.LISTA
(LISTA.LIVRE,HEAP,LISTA.LIVRE[0]);
LISTA.LIVRE[-6]_LISTA.LIVRE[-6]-1;
```

```
LIST_AUXILIAR-LISTA.LIVRE[-10];
 LISTA.LIVRE[-10]_GFIELD(LISTA.LIVRE[LISTA.LIVRE[-10]],
  18,18);
SFIELD(LISTA.LIVRE[AUXILIAR],0,18,GFIELD(I,18,18));
IF GFIELD(I,18,18)=65536 THEN SFIELD(LISTA.LIVRE[
 AUXILIAR],18,18,GFIELD(J,18,18)) ELSE
IF GFIELD(LISTA.LIVRE[J],0,18)=65536 THEN
BEGIN
LISTA.LIVRE[-6]_LISTA.LIVRE[-6]-1;
 AUXILIAR_LISTA.LIVRE[-10];
 LISTA.LIVRE[-10]_GFIELD(LISTA.LIVRE[LISTA.LIVRE[-10]],
  18,18);
 SFIELD(LISTA.LIVRE[AUXILIAR],0,18,GFIELD(J,18,18));
 SFIELD(LISTA.LIVRE[AUXILIAR],18,18,1)
END
                                                    ELSE
SFIELD(LISTA.LIVRE[AUXILIAR],18,18,GFIELD(J,18,18))
END;
```

```
 TYPE EXEMP5.ALG
00010   BEGIN
00020    INTEGER TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP;
00030    READ(TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP);
00040    BEGIN
00050     INTEGER ARRAY LISTA.LIVRE[-10:TAMANHO.DA.LISTA.LIVRE];
00051     STRING ARRAY HEAP [-10:TAMANHO.DO.HEAP];
00052     EXTERNAL PROCEDURE PREPARAR.LISTA.LIVRE.E.HEAP;
00053     EXTERNAL INTEGER PROCEDURE LER.LISTA;
00054     EXTERNAL PROCEDURE IMPRIMIR;
00055     EXTERNAL PROCEDURE IMPRESSAO.DA.LISTA.LIVRE;
00056     EXTERNAL PROCEDURE LISTAGEM.DO.HEAP;
00057     EXTERNAL PROCEDURE RESULTADO.DO.CALCULO.DO.PREDICADO;
00058     EXTERNAL PROCEDURE COMPACTAR.HEAP;
00059     EXTERNAL PROCEDURE RECUPERAR.LISTA;
00060     EXTERNAL PROCEDURE ESTATISTICAS;
00061     EXTERNAL INTEGER PROCEDURE DECODIFICAR.ATOMO.NUMERICO;
00062     EXTERNAL PROCEDURE CODIFICAR.NUMERO;
00063     EXTERNAL BOOLEAN PROCEDURE ATOM, EQ, NUMBERP, NULL;
00064     EXTERNAL INTEGER PROCEDURE CAR, CDR, LIST;
00065     PREPARAR.LISTA.LIVRE.E.HEAP(LISTA.LIVRE,2,TAMANHO.DA.LISTA.LIV
RE,
00070   HEAP,TAMANHO.DO.HEAP);
00075     LISTA.LIVRE[-3]_LER.LISTA(LISTA.LIVRE,HEAP);
00085   NEWLINE(3);
00095     LISTA.LIVRE[-2]_LER.LISTA(LISTA.LIVRE,HEAP);
00105     IMPRIMIR(LISTA.LIVRE,HEAP,CAR(LISTA.LIVRE,LISTA.LIVRE[-3]));
00115     IMPRIMIR(LISTA.LIVRE,HEAP,CDR(LISTA.LIVRE,LISTA.LIVRE[-2]));
00125     IMPRIMIR(LISTA.LIVRE,HEAP,LIST(LISTA.LIVRE,HEAP,CAR(LISTA.LIVR
E,
00135   LISTA.LIVRE[-3]),CDR(LISTA.LIVRE,LISTA.LIVRE[-2])));
00145     NEWLINE(3);
00150     END
00160    END
00170   END

.EXE
LOADING

EXEMP5 2K CORE
EXECUTION
31 20
"(   ( ( ) ) ((\(\  (  ) )   ( ) )"



"  \  "\"(   ( (  ( )  ) ) )"



( ( ) )
```

```
( )

( ( ( ) ) )


END OF EXECUTION - 3K CORE

EXECUTION TIME: 0.19 SECS.

ELAPSED TIME:   1 MINS.  28.72 SECS.

.
```

ALEM DAS FUNCOES JA DESCRITAS COLOCAMOS A DISPOSICAO DO USUARIO ALGUMAS OUTRAS FUNCOES QUE NOS PARECERAM IMPORTANTES DE UM PONTO DE VISTA GERAL.

DUAS DELAS VISAM A CONVERSAO DE VALORES INTEIROS PARA VALORES SIMBOLICOS E VICE VERSA.

AS OUTRAS VISAM UMA MELHOR UTILIZACAO DA MEMORIA.

A) PROCEDURE CODIFICAR.NUMERO(LISTA.LIVRE,HEAP,I,J)

ESTA PROCEDURE APANHA O VALOR INTEIRO ARMAZENADO EM I E CONSTROI SUA EXPRESSAO SIMBOLICA. SE O NUMERO FOR NEGATIVO A ROTINA ACRESCENTA-LHE O SINAL NEGATIVO; O MESMO NAO E FEITO PARA NUMEROS POSITIVOS.

O STRING ASIIM CONSTRUIDO E ARMAZENADO NO ATOMO CUJA CELULA INDICATIVA ESTA APONTADA POR J.

B)INTEGER PROCEDURE DECODIFICAR.ATOMO.NUMERICO(LISTA.LIVRE,HEAP,I)

ESTA PROCEDURE APANHA O VALOR SIMBOLICO DE UM NUMERO, ARMAZENADO NO HEAP E CONSTROI SUA EXPRESSAO BINARIA QUE RETORNA NO IDENTIFICADOR DO PROCEDURE, QUE E UM PROCEDURE DO TIPO INTEIRO.

C) COMPACTAR.HEAP(LISTA.LIVRE,HEAP)

NAS ROTINAS DE LEITURA DE LISTAS JA APRESENTADAS ,TODO ATOMO LIDO, INDEPENDENTE DE TER SIDO LIDO ANTERIORMENTE OUNAO GANHAVA UMA NOVA REPRESENTACAO SIMBOLICA NA MEMORIA. EMBORA ESTA ATITUDE POSSA ESTAR SUJEITA A CRITICAS QUEREMOS RESSALTAR QUE NOSSO PROPOSITO FOI ESSE E FOI ADOTADO CONSCIENTEMENTE ,POIS QUISERMOS TER INSERIDA NO SISTEMA A POSSIBILIDADE DE ALTERAR PEDACOS DE ATOMO, AMPLIANDO DESTA FORMA A POSSIBILIDA-

DE DE MANIPULACAO DE SIMBOLOS. NO ENTAN-
TO ALGUNS PROBLEMAS NAO TEM NECESSIDADE DE TAIS RECURSOS
E, PELO CONTRARIO, TEM REQUISICOES ESTRITAS DE MEMORIA.
VISANDO PODER LIBERAR MEMORIA EMTAIS CASOS
INSERIMOS ESTA ROTINA QUE ELIMINA AS REPETICOES DE ATOMOS NA
MEMORIA, DEIXANDO APENAS UMA COPIA DE CADA TIPO. ESTA ROTINA
E DEMORADA E EM NOSSA OPINIAO SOMENTE DEVE SER USADA QUANDO
NENHUMA DAS OUTRAS FORMAS DE OBTENCAO DE RECURSOS RES-
SOLVER O PROBLEMA DE OBTENCAO DE MEMORIA.

D) PROCEDURE RECUPERAR.LISTA(LISTA.LIVRE,HEAP,I)

ESTE PROCEDIMENTO FAZ A TAREFA DE GARBAGE COLLECTOR DE NOSSO
SISTEMA. AO CONTRARIO DO LISP, POREM, ELE NAO E AUTOMATICO.
DENTRO DE UMSISTEMA COMO O LISP, AS LISTAS IMPRES-
CINDIVEIS AO SISTEMA TINHAM SEUS ENDERECOS GUARDADOS NU-
MA AREA ESPECIAL E O SISTEMA TINHA CONHECIMENTO, A CADA MOMEN-
TO DAS LISTAS QUE ELE PODERIA OU NAO RECUPERAR. NUM SISTEMA
COMO O NOSSO TAL FATO NAO OCORRE E DE UMA OU OUTRA FORMA
DEVEREMOS FORNECER AO PROGRAMA, EM SE DESEJANDO RECUPERAR LIS-
TAS , OS ENDERECOS DAS LISTAS QUE SE DESEJA RECUPERAR.
EMBORA ISSO POSSA PARECER DESVANTAJOSO, PARA NOS A COISA TOMA
FIGURA EXATAMENTE OPOSTA.
TODO O SISTEMA FOI CONCEBIDO E VOLTADO PARA UMA UTILIZACAO
ATRAVES DE TERMINAIS . TODAS AS ROTINAS DE IMPRESSAO VISAM
APENAS FORNECER A UM USUARIO DE TERMINAL O ESQUELETO DO
SISTEMA. O PROCESSO DE RECUPERACAO DE LISTA ACOMPANHA ESSA FI-
LOSOFIA. NA MEDIDA EM QUE O USUARIO TEM ACESSO A TODOS OS DA-
DOS DO SISTEMA ELE PODE CONTROLAR COM MUITO MAIOR EFICIENCIA

OS RECURSOS DELE. UMA SUGESTAO PARA SE TER CONTROLE DOS ENDERECOS DAS LISTAS E ARMAZENANDO-OS NUMA ZONA ESPECIAL DA LISTA LIVRE. OUTRO CONSISTE EM FAZER A IMPRESSAO DO ENDERECO DA LISTA CADA VEZ QUE ELA E FORMADA.

A ROTINA LE OS VALORES INTEIROS APONTADORES PARA AS CABECAS DE LISTAE VAI RECUPERANDO AS LISTAS ATE QUE LHE SEJA FORNECIDO O VALOR 1, MOMENTO EM QUEA RECUPERACAO E ENCERRADAE O CONTROLE RETORNA A FUNCAO LIST QUE VAI CONTINUAR AEXECUCAO DE SUAS TAREFAS. SE NAO HOUVER LISTA RECUPERAVEL, O PROGRAMA DEVERA SER CORTADO E AS DIMENSOES DA LISTA LIVRE AUMENTADAS.

O ENDERECO 0 DA LISTA LIVRE E RESERVADO NESTA ROTINA PARA A LEITURA DOS ENDERECOS E COMO TAL NAO DEVE SER USADO COM OUTROS FINS.

APRESENTAMOS A SEGUIR LISTAGENS DAS ROTINAS E UM EXEMPLO DE SUA UTILIZACAO.

```
PROCEDURE CODIFICAR.NUMERO(LISTA.LIVRE,HEAP,I,J);
VALUE I,J;
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
INTEGER I,J;
BEGIN
INTEGER K,L,M;
INTEGER ARRAY DIGITOS[1:12];
L_SIGN(I);
K_1;
I_ABS(I);
IF I=0 THEN
BEGIN
DIGITOS[K]_48;
K_K+1;
END;
WHILE I#0 DO
BEGIN
DIGITOS[K]_(I REM 10)+ 48;
I_I DIV 10;
K_K+1;
END;
IF L<0 THEN DIGITOS[K]_45 ELSE K_K-1;
L_GFIELD(LISTA.LIVRE[J],18,18);
HEAP[L]_NEWSTRING(K,7);
FOR M_1 UNTIL K DO
HEAP[L].[M]_GFIELD(DIGITOS[K-M+1],29,7);
END;
```

```
INTEGER PROCEDURE DECODIFICAR.ATOMO.NUMERICO
(LISTA.LIVRE,HEAP,I);
VALUE I;
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
INTEGER I;
BEGIN
INTEGER J,K,L;
K_GFIELD(LISTA.LIVRE[I],18,18);
L_0;
FOR J_1 UNTIL LENGTH(HEAP[K]) DO
L_L*10+(HEAP[K].[J]-48);
DECODIFICAR.ATOMO.NUMERICO_L
END;
```

```
PROCEDURE COMPACTAR.HEAP(LISTA.LIVRE,HEAP);
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
BEGIN
INTEGER I,J;
INTEGER ARRAY REFERENCIAL[1:LISTA.LIVRE[-4]-1];
FOR I_1 UNTIL LISTA.LIVRE[-4]-2 DO
FOR J_I+1 UNTIL LISTA.LIVRE[-4]-1 DO
IF REFERENCIAL[J]=0 AND HEAP[I]=HEAP[J] THEN
BEGIN
DELETE(HEAP[J]);
REFERENCIAL[J]_1;
HEAP[J]_HEAP[I]
END
END
```

```
PROCEDURE RECUPERAR.LISTA(LISTA.LIVRE,HEAP,L);
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
INTEGER L;
BEGIN
INTEGER K;
EXTERNAL INTEGER PROCEDURE CAR,CDR;
PROCEDURE GARBAGE.COLLECTOR(LISTA.LIVRE,HEAP,I);
VALUE I;
INTEGER ARRAY LISTA.LIVRE;
STRING ARRAY HEAP;
INTEGER I;
BEGIN
IF I=1 THEN GO TO FIM ELSE
IF GFIELD(LISTA.LIVRE[I],0,18)#65536 THEN
BEGIN
GARBAGE.COLLECTOR(LISTA.LIVRE,HEAP,CAR(LISTA.LIVRE,I));
GARBAGE.COLLECTOR(LISTA.LIVRE,HEAP,CDR(LISTA.LIVRE,I));
LISTA.LIVRE[-6]_LISTA.LIVRE[-6]+1;
LISTA.LIVRE[I]_LISTA.LIVRE[-10];
LISTA.LIVRE[-10]_I
END
                                        ELSE
BEGIN
K_GFIELD(LISTA.LIVRE[I],18,18);
DELETE(HEAP[K]);
LISTA.LIVRE[-6]_LISTA.LIVRE[-6]+1;
LISTA.LIVRE[-5]_LISTA.LIVRE[-5]+1;
LISTA.LIVRE[I]_LISTA.LIVRE[-10];
LISTA.LIVRE[-10]_I
```

```
        END;
FIM:    END;
        NEWLINE(3);
        WRITE (" SUA LISTA LIVRE ESTA ESGOTADA. ");
        NEWLINE(3);
        WRITE(" TENTE RECUPERAR ALGUMA LISTA JA NAO NECESSARIA. ");
        NEWLINE(3);
        WRITE(" PARA TANTO ENTRE COM O ENDERECO DA RAIZ DA LISTA. ");
        NEWLINE(3);
        WRITE(" QUANDO DESEJAR PARAR A RECUPERACAO, BATA 1 . ");
        NEWLINE(3);
LOOP:   READ(LISTA.LIVRE[0]);
        NEWLINE(3);
        IF LISTA.LIVRE[0]=1 THEN GO TO TERM;
        GARBAGE.COLLECTOR(LISTA.LIVRE,HEAP,LISTA.LIVRE[0]);
        GO TO LOOP;
TERM:   END
```

```
TYPE EXEMP7.ALG
  TOTAL OF 730 BLOCKS IN 308 FILES ON DSKA: [12007,12005]

.00010  BEGIN
00020    INTEGER TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP;
00030    READ(TAMANHO.DA.LISTA.LIVRE,TAMANHO.DO.HEAP);
00040    BEGIN
00050     INTEGER ARRAY LISTA.LIVRE[-10:TAMANHO.DA.LISTA.LIVRE];
00051     STRING ARRAY HEAP [-10:TAMANHO.DO.HEAP];
00052     EXTERNAL PROCEDURE PREPARAR.LISTA.LIVRE.E.HEAP;
00053     EXTERNAL INTEGER PROCEDURE LER.LISTA;
00054     EXTERNAL PROCEDURE IMPRIMIR;
00055     EXTERNAL PROCEDURE IMPRESSAO.DA.LISTA.LIVRE;
00056     EXTERNAL PROCEDURE LISTAGEM.DO.HEAP;
00057     EXTERNAL PROCEDURE RESULTADO.DO.CALCULO.DO.PREDICADO;
00058     EXTERNAL PROCEDURE COMPACTAR.HEAP;
00059     EXTERNAL PROCEDURE RECUPERAR.LISTA;
00060     EXTERNAL PROCEDURE ESTATISTICAS;
00061     EXTERNAL INTEGER PROCEDURE DECODIFICAR.ATOMO.NUMERICO;
00062     EXTERNAL PROCEDURE CODIFICAR.NUMERO;
00063     EXTERNAL BOOLEAN PROCEDURE ATOM, EQ, NUMBERP, NULL;
00064     EXTERNAL INTEGER PROCEDURE CAR, CDR, LIST;
00065     PREPARAR.LISTA.LIVRE.E.HEAP(LISTA.LIVRE,2,TAMANHO.DA.LISTA.LIV
RE,
00070   HEAP,TAMANHO.DO.HEAP);
00075     LISTA.LIVRE[-3]_LER.LISTA(LISTA.LIVRE,HEAP);
00085   NEWLINE(3);
00095     LISTA.LIVRE[-2]_LER.LISTA(LISTA.LIVRE,HEAP);
00097      NEWLINE(3);
00100      PRINT(LISTA.LIVRE[-2]); NEWLINE(3);
00105     WHILE DECODIFICAR.ATOMO.NUMERICO(LISTA.LIVRE,HEAP,CAR(LISTA.LI
VRE,
00110    LISTA.LIVRE[-3])) > 30 DO
00113     BEGIN
00115      CODIFICAR.NUMERO(LISTA.LIVRE,HEAP,LISTA.LIVRE[-1] ,CAR(
00116      LISTA.LIVRE,LISTA.LIVRE[-3]));
00117      LISTA.LIVRE[-1]_LIST(LISTA.LIVRE,HEAP,CAR(LISTA.LIVRE,
00118      LISTA.LIVRE[-3]),CDR(LISTA.LIVRE,LISTA.LIVRE[-3]));
00135      LISTA.LIVRE[-3]_CDR(LISTA.LIVRE,LISTA.LIVRE[-3]);
00139     IMPRIMIR(LISTA.LIVRE,HEAP,LISTA.LIVRE[-3]);
00145     END;
00147     IMPRIMIR(LISTA.LIVRE,HEAP,LISTA.LIVRE[-1]);
00150     END
00160    END
00170   END

.EXE
ALGOL: RECUP6
LOADING

EXEMP7 3K CORE
```

```
EXECUTION
31 12
"( 34 35 36 37  38 39 15 )"


"( 1 2 3 4 5 )"


          25


( 35 36 37 38 39 15 )

( 36 37 38 39 15 )

( 37 38 39 15 )

( 38 39 15 )

( 39 15 )

 SUA LISTA LIVRE ESTA ESGOTADA.

 TENTE RECUPERAR ALGUMA LISTA JA NAO NECESSARIA.

 PARA TANTO ENTRE COM O ENDERECO DA RAIZ DA LISTA.

 QUANDO DESEJAR PARAR A RECUPERACAO, BATA 1 .

25

1



( 15 )

( 30 15 )
END OF EXECUTION - 4K CORE
EXECUTION TIME:  0.63 SECS.
ELAPSED TIME:    2 MINS.  10.43 SECS.
```

# II. UMA ABORDAGEM SOB O PONTO DE VISTA DE IMPLEMENTACAO.

## II.1-ANALISE CRITICA SOBRE ALGUMAS IMPLEMENTACOES DO LISP.

ANALISE CRITICA DE ALGUMAS INTERPRETACOES DO LISP

ATER-NOS-EMOS NESTE PARAGRAFO AS IMPLEMENTACOES DO LISP QUE FUNCIONAM DE UM MODO INTERPRETATIVO; EMBORA JA TENHAMOS OUVIDO FALAR DE COMPILADORES LISP, NAO TIVEMOS A MAO NENHUMA DOCUMENTACAO SOBRE ELES. MAIS ESPECIFICAMENTE PRENDER-NOS-EMOS A 2 INTERPRETACOES: A DE MCCARTHY NO MIT, DOCUMENTADA NO LISP 1.5 PROGRAMMER'S MANUAL E A DE GUY AUGIER/SIANG WUN SONG/JORGE STOLFI DISPONIVEL NO B-3500 DA UNIVERSIDADE DE SAO PAULO.

NOSSA ANALISE, POR MOTIVOS VARIOS NAO VAI SER EXAUSTIVA; PROCURAREMOS DESCREVER EM LINHAS GERAIS COMO FUNCIONA UM INTER_ PRETADOR LISP, RESSALTANDO ALGUNS PONTOS DESSE SISTE_ MA QUE NOS PARECEM NEGATIVOS POR DEMANDAR GRANDE TEMPO DE EXECU_ CAO OU UM GASTO EXAGERADO DE MEMORIA.

ADOTAREMOS COMO BASE O SISTEMA BRASILEIRO.

ESSE SISTEMA E FORMADO POR 6 PECAS ESSENCIAIS:

1) ARQUIVOS NO DISCO

2)TABELAS DE SIMBOLOS

3)LISTA LIVRE

4)SUPERVISOR

5)INTERPRETADOR PROPRIAMENTE DITO

6)GARBAGE COLLECTOR

OS ARQUIVOS NO DISCO CONTEM A TABELA DE SIMBOLOS ATO_ MICOS BASICOS DO LLISP ( CAR, CDR, ETC. ), A P-LISTAS (LISTAS DE PROPRIEDADES) ASSOCIADAS A CADA TIPO PARTI_ CULAR DE ATOMO, AS MENSAGENS DE ERRO, ALGUNS OUTROS ARQUIVOS DE MENOR IMPORTANCIA QUE NAO IREMOS CITAR.

AS TABELAS DE SIMBOLOS BASICOS SAO TABELAS, ORDENADAS PELO NUMERO DE CARACTERES COMPONENTES DE SEUS ELEMENTOSE QUE CONTEM OS SIMBOLOS ATOMICOS BASICOS DA LINGUAGEM, POR EXEMPLO, CAR, ATOM, ETC.. ASSOCIADOA A CADA ATOMO TEMOS UMA LISTA QUE NOS FORNECE AS PROPRIEDADES CARACTERISTICAS DAQUELE ATOMO PARTICULAR. POR EXEMPLO A P-LISTA DE CDR CONTEM UM INDICADOR DE SUBROTINA QUE ESPECIFICA A CARACTERISTICA ESPECIAL DESSE ATOMO, O ENDERECO DELA, O NUMERO DE ARGUMENTOS, ETC..

A EQUIVALENCIA ENTRE A TABELA DE SIMBOLOS E AS P-LISTAS E FEITA POR ORDENACAO ENTRE OS NOMES E AS CABECAS DE LISTAS.

A LISTA LIVRE E FORMADA POR CELULAS DE 14 DIGITOS, COM 2 CAMPOS DE 6 DIGITOS PARA ENDERECO E 2 CAMPOS DE 1 DIGITO PARA ESPECIFICAR A NATUREZA DO ATOMO ( NUMERO, SUBROTINA, ETC ) E PARA USO DO COLETOR DE LIXO.

A LISTA LIVRE E INICIALIZADA PELO SUPERVISOR; 2 OUTRAS ROTINAS RETIRAM E DEVOLVEM CELULAS A LISTA LIVRE.E IMPORTANTE RELEMBRAR QUE NO LISP PURO AS CELULAS DA LISTA LIVRE SERVEM PARA ARMAZENAR TANTO DADOS ALFANUMERICOS COMO NUMEROS INTEIROS OU EM PONTO FLUTUANTE,PROGRAMAS,LISTAS DE PROPRIEDADE,ETC..

O SUPERVISOR NO LISP E QUEM FAZ AS INICIALIZACOES DO SISTEMA(ABERTURA DOS ARQUIVOS, PASSAGEM DAS TABELAS DE SIMBOLOS ATOMICOS E P-LISTAS BASICAS PARA A MEMORIA CENTRAL). E QUEM LE TAMBEM O ARQUIVO INICIAL DE CARTOES,ARMAZENANDO-O NO DISCO PARA POSTERIOR RELEITURA E ANALISE E QUEM LIGA AS OPCOES DOS CARTOES DE CONTROLE(TRACE,LISTAGEM DE PROGRAMA, ETC..). UMA VEZ ESTABELECIDAS AS OPCOES DOS CARTOES DE CONTROLE, E INICIADA ENTAO A LEITURA PARA A MEMORIA DO PROGRAMA, JA SENDO FEITA A ESTRUTURACAO

DO MESMO NA FORMA DE LISTA. ESSA LEITURA E LENTA POIS PARA CADA SIM_
BOLO DE BASE ( ( , ) ) OU ATOMO E FEITA UMA CONSULTA A TABELA DE SIM_
BOLOS BASICOS PARA OBTENCAO DO ENDERECO DE SUA P-LISTA OU PARA A
CRIACAO DE UMA NOVA P-LISTA E ATUALIZACAO DA TABELA. A LEITURA
E FEITA POR UM CONJUNTO DE ROTINAS QUE UTILIZA UM STACK PARA A CRIA_
CAO DAS ARVORES NA MEMORIA. A EXPRESSAO PARENTIZADA
DAS LISTAS E TRANSFORMADA NO FINAL NUMA ARVORE BINARIA CONSTANDO APE_
NAS DE APONTADORES. O CONTROLE E ASSUMIDO ENTAO PELO INTERPRETADOR
QUE PASSA A VASCULHAR ESSAS ARVORES SEGUNDO OS CRITERIOS QUE ENUNCIA_
REMOS ABAIXO.

ADOTAREMOS NA DESCRICAO QUE SE SEGUE O FORMALISMO E A NO_
MENCLATURA DE MCCARTHY. O INTERPRETADOR E FORMADO POR 3 FUNCOES PRIN_
CIPAIS: EVALQUOTE, EVAL E APPLY.

A FUNCAO UNIVERSAL EVALQUOTE OBEDECE A SEGUINTE IDENTIDADE:

SEJA F UMA FUNCAO ESCRITA COMO UMA M-EXPRESSAO, E SEJA FN
SUA EQUIVALENTE S-EXPRESSAO. SE F E UMA FUNCAO DE N ARGUMENTOS E
ARGS = ( ARG1 ARG2 ... ARGN ) E A LISTA DAS N S-EXPRESSOES USADAS
COMO ARGUMENTOS, ENTAO:

EVALQUOTE[FN; ARGS]=F[ARG1; ... ARGN]

UM EXEMPLO SEGUE:

F: L[[X; Y]; CONS[CAR[X]; Y]]

FN: (LAMBDA ( X Y) (CONS (CAR X) Y ))

ARG1: (A B)

ARG2: (C D)

ARGS ( (A B) (C D))

EVALQUOTE[(LAMBDA (X Y)(CONS (CAR X) Y)); ((A B) (C D))]=

L[[X; Y]; CONS[CAR[X]; Y]][(A B); (C D)]=

=< A C D >

EVALQUOTE E DEFINIDA USANDO 2 FUNCOES PRINCIPAIS, CHAMADAS EVAL E APPLY. APPLY MANIPULA UMA FUNCAO E SEUS ARGUMENTOS, ENQUANTO EVAL MANIPULA FORMAS. CADA UMA DESSAS FUNCOES TEM AINDA UM OUTRO ARGUMENTO QUE E USADO COMO LISTA DE ASOCIACAO PARA ARMAZENAMENTO DOS VALORES DAS VARIAVEIS E NOMES DE FUNCOES. A DEFINICAO COMPLETA DA FUNCAO VEM ABAIXO:

EVALQUOTE[FN; X]=APPLY[FN; X; NIL]

ONDE

```
APPLY[FN; X; A]=
  [ATOM[FN]_[EQ[FN; CAR]_CAAR[X];
             EQ[FN; CDR]_CDAR[X];
             EQ[FN; CONS]_CONS[CAR[X]; CADR[X]];
             EQ[FN; ATOM]_ATOM[CAR[X]];
             EQ[FN; EQ]_EQ[CAR[X]; CADR[X]];
             T-APPLY[EVAL[FN; A]; X; A]];
   EQ[CAR[FN]; LAMBDA]_EVAL[CADDR[FN]; PAIRLIS[CADR[FN]; X; A]];
   EQ[CAR[FN]; LABEL]_APPLY[CADDR[FN]; X; CONS[CONS[CADR[FN];
                CADDR[FN]]; A]]]
EVAL[E; A]=[ATOM[E]_CDR[ASSOC[E; A]];
           ATOM[CAR[E]]_
                [EQ[CAR[E], QUOTE]_CADR[E];
                 EQ[CAR[E]; COND]_EVCON[CDR[E]; A];
                 T_APPLY[CAR[E]; EVLIS[CDR[E]; A]; A]];
          T-APPLY[CAR[E]; EVLIS[CDR[E]; A]; A]]
```

AS DEFINICOES DE PAIRLIS, ASSOC, EVCON, EVLIS SEGUEM:

PAIRLIS[X;Y;A]=[NULL[X]_A;CONS[CONS[CAR[X];CAR[Y]];
PAIRLIS[CDR[X];CDR[Y];A]]]

ASSOC[X;A]=[EQUAL[CAAR[A];X]_CAR[A];T_ASSOC[X;CDR[A]]]

EVCON[C;A]=[EVAL[CAAR[C];A]_EVAL[CADAR[C];A];
T-EVCON[CDR[C];A]]

EVLIS[M;A]=[NULL[M]_NIL;
T_CONS[EVAL[CAR[M];A];EVLIS[CDR[M];A]]]

OUTRAS FUNCOES USADAS SEGUEM ABAIXO:

CAAR[X]=CAR[CAR[X]]

CADR[X]=CAR[CDR[X]]; ETC.;

EQUAL[X;Y]=[ATOM[X]_[ATOM[Y]_EQ[X;Y]; T_F];
EQUAL[CAR[X];CAR[Y]]_EQUAL[CDR[X];CDR[Y]];
T_F]

QUOTE E UMA FORMA ESPECIAL QUE PREVINE QUE SEU ARGUMENTO SEJA CALCULADO.

ALGUNS EXEMPLOS DE ATUACAO DO INTERPRETADOR SEGUEM ABAIXO:

SE FORNECEMOS A FUNCAO EVALQUOTE O PAR:

CAR ( ( A B ) )

ESTA PASSARA A APPLY

CAR ( ( A B ) ) NIL

QUE DEVOLVERA O VALOR A.

SE FORNECEMOS A EVALQUOTE O PAR

( LAMBDA ( X Y ) ( CONS X Y ) ) ( A B )

ELA PASSARA A APPLY O TRIPLETO

( LAMBDA ( X Y ) ( CONS X Y ) ) ( A B ) NIL

A FUNCAO APPLY IRA ASSOCIAR ESTAS VARIAVEIS E FORNECER A FUNCAO E A LISTA DE ASSOCIACAO PARA EVAL.

EVAL RECEBERA O SEGUINTE DUPLETO:

( CONS X Y ) ( (X. A) (Y. B) )

E DEVOLVERA PARA APPLY

CONS ( A B ) ( (X. A) (Y. B) )

A EXECUCAO DE APPLY LEVERA FINALMENTE A EXECUCAO DA FUNCAO CONS OBTENDO-SE O VALOR FINAL ( A . B ).

UMA VEZ CALCULADA A FUNCAO, O CONTROLE PASSA AS ROTINAS DE IMPRESSAO QUE CUIDAM DE PASSAR AO MEIO EXTERIOR OS RESULTADOS DA COMPUTACAO.

OS EXEMPLOS APRESENTADOS, EMBORA BASTANTE SIMPLES JA PERMITEM TER UMA IDEIA DA LENTIDAO DO PROCESSO INTERPRETATIVO, QUE IRA CONSTRUINDO AS LISTAS A CADA PASSO E TAMBEM DAS EXIGENCIAS DE MEMORIA QUE SAO REQUERIDAS DURANTE O PROCESSO.

A PECA FINAL DO SISTEMA E BASTANTE IMPORTANTE DO PONTO DE VISTA DOS TEMPOS DE EXECUCAO E O GARBAGE COLLECTOR. E ELE QUEM NOS PERMITE RECUPERAR AS LISTAS JA NAO NECESSARIAS E COMO NO LISP TUDO( PROGRAMAS E DADOS ) ESTA ESTRUTURADO EM FORMA DE LISTAS AS NECESSIDADES DE MEMORIA SAO GRANDES; O GARBAGE E FREQUENTEMENTE SOLICITADO VARIAS VEZES DURANTE A EXECUCAO DE UM PROGRAMA.

EXISTEM VARIAS IDEIAS SOBRE GARBAGE COLLECTION: NO LISP O USUAL E TERMOS GARBAGE COLLECTORS DE 2 PASSOS, ISTO E, NUMA PRIMEIRA PASSAGEM O GARBAGE ASSINALA AS CELULAS QUE JA NAO SAO MAIS NECESSARIAS E NUM SEGUNDO PASSO E FEITA A RECUPERACAO PROPRIAMENTE DITA. A RECUPERACAO NOS SISTEMAS LISP TRADICIONAIS E AUTOMATICA. QUANDO OCORRE

O ESGOTAMENTO DAS DISPONIBILIDADES DA LISTA LIVRE , O CONTROLE E E FORNECIDO AO COLETOR DE LIXO QUE COM BASE NUM ARMAZENAMENTO PREVIO DAS LISTAS UTEIS PASSA A ASSINALAR AS CELULAS DO SISTEMA QUE PERTENCEM A ESTAS LISTAS. UMA VEZ ESGOTADAS AS CELULAS INDICATIVAS DE LISTAS UTEIS, E FEITA UMA SEGUNDA PASSAGEM ,MOMENTO EM QUE E FEITA EFETIVAMENTE A RECUPERACAO DAS CELULAS COM O SEU RETORNO A LISTA LIVRE. VARIOS ALGORITMOS EXISTEM PARA A COLECAO DE LIXO TODOS TENDO COMO PREOCUPACAO FUNDAMENTAL O FATO DE QUE A MEMORIA UTILIZAVEL DO SISTEMA ESTA ESGOTADA NO MOMENTO DE EXECUCAO DA RECUPERACAO. EM NOSSO SISTEMA O ENFOQUE E DIFERENTE E EMBORA TENHAMOS UMA FUNCAO QUE FACA A RECUPERACAO DE LISTAS, AS AREAS DESTINADAS A LISTA LIVRE E AO HEAP SAO DIVERSAS E A ROTINA SOMENTE E ACIONADA QUANDO A LISTA LIVRE E ESGOTADA.

NOSSA LISTA LIVRE FORNECE CELULAS APENAS PARA AO ARMAZENAMENTO DE APONTADORES. NO ARRAY LISTA LIVRE TEMOS APENAS O ESQUELETO DAS LISTAS ; UMA RECUPERACAO DE CELULAS VISA APENAS OTIMIZAR O GASTO DE CELULAS PARA A MONTAGEM DE LISTAS, LIBERANDO AREA PARA O RESTO DA MEMORIA, AREA ESSA QUE PODERA SER UTILIZADA PARA ARMAZENAMENTO DE ATOMOS, PROCEDIMENTOS RECURSIVOS, ETC..

## II.2- VANTAGENS E DESVANTAGENS DA IMPLEMENTACAO DO LISP-ALGOL.

VANTAGENS E DESVANTAGENS DA IMPLEMENTACAO DO LISP-ALGOL

SEGUNDO NOSSO PONTO DE VISTA, VARIAS SAO AS VANTAGENS QUE UMA IMPLEMENTACAO COMO A REALIZADA TEM SOBRE O LISP TRADICIONAL.

A QUE NOS PARECE MAIS SIGNIFICATIVA E A POSSIBILIDADE ILIMITADA DE INSERCAO DE NOVAS ROTINAS CONSTRUIDAS EM LINGUAGEM DE ALTO NIVEL, SEM OS INCOMODOS E EXTENSAO DAS ROTINAS ESCRITAS EM UMA LINGUAGEM DE MONTAGEM. O SISTEMA E NESSE SENTIDO ILIMITADO.

ACOMPANHANDO ESSA PRIMEIRA VANTAGEM TEMOS A DE PODER CARREGAR NA MEMORIA PARA CADA PROGRAMA PARTICULAR APENAS AQUELAS ROTINAS EFETIVAMENTE USADAS PELO PROGRAMA EM QUESTAO. ISSO NOS PERMITE, AO MESMO TEMPO, UMA COEXISTENCIA ENTRE UM SISTEMA DE TAMANHO ILIMITADO COM AS LIMITACOES NATURAIS DE MEMORIA DE UM COMPUTADOR. ALEM DISSO, A POSSIBILIDADE DE COMPILACAO SEPARADA DO PROGRAMA PRINCIPAL NOS PERMITE O ARMAZENAMENTO DE ROTINAS EM FORMATO RELOCAVEL QUE IRAO APENAS SER CARREGADAS NA MEMORIA QUANDO DA EXECUCAO DE UM PROGRAMA.

A POSSIBILIDADE DE IMPRESSAO DA LISTA LIVRE E DO HEAP, ATRAVES DAS ROTINAS ESPECIAIS MONTADAS COM ESSA FINALIDADE NOS PERMITEM UM CONTROLE TOTAL DA SITUACAO DAS LISTAS EM QUALQUER MOMENTO DA EXECUCAO DE UM PROGRAMA. ESSA POSSIBILIDADE E EXTREMAMENTE UTIL NUM SISTEMA DE TERMINAIS, POIS NOS PERMITE A TOMADA DE PROVIDENCIAS VARIAS AO TERMINAL EM FUNCAO DOS RESULTADOS VISTOS DURANTE A EXECUCAO.

OUTRO PONTO QUE FIZEMOS QUESTAO DE DEIXAR A CONTROLE DO USUARIO E O DIMENSIONAMENTO DE SSUA LISTA LIVRE E SEU HEAP PODENDO ELE DESSA FORMA EM FUNCAO DE SUAS PREVISOES RESERVAR A ME-

MORIA MAIS ADEQUADA AO SEU PROGRAMA. COMO A PREPARACAO DA LISTA LI_
VRE DEMANDA UMA RAZOAVEL PERDA DE TEMPO, ESSE TEMPO E GANHO SE UMA
MENOR RESERVA E FEITA.

NA CONCEPCAO GERAL DO SISTEMA PREOCUPAMO-NOS COM UTILIZAR
OTIMAMENTE A MEMORIA EM VARIOS SENTIDOS. A CONCEPCAO DO CAR E DO CDR
NOS PARECEU FUNDAMENTAL: RESERVAMOS UMA PALAVRA PARA CADA CELU_
LA. CADA CELULA GUARDA 2 APONTADORES COM 16 BITS PERMITINDO EN_
DERECAMENTOS ATE 64 K. DOS 2 BITS RESTANTES EM CADA MEIA PALAVRAUM
E USADO PARA INDICADOR DE ATOMOENQUANTO O OUTRO ESTA LIBERADO PARA
QUALQUER OUTRA UTILIZACAO. OS ATOMOS, ATRAVES DO CONCEITO JA IMPLE_
MENTADO DE STRINGS UTILIZAM TAMBEM COMPACTAMENTE A MEMORIA; PODEMOS
TER ATE 5 SIMBOLOS ALFANUMERICOS POR PALAVRA.

NOSSO SISTEMA TEM UMA OUTRA VANTAGEM ADICIONAL SO_
BRE O LISP QUE E A POSSIBILIDADE DE FAZER ALTERACOES NOS SIMBOLOS
DOS ATOMOS. SUA CONCEPCAO TAMBEM TEM COMO PREOCUPACAO FUNDAMENTAL A
MANIPULACAO DE STRINGS; ISSO NAO IMPEDE POREM A EXECUCAO DE OPE_
RACOES COM ATOMOS NUMERICOS ATRAVES DA UTILIZACAO DAS ROTINAS DE
CONVERSAO . COMO NO LISP ENTRETANTO TAL PROCEDER E INEFICIENTE:
SUA UTILIZACAO NAO DEVE SE CONSTITUIR UMA TONICA DO SISTEMA MAS UMA
SIM UM ACESSORIO QUE PODE SER UTILIZADO MAS CUJO USO IMPLICA DISPEN_
DIO EXAGERADO DE TEMPO.

A RECUPERACAO NAO AUTOMATICA DAS LISTAS PERMITE UM MELHOR
CONTROLE DO ESPACO QUE SE DESEJA TER; NO LISP TRADICIONAL, EM GERAL
AS RECUPERACOES ERAM GLOBAIS E AS VEZES , EMBORA SE TENHA NECES_
SIDADE DE MEMORIA, NAO SE TEM NECESSIDADE DE MUITA MEMORIA. NO SIS_
TEMA PROPOSTO RECUPERA_SE O NUMERO DE LISTAS QUE SE QUEIRA; TAO LOGO
CESSE A RECUPERACAO O CONTROLE E DEVOLVIDO A ROTINA QUE RETIRA CE_

LULAS DA LISTA LIVRE. SE HOUVER NECESSIDADE DE MAIS CELULAS TAL INFORMACAO E IMPRESSA NA IMPRESSORA DO TERMINAL.

ALGUNS ASPECTOS QUE NOS PARECERAM NEGATIVOS NO SISTEMA E QUE NAO PUDEMOS EVITAR FORAM AS DECLARACOES EXTERNAL E AS PASSAGENS DE PARAMETROS PELA LISTA LIVRE.

AS DECLARACOES EXTERNAL DO INICIO DO PROGRAMA PRENDEM-SE A CARACTERISTICAS DO COMPILADOR ALGOL UTILIZADO. VISTO SER UM COMPILADOR DE UM UNICO PASSO O APARECIMENTO DE UM IDENTIFICADOR NAO PREVIAMENTE DECLARADO LEVARIA A UMA MENSAGEM DE ERRO. ESSE FATO SE RETE NA TENTATIVA DE USAR VARIAVEIS GLOBAIS EM PROCEDURES COMPILADOS EXTERNAL. ANTE OUTRA SOLUCAO QUE SERIA PASSAR ESSAS VARIAVEIS COMO PARAMETROS FORMAIS OPTAMOS POR MANTE-LOS GLOBAISPASSANDO A LISTA LIVRE "BY NAME". O SISTEMA FOI PROJETADO PARA SER UTILIZADO EM TERMINAIS; EMBORA NAO SEJA PROIBITIVA SUA UTILIZACAO EM BATCH ALGUNS RECURSOS DO SISTEMA VER-SE-IAM DESLOCADOS.

OS ASPECTOS APRESENTADOS SAO AQUELES QUE NOS FOI DADO PERCEBER; ESPERAMOS QUE AS CRITICAS E SUGESTOES PERMITAM UMA MELHORIA E O REFINAMENTO DO SISTEMA.

# BIBLIOGRAFIA


LISP 1.5 PROGRAMMER'S MANUAL; THE COMPUTATION
CENTER AND RESEARCH LABORATORY; M. I. T.

LINGUAGEM LISP E SEU INTERPRETADOR; SIANG WUN SONG

CONSTRUCAO DO INTERPRETADOR LISP NO SISTEMA B-3500; GUY AUGIER,
SIANG WUN SONG, JORGE STOLFI

MATHEMATICAL LANGUAGES HANDBOOK; DECSYSTEM HANDBOOH SERIES

COMMUNICATIONS OF THE ACM; VOLUME 6/NUMBER9/SEPTEMBER 1963;
SYMMETRIC LIST PROCESSOR; J. WEIZENBAUM

COMMUNICATIONS OF THE ACM; VOLUME 3/NUMBER 4/APRIL 1960;
RECURSIVE FUNCTIONS OF SYMBOLIC EXPRESSIONS AND THEIR COMPU_
TATION BY MACHINE, PART I; JOHN MCCARTHY

PROGRAMMATION NON NUMERIQUE LISP 1.5; D. RIBBENS

LISP 1.5 PRIMER; CLARK WEISSMAN

REVUE FRANCAISE DE TRAITEMENT DE L'INFORMATION; 4EME TRIMESTRE 1965;
DEFINITION DE PROCEDURES LISP EN ALGOL; JACQUES COHEN ET
NGUYEN-HUU-DUNG

REVUE FRANCAISE DE TRAITEMENT DE L'INFORMATION; 4EME TRIMESTRE 1966;
REMARQUES SUR LES PROCEDURES LISP EN ALGOL; JACQUES COHEN

THE ART OF COMPUTER PROGRAMMING; DONALD KNUTH

LIST PROCESSING; J. M. FOSTER

A VIEW OF PROGRAMMING LANGUAGES; B. A. GALLER AND A. J. PERLIS

COMMUNICATIONS OF THE ACM; VOLUME 15/NUMBER 15/JULY 1972;
PROGRAMMING LANGUAGES : HYSTORY AND FUTURE; JEAN SAMMET;
PROGRAMMING SYSTEMS AND LANGUAGES; SAUL ROSEN