[MAC0211] Laboratório de Programação I
Aula 26
Revisão para a Prova 2

Kelly Rosa Braghetto

DCC–IME–USP

18 de junho de 2013

Revisão para a prova 2 (dia 20/06)

Matéria da prova

- Introdução à análise léxica
- Processamento de macros com o CPP e o M4
- Expressões regulares
- Processamento de textos com o Awk
- Geração de analisadores léxicos com o Flex
- Geração de analisadores sintáticos com o Bison
- A arte e a ciência da depuração

Introdução à análise léxica (aulas 15 e 16)

Conhecimentos esperados

- Etapas envolvidas em um proceso de compilação (análises léxica, sintática e semântica)
- Notações prefixa, infixa e pós-fixa
- Implementação de uma calculadora que usa notação pós-fixa ou infixa

Processamento de macros (aula 17)

Conhecimentos esperados

- Pré-processadores
- Macros (definição e uso)
- Processamento de macros na linguagem C (com o CPP)
- Processamento de macros com o M4

Expressões regulares (aulas 18 – 19)

Conhecimentos esperados

- Expressões *glob*
- Epressões regulares básicas
- **Expressões regulares estendidas**

Processamento de textos com o Awk (aulas 19–20)

Conhecimentos esperados

- Funcionalidades do Awk
- Formato de um *script* em Awk
- Como o Awk processa um texto de entrada
- Prático: Manipulação de arquivos texto usando o Awk

Geradores de analisadores léxicos – Flex (aulas 20–22)

Conhecimentos esperados

- Propósito de uma analisador léxico
- Propósito de um gerador de analisador léxico
- Estrutura de um arquivo de entrada do Flex
- Definição de regras léxicas no Flex
- Prático: Implementação de analisadores léxicos usando Flex

Geradores de analisadores sintáticos – Bison (aula 22)

Conhecimentos esperados

- Propósito de um analisador sintático
- Propósito de um gerador de analisador sintático
- Estrutura de um arquivo de entrada do Bison
- Definição de regras gramaticais no Bison
- Prático: Implementação de analisadores gramaticais usando o Bison

Sistemas de controle de versão (aula 23)

Conhecimentos esperados

- Propósitos desses sistemas
- Principais características
- Arquitetura centralizada $\times$ distribuída

A arte e a ciêncida da depuração (aula 24)

Conhecimentos esperados

- Fatores que levam a erros de programação
- Técnicas para garantir a integridade de software
- "Dicas" para se evitar os erros
- "Dicas" para a depuração (= identificação de erros)

O que mais foi visto em aula...

... mas **não** cai na prova

- Autoconf (aula 18)
- Introdução ao Git (aulas 23 e 24)
- Testes – parte inicial (aula 25)

Alguns exercícios

(Provenientes de provas de anos anteriores dos profs Gubi e Fabio)

Tema: Análise Léxica (notações e calculadoras)

1. Qual a relação entre o número de chamadas das funções `expr`, `termo` e `fator` na calculadora infixa (recursiva)?
2. Como implementar na calculadora infixa recursiva o operador $\wedge$ (potência)? A precedência desse operador é maior que a dos operadores de divisão e multiplicação.
3. Explique quais alterações você faria na calculadora infixa (recursiva) para implementar os operadores $>>$ e $<<$ (*bit-shift-right* e *bit-shift-left*) com precedência maior que qualquer outro operador.

Alguns exercícios

(Provenientes de provas de anos anteriores dos profs Gubi e Fabio)

Tema: Processamento de Macros

1. Discuta as vantagens e desvantagens de se definir funções em C como macros.

Alguns exercícios

(Provenientes de provas de anos anteriores dos profs Gubi e Fabio)

Tema: Expressões Regulares

1. Escreva uma expressão regular (usando a sintaxe das expressões estendidas) que reconheça:
   - 1.1 cadeias que possuam somente letras maiúsculas e minúsculas alternadas.
   - 1.2 cadeias que tenha todas as vogais pelo menos uma vez e em ordem.
   - 1.3 todos os números inteiros positivos menores que 3000.
   - 1.4 todos os números hexadecimais que podem ser representados com um byte.
   - 1.5 exatamente 100 cadeias diferentes.
   - 1.6 exatamente 101 cadeias diferentes.

Alguns exercícios

(Provenientes de provas de anos anteriores dos profs Gubi e Fabio)

Tema: Awk

1. Escreva um script em `awk` que lê um arquivo da entrada padrão e gera uma tabela de todas as palavras de comprimento maior do que 3 caracteres, onde cada linha contém a palavra na primeira posição, seguida por um ':' seguido pelos números das linhas onde a palavra ocorre. Por palavra, entenda uma sequência de letras. Depois, responda as perguntas abaixo (justificando suas respostas):
   1.1 As linhas aparecerão em ordem?
   1.2 As palavras aparecerão em ordem?
2. Para cada uma das chamadas do `awk` abaixo, explique o que ela faz.

```
2.1 awk 'length($0) > 80'
2.2 awk '{ if (NF > 0) print }'
2.3 awk '{ printf "%d:  %s\n", NR, $0 }'
2.4 awk '/banana/ { print $2 $1 }'
```

Alguns exercícios

(Provenientes de provas de anos anteriores dos profs Gubi e Fabio)

Tema: Flex

1. Escreva uma especificação em `flex` para um filtro que copia um texto da entrada para a saída, trocando:
   - todas as sequências consecutivas de espaços e TABs por um único espaço,
   - todos os espaços em branco nos finais das linhas por nada (remoção) e
   - todas as ocorrências da palavra usuário pelo nome do usuário devolvido pela função `char* getlogin()`. Para usar essa função, é preciso incluir o arquivo de cabeçalho `unistd.h`.
2. Escreva uma especificação em `flex` para um filtro que copia um texto, com a diferença que todos os números contidos na entrada são apresentados na saída divididos por 1000.

Alguns exercícios

Tema: Bison

1. Escreva um *scanner* em `flex` + um *parser* em `bison` para processar um arquivo contendo dados de ex-alunos do IME que possui o seguinte formato esperado:

```
MAC     3132425     Ana Maria Machado              MAC123-4 MAT325-6 MAC211-4
MAT     2128902     Carlos Drummond de Andrade     MAT325-4
MAE     2871218     Cecilia Meirelles              MAE565-6 MAC122-4 MAC211-4 MAE258-6
MAE     2329803     Fernando Pessoa                MAC211-4 MAE258-6
MAC     2389212     Guimaraes Rosa                 MAC122-4 MAC211-4 MAE565-6 MAT232-2
MAC     2328372     Vinicius de Moraes             MAE454-6 MAT232-2
```

   Nesse arquivo, cada linha contém os dados de um aluno. Os campos estão separados por uma ou mais tabulações. O primeiro campo contém a sigla do departamento do aluno (que pode ser MAC, MAT ou MAE); o segundo contém o número USP do aluno; o terceiro, o nome; e o quarto é a lista dos códigos das disciplinas cursadas pelo aluno (separados por espaços). O código de uma disciplina é composto pela sigla do departamento que a ofereceu, seu número e, separado por um '-', o número de créditos da disciplina (que está entre 1 e 9). O seu parser deve mostrar, para cada aluno, o total de créditos cursados. E, no final do processamento do arquivo, deve mostrar o número de alunos contabilizados em cada um dos 3 departamentos.

Alguns exercícios

(Provenientes de provas de anos anteriores dos profs Gubi e Fabio)

Tema: Depuração

1. O que devemos fazer para depurar um programa que apresenta um determinado erro (*bug*) só de vez em quando, indeterministicamente? Dê um exemplo.
2. Escreva um programa (curto) que tenha um *bug* tal que ele tenha um comportamento diferente cada vez que for rodado. Justifique.