Departamento de computação e sistemas – DECSI

Professor: Gleiph Ghiotto Lima de Menezes

Disciplina: CSI476 – Fundamentos Teóricos da Computação

Trabalho de Prático

---

1. Informações gerais

Nessa seção serão apresentadas informações gerais de como será a entrega do trabalho e outras instruções que devem ser seguidas

1.1. Equipe de desenvolvimento

O trabalho será desenvolvido em grupo de até três pessoas. Em hipótese alguma será permitido uma equipe com quatro ou mais alunos.

1.2. Escolha da linguagem de programação

A linguem de programação é livre. Portanto, o aluno pode escolher a linguagem que tiver mais familiaridade. Vale ressaltar que as estruturas de dados deverão ser implementadas pelo grupo, exceto estruturas mais simples como: listas, filas, árvores e etc.

1.3. Artefatos a serem entregues

Os artefatos a serem entregues são:

- Código fonte
- Arquivo de construção (makefile, ant, maven, shell script, etc.)
- Documentação do trabalho em formato pdf contendo
    - Descrição da estrutura do projeto (classes, pacotes e métodos relevantes)
    - Descrição de como construir e executar o projeto
    - Descrição das estruturas de dados utilizadas na implementação
    - Descrição da divisão das tarefas por componentes do grupo

Antes de enviar o trabalho verifique:

- Se o código-fonte compila
- Se todos os arquivos fonte tem comentário com o nome e matrícula dos componentes do trabalho
- O arquivo de documentação tenha a identificação dos alunos
- O arquivo compactado tenha a identificação do(s) aluno(s)

1.4. Critério de avaliação

Esse trabalho terá uma avaliação com nota de 0 a 20. Sendo que a primeira fase terá peso 8 e a segunda peso 12. A avaliação considerará os seguintes aspectos:

- Organização dos arquivos em níveis físico e lógico

- Legibilidade do código, *i.e.*, nomes mnemônicos para variáveis, métodos e comentários úteis
- Entrevista

Trabalhos que não forem entregues até a data estipulada serão penalizados em um ponto por dia em atraso.

2. Especificação técnica

O trabalho consiste em implementar um analisador de *strings*, o qual divide a entrada em uma sequência de subpalavras. Para tal, o usuário definirá um conjunto de "*tags*" (marcadores) que serão usados pelo programa para dividir a *string* de entrada. Por exemplo, na Tabela 1 há um conjunto de marcadores e uma breve descrição da linguagem representada por cada uma delas. Supondo o texto de entrada "CSI476 =1000 /* Fundamentos Teóricos da Computação é nota 1000! */", o programa devera apresentar a seguinte resposta da divisão desse texto conforme as *tags* da Tabela 1:

VAR SPACE EQUALS SPACE INT SPACE COMMENT

Tags	**Linguagem que a *tag* descreve**
INT	Números inteiros
SPACE	Sequência de espaços em branco
VAR	Sequência de símbolos alfanuméricos com o primeiro símbolo sendo uma letra
EQUALS	O símbolo "="
COMMENT	Qualquer sequência contida entre os símbolos "*" e "*/"

Tabela 1. Exemplos de tags.

Tag emitida	**Texto correspondente**
VAR	CSI476
SPACE	␣
EQUALS	=
SPACE	␣
INT	1000
SPACE	␣
COMMENT	/* Fundamentos Teóricos da Computação é nota 1000! */

Tabela 2. Distribuição de *tags* para o exemplo anterior.

2.1. Linguagem de especificação de *tags*

As *tags* são definidas, uma por linha, seguindo o padrão:

NOME_TAG: DEFINICAO_TAG

Especifica-se o nome da *tag*, por exemplo VAR, seguindo por dois pontos e, após a definição da linguagem denotada pela *tag*. A definição da *tag* será feita usando expressões regulares em notação polonesa inversa, assim definidas:

1. $a \in \Sigma, \lambda\ e\ \emptyset$ são expressões regulares que denotam as linguagens $\{a\}, \{\lambda\}\ e\ \emptyset$;

2. Se e_1 e e_2 são expressões regulares, então e_1e_2+, $e_1e_2.$ e e_1 * são expressões regulares que denotam, respectivamente, a união das linguagens denotadas por e_1 e por e_2 , a concatenação da linguagem denotada por e_1 com a linguagem denotada por e_2 e o fecho de *Kleene* da linguagem denotada por e_1.

Por exemplo, a expressão regular em notação polonesa inversa ab.ba.+* corresponde a expressão regular (ab + ba) , que denota a linguagem {ab, ba}*. Assim, podemos especificar a *tag* INT, da Tabela 1, dessa forma:

INT: 01+2+3+4+5+6+7+8+9+01+2+3+4+5+6+7+8+9+*.

Ressalta-se que entre o nome da *tag* e o símbolo de dois pontos não há espaços. Entre o símbolo de dois pontos e a expressão regular que descreve a linguagem da *tag* há exatamente um espaço.

2.2. Execução do programa

O programa deverá ser executado pelo terminal (linha de comando) e, quando executado, entrara em modo de interpretação. Toda interação entre o usuário e o programa dar-se-á no modo de interpretação, no qual o usuário poderá: especificar as *tags* conforme definido na Seção 2.1; executar o comando para classificar *strings* de entrada conforme as tags informadas; e executar demais comandos, como para sair do programa, dentre outros. A Figura 1 mostra um exemplo de funcionamento do programa. Durante a execução do programa, o usuário definiu as tags INT, EQUALS e VAR, uma por linha. Em seguida usou o comando “:p” para realizar a classificação da entrada informada na mesma linha “x=1230". Após esse comando, o programa emitiu na tela a divisão da entrada, tendo como saída a linha “VAR EQUALS INT”. Na sequência, o usuário digitou o comando “:q" para encerrar o programa.

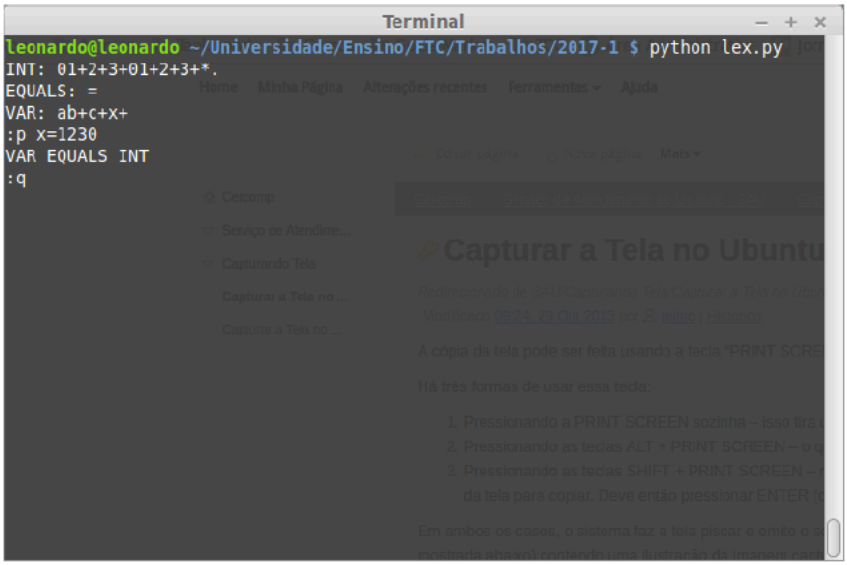

Figura 1. Exemplo de execução do programa.

Em suma, o modo de interpretação aguarda definições de *tags* ou comandos. Os comandos começam com o símbolo de dois pontos seguido por um caractere e é sucedido pelo seu parâmetro, caso exista, separado por um espaço. A relação de comandos, descrição e exemplo de uso estão descritos na Tabela 3. O comando ":f" é utilizado para realizar a divisão do texto contido em um arquivo. O funcionamento desse comando é semelhante ao do comando ":p". A diferença entre ambos é que no comando ":p" o parâmetro é a entrada a ser dividida, enquanto que no comando ":f" a entrada esta contida em um arquivo, cujo caminho é dado como parâmetro. O comando ":l" é utilizado para carregar definições de tags contidas em um arquivo texto. O parâmetro desse comando é o caminho do arquivo contendo as definições de *tags*. As definições de *tags* seguem a mesma sintaxe do modo de interpretação, com uma definição por linha. Finalmente, ":t" é o comando utilizado para listar as tags válidas definidas.

Comando	Descrição	Exemplo
:f	Realiza a divisão da *string* do arquivo informado	:f input.txt
:l	Carrega um arquivo com a definição de *tags*	:l tags.lex
:o	Especifica o caminho do arquivo de saída para a divisão de *tags*	:o output.txt
:p	Realiza a divisão de *tags* da entrada informada	:p x=1230
:q	Sai do programa	:q
:s	Salva as *tags* no arquivo especificado	:s arquivo.txt
:t	Lista as *tags* válidas definidas	:t

Tabela 3. Descrição dos comandos do programa.

A sequência das *tags* referente a divisão do texto de entrada é escrita na saída padrão (terminal), separadas por espaço. No entanto é possível alterar onde a saída será escrita com o comando ":o", passando como parâmetro o caminho do arquivo no qual a saída devera ser escrita. O programa é encerrado após executar o comando ":q". Para salvar as *tags* no escopo da execução do programa é utilizado o comando ":s", cujo parâmetro é o caminho do arquivo de texto no qual as definições serão escritas.

2.3. Emissão de avisos, erros e informações ao usuário

Mensagens de erros, avisos e informações devem ser emitidas pelo programa. O início de cada mensagem terá um marcador [INFO], [ERROR] ou [WARNING] indicando se é uma informação, uma mensagem de erro ou mensagem de aviso, respectivamente. As mensagens de aviso serão emitidas após comandos que não têm saída para informar ao usuário que foi corretamente executado. Por exemplo, apos executar o comando ":l" o programa emitira na saída padrão a mensagem "[INFO] As definições de *tags* foram carregadas".

As situações de erros são diversas, desde erros na entrada dos dados a erros na execução do programa. Destaca-se algumas situações que devem ser tratadas: erros na definição da *tag*; erros no comando; e erros na divisão da entrada. Deve-se verificar se há erros na especificação das *tags*, o qual pode ocorrer na definição do nome da *tag* ou com uma expressão regular

malformada (errada). Erros nos comandos podem ocorrer por falta de parâmetros, parâmetros errôneos ou comandos inválidos. Erro na classificação acontecera se não existir nenhuma tag para classificar alguma porção da entrada. Portanto, a divisão somente terminara com sucesso caso toda a entrada tenha sido dividida (classificada) com alguma *tag*.

O conjunto das palavras descritas por duas *tags* podem ter interseção. Por exemplos, suponha as definições das tags PAR e AB:

AB: a*b*
PAR: ab.ba.+aa.+bb.+*

Note que a entrada “aabbbbaa”, por exemplo, tem prefixo que pertence ao conjunto descrito por ambas as *tags*. O programa deverá identificar tais situações e emitir um aviso na saída padrão. Para esse exemplo, em particular, o programa devera emitir a mensagem “[WARNING] Sobreposição na definição das *tags* PAR e AB". Nessa situação de sobreposição, quando a classificação da entrada for realizada, o programa dará preferência para a *tag* que foi definida primeiro. Assim, nesse exemplo, a saída será “AB AB", sendo que o prefixo “aabbbb” casa com a primeira emissão da *tag* AB e o sufixo “aa” com a segunda emissão.

2.4. Alfabeto de entrada e caracteres especiais

O alfabeto da linguagem de expressão regular e o conjunto de caracteres especificado pela tabela ASCII (códigos 32 ao 126 da tabela ASCII)[2], exceto os caracteres de controle, e os caracteres especiais para quebra de linha. A fim de definir linguagens (ou *tags*) com caracteres especiais, que usam os meta-símbolos da linguagem ou a cadeia vazia (λ), serão usados símbolos de escape. Os símbolos de escape são especificados usando o símbolo \ seguido por um caractere. A tabela 4 apresenta todos os códigos de escape que devem ser tratados na definição das expressões regulares.

Código de escape	**Caractere representado**	**Representação decimal do código ASCII**
\n	Quebra de linha	10
\\	\	92
*	*	42
\.	.	46
\+	+	43
\l	λ	955

Tabela 4. Código de escape para a linguagem de expressões regulares.

3. Entrega do trabalho

A entrega do trabalho será dividida em duas etapas: na primeira etapa será feita uma entrega parcial e, na segunda, o trabalho completo. Os artefatos que devem ser entregues e a forma de avaliação em ambas as etapas estão detalhadas nas Seções 1.3 e 1.4. A primeira entrega tem por finalidade avaliar o caminho que está sendo seguido no desenvolvimento do trabalho e dar

orientações para o melhor andamento do mesmo. Portanto, nessa etapa devera estar pronto as seguintes funcionalidades:

- Interface de interação com o usuário: essa interface deve facilitar a utilização do usuário e indicar as funcionalidades que ainda não foram implementadas para o usuário;
- Módulo de leitura e estrita de arquivos ou saída padrão para cada uma das funcionalidades;
- Módulo de análise das definições de *tags*: aceitando as tags, quando são definidas corretamente; e rejeitando, caso contrário;
- Emissão de avisos e erros em comandos e nas definições de *tags*;
- Projeto da estrutura de dados a serem usadas na solução do problema.

É válido ressaltar que todas as decisões de projeto devem estar documentadas no relatório.

A divisão da entrada em *tags*, os erros na classificação e aviso de sobreposição de *tags* não precisam ser entregues nessa etapa. Assim, a funcionalidade principal (divisão da entrada conforme as *tags*), seus respectivos erros e avisos deverão estar prontos para serem entregues na segunda etapa do trabalho. Todo o software com as funcionalidades corretamente implementadas será novamente avaliado na segunda etapa. Portanto, a segunda etapa consiste na entrega final do trabalho.

A entrega será realizada via Moodle. **Sendo o limite de entrega da primeira etapa o dia 02/12/2017 e 28/01/2018 o limite de entrega da segunda etapa.** É de responsabilidade do aluno conferir se o trabalho foi corretamente enviado. Portanto, após enviar o trabalho, verifique o anexo no Moodle, baixe-o, compile o trabalho e tente executá-lo.