Metodologia

Neste capitula é informado quais os métodos e ferramentas foram utilizados para desenvolver o interpretador e a interface, a forma de abordagem para a coleta de dados, o cenário e os indivíduos participantes.

A metodologia seguida para se obter dados sobre a utilização e a funcionalidades para os alunos e para o professor da matéria de compiladores, é a utilização de um questionário elaborado para ser respondido após o uso do software, este questionário está no apêndice.

As tecnologias utilizadas para o desenvolvimento do interpretador, são: linguagem de programação C++, QT Creator, JFLAP, e as metodologias de criação de um compilador, Análise Léxica, Análise Sintática e Analise Semântica.

5.1 LINGUAGEM C++

A linguagem utilizada para o desenvolvimento deste software é C++, que é a predominante no programa QT Creator, e por se diferenciar pouco da linguagem C, a base no ensino no curso de ciências da computação. Esta linguagem também possui bibliotecas que auxiliam no desenvolvimento, como o regex, que é utilizado neste trabalho para tratar expressões regulares e assim tornar o código melhor escrito sem necessidade de repetição de regra.

Na figura abaixo é ilustrado a função descrita acima, nela é chamada a função regex\_match, está função verifica se a letra passada, a variável line[aux], faz parte da gramatica enviada junto, regex("[a-zA-Z\_]").

5.2 QT CREATOR

QT Creator é um ambiente de desenvolvimento integrado (IDE), criado por uma empresa norueguesa Trolltech, que visa facilitar e ajudar no desenvolvimento de softwares, permitindo criar sistemas para plataformas múltiplas(<https://wiki.qt.io/Main>).

A principal razão da escolha desta IDE foi a facilidade de desenvolvimento, devido a possuir um compilador integrado, uma depuração excelente que possibilita que o desenvolvedor encontre o erro no código produzido, e o corrija rapidamente, além de ter familiaridade com está framework por já utilizá-la em trabalhos da universidade.

Na figura abaixo é ilustrado a interface do Qt Creator, e as opções de funcionalidades no lado esquerdo, sendo o ícone de triangulo deitado verde a função de compilação e o ícone de triangulo verde junto a um inseto o de depuração.

Figura(qtCREATOR).

Este software da a possibilidade de elaborar a interface, com componentes básicos prontos necessitando apenas configura-los, assim centralizando tanto a Back-End (parte logica do software) como o Front-End (interface).

Na figura abaixo é ilustrado a interface de do Qt Creator para a criação de interfaces, no lado esquerdo se encontra componentes já criados, sendo necessários apenas arrastar e configura-los.

Figura(qt)

5.3 JFLAP

JFLAP é um software gratuito educacional desenvolvido na linguagem JAVA por Susan H. Rodger, o principal uso deste framework é na criação de autômatos finitos não determinísticos, maquinas de Turing e vários tipos de gramatica. Este software foi utilizado para criar todos os autômatos da linguagem D+, utilizados para ilustrar na interface do interpretador por onde o código passou.

Na figura abaixo é ilustrado as opções que o software oferta para o usuário, desde autômato finito, gramáticas, máquina de Turing entre outros. Neste trabalho foi utilizado apenas a funcionalidade de autômato finito.

Figura JFLAP

Na figura abaixo é ilustrado o quadro que a framework abre para a criação dos autômatos.

Autômatos

A criação dos autômatos da linguagem d+ foram criados para facilitar no desenvolvimento da análise léxica e posteriormente para a ilustração do percurso do código escrito. Foi desenvolvido uma função para cada estado do autômato.

Foram criado um total de 20 autômatos, exportados como png para utilizá-los na interface, e assim ilustrar para quem estiver utilizando o interpretador.

Na figura abaixo é ilustrado um autômato criado na ferramenta JFLAP. Ele demonstra o caminho de validação de um identificador ou palavra reservada, onde sua gramática permite letras de A à Z tanto minúsculas ou maiúscula.

Figura JFLAPAutomato

5.4 ESTRUTURAS ADOTADAS NO DESENVOLVIMENTO

Neste capitulo são tratadas as estruturas utilizadas para o desenvolvimento da lógica da compilação, programação estruturada, Análise léxica, Análise Sintática, e Análise semântica.

Programação estruturada

A estrutura de programação estruturada foi a escolhida no desenvolvimento do software, por conta de ser a base ensinada no curso de ciências da computação, e com isso ter maior familiaridade com esta estrutura.

A programação estruturada tem como base três mecanismos em que os blocos de códigos se interligam, os mecanismos são: sequência, seleção e iteração \cite{ivan2003}.

A sequência é o fluxo que o código toma de acordo com a ação \cite{ivan2003, um exemplo é o fluxo que o interpretador desenvolvido toma quando é clicado no botão de Run, ele carrega o código inserido, efetua a análise léxica em seguida a análise sintática e por fim a analise semântica.

Figura exemplo de sequência.

A seleção é a verificação do caminho que será tomado com uma verificação do comando IF \cite{ivan2003, um exemplo encontrado no código, é uma verificação que se faz andes de iniciar a análise sintática, caso haja um erro na analise léxica, ele não prossegue para a análise sintática.

Figura codigo

A iteração é o mecanismo de repetição, enquanto uma condição for atendida ela é executa, um exemplo encontrado no código é na função state01, nesta função existe uma estrutura de repetição, while, em que e verificado se a letra de uma variável satisfaz a uma regra, caso sim, refaz os passos contidos dentro desta estrutura, ao contrário segue o código abaixo dela.

Análise Léxica

Para o desenvolvimento da analise léxica, foram utilizados os autômatos criados, em apêndice, como guias, o autômato do estado 0 é o que seleciona o caminho que aquele caractere ira percorrer, o mesmo é feito na função state00, ao encontrar o caminho é atribuído um valor para a variável state sendo o valor o número do próximo estado, e assim prossegue para o estado subsequente.

Na figura abaixo é ilustrado a função state00, nela é verificado a letra que o programa está analisando, caso faça parte da gramatica, é direcionada para o caminho correto.

A função reservWorks contém a lógica para verificar se aquele lexema formado é uma palavra reservada, a função possui um laço de repetição, for, em que percorre uma matriz criada que contém todas as palavras reservadas da linguagem D+, casso seja igual a alguma delas, é atribuído o token da palavra reservada, do contrário o token é atribuído como identificador. Na figura abaixo é ilustrado a função reservWorks.

Após a análise obter os valores de lexema e token, é chamada a função queueValue, que atribui os valores do lexema e token em uma estrutura de fila, está fila é utilizada para passar estes valores para uma string que é utilizada para ilustrar em um campo text na interface do software, na figura 27 é ilustrada a saída deste string na aba Análise léxica.

Para identificar por qual autômato o código passou, e utilizando de variáveis globais booleanas, em que ao acessar um estado, esses funções criadas como state01, state02, são atribuídos true para a variável correspondente aquele estado, como stateAutomato1. No fim de toda compilação é selecionada a imagem de todos os autômatos para as labels da interface, caso aquela variável do autômato for true ele troca a imagem preto e branco para a colorida.

Na figura abaixo é ilustrado os dois casos, um autômato em preto e branco, quando não é acessado e outro colorido, quando é acessado.

Figura automato.

Analise Sintática

Para o desenvolvimento da analise sintática, foi utilizado a criação de uma matriz que armazena todos os tokens e lexemas encontrados após a análise léxica, esta matriz é utilizada para verificar a sequencia de tokens, e validados se estão a ordem correta.

Após finalizar a analise léxica, o software chama em sequencia a rotina para a analise sintática, está começa a verificação na posição 0 da matriz tokensLexemasTable, e verificando em sequencia da matriz utilizando das regras da gramatica.

Na tabela abaixo e ilustrado as funções criadas referentes a cada regra, seguindo uma logica para verificar se o próximo elemento da matriz corresponde com o elemento esperado.

Tabela das funcoes

Cada função criada na analise sintática, soma mais um na variável tamanho, que percorre a matriz e valida com o token aguardado para aquela regra.

Na figura abaixo ilustra a função DV que corresponde a regra decl-var, nela é efetuada uma chamada para função ET e em seguida a função LV, após estas chamadas ocorrerem e não retornarem erro, ele valida se possui o lexema de ( ; ), casso possua, e retornado uma mensagem de sucesso ao log, ao contrário retorna a mensagem de erro, a variável keySintatico funciona como uma chave, sempre que é encontrado um erro no código, é atribuído o valor 1, assim o software para de efetuar a analise sintática, por já ter encontrado erro.

Figura funçãoDV

Arvore Sintática

Para a criação da arvore sintática é utilizado de uma variável global chamada NIVEL, ela é responsável por informar quantos caracteres “\t” deve possuir aquele trecho da arvore, cada regra acessada na análise sintática, é chamada a função treeSintatica, onde é implementada a arvore. Sempre que a análise sintática acessa uma regra, é atribuída a variável treeTerminal uma string que tem como valor o nome da regra que aquela função é correspondente, esta variável é utilizada na função treeSintatica para adicionar aquela regra a arvore.

Na figura abaixo é ilustrada a função treeSintatica, esta função possui uma estrutura de repetição while, que incrementa o caractere “\t” o número de vezes que a variável NIVEL possui como valor, esta parte da função implementa a estrutura da arvore, após este passo, é atribuído o caractere “|” para servir como referencia a qual aquela palavra pertence ao ramo pai, por fim é atribuído o token a arvore, caso a variável treeTerminal for diferente de vazia, é impresso aquilo que esta variável possui.

Figura treeSintatica

Na figura abaixo é ilustrado a saída da arvore de um código que efetua uma declaração de var, nela é possível ver por quais regras foram utilizadas na montagem.

Gramatica

A gramatica completa é apresentada na interface para que facilite no entendimento da linguagem, assim pode ser usado como consulta ou para retirar duvidas de uma regra especifica.

Na figura abaixo e ilustrado a aba de gramatica contendo um trecho das regras da linguagem d+.

Figura gramatia

Após compilar o código inserido na interface, e adicionado na aba da gramatica as regras utilizadas. Para desenvolver este comportamento, e utilizado de uma variável global, similar ao do autômatos descrita abaixo, porem para gramatica, quando na analise sintática aquela regra especifica e acessada, a variável global da quela regra, setada como false, e atribuída true, e verificada no final pra incrementar na string que e passada para a interface.

Na figura abaixo inlustra a função V, que e a regra de VAR, nela e atribuído o valor true, na figura abaixo e ilustrado o código que verifica se aquela regra foi acessada pela varaivel global atribuída, caso seja true, e incrementado aquela string na string que e passada para a interface caso contrario não faz nada.

Figura gramaticaUtilizada

Figura função VAR

Figura ifdevalidacao da gramatica.

Analise semântica

O desenvolvimento para a analise semântica, foi na criação da tabela de símbolos, que e um dos passso dentro desta, a tabela de símbolos e criada na fase de analise léxica, na parte de atribuição, e verificado se aquele token e um identificador, caso seja, e percorrido a matriz para traz, se houver alguma estrutura de declaração e adicionada aquele identificador a tabela de símbolos, caso contrario, e entendido que pode ser um comando de atribuição ou outro que utiliza de identificadores.

Na figura abaixo e ilustrado o código que implementa a solução descrita no parágrafo acima.

Figura da verificação de identificadores.

Interface

Na criação da interface gráfica foi utilizando como base o editor de texto básico notepad, e editores de desenvolvimentos simples como code block e visual studio code.

Na barra superior do programa foi incrementando funcionalidades simples mas praticas como, novo, que cria um arquivo novo, abrir, que abre uma janela para que possa procurar algum arquivo do formato d+ e seja carregado no software, salvar, run, que efetua os passos das análises no código escrito na framework, desfazer e refazer.

Na figura abaixo é ilustrado a barra onde se encontra as funcionalidades descritas acima. Todos os ícones foram retirados do site [https://www.flaticon.com/https://www.flaticon.com/](https://www.flaticon.com/)

Figura da barra

A interface do software abrange todas as saídas das analises, para melhor aproveitamento de espaço da tela e organizar, é utilizando função de abas similar ao dos navegadores como google chrome, nestas abas é possível selecionar o que quer ver.

Na aba editor é o local em que insere o código d+, foi deixado um fundo acinzentado para auxiliar e destacar o local de escrita do código, na aba codigo numerado, é uma copia do codigo inserido porem numerado, para que assim seja mais fácil encontrar o erro acusado pelo log da analise sintática.

Na figura abaixo é ilustrado as duas abas do editor, no canto esquerdo a aba do editor e no canto direito a aba do codigo numerado.

Figura editor

Na figura abaixo é ilustrado uma visão geral do software.

Figura interface