Arquivos de códigos-fonte:

- lex.py = Contém o analisador léxico, que tem como função validar caracter por caracter e traduzir em uma sequência de tokens léxicos (Ferramenta).

- yacc.py = contém o analisador sintático que tem a função de validar se as sentenças estão respeitando a gramática(Ferramenta).

- declara.py = Declaração de funções e variaveis.

- errors.py = Declaração de mensageria de Erros.

- grammmar.py = contém a Gramática da Linguagem;

- main.py = Contém a ordem de execução dos analisador do compilador.

- mylexer.py = Contém o analisador léxico, que tem como função validar caracter por caracter e traduzir em uma sequência de tokens léxicos - (Específico do nosso trabalho);

- teste.txt = arquivo onde deve ser colocada as sentenças para realizar as validações;

Fases de Desenvolvimentos e Requisitos:

Etapa 1 - (Finalizado):

- Devem reconhecer os tipos: char – int – float;
- O char pode ser um caractere ou uma cadeia;
- Identificadores: de acordo com as regras da Linguagem C (iniciam por letras ou _, depois do segundo caractere pode ser número, letra ou _ e o único caractere especial é o_);
- Podes ser declarados individualmente ou por uma lista (sendo lista separados por vírgulas);
- A finalização de cada declaração será por ponto-e-vírgula (;);

Etapa 2 - (Finalizado):

- Comandos de seleção: if e switch-case;
- Sintaxe: if(condição){<comandos>} else {<comandos>};
- **Comando if-else;**
  - O comando else é opcional (assim, como na linguagem C);
  - Os comandos serão apenas matemáticos, ou seja, operações matemáticas simples: soma, subtração, multiplicação e divisão (operadores matemáticos), que serão atribuídos, por meio dos operadores de atribuição a uma variável;
- Sintaxe: switch(variável){case1:{<comandos> break;} case2: case3: {<comandos> break;} default: {<comandos>}};
- **Comando switch-case;**
  - Mesmas informações do comando if na questão dos comandos;
  - Os cases podem ser unitários ou até a quantidade de três;
  - O comando default é opcional (assim, como na linguagem;

Etapa 3 (Em andamento):

- Comandos de seleção: while e for;
- Sintaxe: while(condição){<comandos>};
- **Comando while;**

    - Os comandos serão apenas matemáticos, ou seja, operações matemáticas simples: soma, subtração, multiplicação e divisão (operadores matemáticos), que serão atribuídos, por meio dos operadores de atribuição a uma variável;
- Sintaxe: for(ini_variável; condição; incremento/decremento){<comandos>};
- **Comando for;**
    - Os comandos serão apenas matemáticos, ou seja, operações matemáticas simples: soma, subtração, multiplicação e divisão (operadores matemáticos), que serão atribuídos, por meio dos operadores de atribuição a uma variável;
    - A inicialização será por meio de uma atribuição.

Etapa 4 (Finalizado):

Operadores Matemáticos

Operador	Exemplo	Comentário
+	x + y	Soma x e y
-	x – y	Subtrai y de x
*	x * y	Multiplica x e y
/	x / y	Divide x por y

Operadores de Atribuição

Operador	Exemplo	Comentário
=	x = y	Atribui o valor de y a x
+=	x +=y	Equivale a x = x + y
-=	x -= y	Equivale a x = x – y
*=	x *= y	Equivale a x = x * y
/=	x/=y	Equivale a x = x / y

Operadores Relacionais

Operador	Exemplo	Comentário
==	x == y	O conteúdo de x é igual ao de y
!=	x != y	O conteúdo de x é diferente de y
<=	x <= y	O conteúdo de x é menor ou igual ao de y
>=	x >= y	O conteúdo de x é maior ou igual ao de y
<	x < y	O conteúdo de x é menor que o de y
>	x > y	O conteúdo de x é maior que o de y

- Apresentação e explicações do processo de construção do Trabalho, com horário marcado e tempo de apresentação de 15 minutos.

Gramática:

G = ({ })

P ::= {

<declaracao> ::= <declaracao_variavel> | <comparacao>

<declaracao_variavel>::= <tipo_primitivo><id_identificadores>;

<comparacao>::= IF(<condicao>){<comando>} | IF(<condicao>){<comando>}ELSE{<comando>} | SWITCH(condicao){CASE:{<comando>BREAK}} | SWITCH(<identificador>){CASE:{<comando>BREAK}DEFAULT:<comando>}

<comando>::= <identificador>+<identificador> | <identificador>-<identificador> | <identificador>/<identificador> | <identificador>*<identificador> | <identificador> = <identificador> | <identificador>+=<identificador> | <identificador>-=<identificador>|<identificador>*=<identificador>| <identificador>/=<identificador>

<condicao>::= <identificador><<identificador> | <identificador>><identificador> | <identificador>==<identificador> | <identificador>!=<identificador> | <identificador><=<identificador> | <identificador> =<identificador>

<tipo_primitivo> ::= INT| FLOAT| CHAR

<id_identificadores>::= <identificador> | <identificador>,<id_identificadores>

<identificador> ::= id
}

Sentenças no Formato que serão válidos:

Declaração de Variável(≤**declaracao_variavel>**):

ex: float b,a;

<tipo_primitivo><id_identificadores>;

Condicional IF(**<comparacao>**);

- A duas formas, utilizando o if sem o else;

```
IF (a==b){
    a=1;
}
```

IF(<condicao>){<comando>}

- Utilizando o if com o else;

```
IF (a==b){
        a=1;
}ELSE{
        a=2;
}
```

IF<condicao>{<comando>}ELSE{<comando>}

Condicional Switch ;

- A duas formas, utilizando o Switch sem o default;

```
SWITCH(a){
    CASE:{
       a=1;
    BREAK
}
```

SWITCH(<identificador>){CASE:{<comando>BREAK}

- A duas formas, utilizando o Switch com o default;

```
SWITCH(c){
```

```
    CASE:{
        a=1;
BREAK
    }DEFAULT:{
        b=1;
    }
}
```

SWITCH(<*identificador*>){CASE:{<comando>BREAK}DEFAULT:<comando>}

Palavra Token	**Expressão Regular Correspondente**
Palavras Reservadas	IF, ELSE, SWITCH, CASE, DEFAULT, BREAK, INT, FLOAT, CHAR
Identificadores	(a..z), (a..z)(0..9), _(a..z), _(a..z)(0..9)
Operadores	+ - / * == != > >= < <=
Delimitadores	; , ( ) { }