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.

As tecnologias utilizadas para o desenvolvimento do interpretador, são: linguagem de programação C++, QT Creator, JFLAP, 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.

A estrutura adota para o projeto é a programação estruturada, ...

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. (http://www.jflap.org/). 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 16 é ilustrado um autômato criado no JFLAP, ele é apresentado na interface desenvolvida quando aquele caminho de autômato é acessado pela analise léxica. Na figura 17 é ilustrado um autômato que não foi acessado na análise léxica).

Na figura abaixo é ilustrado as opções que o software oferta para o usuário, desde autômato finito, gramaticas, 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.

Figura JFLAPAutomato

5.4 ESTRUTURAS ADOTADAS NO DESENVOLVIMENTO

Nesta sessão, são tratadas as estruturas utilizadas para o desenvolvimento da lógica da compilação, Analise léxica, Analise Sintática, e analise semântica.

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.

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.

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 aquela letra ira percorrer, e assim obtendo o lexema e validando se faz parte da gramatica da linguagem, constatando se é uma palavra reservada ou um identificador.

Na figura abaixo ilustra a função do estado 0, verificando se aquela letra que esta validando, satisfaz alguma regra nela incorporada, como o if de validação de letra ou o switch case para verificar se é algum caractere que não seja letra nem número mas são validos, como +.

Após validar o caractere e obter a palavra completa, é chamada a função de reserveWord, em que verifica se o lexema encontrado é uma palavra reservado, as palavras reservadas estão armazenadas em uma matriz de tamanho 45 por 2, em apêndice, está matriz é percorrida através de um for, e comparando se aquela palavra é igual ou lexema, caso seja, é atribuído o id daquela palavra para a variável token, ao contrário é atribuído ao token o id de identificador.

Na figura abaixo é ilustrado a função reservWord.

Após a análise obter os valores de lexema e token, e chamada a função queueu, que atribui os valores do lexema e token em uma fila, que é retirado no final da análise para uma string, esta é utilizada para ilustrar em um campo text na interface do software.

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.

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 é apresentada por padrão, esse trecho é utilizado apenas um