



Universidade Federal de Itajubá

Compiladores

Linguagem Val

Grupo

André Arantes

Lívia Granato

Victor Pereira





Universidade Federal de Itajubá

Sobre a Linguagem





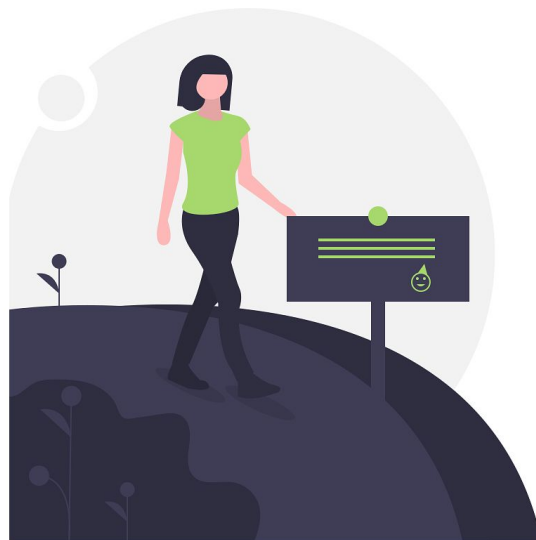
Sobre

Criada com o intuito de facilitar a programação:

- ✓ Estrutura baseada em Tags
- ✓ Palavras reservadas de fácil entendimento e memorização.

Código de uso geral

Se assemelha ao **C** (no sentido de ser uma linguagem estruturada, procedural e de propósito geral) e **Python** (não há necessidade de “;” ao final de cada linha de comando e declaração de variáveis).





Documentação



1. Expressões Regulares

Expressão regular provê uma forma concisa e flexível de identificar cadeias de caracteres particulares, palavras ou padrões, criando uma forma de ser interpretada por um processador.

início	<MAIN></MAIN>
função	<FUNCTION></FUNCTION>
quebra de linha	\n
real	("-")? {dígito}+("." {dígito})*
string	"({letra}+{dígito})*"
letra	{a-zA-Z}
dígito	{0-9}
booleano	{falso} {vdd}
entrada	>> variável
saída	<<
atribuição	()
condicional	<SE></SE>
desvio condicional	<SENAO></SENAO>
desvio condicional encadeado	<ESE></ESE>
repetição	"<RT></RT>" "<ENQ></ENQ>"
lógico	("E" "OU" "I")
variável / identificador	{letra}+({letra} {dígito})*



Soma	" + "
Subtração	" - "
Multiplicação	" * "
Divisão	" / "
Exponencial	" ^ "
Resto da divisão (mod)	" % "
Concatenação (string - string ou string-variável)	" + "

2. Operações Aritméticas

São os símbolos definidos para a realização de **operações matemáticas** em um programa como soma, subtração e outros.



Igual	=
Menor	<
Menor-Igual	<=
Maior	>
Maior-Igual	>=
Diferente	?

3.

Operações Relacionais

São os símbolos definidos para a realização de **operações relacionais** em um programa como igual, menor, maior, etc.



4. Símbolos Especiais

São símbolos que podem ser utilizados na linguagem para a definição de comentários no código, abertura e fechamento de funções, etc.


separador	"." (casas decimais) ".,", (estrutura repetição) ".,", (em vetores)
abre e fecha parênteses	"(" ")" (indica prioridade nas operações aritméticas)
comentário mesma linha	"##"
comentário várias linhas	"# {texto} #"
abertura função (tag)	<
fechamento função (tag)	>
finalização função (tag)	<\
abre e fecha aspas (escrever string)	" "
abre e fecha colchetes	[] (delimita condição e estrutura de repetição)



5.

Blocos de Comando

Relacionam-se quanto às tags que indicam o **início** e **fim** de um programa na linguagem de programação.

início	<code><main></code> (trocar "main" por qualquer outra função)
fim	<code></main></code> (trocar "main" por qualquer outra função) 



Implementação



Ferramentas utilizadas:

- Linguagem: Java 8
- Bibliotecas: jflex-1.6.1 e java-cup-11a
- IDE: Netbeans 8.2
- Analisador Léxico: JFlex
- Analisador Sintático: Java CUP





Etapas:

1. Definir os Tokens em um arquivo .flex.
2. Gerar as classes Java necessárias para a análise Léxica.
3. Definir as árvores sintáticas em um arquivo .cup.
4. Gerar as classes Java necessárias para a análise Sintática.
5. Carregar um programa.
6. Realizar a análise léxica e sintática.





Dificuldades





Dificuldades de Implementação:

- Entender como as bibliotecas funcionam.
- Gerar as árvores sintáticas.
- Tratamento e identificação de erros.





Exemplos de Código



Exemplos

Função Soma e Interação com usuário

1

Programa simples com entrada e saída de dados com interação do usuário.

Entrada e Saída

1.

```
<MAIN>  
  << "Por favor, insira o primeiro valor:"  
  >> v1  
  << "Por favor, insira o segundo valor:"  
  >> v2  
  << "A soma e:" + (v1 + v2)  
</MAIN>
```




Exemplos

1

Funções, Condicionais e Laços de Repetição

Programa de complexidade maior utilizando a declaração de funções, condicionais e laços de repetição para verificar os números de um

2.

Estrutura RT

```
<FUNCTION> vasculhaVector(vetor)
  par(0)
  impar(0)
  <RT> [i(0); i < 10; i++]
    <SE> [ i % 2 = 0]
      par++
    </SE>
    <SENAO>
      impar++
    </SENAO>
  </RT>

  <SE> [par > impar]
    RETORNA "0 seu vetor tem mais numeros pares"
  </SE>

  <ESE> [par = impar]
    RETORNA "0 seu vetor tem a mesma quantidade de numeros pares e impares"
  </ESE>

  <SENAO>
    RETORNA "0 seu vetor tem mais numeros impares"
  </SENAO>
</FUNCTION>

<MAIN>
  vetor(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
  << vasculhaVector(vetor)
</MAIN>
```



Exemplos

1

Funções, Condicionais e Laços de Repetição

Programa de complexidade maior utilizando a declaração de funções, condicionais e laços de repetição para verificar os números de um vetor.

Estrutura ENQ

3.

```
<FUNCTION> vasculhaVetor(vetor)
  par(0)
  impar(0)
  i(0)
  <ENQ> [i < 10]
    <SE> [ i % 2 = 0 ]
      par++
    </SE>
    <SENAO>
      impar++
    </SENAO>
    i++
  </ENQ>

  <SE> [par > impar]
    RETORNA "0 seu vetor tem mais numeros pares"
  </SE>
  <ESE> [par = impar]
    RETORNA "0 seu vetor o mesmo numero de numeros pares e impares"
  </ESE>
  <SENAO>
    RETORNA "0 seu vetor tem mais numeros impares"
  </SENAO>
</FUNCTION>

<MAIN>
  vetor (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
  << vasculhaVetor(vetor) ##printa o retorno
</MAIN>
```



Universidade Federal de Itajubá



Vídeo



Obrigado!

Dúvidas, perguntas
e sugestões?

