Relatório de Desenvolvimento do Trabalho Final de Compiladores

Jediael Morais de Souza (RA: 171253892) e Matheus Henrique Patrizi (RA: 171251377)

1. Desenvolvimento

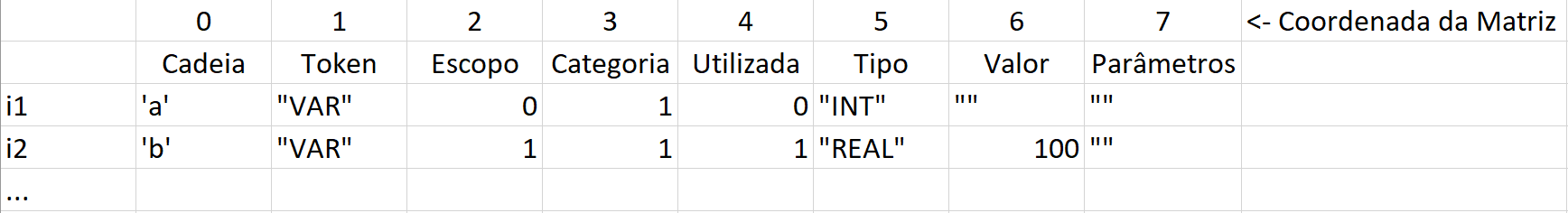
Trabalho desenvolvido em Javascript, HTML e CSS.

Sistema operacional utilizado foi o Windows 10, e todos os testes foram feitos no Google Chrome.

1.1 Desenvolvimento do Tratamento de Erros da Análise Sintática

* 1. Desenvolvimento da Análise Semântica

O desenvolvimento da análise semântica do trabalho foi realizada utilizando uma abordagem que conta com uma tabela de símbolos onde as variáveis e procedimentos são inseridos, assim como suas características. Esta tabela foi implementada como uma matriz JavaScript. Segue sua estrutura:



A cadeia representa o “nome” da variável (ou procedimento), o segundo item é o token, seguido pelo escopo, que foi representado numericamente, sendo 0 o escopo “global” e a cada escopo fechado esse número aumenta. Categoria = 1 significa que se trata de uma variável, e Categoria = 2 é um procedimento. Utilizada = 1 significa que a variável é usada em algum ponto do programa, e 0 ela não é usada. O tipo armazena se a variável é inteira, real ou boolean, sendo que procedimentos não tem esse tipo. Valor armazena o valor da variável, e se ela não tiver nada, armazena uma string vazia. E por fim, Parâmetros armazena os tipos dos parâmetros de um procedimento (Ex: [“INT”, “REAL”, “INT”]), e se for uma variável, ou o procedimento não tiver parâmetros, armazena uma string vazia também.

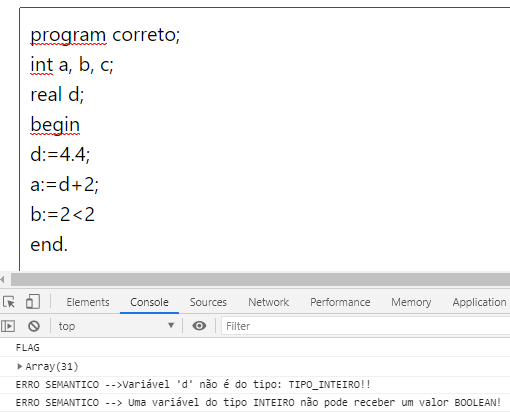
A seguir, algumas considerações particulares da semântica elaborada para este trabalho:

* Só pode-se realizar operações com variáveis de mesmo tipo (INTxINT, REALxREAL, BOOLEANxBOOLEAN, INTxNUM\_NAT...)
* Como não se pode declarar novas variáveis dentro de um comando ‘if’, ‘while’, ou de um comando composto, apenas PROCEDIMENTOS determinam um novo escopo no código.
* Se uma variável do tipo Boolean recebe um valor numérico (Ex: Sendo 'a' e 'b' numéricas, a+b, a+2, a, b, 2, etc...), não será acusado erro, pois este número será tido como TRUE, a menos que o valor inserido na variável seja '0 ou negativo', neste caso, será tido como FALSE.
* A análise semântica considera que o código está sintaticamente correto, pois não faz sentido analisar semanticamente algo que está sintaticamente bagunçado, por isso, erros sintáticos podem (e provavelmente vão) acarretar erros semânticos).
* Os códigos de teste devem ser copiados (ou digitados) até exatamente o ponto final que determina o final do código. Se houver algum espaço ou quebra de linha após o ponto final, ocorrerá um erro (este ponto vale tanto para a análise léxica, sintática ou semântica).
* O escopo definido por esta implementação não permite que variáveis com mesmo nome sejam declaradas em escopos diferentes, ou seja, uma variável local não pode ter o mesmo nome que uma global (ou de um escopo acima).
* A análise semântica é realizada em conjunto com a sintática, sendo que seus métodos são chamados pelos procedimentos sintáticos em meio à análise destes.
* Para visualizar os erros da análise semântica, é necessário abrir o console (apertar tecla F12 e depois clicar na aba “Console”).

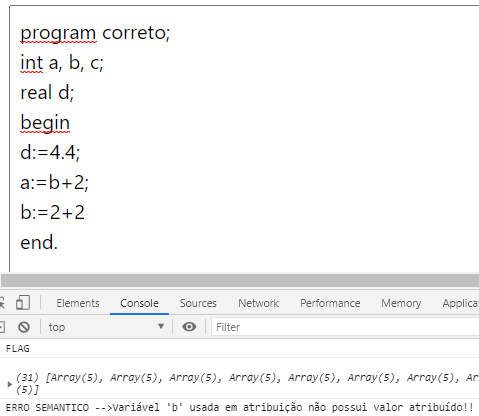
Casos de Teste

A seguir, os códigos que foram usados para testar o programa. Destes, todos obtiveram resultados corretos. (Todos estes códigos de teste estão em um arquivo na pasta do projeto (casos\_de\_teste2.txt) para serem usados).

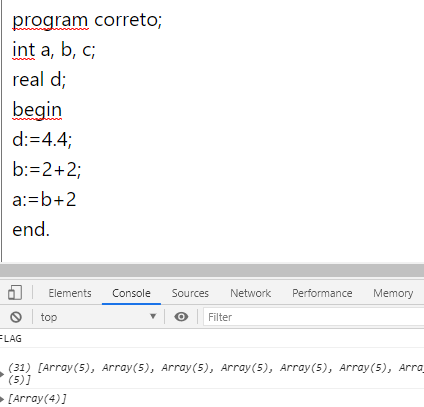
1. **(COM ERRO SEMÂNTICO) Neste, atribui-se à variável ‘a’ (INT) uma soma entre uma variável REAL e um número natural.**

****

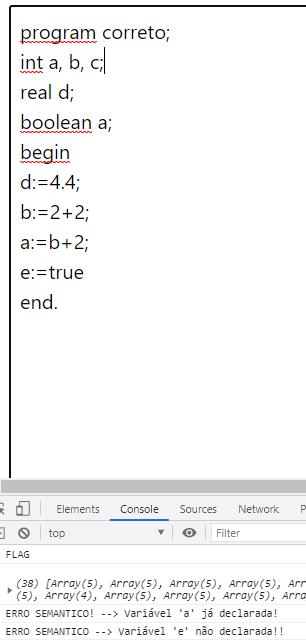
1. **(COM ERRO SEMÂNTICO) Neste, a variável ‘b‘ (INT) é usada na atribuição da variável ‘a’ (INT), porém, ‘b’ não possui valor.**

****

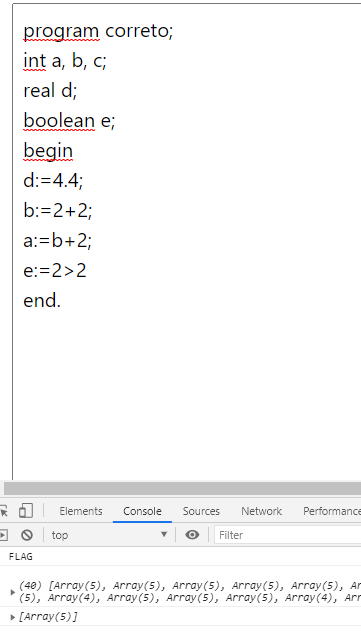
1. **(SEM ERROS SEMÂNTICOS)**

****

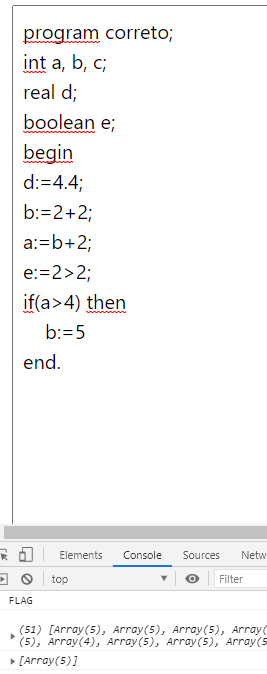
1. **(COM ERROS SEMÂNTICOS) Neste, tenta-se declarar novamente uma variável ‘a’, e tenta-se usar uma variável ‘e’ que não existe.**

****

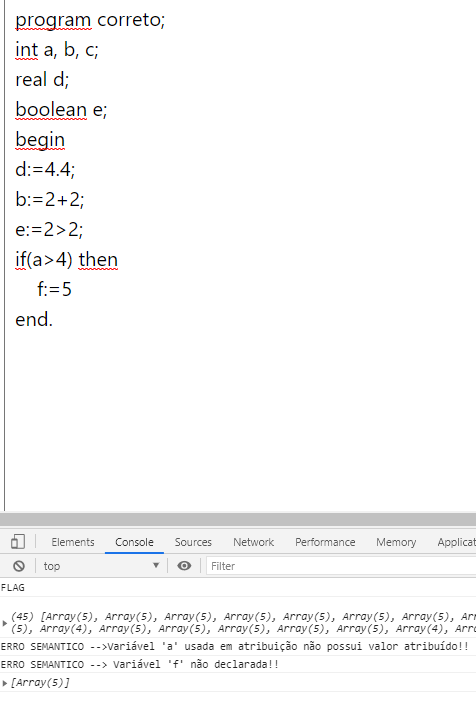
1. **(SEM ERROS SEMÂNTICOS) Acréscimo de boolean**

****

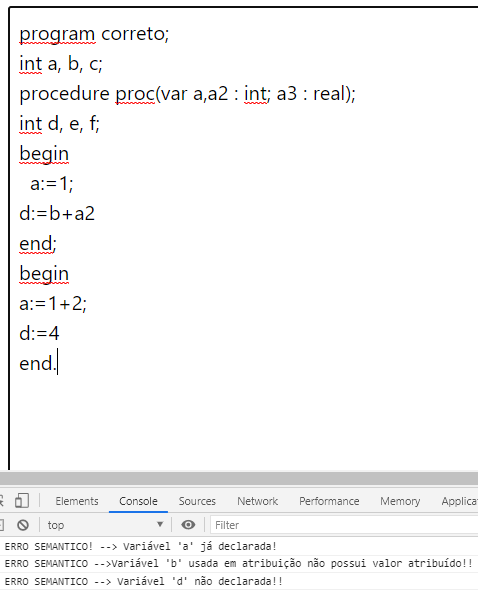
1. **(SEM ERROS SEMÂNTICOS) Acréscimo do ‘if’**

****

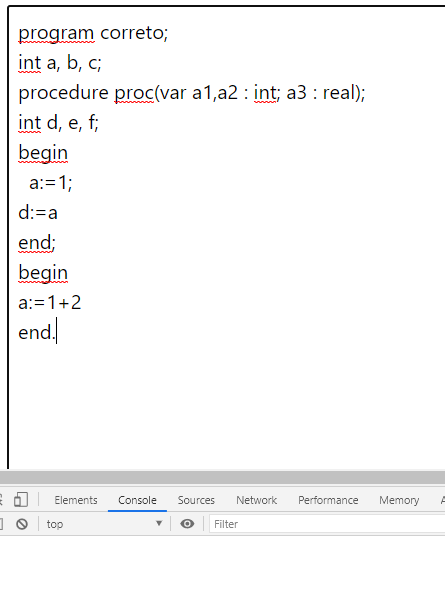
1. **(COM ERROS SEMÂNTICOS) Neste, o ‘if’ tenta usar uma variável ‘a’ que não tem valor, e dentro do ‘if’ tenta-se usar uma variável ‘f’ que não foi declarada.**

****

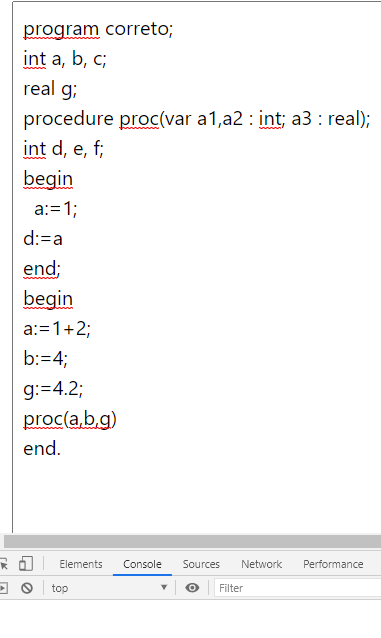
1. **(COM ERROS SEMÂNTICOS) Neste, um dos parâmetros declarados no procedimento tem o mesmo nome ‘a’ de uma variável já declarada. Além disso, dentro do procedimento tenta-se usar uma variável ‘b’ que não tem valor, e por fim, fora do procedimento, tenta-se usar uma variável ‘d’ que pertence ao escopo dele.**

****

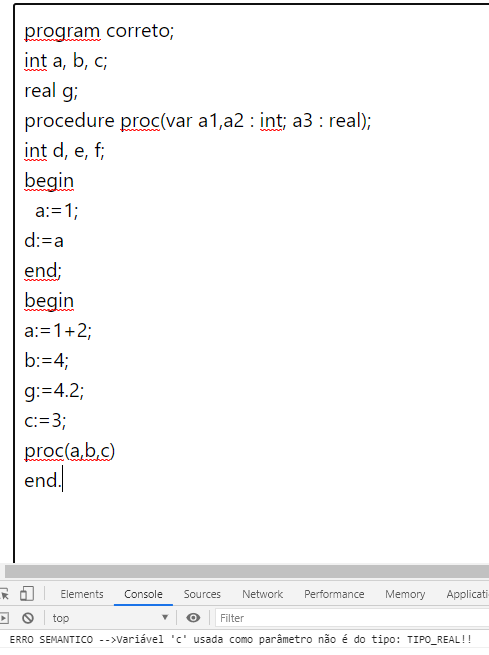
1. **(SEM ERROS SEMÂNTICOS) Acréscimo do procedimento**

****

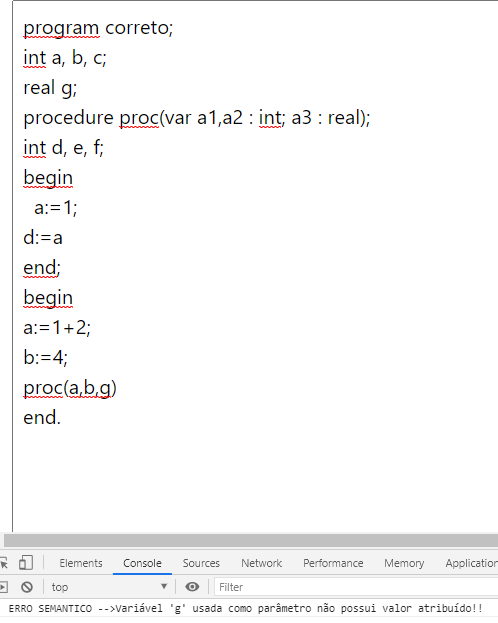
1. **(SEM ERROS SEMÂNTICOS) Acréscimo da chamada de procedimento**

****

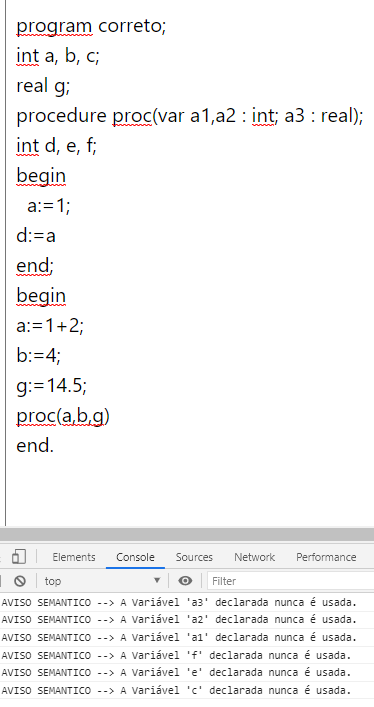
**11. (COM ERROS SEMÂNTICOS) Neste, tenta-se passar uma variável ‘c’ (INT) como parâmetro para o procedimento ‘proc’ quando ele espera que seu terceiro parâmetro seja REAL.**

****

**12.(COM ERROS SEMÂNTICOS) Neste, o terceiro parâmetro do procedimento ‘proc’ recebe uma variável REAL, porém, esta não tem valor.**

****

**13. (COM AVISOS SEMÂNTICOS) Neste, várias das variáveis declaradas, tanto no procedimento quanto fora dele, não são utilizadas.**

****

**OBS: Nos outros exemplos também existem variáveis que não foram utilizadas, porém, esta parte foi implementada depois, então elas não foram detectadas. Mas ao testar-se agora, funcionará normal.**