

## 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	<code>int</code>	Aritméticas, lógicas e relacionais
Ponto flutuante	<code>float</code>	Aritméticas, lógicas e relacionais
Caractere	<code>char</code>	Relacionais
Booleano	<code>bool</code>	Relacionais e lógicas
Cadeias de caracteres	<code>string</code>	Concatenação e relacionais
Arranjos	<code>tipo nome[tamanho]</code>	Concatenação

### Constantes literais

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

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
```

Tipo das operações são definidos por expressão ou operador a operador

Os operadores E e OU (&& e ||) são avaliados em curto-circuito

### Sintaxe e exemplo de estruturas de controle

Todos os blocos devem possuir chaves ({})

### Comando de seleção

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

## **Comandos de iteração**

### **Controle lógico**

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

### **Controle por contador**

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

## **Instrução de desvio incondicional**

```
goto 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.