Sumário

Estrutura geral de um programa

A linguagem admite escopo global e as variáveis são declaradas ao especificar o tipo e o nome em qualquer parte do programa. Funções são declaradas especificando o tipo de retorno e a lista de parâmetros.

A execução é iniciada a partir da função `init()` e as funções são definidas em qualquer parte especificando o tipo de retorno, o nome e a lista de parâmetros.

Nomes

Os nomes são sensíveis à caixa e tem tamanho máximo de 100 caracteres alfanuméricos iniciando com uma letra.

Os nome são reconhecidos pela expressão regular `[a-zA-Z][a-zA-Z0-9]`, são permitidos caracteres alfanuméricos exceto o primeiro que deve ser uma letra.

Tipos e estruturas de dados

Forma de declaração

Os tipos de dados predefinidos e suas operações suportadas são:

Tipo	**Declaração**	**Operações**
Inteiro	`int nome;`	Aritméticas e relacionais
Ponto flutuante	`float nome;`	Aritméticas e relacionais
Caractere	`char nome;`	Relacionais
Booleano	`bool nome;`	Relacionais e lógicas
Cadeias de caracteres	`string nome;`	Relacionais e concatenação
Arranjos	`tipo nome[tamanho];`	Concatenação

Operadores

Aritméticos	*+, -, *, /, % (módulo), - (unário)*
Relacionais	*==, !=, <=, >=, <, >*
Lógicos	*!, and, or*
Concatenação	*++*

Constantes literais

```
int         [0-9]+
float       [0-9]+.[0-9]+
char        '[a-zA-Z]'
bool        true|false
string      "[a-zA-Z]*"
Arranjos    { el1, el2,...}
```

Declarações

Os tipos inteiro, ponto flutuante, caractere, booleano e string são declarados seguindo o formato *tipo nome;*

As constantes nomeadas são precedidas pela palavra reservada *const* e em seguida deve ser declarada a variável.

Arranjos

Os arranjos são declarados na forma *tipo nome[tamanho];*

Os elementos dos arranjos são referenciados por índices listados entre colchetes e são armazenados em posições contíguas na memória.

Equivalência de tipos e coerção

A equivalência de tipos é por nome e é permitido conversões de tipo explícitas *(cast)*. A coerção admitida é de *int* para *float.*

Atribuição e expressões

O operador de atribuição é o símbolo `'='` onde a expressão do lado direito é atribuída à variável alvo do lado esquerdo.Para as cadeias de caracteres deve ser utilizadas aspas (`"`) para atribuição:

```
string nome = "valor";
```

Os operadores `and` e `or` são avaliados em curto-circuito e os tipos das operações são definidos de acordo com a variável alvo.

Precedência

Precedência da mais alta para a mais baixa

```
- unário
*, /, %
+ e - binários
<, >, <=, >=
==, !=
and, or
= atribuição
```

Associatividade

```
Esquerda:  *, /, %, + e - binários
           <, >, <=, >=, ==, !=
           and, or
Direita: - unário, = atribuição
```

Sintaxe e exemplo de estruturas de controle

Todos os blocos devem possuir chaves (`{}`)

Comando de seleção

O comando de seleção é o `if` que seleciona um caminho baseado em cada condição. Para mais de uma condição é usado o `else if` e o bloco opcional `else` que é selecionado caso nenhuma das condições sejam atendidas.

```
if(condição) {
    ...
} else if(condição) {
    ...
} else {
```

```
    ...
}
```

Comandos de iteração

Controle lógico

Enquanto a condição não for verdadeira o bloco de código é executado.

```
while (condição){
    ...
}
```

Controle por contador

A inicialização de variáveis é feita em *var_int* e a *condição* é testada antes de cada iteração, o *incremento* é feito no final da iteração.

```
for (var_int; condição; incremento){
    ...
}
```

Desvio incondicional

break
Permite encerrar um loop

```
while (condição){
    ...
    break;
    ...
}
```

Instruções de entrada e saída

A instrução de leitura da entrada padrão é realizada especificando as variáveis em que serão atribuídos os valores lidos.

```
read  var1, var2,...;
```

Na instrução de saída padrão deve ser especificado o texto formatado e as variáveis ou constantes que vão ser substituídas

```
print "%2d %.2f", var1, cons1;
```

Subprogramas

As funções são declaradas no corpo do programa e seguem a forma:

```
tipo nome(parâmetros){
      ...
}
```

Os procedimentos são declarados como funções. A palavra reservada `proc` é usada para declarar os procedimentos.

```
proc nome(parâmetros){
      ...
}
```

Métodos de passagem de parâmetros

As funções implementam o modo de entrada e saída, os parâmetros são passados por valor-resultado.

Funções não podem ser passadas como parâmetro.

Exemplos

Alô mundo

```
int init(){
      print “Alô mundo”;
      return 0;
}
```

Série de Fibonacci

```
int init(){
      int a, b, i, aux, limite;
      a = 0;
      b = 1;

      print "Digite um número limite: ";
      read "%d", &limite;

      i = 0;
```

```
        while(i < limite){
            aux = a + b;
            a = b;
            b = aux;
            print "%d, ", auxiliar;
            i = i + 1;
        }

        return 0;
    }
```

Shell sort

```
    int init(){

        return 0;
    }
```

Enumeração com as categorias dos tokens

```
enum class Category {
    Init=1, Integer, Float, Char, Boolean,
    PtVg, Pt2, Vg, AbPar, FePar, AbCol, FeCol, AbChav, FeChav,
    Procedure, Return, Break,
    Read, Write,
    For, While, If, ElseIf, Else,
    OpEq, OpMaior, OpMenor, OpMaiorEq, OpMenorEq, OpDifer,
    OpMais, OpMenos, OpMult, OpDiv, OpMod,
    OpAtr,
    Id, CteInt, CteFloat, CteChar, CteBool, CteStr, Eof
};
```