UNIVERSIDADE FEDERAL
DE ALAGOAS

Universidade Federal de Alagoas
Instituto da Computação
Curso de Ciência da Computação
Compiladores

Especificação da Linguagem

Aluno:
Romário Oliveira Pantaleão

Professor:
Alcino Dall'Igna Júnior

Julho
2018

Conteúdo

1 Introdução

1.1 Linguagem ROP

Linguagem ROP criada para a implementação dos analisadores léxico e sintático, com objetivo de ganhar conhecimento sobre compiladores.

1.2 Objetivo

O objetivo desse trabalho é definir as especificações da linguagem apresentada , introduzindo assim a linguagem que será usada nos próximos trabalhos da disciplina, tendo como um dos objetivos da linguagem a facilidade de leitura e escrita, facilitando a criação de programas.

2 Estrutura geral de um programa

2.1 Bloco de execução

Como na linguagem c++, nosso bloco de execução será composto por chaves {} , que delimitarão o inicio e o fim de um bloco de execução. A identação será opcional, os espaços em branco serão ignorados.

2.2 Declaração de variáveis

A declaração de variáveis pode ser feita em qualquer parte do programa, contanto que seja feita antes de que essa variável seja usada. Uma variável criada dentro de um bloco de execução só será acessada dentro do mesmo. Uma variável poderá ter no máximo 64 caracteres e não poderá começar com um digito ou seja [0-9]. Variáveis podem conter os caracteres alfanuméricos [0-9] , [a-z] , [A-Z].

2.3 Exemplo de estrutura

```
main[]{
  int message ;
    print [message] ;
}
```

3 Especificação de tipos

3.1 Inteiro

- Tipo primitivo
- Palavra reservada : int
- Operações

  1. Aritmeticas : adição, subtração, multiplicação, divisão, unários, incremento e decremento.
  2. Relacionais : maior , maior ou igual, menor, menor ou igual, igual , diferente.
  3. Concatenação com caracteres e cadeia de caracteres

3.2 Ponto Flutuante

- Tipo primitivo
- Palavra reservada: float
- Operações
    1. Aritméticas : adição, subtração, multiplicação, divisão, unários, incremento e decremento.
    2. Relacionais : maior, maior ou igual, menor, menor ou igual, igual , diferente.
    3. Concatenação com caracteres e cadeia de caracteres

3.3 Caractere

- Tipo primitivo (caracteres da tabela ascii)
- Palavra reservada: char
- Operações
    1. Relacionais: maior, maior ou igual, menor menor ou igual, igual, diferente.
    2. Concatenação com caracteres e cadeia de caracteres

3.4 Booleano

- Tipo primitivo
- Palavra reservada bool
- Operações:
    1. Lógica: negação, conjunção, disjunção insclusiva, disjunção exclusiva
    2. Relacionais: igual , diferente.
    3. Concatenação com caracteres e cadeia de caracteres

3.5 Cadeia de Caracteres

- Tipo não primitivo
- Palavra reservada string
- Operações:
    1. Relacionais: igual, diferente.
    2. Concatenação com caracteres e cadeia de caracteres

3.6 Arranjos Unidimensionais

- Tipo não primitivo
- Sintaxe "tipo"(quantidade)

4 Especificação de constantes literais

4.1 Inteiros

Valores inteiros de até 32 bits.
Permite coerção para float, caractere de acordo com a tabela ascii e string.

4.2 Float

Valores de ponto flutuante de até 32 bits ou maiores com formato de notação ciêntífica.
Permite coerção para inteiro e string

4.3 Booleano

Palavras reservadas true ou false.
Permite coerção para inteiro e string.

4.4 Cadeia de Caracteres

Sequencia de caracteres entre aspas duplas

4.5 Caracteres

Qualquer caractere da tabela ascii.
Permite coerção para Inteiro de acordo com seu valor na tabela ascii.

5 Conjunto de Operadores

5.1 Aritméticos

```
Aditivo : +
Subtrativo e unario negativo : -
Multiplicativo : *
Divisor : /
Incremento : ++
Decremento : --
```

5.2 Relacionais

```
Numericos , caracteres e cadeia de caracteres : = , >= , < , <= , !=
Booleanos : = , !=
```

5.3 Lógicos

```
Booleanos : &, !, ^, |
Negacao : !
Conjuncao : &&
Disjuncao : ||
```

5.4 Concatenação

```
Para todos os tipos que suportam : +=
```

6 Instruções

6.1 Condicional de uma via

1. Palavra reservada if

2. if[ExprLog] {bloco de execução}

6.2 Condicional de duas vias

1. Palavras reservadas if e else/ simbolos '?' e ':'
2. if[ExprLog] {bloco de execução } else { bloco de execução}
3. .[ExprLog] ? {bloco para verdadeiro} : {bloco para falso}

6.3 Estrutura iterativa com controle lógico

1. Palavra reservada while
2. while[ExprLog] {bloco de execução}

6.4 Estrutura iterativa controlada por contador

1. Palavra reservada for
2. for[varInt ; expArit; expArit; expArit] bloco de execução
3. Onde a primeira expArit é o inicio e a segunda é o fim
4. A terceira expArit é o tamanho do passo, caso omitido tem o padrão 1

6.5 Entrada e saída

1. Entrada(Palavra reservada scan): scan[lista de variaveis separadas por virgula];
2. Saida(Palavra reservada print): print[lista de variáveis separada por virgula];
   caso venha 1 número após a virgula, o valor da variável anterior será formatado com a quantidade do número ex: seja a um inteiro = 1 e b um float = 1 print[a , 2, b , 3] Saída : 01 e 1.000
   Não são permitidas constantes nos prints, só variáveis e strings.
   Caso o número seja muito grande será usada a notação ciêntifica na saída.

6.6 Atribuição

1. Operador =
   "tipo da variável"= "valor";

7 Funções

1. A função main será a primeira função chamada ao iniciar o programa
2. Inicia com a palavra reservada do tipo que será retornado, ou a palavra reservada void(caso não retorne nada)
3. Após o retorno ser definido o nome será definido da mesma forma que o nome de uma variável
4. seguido de um bloco de parâmetros [tipo nome, tipo nome ...]
5. seguido de um bloco de execução {}
6. No final a palavra reservada return, seguida da variável a ser retornada e ponto e virgula.

7. Ex:

```
void funcExemplo[int parametro1, char parametro2, float parametro3] {
   // executa bloco
}
```

8 Programa Alô Mundo

```
void main[] {
   string message = "alo mundo" ;
     print(message) ;
 }
```

9 Programa Fibonacci

```
void main[] {
   int n ;
     scan[n] ;
     int left = 0 ;
     int right = 1 ;
     if[n > 0]{
      print[left] ;
     }
     if[n > 1]{
      print[right] ;
     }
     while[n > 2] {

      n-- ;
     }
 }
```

10 Programa ShellSort

```
int vetor(100);
int n;

void shellSort[]{
   int h = 1 ;
    while[h < n]{
      h = h * 3 + 1 ;
    }
    h = h/3 ;
    int temp, j ;
    while[h > 0]{
      for[i = h; i < n; i++]{
         temp = a(i) ;
            j = i ;
            while[j >= h && a(j-h) > temp]{
               a(j) = a(j-h) ;
                j = j - h ;
            }
            a(j) = temp ;
      }
      h = h / 2 ;
```

```
    }

}

void main[]{

    int i ;
    scan[n] ;
    for[i ; 0 ; n]{
      scan[vetor(i)] ;
    }
    shellSort[];
    for[i = 0; i < n; i++]{
      print[vetor(i)] ;
        print[" "] ;
    }
}
```