

1. REQUISITOS E EXECUÇÃO DO PROJETO

2. SOBRE A LINGUAGEM CRIADA

A linguagem criada segue uma sintaxe baseada no C++ e contém:

- Comentários
- Declarações de variáveis
- Atribuições de variáveis
- Comando de impressão e leitura
- Condições
- Repetições

Os arquivos para serem executados devem possuir a extensão “.medjed”.

A análise léxica e sintática é feita pelo ANTLR, porém o projeto possui um arquivo chamado Lexico.java e Sintatico.java que reescreve o código do ANTLR, para permitir que apareça uma mensagem de erro léxico ou sintático em português (somente a mensagem de que ocorreu um erro irá aparecer em português, o erro em si não está traduzido).

Além disso contem um arquivo Semantico.java, onde é feita toda a análise semântica.

Após a análise semântica o Interpretador.java executa os comandos necessários. A única exceção de comando que o Interpretador não executa são as repetições **WHILE** e **FOR**, isso ocorre pois não foi possível armazenar quais comandos deveriam ser repetidos corretamente. Porém mesmo que esses dois comandos não sejam interpretados, a análise semântica ainda ocorre.

Se caso possuir algum erro, o compilador irá dizer quais erros ocorreram, caso contrário será exibido que não ocorreu nenhum erro e os comandos serão executados corretamente.

Exemplo de código compilado na linguagem com erros léxicos, sintáticos e semânticos:

```
BeginPlay()
{
    @
    int X = 50.1;

    cout << Y;

    cout << "Teste"
}
```

Usando arquivo padrão: testes/Entrada.medjed

Foram encontrados erros léxicos na execução do programa:

Erro na linha 3, coluna 4: token recognition error at: '@'

Foram encontrados erros sintáticos na execução do programa:

Erro na linha 9, coluna 0: extraneous input '}' expecting {';', '<','>'}

Foram encontrados erros semânticos na execução do programa:

Erro na linha 4, coluna 8: a variável 'X' foi declarada com o valor incorreto!

Erro na linha 6, coluna 12: a variável 'Y' não foi declarada!

Process finished with exit code 0

Exemplo de código compilado na linguagem que não possui nenhum tipo de erro:

```
BeginPlay()
{
    int X, VariavelInt, A;
    int Y = 50;
    float Z = 50.0, VariavelFloat;
    string MinhaString = "Esse é um exemplo de STRING";
    bool VariavelBool = true;

    VariavelInt = 50, A = 20;
    cout << VariavelInt << " - " << A << " = " << VariavelInt - A;

    cout << "\n\nO valor de: X + Y é: " << X + Y << "\n"; // Isso irá
    gerar um valor inesperado, pois a variável X não foi inicializada
    cout << "O valor de Y + Z é: " << Y + Z << "\n"; // Isso irá mostrar
    o valor '100' e não '100.0', pois como as casas decimais do resultado
    são '0' ele irá converter para INTEIRO
    cout << "\n" << MinhaString;
    cout << "\nE esse um exemplo de booleano: " << VariavelBool;

    cout << "\n\nDigite um valor inteiro: ";
    cin >> X;

    if(X > 50)
    {
        cout << "O valor digitado '" << X << "' é maior que 50";
    }
    else
    {
        cout << "O valor digitado '" << X << "' é menor que 50";
    }

    cout << "\n\n50 + 90 * 40 / 2 = " << 50 + 90 * 40 / 2; // 0
    resultado deve ser 1850
    cout << "\n(50 + 90) * 40 / 2 = " << (50 + 90) * 40 / 2; // 0
    resultado deve ser 2800
    cout << "\n(50 + (90 * 40)) / 2 = " << (50 + (90 * 40)) / 2; // 0
    resultado deve ser 1825
}
```

```

Usando arquivo padrão: testes/Entrada.medjed

Nenhum erro léxico encontrado!
Nenhum erro sintático encontrado!
Nenhum erro semântico encontrado!

50 - 20 = 30

O valor de: X + Y é: 16307
O valor de Y + Z é: 100

Esse é um exemplo de STRING
E esse um exemplo de booleano: true

Digite um valor inteiro: 40
O valor digitado '40' é maior que 50

50 + 90 * 40 / 2 = 1850
(50 + 90) * 40 / 2 = 2800
(50 + (90 * 40)) / 2 = 1825

(programa BeginPlay() { (bloco (declaracao (tipo_variavel int) (variavel X) , (variavel VariavelInt) ,
Process finished with exit code 0

```

Devido a sintaxe e aos comportamentos serem semelhantes ao do C++, o código dentro de “BeginPlay()” pode ser facilmente adaptado e executado em um compilador de C++ e exibir um resultado semelhante ao de cima.

<pre> main.cpp 1 #include <iostream> 2 using namespace std; 3 4 int main() 5 { 6 int X, VariavelInt, A; 7 int Y = 50; 8 float Z = 50.0, VariavelFloat; 9 string MinhaString = "Esse é um exemplo de STRING"; 10 bool VariavelBool = true; 11 12 VariavelInt = 50, A = 20; 13 cout << VariavelInt << " - " << A << " = " << VariavelInt - A; 14 15 cout << "\n\nO valor de: X + Y é: " << X + Y << "\n"; // Isso irá gerar um 16 cout << "O valor de Y + Z é: " << Y + Z << "\n"; // Isso irá mostrar o val 17 cout << "\n" << MinhaString; 18 cout << "\nE esse um exemplo de booleano: " << VariavelBool; 19 20 cout << "\n\nDigite um valor inteiro: "; 21 cin >> X; 22 23 if(X > 50) 24 { 25 cout << "O valor digitado '" << X << "' é maior que 50"; 26 } 27 else 28 { 29 cout << "O valor digitado '" << X << "' é menor que 50"; 30 } 31 32 cout << "\n\n50 + 90 * 40 / 2 = " << 50 + 90 * 40 / 2; // O resultado dev 33 cout << "\n(50 + 90) * 40 / 2 = " << (50 + 90) * 40 / 2; // O resultado de 34 cout << "\n(50 + (90 * 40)) / 2 = " << (50 + (90 * 40)) / 2; // O resultad 35 36 return 0; 37 } </pre>	<pre> 50 - 20 = 30 O valor de: X + Y é: 900140612 O valor de Y + Z é: 100 Esse é um exemplo de STRING E esse um exemplo de booleano: 1 Digite um valor inteiro: 40 O valor digitado '40' é menor que 50 50 + 90 * 40 / 2 = 1850 (50 + 90) * 40 / 2 = 2800 (50 + (90 * 40)) / 2 = 1825 ...Program finished with exit code 0 Press ENTER to exit console. </pre>
---	---

Compilado com: https://www.onlinegdb.com/online_c++_compiler

3. ESTRUTURA BÁSICA DO PROGRAMA.

O programa deve obrigatoriamente começar com “BeginPlay()”, seguido da abertura de chaves que deve ser fechada no final do programa. Dentro das chaves é possível declarar variáveis, atribuições, condições, repetições e comandos de impressão e leitura.

Exemplo:

```
BeginPlay()
{
    // Declarações devem ser incluídas aqui
}
```

3.1 COMENTÁRIOS

Os comentários podem ser feitos utilizando “//”. Não é possível fazer comentários de múltiplas linhas, somente de linha única.

Exemplo:

```
// Exemplo de comentário  
// Esse é outro comentário
```

3.2 DECLARAÇÕES DE VARIÁVEIS

As variáveis podem ser declaradas em qualquer parte do código, porém elas possuem escopo, por tanto não é possível declarar a mesma variável dentro do mesmo escopo.

A declaração pode ser feita sem inicializar o valor (mesmo que não seja inicializado, a variável ainda terá um valor padrão) ou declaração com inicialização direta. Além disso é permitido declarar mais de uma variável por linha, desde que elas sejam do mesmo tipo.

Exemplo:

```
tipo nome_da_variavel;  
tipo nome_da_variavel1, nome_da_variavel2, nome_da_variavel3;  
  
tipo nome_da_variavel = valor;  
tipo nome_da_variavel1 = 50, nome_da_variavel2 = 40;
```

O nome da variável deve começar obrigatoriamente com uma letra (maiúscula ou minúscula), podendo então ser seguido de qualquer letra ou número. Os tipos e valores aceitos são:

- **INT** - números inteiros positivos ou negativos ('-'?[0-9]+).
- **FLOAT** - números decimais positivos ou negativos, deve ser utilizado ponto em vez de vírgula ('-'?[0-9]+'.[0-9]+).
- **BOOL** – valor booleano verdadeiro ou falso ('true' | 'false'). Diferente de C++ e outras linguagens, o valor 0 e 1 não é aceito.
- **STRING** – qualquer tipo de texto, desde que esteja entre aspas. A quebra de linha “\n” também é aceita (" (~[\"\\r]|\n')* ").

Exemplo:

```
int X, Y = 50, Z = 20;  
  
float A, B, C = 50.1 + 10;  
  
string Texto = "Texto";  
  
bool booleano = true, booleano2 = false;
```


3.3 ATRIBUIÇÕES DE VARIÁVEIS

As variáveis também podem ter seu valor atribuído após a declaração, para isso é necessário informar o nome da variável, seguido do operador de atribuição “=” e do valor. É possível fazer varias atribuições na mesma linha e e as variáveis que estão tendo o valor atribuído não precisam ser do mesmo tipo.

Exemplo:

```
int X, Y;  
float Z;  
  
X = 50, Y = 20, Z = 50.1;
```

3.4 IMPRESSÃO DE VALORES COM COUT

Para imprimir se utiliza “cout” no início, seguido do operador “<<” e do valor que irá ser impresso. É possível imprimir mais de um valor por linha e o ponto e vírgula é necessário no final.

Exemplo:

```
cout << "Exemplo do uso de cout\n";

cout << "É possível imprimir textos, números e expressões aritméticas: " << 50 + 40 << "\n";

cout << "A impressão de variáveis também é aceita: " << X << " + " << Y << " = " << X + Y << "\n";
```

OBS: se tentar realizar a impressão de uma variável/valor INT somado com uma variável/valor FLOAT, caso as casas decimais sejam 0, o programa irá exibir o número como sendo inteiro, caso contrário irá exibir como um número decimal, assim exibindo as casas decimais. Isso é semelhante a C++.

```
A soma de 50 + 50.0 irá exibir: 100
A soma de 50 + 50.1 irá exibir: 100.1
```

3.5 LEITURA DE VALORES COM CIN.

A leitura de valores é feita utilizando “cin” no início, seguido do operador “>>” e da variável que irá receber o valor. É possível passar mais de uma variável que irá receber determinado valor e o usuário terá que digitar todos eles.

Exemplo:

```
cout << "\n\nDigite um valor inteiro para X e depois para Y: ";  
cin >> X >> Y;
```

Caso o valor que seja digitado não seja compatível com o valor da variável, o programa será encerrado automaticamente exibindo erro sem executar os comandos seguintes, diferente de C++ que aceita a leitura de um valor mesmo que seja o tipo incorreto da variável, fazendo com que o valor seja convertido para algum valor padrão.

```
Digite um valor inteiro: aaa  
  
O programa foi encerrado pois o valor digitado é inválido para a variável 'X' de tipo (INT)!
```

3.6 CONDIÇÕES COM IF E ELSE

As condições podem ser feitas utilizando **IF**, seguido da abertura de parênteses, que deve conter dentro uma expressão booleana (**valor/variável, operador de comparação, valor/variável**), seguindo então de abre chaves, podendo conter declarações, atribuições e outras coisas dentro, sendo necessário fechar as chaves no final do bloco.

O **ELSE** não é obrigatório, mas também pode ser utilizado e será executado caso a condição do IF seja falsa.

ELSE IF(condição) não é aceito pelo programa.

Os operadores de comparação aceitos são:

- == (igual)
- != (diferente)
- > (maior)
- >= (maior ou igual)
- < (menor)
- <= (menor ou igual)

O tipo **INT** e **FLOAT** podem utilizar qualquer um dos operadores, enquanto **BOOL** e **STRING** só podem utilizar o operador '==' ou '!=

Exemplo:

```
int X = 5;

if(10 / 2 * 3 > X)
{
    cout << "É verdadeiro\n";
}
else
{
    cout << "É falso\n";
}

if("Texto" == "Texto")
{
    cout << "Texto é igual a Texto";
}
```


3.7 REPETIÇÃO WHILE

3.8 REPETIÇÃO FOR