MINI LANG - DOCUMENTAÇÃO DO USUÁRIO

github do projeto com melhor visualização da documentação e do que foi feito:

https://github.com/Lucas-magalhaes1/MiniLangProject

MiniLang é uma linguagem de programação didática, inspirada em linguagens como C e Java, com foco em simplicidade e clareza. Ela permite criar programas com estruturas comuns como variáveis, condicionais, repetições, entrada/saída, funções, entre outros.

---
1. Comentários

Comentários de linha são definidos com `//`.

Exemplo:
```
// Este é um comentário
```

---
2. Tipos de dados

A linguagem suporta os seguintes tipos:

- int   → números inteiros
- string → textos
- boolean → valores lógicos (true ou false)

Exemplo:
```
int idade = 20;
string nome = "Lucas";
boolean ativo = true;
```

Também é possível declarar constantes com `const`:

```
const int LIMITE = 10;
```

---
3. Identificadores

Nomes de variáveis, funções e parâmetros devem:

- Começar com letra
- Podem conter letras, números e `_` (underline)

Exemplo:
```
int contador_1;
```

---
4. Declaração e atribuição de variáveis

Declaração:
```
  int idade;
  string nome;
```

Com atribuição:
```
  int idade = 18;
  idade = idade + 1;
  nome = "Ana";
```

Também é possível usar `++` e `--`:
```
  idade++;
```

---
5. Entrada de dados

Comandos:
- `getInt(nomeDaVariavel);`
- `getText(nomeDaVariavel);`

Exemplo:
```
  int idade;
  getInt(idade);

  string nomes;
  getText(nomes);
```

---
6. Saída de dados

Use `write(...)` para imprimir dados.

Exemplo:
```
  write("Olá, mundo");
  write("Idade: " + idade);
```

---
7. Comandos de decisão (if / else)

Sintaxe:
```
  if (<condição>) {
    // comandos
  } else if (<condição>) {
    // comandos
  } else {
```

```
  // comandos
}
```

Operadores lógicos disponíveis:
```
==, !=, <, >, <=, >=, &&, ||, !
```

Exemplo:
```
if (idade >= 18) {
  write("Adulto");
} else {
  write("Menor de idade");
}
```

---
8. Estruturas de repetição

→ FOR:
```
for (int i = 0; i < 5; i++) {
  write("i = " + i);
}
```

→ WHILE:
```
int i = 0;
while (i < 3) {
  write(i);
  i++;
}
```

→ FOREACH:
```
string nomes;
getText(nomes);

foreach (string nome in nomes) {
  write("Olá, " + nome);
}
```

→ CONTROLE:
- `break;` → sai do laço
- `continue;` → pula para a próxima iteração

---
9. Funções e procedimentos

As funções podem retornar `int`, `string`, `boolean`, ou `void`.

Exemplo com retorno:
```
int somar(int a, int b) {
```

```
  return a + b;
}
```

Exemplo `void`:
```
void saudacao(string nome) {
  write("Olá, " + nome);
}
```

Chamando função:
```
int resultado;
resultado = somar(2, 3);
```

---
10. Estrutura principal (main)

Todo programa deve ter uma função principal:

Exemplo:
```
int main() {
  write("Programa iniciado");
  return 0;
}
```

---
11. Inclusão de bibliotecas

Use `#include <nome>` para incluir bibliotecas (fictícias ou decorativas).

Exemplo:
```
#include <utils>
#include <io>
```

---
12. Operadores e separadores

Operadores aritméticos:
```
+, -, *, /
```

Operadores lógicos:
```
==, !=, <, >, <=, >=, &&, ||, !
```

Separadores:
```
;   → fim de comandos
,   → separação de parâmetros
()  → argumentos e expressões
{}  → blocos de código
[]  → reservado para arrays futuros
```

---
13. Exemplo completo

```
#include <utils>
#include <io>

int main() {
  string nomes;
  getText(nomes);

  foreach (string nome in nomes) {
    if (nome == "admin") {
      write("Bem-vindo, administrador");
    } else {
      saudacao(nome);
    }
  }

  return 0;
}

void saudacao(string nome) {
  write("Olá, " + nome);
}
```

Fim da documentação.