Relatório Trabalho de Compiladores
Nome dos alunos:

Pedro Henrique Cabral Moreira
Paulo Eduardo Pereira Carvalho
Menção honrosa: Camily Gonçalves de Bem

Link github: https://github.com/PauloEduCarvalho/trabalhoCompiladores

Visão Geral da Linguagem

Nome da Linguagem: CPP - (Camily Pedro Paulo)

Descrição Geral:

A linguagem CPP foi criada com inspiração em 3 grandes estudantes de computação: Camily, Pedro e Paulo. Conta com uma sintaxe mais simples e possui as palavras reservadas em português.

A linguagem possui os seguintes elementos:

- Tipos de dados básicos: `int`, `real`, `bool`, `letra`, `palavra`.
- Comandos básicos: Atribuição("="), leitura de entradas("entrada"), e escrita de saídas("saída").
- Operações aritméticas: Soma("+"), subtração("-"), multiplicação("*") e divisão("/").
- Operações relacionais: Comparações de igualdade("=="), diferença("!="), maior(">") e menor("<").
- Estruturas de controle: Condicional ("`se`", "`senão se`" e "`senão`") e laços de repetição ("`enquanto`").
- Suporte a funções com passagem de parâmetros e retorno ou sem retorno, com a palavra reservada "retornar".

Características principais:

- É uma linguagem fortemente tipada, exigindo que as variáveis sejam declaradas com tipos explícitos.
- Suporta controle de fluxo através de condicionais e repetições.
- Tem suporte a funções com recursividade, como no exemplo clássico de fatorial.
- Sua sintaxe foi criada para ser simples, buscando reduzir a quantidade de operadores complexos.

Definição Léxica:

Lexemas aceitos pela linguagem conforme a tabela

Categoria	Lexema	Padrão
Palavras-chave	int, real, bool, letra, palavra, entrada, saida, se, senão, enquanto, faça, retornar, ou, e	Palavras reservadas fixas

Operadores	=, +, -, *, /, >, <, ==, !=	Operadores aritméticos e relacionais
Delimitadores	`	“,”, “()” “||” “{}”
Números inteiros	Ex: 123	[0-9]+
Números reais	Ex: 12.34	[0-9]+\.[0-9]+
Booleanos	true, false	Valores booleanos pré-definidos
Caracteres	Ex: ‘a’	[a-zA-z]
Strings	Ex: “texto”	“\”.*?\””
Identificadores	Ex: nome_var	[a-zA-Z_][a-zA-z0-9]
Comentários	Ex: // comentário	Comentário de uma linha

Análise Léxica

```
/*
 Regras Léxicas
*/
DEC        : 'DECLARACOES';
ALG        : 'ALGORITMO';

// Palavras-chave
TIPO       : 'int' | 'real' | 'bool' | 'letra' | 'palavra';
ENTRADA    : 'entrada';
SAIDA      : 'saida';
SE         : 'se';
SENAO      : 'senão';
ENQUANTO   : 'enquanto';
FAZER      : 'faça';
RETORNAR   : 'retornar';
OU         : 'ou';
E          : 'e';

// Operadores
OP_ARIT    : '+' | '-' | '*' | '/';
OP_COND    : '>' | '<' | '==' | '!=';
OP_ATR     : '=';

// Outros símbolos
COLON      : ':';
PIPE       : '|';
VIRGULA    : ',';
LPAREN     : '(';
RPAREN     : ')';
```

```
LCHAVE      : '{';
RCHAVE      : '}';
PONTO_VIRGULA : ';';

// Literais
INT         : DIGITO+ ;
REAL        : DIGITO+ '.' DIGITO+ ;
BOOL        : 'verdadeiro' | 'falso' ;
PALAVRA     : '"' (~'"')* '"' ;

// Identificadores
ID          : LETRA (DIGITO | LETRA)* ;

// Fragmentos
fragment DIGITO : [0-9];
fragment LETRA : [a-zA-Z];

// Ignorar espaços em branco e novas linhas
WS          : [ \t\r\n]+ -> skip;
```

Exemplos de uso da linguagem

fatorial:

```
fatorial | int n | {
        se | n == 0 ou n == 1 | faça {
                retornar 1;
        } senão faça {
                retornar n * fatorial| n-1|;
                }
}
```

fibonacci:

```
fibonacci | int n | {
        se | n == 0 | faça {
                retornar 0;
        } senão se | n == 1 | faça {
                retornar 1 ;
        }senão faça{
                retornar fibonacci | n - 1 | + fibonacci | n - 2 |;
        }
}
```

Análise Sintática

Nessa parte do trabalho desenvolvemos a gramática livre de contexto da linguagem CPP, nessa gramática formalizamos descrição da repetição, da condicional, definição de funções, chamadas de função, parâmetros de função, etc.

Definição sintática da linguagem CPP

```
grammar CPP;

/*
 Regras Sintáticas
*/
// Programa principal
programa
    : funcao+ ;

// Definição de função
funcao
    : ID PIPE parametros PIPE LCHAVE comando* RCHAVE ;

// Parâmetros de função
parametros
    : (tipo ID (VIRGULA tipo ID)*)? ;

// Comandos gerais
comando
    : atribuicao
    | condicional
    | enquanto
    | retorno
    | chamadaFuncao PONTO_VIRGULA
    | expressao PONTO_VIRGULA ;

// Comando de atribuição
atribuicao
    : ID OP_ATR expressao PONTO_VIRGULA ;

// Comando de retorno
retorno
    : RETORNAR expressao PONTO_VIRGULA ;

// Estruturas de controle
condicional
    : SE PIPE condicao PIPE FAZER LCHAVE comando* RCHAVE SENAO?
      (SENAO FAZER LCHAVE comando* RCHAVE
      | SENAO SE PIPE condicao PIPE FAZER LCHAVE comando* RCHAVE)? ;

// Estrutura de repetição
enquanto
```

```
    : ENQUANTO PIPE condicao PIPE FAZER LCHAVE comando* RCHAVE ;

// Condições e expressões
condicao
    : expressao operadorComparacao expressao
    | expressao ;

// Expressões e termos
expressao
    : termo ((OP_ARIT) termo)* ;

termo
    : fator ((OP_ARIT) fator)* ;

fator
    : INT
    | REAL
    | BOOL
    | PALAVRA
    | ID
    | LPAREN expressao RPAREN
    | chamadaFuncao ;

// Chamadas de função
chamadaFuncao
    : ID PIPE argumentos PIPE ;

// Argumentos de funções
argumentos
    : expressao (VIRGULA expressao)* | ;

// Tipos e operadores
tipo
    : TIPO ;

operadorComparacao
    : OP_COND ;
```