PCS2056 – Linguagens e Compiladores

2ª parte do projeto

Sintaxe formal

Alan Raso – 8041511. Descrição informal da linguagem

A linguagem definida tem uma sintaxe muito semelhante a da linguagem C, com apenas algumas diferenças. As palavras reservadas, sendo abreviações ou não, estão na língua portuguesa. Além disso, a atribuição é feita utilizando o sinal “:=” ao invés de “=”, e a comparação de expressões utiliza “=” ao invés de “==”. Outra modificação é a declaração de funcões, que no C é feita primeiramente explicitando o tipo de retorno, ou vazio, caso não haja, depois o nome da função, seguido da declaração dos parâmetros, por exemplo: “int fatorial(int x)”. A nova linguagem possui a palavra reservada “func” fixa no início, depois o nome da função, a declaração dos parâmetros e por último o tipo de retorno entre parêntesis, por exemplo: “func fatorial(int x)(int)”.

1.1 Declaração de variáveis

int *var*; caract var; pflut var; bool var;

int var[3]; caract var[3][3]; bool var[3][3][3];

comp estrutura { *declaração de variáveis* }

1.2 Declaração de tipos

deftipo *novotipo* int;

deftipo comp { *declaração* *de* *variáveis* } estrutura;

1.3 Atribuição

a := expressão;

1.4 Funções

func função (int parametro)(int retorno) { *código;* retorna 1; }

a := fatorial(2);

func imprimir(caract c)() { *código* }

imprimir(‘a’);

1.5 Condição

se (expressão booleana) { *código* }

se (expressão booleana) { *código* } senão { *código* }

se (expressão booleana) { *código* } senão se { *código* } senão { *código* }

1.6 Loop

enquanto (*condição*) { *código* }

para (*atribuição*; *condição*; *atribuição*) { *código* }

2. Exemplo de programa

Programa que declara a função fatorial e calcula o fatorial do número dado:

func fatorial(int x)(int) {

se (x = 1) {

retorna 1;

}

retorna x\*fatorial(x-1);

}

func main()() {

int n; int fat;

scanf(“%d”, &n);

fat = fatorial(n);

printf(fat);

}

3. Descrição da linguagem em BNF

<Tipo> ::= bool|caract|int|pflut

<TipoRetorno> ::= (<Tipo>) | ()

<Dig> ::= 0 | 1 | 2 | ... | 9

<Letra> ::= A | B | ... | Z | a | b | ... | z

<LetraDig> ::= <Letra><LetraDig> | <Dig><LetraDig> | <Letra> | <Dig>

<ID> ::= <Letra><LetraDig> | <Letra>

<MLista> ::= [<ExpAritimetica>]<MLista>|[<ExpAritimetica>]

<IDM> ::= <ID><MLista>|<ID>

<DigLista> ::= <Dig><DigLista> | <Dig>

<Num> ::= <DigLista> | 0.<DigLista> | <DigLista>.<DigLista>

<Potencia> ::= <ChamadaFuncao> | <IDM> | <Num> | (<ExpAritimetica>)

<Fator> ::= <Fator>\*\*<Potencia> | <Potencia>

<Termo> ::= <Fator>\*<Termo> | <Fator>/<Termo> | <Fator>

<ExpAritimetica> ::= <Termo>+<ExpAritimetica> | <Termo>-< ExpAritimetica > | <Termo> | -<Termo>+< ExpAritimetica > | -<Termo>-< ExpAritimetica > | -<Termo>

<FatorBool> ::= <IDM> | !<ExpBool> | (<ExpBool>)

<TermoBool> ::= <FatorBool>&<TermoBool> | <FatorBool>

<ExpBool> ::= <TermoBool>“||”<ExpBool> | <TermoBool>

<Exp> ::= <ExpBool> | <ExpAritimetica>

<Atribuicao> ::= <IDM>:=<Exp>

<CondicaoSimples> ::= se(<ExpBool>){<ComandoLista>}

<Condicao> ::= <CondicaoSimples>senao <Condicao> | <CondicaoSimples>senao{<ComandoLista>} | <CondicaoSimples>

<Iteracao> ::= enquanto(<ExpBool>){<ComandoLista>} | para(<Atribuicao>;<ExpBool>;<Atribuicao>){<ComandoLista>}

<ParamLista> ::= <Exp>,<ParamLista> | <Exp>

<ChamadaFunc> ::= <ID>(<ParamLista>) | <ID>()

<Retorno> ::= retorna <Exp>

<Declaracao> ::= <Tipo> <IDM>

<Comando> ::= <Declaracao>; | <Atribuicao>; | <Condicao> | <Iteracao> | <ChamadaFunc>; | <Retorno>;

<ComandoLista> ::=<Comando><ComandoLista> | <Comando>

<ParamListaDef> ::= <Declaracao>,<ParamListaDef>|<Declaracao>

<Funcao> ::= func <ID> (<ParamListaDef>)<TipoRetorno>{<ComandoLista>} | func <ID> ()<TipoRetorno>{<ComandoLista>}

4. Descrição da linguagem em Wirth

Tipo = bool|caract|int|pflut.

Dig = “0” | “1” | “2” | ... | “9”.

Letra = “A” | “B” | ... | “Z” | “a” | “b” | ... | “z”.

ID = Letra{Letra|Dig}.

MLista = {“[“ExpAritimetica“]”}.

IDM = ID MLista.

Num = “.”Dig{Dig}|Dig{Dig}[“.”Dig{Dig}].

Potencia = ChamadaFuncao|IDM|Num|“(”ExpAritimetica“)”.

Fator = Potencia{“\*\*”Potência}.

Termo = Fator{(“\*”|“/”)Fator}.

ExpAritimetica = [“-”]Termo{(“+”|“-”)Termo}.

FatorBool = IDM|“!”ExpBool|“(”ExpBool“)”.

TermoBool = FatorBool{“&”FatorBool}.

ExpBool = TermoBool{“||”TermoBool}.

Exp = ExpBool|ExpAritimetica.

Atribuicao = IDM“:=”Exp.

CondicaoSimples = se“(”ExpBool“)”“{”ComandoLista“}”.

Condicao = CondicaoSimples“senao”Condicao|

­CondicaoSimples“senao”“{“ComandoLista“}”|CondicaoSimples

Iteracao = “enquanto”“(”ExpBool“)”{ComandoLista}| para“(”Atribuicao“;”ExpBool“;”Atribuicao“)”“{”ComandoLista“}”

ParamLista = Exp[“,”ParamLista]

ChamadaFunc = <ID>“(”[ParamLista]“)”

Retorno = retorna Exp

Declaracao = Tipo IDM

Comando = (Declaracao|Atribuicao|ChamadaFunc|Retorno)“;” | Condicao | Iteracao

ComandoLista = Comando{Comando}

Funcao = “func”ID“(”[Declaracao{“,”Declaracao}])([Tipo])“{”ComandoLista“}”