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.

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

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

Expressão regular para os nomes: `[a-zA-Z][a-zA-Z0-9]`

Tipos e estruturas de dados

Os tipos de dados predefinidos e suas operações suportadas são:

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

Os tipos são declarados seguindo o formato `tipo nome;`

Para as cadeias de caracteres a declaração é feita usando a palavra reservada `string`. As strings suportam a operação de concatenação (`++`) e deve ser utilizadas aspas (`"`) para atribuir uma string

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

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

Os elementos dos arranjos são referenciados por índices listados entre colchetes e são armazenados

Forma de armazenamento em 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`.

As constantes nomeadas são precedidas pela palavra reservada `const`

Atribuição e expressões

O operador de atribuição é o símbolo `'='` onde a expressão do lado direito é atribuída à variável alvo do lado esquerdo.

Precedência da mais alta para a mais baixa

```
- unário
*, /, %
+ e - binários
<, >, <=, >=
==, !=
&&, ||
= atribuição
```

Associatividade

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

Os operadores E e OU (`&&` e `||`) são avaliados em curto-circuito.
Os tipos das operações são definidos de acordo com a variável alvo.

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){
      ...
}
```

Instrução de desvio incondicional

```
goto Rótulo
```

Transfere a execução para a posição identificada pelo *Rótulo*

Subprogramas

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

```
<tipo> nome(parametros){
      instruções
}
```

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.