Universidade Federal de Alagoas - UFAL
Instituto de Computação - IC

Karla Elisabeth Cristo dos Santos
Roland dos Santos Gonçalves Sobrinho

EBNF Terminais

2 de Outubro de 2013

Universidade Federal de Alagoas - UFAL
Instituto de Computação - IC

Karla Elisabeth Cristo dos Santos
Roland dos Santos Gonçalves Sobrinho

EBNF Terminais

Trabalho Acadêmico solicitado pelo Prof. Alcino Dall Igna Júnior, com vistas à obtenção parcial de nota da disciplina Compiladores do Curso de Bacharel em Ciência da Computação — UFAL

2 de Outubro de 2013

Sumário

1 Nomes, Vinculações, Verificação de Tipos e Escopo

1.1 Formas de nome

Relativo as questões de projeto, temos:

- Tamanho do nome: ilimitado
- É permitida a colocação do caractere de conexão: "_"
- Não é *case sensitive*
- As palavras especiais são todas palavras reservadas. Listadas abaixo

Palavra Reservada	Descrição
int	Utilizada para declaração de inteiros
float	Utilizada para declaração de ponto flutuante
char	Utilizada para declaração de caracteres
if	Referente a instrução de seleção
else	Referente a instrução de seleção de duas vias
break	Desvio incondicional não rotulado
for	Instrução de repetição com contador
while	Instrução de repetição lógica
input	Função de entrada de dados
output	Função de saída de dados
const	Utilizada para declaração de constantes
function	Utilizada para declaração de funções
return	Retorna o processo realizado pela função
not	Negação logica
and	E lógico
or	OU lógico

Tabela 1: Palavras Reservadas

1.2 Variáveis

Nessa linguagem, a declaração é sempre explícita.
As variáveis são dinâmicas na pilha.
É permitida a conversão implícita e explícita de tipos. É permitida conversões de alargamento, permitindo expressões de modo misto.
A compatibilidade de tipos é dada pela compatibilidade de nome.

1.3 Escopo

O escopo é estático. Os blocos são delimitados por "{}"

1.4 Constantes nomeadas

As constantes são declaradas com a palavra reservada *const* seguida de seu tipo.

2 Tipos de Dados

Tipos númericos: **Inteiro e Ponto flutuante**, variáveis para esses tipos devem ser declaradas com as palavras reservadas *int* e *float*, respectivamente. Com os tipos numéricos é permitidas as operações aritméticas "*", "/", "+", -", bem como operações relacionais.

Variável do tipo **caractere** deve ser declarada pela palavra reservada *char*. O tipo *char* usa a codificação ASCII. Pode-se utilizar os operadores relacionais.

A **cadeia de caracteres** é definida como na linguagem C, utilizando um array unidimensional de caracteres. São permitidas as operações comparar, copiar, que utilizam as palavras reservadas *srtcmp* e *strcop*, respectivamente, e concatenar que é feita com o operador "+".
As cadeias de caracteres são finalizadas com o caracter especial nulo "\0"

Para copiar uma cadeia de caracteres para outra, utilizamos a operação strcop:

strcop(char1[], char2[])

o valor de char2 será copiado para char1
A função terá retorno 1 se a cópia for concluída com sucesso, caso contrário será retornado 0.

strcmp(char1[],char2[])

Ao passar cada parâmetro, será verificado se as cadeias de caracteres são iguais, o retorno será 0 se as cadeias de caracteres diferem, 1 se forem iguais.

Os **arranjos unidimensionais** devem ser declarados de acordo com a seguinte sentença: <tipo><identificador>[]
O tipo pode ser inteiro, ponto flutuante ou caracter. O arranjo tem sua posição inicial em 0.

3 Operadores

Operadores Ariméticos

Operação	Operador
Multiplicação	*
Divisão	/
Soma	+
Subtração	-

Tabela 2: Operadores Aritméticos

Operadores Relacionais

Operação	Operador
Igual	==
Diferente	!=
Maior que	>
Menor que	<
Maior que ou igual	>=
Menor que ou igual	<=

Tabela 3: Operadores Relacionais

Operadores lógicos

Operação	Operador
negação	not
conjunção	and
disjunção	or

Tabela 4: Operadores lógicos

Concatenação da cadeia de caracteres

Será utilizado o operador "+"

Ordem de precedência:

Precedência	Operador
Mais Alta	*, /, not
	+, - (unário)
	+, - (binário)
	==, !=, <, >, <=, >=
	and
	or, + (operador de concatenação)
Mais Baixa	= (operador de atribuição)

Tabela 5: Precedência

A associatividade de todos os operadores é da esquerda para direita, exceto os operadores relacionais que são não associativos.

4 Instruções

A estrutura condicional de uma via terá a seguinte forma

```
if(expressão booleana){
instrução
}
```

A estrutura condicional de duas vias terá a seguinte forma

```
if(expressão booleana){
instrução
}
else{
instrução
}
```

O aninhamento será feito de acordo com a ordem dos delimitadores ""A expressão será sempre booleana.

Estrutura iterativa com controle lógico
O controle será pré-teste e será feito por uma instrução separada

a estrutura terá a seguinte forma

```
while(expressão booleana){
instrução
}
```

Estrutura iterativa controlada por contador

O tipo da variável do laço será um inteiro positivo e a verificação da condição será feita uma vez para cada iteração

A estrutura tem a seguinte forma

```
for(expressao_1 ; expressão_2 ; expressão_3){
instrução
}
```

Onde expressao_1 determina a inicialização da variável do laço expressao_2 determina a condição de término, será avaliada uma vez para cada iteração. expressao_3 determina a forma da iteração da variável do laço

a variável do laço pode ser alterada no corpo do laço.

Entrada e saída São funções que através dos parâmetros determinam o tipo de dado da entrada ou saída. Será utilizada as palavras reservadas: *input* e *output*.

Sintaxe

Input("%tipo",identificador)

Output("%tipo",identificador)

Onde o tipo será representado da seguinte forma:

- d - representa um inteiro
- f - representa um ponto flutuante
- c - representa um caracter;

5 Atribuição

Será utilizado o operador: "="

A sintaxe geral é da forma:

<variavel_alvo><operador_de_atribuição><expressão>

6 Funções

Será utilizada a palavra reservada *function* para declaração de funções e a palavra reservada *return* para indicar o seu retorno A passagem dos parâmetros é feita por valor
Os tipos de parâmetros reais são verificados em relação aos tipos de parâmetros formais.
As variáveis locais são estaticamente alocadas.

Sintaxe

```
function identificador(paramentros)
{

instrução...

return identificador;

}
```

O retorno definirá o tipo da função. Se a palavra *return* for omitida a função não terá um retorno.

7 Algoritmos

Segue alguns exemplos de algoritmos na linguagem:
Alo Mundo!

```
main()
{
output("Alo Mundo!");
}
```

Serie de Fibonacci iterativa

```
function fibonacci(int n)
{
     int i = 0;
     int j = 1;
     int k;
     for(k = 1; k < n; k = k + 1){
             output("%d ",i);
             int t = i + j;
             i = j;
             j = t;
     }
     return j;
}
```

ShellSort

```
function shellsort(int vet[], int size) {
    int i , j , value;
    int gap = 1;
    while(gap < size) {
        gap = 3*gap+1;
    }
    while ( gap > 1) {
        gap = gap/3;
        for(i = gap; i < size; i = i + 1) {
            value = vet[i];
            j = i - gap;
            while (j >= 0 && value < vet[j]) {
                vet [j + gap] = vet[j];
                j = j - gap;
            }
            vet [j + gap] = value;
        }
    }
}
```