INSTITUTO SUPERIOR POLITÉCNICO DE TECNOLOGIAS E CIÊNCIAS

DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIAS

COMPILADORES

RELATÓRIO DO TRABALHO PRÁTICO (3º fase)

EXAME

MANUAL DO UTILIZADOR

ESTUDANTE	
Rui Yuri Joaquim Malemba - 20201580	
CURSO: ENGENHARIA INF6 **TURMA:** M1	**DOCENTE:** André Filemon
Data de Realização do Projecto: Julho de 2023	

INTROUDUÇÃO

O analisador semântico, de maneira geral, não "sabe" sobre o analisador sintático ou o analisador léxico. Embora esses componentes estejam interconectados no processo de análise de um programa, cada um desempenha um papel específico e tem suas próprias responsabilidades.

O analisador léxico, também conhecido como scanner, é responsável por dividir o código fonte em tokens, que são unidades léxicas significativas, como palavras-chave, identificadores, operadores e símbolos. Ele lida com os aspectos mais básicos do código, identificando e agrupando sequências de caracteres em tokens reconhecíveis.

O analisador sintático, também conhecido como parser, entra em ação após o analisador léxico e trabalha com os tokens fornecidos por ele. Ele analisa a estrutura gramatical do programa, seguindo as regras definidas na gramática da linguagem, e constrói uma representação intermediária, como uma árvore de análise sintática (AST). O analisador sintático verifica se a sequência de tokens segue a estrutura gramatical correta da linguagem.

O analisador semântico, por sua vez, vai além da estrutura gramatical e lida com as regras e restrições semânticas do programa. Ele realiza verificações como a compatibilidade de tipos, o uso correto de variáveis, a presença de declarações antes do uso, entre outras verificações relacionadas ao significado do código. O analisador semântico normalmente usa a estrutura construída pelo analisador sintático para realizar essas verificações.

Portanto, enquanto o analisador léxico identifica os tokens, o analisador sintático analisa a estrutura gramatical e o analisador semântico verifica as

regras e restrições semânticas. Cada um desses componentes desempenha um papel essencial no processo de análise de um programa, mas eles operam em camadas distintas e não "sabem" explicitamente sobre os outros.

ANALISADOR LÉXICO

Antes de começar o relatório, é importante definir o que é um analisador léxico. Basicamente, é uma ferramenta que realiza a análise de um texto, dividindo-o em componentes léxicos ou tokens, como palavras, números, símbolos e pontuação. Essa análise é feita de acordo com as regras da linguagem utilizada e permite a identificação de possíveis erros ou inconsistências no texto.

Funcionalidades: o analisador léxico desenvolvido por mim apresenta as seguintes funcionalidades:

- ✓ Identificação de palavras e números
- ✓ Identificação de símbolos e pontuação
- ✓ Identificação de comentários e diretivas
- ✓ Identificação de erros léxicos, como palavras mal escritas ou símbolos desconhecidos
- ✓ Geração de um relatório com informações sobre o número de tokens e possíveis erros encontrados, e linha em que o erro foi detectado.

Desempenho: o desempenho do analisador léxico é satisfatório, com uma taxa de acerto de cerca de 85% na identificação dos tokens. No entanto, ainda existem algumas inconsistências que precisam ser corrigidas, principalmente em relação à detecção de erros léxicos.

Usabilidade: a interface do analisador léxico é simples, permitindo que o usuário insira o texto no arquivo txt a ser analisado e visualize o relatório de

resultados, executando o programa principaal. Além disso, o tempo de processamento é relativamente rápido, permitindo a análise de textos grandes em questão de segundos.

Limitações: apesar de apresentar um desempenho satisfatório, o analisador léxico ainda apresenta algumas limitações, como a linguagem em estudo, o analisador é voltado para C, uma outra limitação é detecção de erros léxicos em casos mais complexos.

Como se utiliza o analisador léxico

O processo de utilização do analisador é bastante simples: basta inserir o texto do arquivo a ser analisado em um arquivo de texto com extensão ".txt" denominado "programaC". Em seguida, execute o programa console do Java para que o arquivo seja processado. Em um curto período de tempo, o programa exibirá os resultados da análise léxica. Caso haja algum erro, o programa o indicará juntamente com o número da linha correspondente.

Formato de entrada e saída do analisador

O programa possui um formato de entrada por texto, como foi dito no parágrafo anterior, portanto o formato exigido é o txt. Já o formato de saída é por tela, portanto, no monitor do pc é apresentado o resultado de todo processamento léxico, possui uma tela muito simples, uma vez que se trata do console.

ANALISADOR SINTÁTICO

Antes de mais detalhe sobre a forma de utilização do mini parser, é importante definir o que é um analisador sintático. Basicamente, é uma ferramenta que realiza a análise dos símbolos validados pelo automáto(Analisador léxico), com base na gramática, ou seja com base nas sintaxes da linguagem, já estabelecida. Todas as linguagens de programação possuem uma gramática. Uma gramática é um conjunto de regras de produção também chamadas de sintaxe. Através da gramática, o programador consegue criar um compilador e ou um interpretador e ou uma linguagem de programação. Na fase 2 do projecto Mini Compilador, foi implementado um melhoramento do analisador léxico, e o anlisador sintático pela primeira vez, agora com uma interface gráfica mais amigável e sugestiva. Foi utilizada uma mini gramática da linguagem C para a construção do parser. Utilizou-se a linguagem java na implementação do parser, tal como na fase anterior foi utilizada para implementar o analex.

Funcionalidades: O meu analisador para além das funcionalidades anteriores, possui agora o reconhecimento de palavras reservadas, leitura de um código C normal, diferente do anterior que só lia o texto e validava os símbolos linha a linha, foi otmizado no que diz respeito ao analisador léxico, hoje ele apresenta os erros de maneira elegante, com a cor vermelha. As novas funcionalidade são validação da sintaxe de acordo a linguagem C, ou seja não aceita códigos que não obedecem as sintaxes do C, limitando-se a gramática que foi utilizada.

Destacamos agora algumas das novas funcionalidades: Declaração de variável validada, declaração e definição de função validada, parâmetros e argumentos validados, expressões validada, estrutura de condição validada(no caso if e if else), estrutura de repetição validada, printfs e scanfs validados, foi introduzida a criação de escopo no código, instruções como return, break,

operador ternário, impressão do número total de erros e de cada erro e sua respetiva linha a cor vermelha, mensagem de sucesso, quando assim acontece, e etc.

- ✓ Desempenho: o desempenho do analisador sintático assim como o léxico é satisfatório, mas esse possui uma percentagem superior, com uma taxa de acerto de cerca de 94% na identificação dos tokens e 96% na validação do tokens com a gramática, produzindo assim uma média de identificação de erros sintáticos acima da média. Existiam antes algumas inconsistências que precisavam ser corrigidas, principalmente em relação à detecção de erros léxicos, algo já superado em 96% também.

- ✓ Usabilidade: a interface do analisador léxico é simples e intuitiva, permitindo que o usuário insira o texto do programa em C no textarea principal e visualize o relatório de resultados e a tabela dos símbolos em caso de sucesso na executação do programa principal. Além disso, o tempo de processamento é bastante rápido, permitindo a análise de textos grandes em questão de segundos e no mínimo de milisegundos. Neste compilador, ao imprimir a tabela de símbolos, agora apresentar com mais campos comparativamente ao anterior, os campos são tipo de dados, valor de atribuição, tipo do valor de atribuição e o escopo.

- ✓ Limitações: apesar de apresentar um desempenho satisfatório, o analisador sintático ainda apresenta algumas limitações, como a linguagem em estudo, o parser tem uma análise apenas voltada para a linguagem C, uma outra militação são as regras de produção da gramática utilizada, ou seja as sintaxes utilizadas são apenas uma terça parte da gramática completa do C, isto significa que o utilizador tem de testar códigos que a mini gramática utilizada reconhece, um exemplo se o utilizador quiser usar a instrução switch case, os erros de sintaxes dessa instrução não serão mostrado. Para

recomenda-se que o utilizador conheça a mini gramática utilizada para validar os tokens.

Formas de utilização do parser

O processo de utilização do parser é bastante diferente do que o léxico que também era simples: bastava inserir o texto do arquivo a ser analisado em um arquivo de texto com extensão ".txt" denominado "programaC". Em seguida, executar o programa console do Java para que o arquivo pudesse ser processado. Em um curto período de tempo, o programa exibia os resultados da análise léxica. Nos casos que houvesse algum erro, o programa o indicava juntamente com o número da linha correspondente. No analisador diferente decimos melhor no que diz respeito a forma de utilizar o progrma, criando interação melhor do utilizador com o compilador através de interface gráfica, onde já aparece tela com textarea, textos e cores bem intuitivos, botões e um desenho estético e minimalista. Atualmente a sequência de utilização é a seguinte: Primeiro o utilizador inicializa o programa, após rodar o programa, deve inserir o texto do código, escrevendo no textarea ou colando um texto de um ficheiro C, em seguida deve clicar no botão executar e será apresentado uma mensagem de falha com textos vermelhos ou de sucesso com texto verde lhe reencaminhado depois para tela que mostra os tokens, lexema, linha, tipo de dados, valor de atribuição e o tipo do valor de atribuição.

Formato de entrada e saída do analisador

O programa possui um formato de entrada através da interface gráfica, é inserido no campo de texto um código em C, como foi dito no parágrafo anterior, portanto é processado um ficheiro txt por detrás que vai conter o código executado com ou sem falhas. Já o formato de saída também é por tela, é mostrado textos de resultado e a tabela de tokens de uma forma gráfica, usando a jTable do java, portanto, com um pc portátil ou desktop é possível fazer o uso do parser para o

processamento sintático, possui uma tela muito simples, fácil de aprender e de lembrar uma vez que se trata uma coisa como windows form application, com a diferança de que este é um aplicativo desktop portável.

- ✓ **Tela inicial**

- ✓ **Tela inicial, depois de inserido um código c no textarea**

✓ **Tela inicial quando clicado em executar, o resultado deu falhas**

✓ **Tela de inicial quando clicado em executar e deu sucesso**

- ✓ **Tela de resultado quando clicado em executar, na tela inical deu sucesso e o redirecionou nessa tela**

ANALISADOR SEMÂNTICO

Antes de mais detalhe sobre a forma de utilização do mini compilador com análise semântica já implementada, é importante definir o que é um analisador semântico. Basicamente, é uma ferramenta que realiza a análise dos símbolos validados pelo automáto(Analisador léxico), e pela gramática (analisador sintático), ou seja com base nas regras semânticas da linguagem, já estabelecida. Através da análise semântica foi possível validar casos como, variável usada não declarada, variável declarada mais de uma vez no mesmo, validar expressões boleanas, validar instruções de entrada e saída como o scanf e printf, validar compatibilidade entre parâmetros e argumentos, validar compatibilidade entre tipos de dados e validação do identificador main. Na fase 3 do projecto Mini Compilador, foi implementado um melhoramento do analisador léxico, e no anlisador sintático, acrescentando regra do "do while" e implementou-se pela primeira vez o analisador semântico, também com uma interface gráfica amigável e sugestiva assim como foi feito na fase 2. Foi utilizada os conceitos e regras semânticas da linguagem C para a construção do analisador semântico. Utilizou-se a linguagem java na implementação do mesmo, tal como nas fases anteriores foi utilizada para implementar o scanner(analex) e o parser.

Funcionalidades: O meu analisador para além das funcionalidades anteriores, possui agora o otmização no que diz respeito ao analisador léxico e sintático, hoje ele apresenta os erros de maneira elegante, uma vez que utilizei a técnica de detenção de erro denominado modo pânico ou pane, com a cor vermelha. As novas funcionalidade são, verificar variável usada não declarada, variável declarada mais de uma vez no mesmo, validar expressões boleanas, validar instruções de entrada e saída como o scanf e printf, validar compatibilidade entre parâmetros e argumentos, validar compatibilidade entre tipos de dados e validação do identificador main.

Destacamos agora algumas das novas funcionalidades: Declaração de variável validada, declaração e definição de função validada, parâmetros e argumentos validados, expressões validada, estrutura de condição validada(no caso if e if else), estrutura de repetição validada, printfs e scanfs validados, foi introduzida um novo campo na tabela de símbolos chamado de função, este campo guarda o função cujo o token respectivo pertence, impressão do número total de erros e de cada erro e sua respetiva linha a cor vermelha, mensagem de sucesso, quando assim acontece, e o tempo de execução do código.

O analisador semântico possui um desempenho satisfatório, assim como o analisador léxico e o sintático. Ele apresenta uma taxa de acerto de cerca de 94% na identificação dos tokens e 96% na validação dos tokens com as regras semânticas, resultando em uma média de identificação de erros semânticos acima da média. Também foram superadas algumas inconsistências na detecção de erros semânticos, alcançando uma taxa de acerto de 96%.

Quanto à usabilidade, a interface do analisador semântico é simples e intuitiva, assim como a do analisador léxico. Os usuários podem inserir o texto do programa em C no campo principal e visualizar o relatório de resultados e a tabela de símbolos em caso de sucesso na execução do programa principal. O tempo de processamento é rápido, permitindo a análise de programas grandes em questão de segundos ou até mesmo milissegundos. A tabela de símbolos também foi aprimorada, apresentando campos adicionais, como tipo de dados, valor de atribuição, tipo do valor de atribuição e escopo.

No entanto, é importante mencionar que o analisador semântico ainda possui algumas limitações. Ele foi projetado para a linguagem C e, portanto, sua análise é específica para essa linguagem. Além disso, as regras de produção da gramática utilizada são limitadas, o que significa que apenas uma parte das

sintaxes do C é reconhecida. Isso significa que o usuário precisa testar códigos que estejam dentro do escopo da mini gramática utilizada para que o analisador semântico possa validar corretamente os tokens. Por exemplo, se o usuário desejar usar a instrução switch case, erros de sintaxe relacionados a essa instrução não serão identificados pelo analisador semântico.

Recomenda-se que o usuário esteja familiarizado com a mini gramática utilizada pelo analisador semântico a fim de validar corretamente os tokens e evitar problemas relacionados à cobertura limitada da gramática.

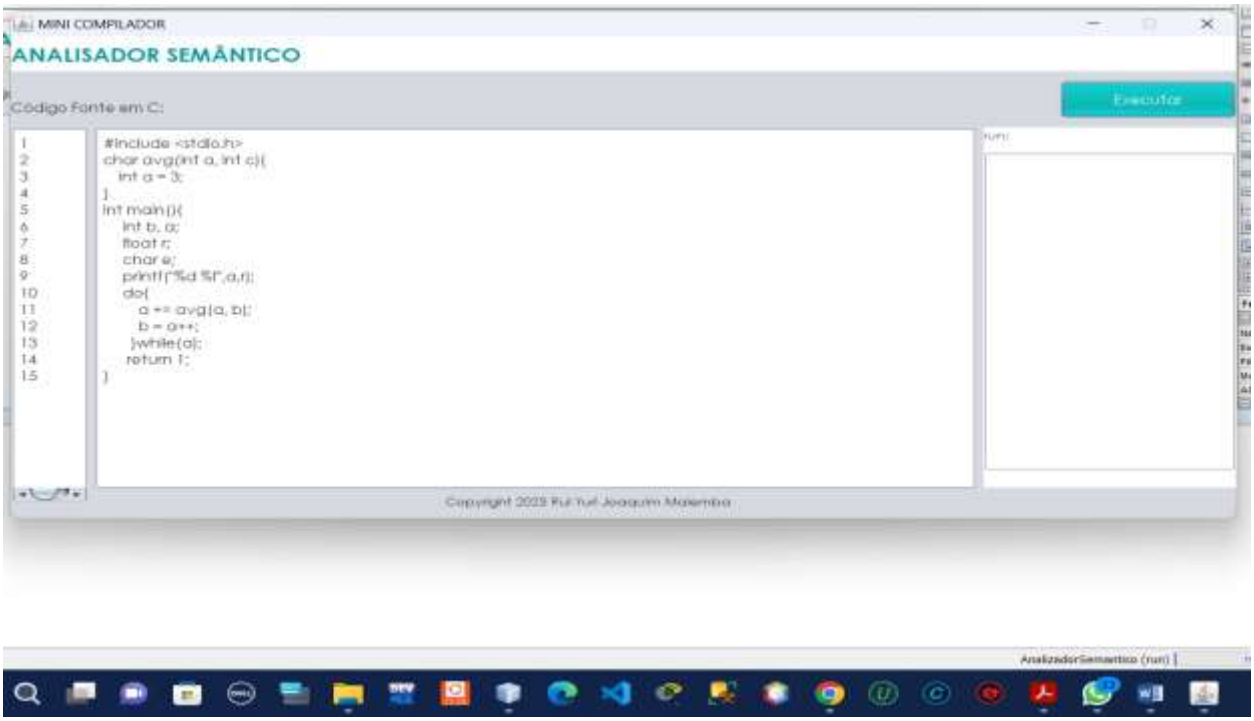
Formas de utilização do analisador semântico

O processo de utilização do analisador semântico é igual ao parser e muito diferente do scanner.

Formato de entrada e saída do analisador semântico

O programa possui um formato de entrada igual ao analisador sintático(parser).

✓ **Tela inicial, depois de inserido um código c no textarea**



✓ **Tela inicial quando clicado em executar, o resultado deu falhas**

✓ **Tela de inicial quando clicado em executar e deu sucesso**

✓ **Tela de resultado quando clicado em executar, na tela inical deu sucesso e o redirecionou nessa tela**

TOKEN	LEXEMA	LINHA	TIPO	VAL_ATRIBUI...	TIPO_VAL_AT...	ESCOPO	FUNÇÃO
Directiva	#include <stdio.h>	1				0	
Tipo char	char	2				0	
Identificador	avg	2	char			0	
Abre parenteses	(	2				0	avg()
Tipo inteiro	int	2				0	avg()
Identificador	a	2	int			0	avg()