Software: O terror das férias.

Alunos: Gabriel Pereira e Sofia Rovaris.

1. DESCRIÇÃO DA LINGUAGEM

A linguagem utilizada para o desenvolvimento do trabalho é um subconjunto de C. Os tokens reconhecidos pela mesma são os indicados na tabela abaixo:

TOKEN	CATEGORIA	TOKEN	CATEGORIA
tok_while	Palavra reservada	tok_do	Palavra reservada
tok_for	Palavra reservada	tok_if	Palavra reservada
tok_else	Palavra reservada	tok_printf	Palavra reservada
tok_scanf	Palavra reservada	tok_void	Palavra reservada
tok_return	Palavra reservada	tok_main	Palavra reservada
int	Tipo de dado	double	Tipo de dado
%d	Representação de tipo de dado	%lf	Representação de tipo de dado
(	Símbolo	)	Símbolo
{	Símbolo	}	Símbolo
,	Símbolo	;	Símbolo
+	Operador	-	Operador
*	Operador	/	Operador
=	Operador	&&	Operador
||	Operador	==	Operador
!=	Operador	Texto constante	Texto
Número real	Número	Identificador	Identificador

2. EXPRESSÕES REGULARES

As seguintes expressões foram utilizadas para reconhecer classes de tokens específicas:

1. Para reconhecer um texto a expressão deve conter uma aspa no início e uma no final, podendo conter qualquer letra, número, caractere, espaço e símbolo entre elas.

```
({aspas}{1})({letra}|{numero}|{caractere}|{espaco}|{simbolos})*({aspas}{1})
```

2. Para reconhecer um número real a expressão deve conter pelo menos um número, podendo este ser seguido por mais números ou um ponto (indicando um número decimal).

```
({numero})+(".")?({numero})*
```

3. Para reconhecer um identificador a expressão deve conter ou uma letra ou *underline*, e pode ser seguido por números, letras ou *underline*.

```
({letra}|"_")({letra}|"_"|{numero})*
```

3. FUNCIONAMENTO DO SOFTWARE

O software funciona de maneira bem simples. Ao executá-lo, solicitará a entrada do nome de um arquivo, que será lido. O software irá realizar uma análise léxica no código contido no arquivo, reconhecendo as classes de tokens e apontando os erros.

Sobre o desenvolvimento do software: Na primeira parte delimitada por %{ }% são declaradas as variáveis auxiliares, e as bibliotecas. Logo abaixo, são definidas as expressões regulares, e entre os símbolos de %% ficam declarados as regras que consistem em padrões que serão encontrados e as ações correspondentes que serão executadas. Por último temos as funções auxiliares e o main. O Lex oferece algumas variáveis e funções que auxiliam na análise léxica. A variável yyin é do tipo ponteiro para arquivo e foi utilizada para abrir o arquivo texto de entrada. A função yylex() lê a entrada apontada pela variável yyin e procura um padrão de correspondência na seção de regras, e quando encontra executa a ação indicada.

4. TRATAMENTO DE ERROS

Para tratar os erros, criaram-se expressões para identificar cada tipo de erro, entre eles: má formação, fins inesperados, e extrapolação dos limites de dados.

Abaixo encontram-se as expressões regulares escritas para reconhecer os erros léxicos:

1. A expressão a seguir reconhece que um texto é mal formado quando há a presença de apenas uma aspa.

```
({aspas}{1})({letra}|{numero}|{caractere}|{espaco}|{simbolos})*
```

2. As próximas três expressões reconhecem erros relacionados a classe de tokens número.

a) Identifica se há uma letra, caractere ou *underline* antes do ponto, identificando um número mal formado.

```
({numero})+({letra}|{caractere}|"_")+(".")?({numero})*
```

b) Identifica se há uma letra, caractere ou *underline* depois do ponto, identificando um número mal formado.

```
({numero})+(".")?({letra}|{caractere}|"_")+({numero})*
```

c) Verifica se o número excede o limite de caracteres permitidos.

```
({numero})+(".")?({numero}){9,}
```

3. As outras três expressões reconhecem se há erros relacionados a classe de tokens identificador.

a) A primeira expressão reconhece se existe um caractere especial antes das letras e números que formam o identificador.

```
({caractere})+({letra}|"_")({letra}|"_"|{numero})*
```

b) A segunda reconhece se existe um caractere especial no meio do identificador.

```
({letra}|"_")({letra}|"_"|{numero})*({caractere})+({letra}|"_"|{numero})*
```

c) E a última verifica se o identificador tem um nome que ultrapassa os limites de caracteres.

```
({letra}|"_")({letra}|"_"|{numero}){128,}
```

5. CONFIGURAÇÕES E FERRAMENTAS

Para desenvolver o software foi utilizado o Flex (Fast Lexical Analyzer Generator), uma ferramenta para gerar analisadores léxicos.

Para compilar o software é necessário fazer o download de dois programas: o flex e o bison. Após o download, deve-se incluir os diretórios na variável PATH. Há um tutorial para a instalação no Stack Overflow, link para acesso: https://stackoverflow.com/questions/5456011/how-to-compile-lex-yacc-files-on-windows.

Depois da instalação e configuração finalizada, basta compilar o código com os seguintes comandos:

1. flex .\analisador_lexico.l
2. gcc lex.yy.c -o analisador_lexico.exe
3. ./analisador_lexico

6. REFERÊNCIAS

Para instalar o Flex: https://stackoverflow.com/questions/5456011/how-to-compile-lex-yacc-files-on-windows. Acesso em: 18 de janeiro de 2022.

Documentação do Lex utilizada para entender o funcionamento da ferramenta: https://silcnitc.github.io/lex.html#navstructure. Acesso em: 18 de janeiro de 2022.

Documento utilizado para verificar formatos, definições e condições de variáveis e regras: https://www.iith.ac.in/~ramakrishna/Compilers-Aug14/doc/flex.pdf. Acesso em: 20 de janeiro de 2022.

Vídeo utilizado de referência para ler um arquivo txt usando o Flex: https://www.youtube.com/watch?v=JHIMT5OiOJQ. Acesso em: 18 de janeiro de 2022.