Universidade Federal do Espírito Santo
Departamento de Informática
Estruturas de Dados 2017/1
Profa. Claudine Badue
Trabalho 2

Indexação e Busca

1. Objetivo

O objetivo deste trabalho é projetar e implementar um sistema de programas para indexação e busca em arquivos de texto.

2. Problema

Parte 1 : Indexação [Ziviani, 2011; Capítulo 5, Exercício 19]

Várias aplicações necessitam de um relatório de referências cruzadas. Por exemplo, a maioria dos livros apresenta um índice remissivo, que corresponde a uma lista alfabética de palavras-chave ou palavras relevantes do texto com a indicação dos locais no texto onde cada palavra-chave ocorre. Na verdade, o índice remissivo é um **arquivo invertido**.

Como exemplo, suponha um arquivo contendo um texto constituído por:

Linha 1: Good programming is not learned from
Linha 2: generalities, but by seeing how significant
Linha 3: programs can be made clean, easy to
Linha 4: read, easy to maintain and modify,
Linha 5: human-engineered, efficient, and reliable,
Linha 6: by the application of common sense and
Linha 7: by the use of good programming practices.

Assumindo que o índice remissivo seja constituído das palavras-chave:

programming, programs, easy, by, human-engineered, and, be, to,

o programa para criação do índice deve produzir a seguinte saída:

and	4 5 6
be	3
by	2 6 7
easy	3 4
human-engineered	5
programming	1 7
programs	3
to	3 4

Note que a lista de palavras-chave está em ordem alfabética. Adjacente a cada palavra está uma lista de números de linhas, um para cada vez que a palavra ocorre no texto. Uma estrutura de dados desse tipo é conhecida como **arquivo invertido**. As listas do arquivo invertido são conhecidas como **listas invertidas**. O arquivo invertido é um mecanismo muito utilizado em arquivos constituídos de texto, como as **máquinas de busca na Web**.

Projete um sistema para produzir um índice remissivo. O sistema deverá ler um número arbitrário de palavras-chave, que deverão constituir o índice remissivo, seguido da leitura de um texto de tamanho arbitrário, que deverá ser esquadrinhado à procura de palavras que pertençam ao índice remissivo. Para extrair as palavras de um texto, utilize o procedimento mostrado no Programa 1.

Programa 1: Procedimento para extrair palavras de um texto.

```
/* Informar nomes dos arquivos teste.txt e alfabeto.txt na linha de
comando */
/* Exemplo: a.out teste.txt alfabeto.txt
*/

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#define MAXALFABETO 255
#define TRUE  1
#define FALSE 0

typedef short  TipoAlfabeto[MAXALFABETO + 1];
FILE *ArqTxt, *ArqAlf;
TipoAlfabeto Alfabeto;
char Palavra[256];
char Linha[256];
int i;
short aux;

void DefineAlfabeto(short *Alfabeto)
{ char Simbolos[MAXALFABETO + 1];
  int i, CompSimbolos;
  char *TEMP;
  for (i = 0; i <= MAXALFABETO; i++)
    Alfabeto[i] = FALSE;
  fgets(Simbolos, MAXALFABETO + 1, ArqAlf);
  TEMP = strchr(Simbolos, '\n');
  if (TEMP != NULL) *TEMP = 0;
  CompSimbolos = strlen(Simbolos);
  for (i = 0; i < CompSimbolos; i++)
    Alfabeto[Simbolos[i]+127] = TRUE;
  Alfabeto[0] = FALSE;   /* caractere de codigo zero: separador */
}

int main(int argc, char *argv[])
{ ArqTxt = fopen(argv[1], "r");
  ArqAlf = fopen(argv[2], "r");
  DefineAlfabeto(Alfabeto);   /* Le alfabeto definido em arquivo */
  aux = FALSE;
  while (fgets(Linha, 256, ArqTxt) != NULL)
    { for (i = 1; i <= strlen(Linha); i++)
        { if (Alfabeto[Linha[i-1]+127])
            { sprintf(Palavra + strlen(Palavra), "%c", Linha[i-1]);
              aux = TRUE;
            }
          else
          if (aux)
            { puts(Palavra);
              *Palavra = '\0';
              aux = FALSE;
            }
        }
    }
  if (aux)
  { puts(Palavra);
    *Palavra = '\0';
  }
  fclose(ArqTxt);
```

```
  fclose(ArqAlf);
  return 0;
}
/* End. */
```

Para implementar o índice, você deve utilizar duas estruturas de dados distintas:

a) Árvore binária de pesquisa;
b) Tabela *hash* com endereçamento aberto e *hashing* linear para resolver **colisões**.

Observe que, apesar de o *hashing* ser mais eficiente do que árvores de pesquisa, existe uma desvantagem na sua utilização: após atualizado todo o índice remissivo, é necessário imprimir suas palavras em ordem alfabética. Isso é imediato em árvores de pesquisa, mas, quando se usa *hashing*, isso é problemático, sendo necessário ordenar a tabela *hash* que contém o índice remissivo.

Para testar seu programa, utilize o exemplo anterior. Utilize também o texto e as palavras-chave disponíveis nos arquivos http://www.inf.ufes.br/~claudine/Cursos/2017_1_Estruturas_Dados_II/Trabalho_2/Texto.txt e http://www.inf.ufes.br/~claudine/Cursos/2017_1_Estruturas_Dados_II/Trabalho_2/Palavras_Chave.txt, respectivamente.

Parte 2 : Busca

Projete também um sistema para buscar no índice remissivo. O sistema deverá ler uma consulta (conjunto de palavra-chave do teclado separadas por espaço em branco), buscar as palavras-chave da consulta no índice remissivo, recuperar as listas invertidas das palavras-chave, fazer a interseção das listas invertidas e imprimir o conteúdo de cada linha resultante da interseção das listas.

Para testar seu programa, utilize as seguintes consultas: “Computer Science”, “Computer scientists”, “computers”, “computer systems”,” software packages”.

3. Instruções de Desenvolvimento

a) Este trabalho deverá ser desenvolvido em grupos de dois alunos.
b) O programa deste trabalho deverá ser implementado usando a linguagem C.
c) O programa deverá também ser modularizado. Crie arquivos de extensão .c e .h para cada módulo do programa. Crie arquivos exclusivos para definir tipos abstratos de dados.
d) O programa deverá também ser compilado e executado com o aplicativo make e as regras all e run. O programa deverá ser compilado com o comando make all e o teste executado com o comando make run.

4. Instruções de Entrega

a) Para credenciar o representante do grupo no sistema de submissão de trabalhos, envie um e-mail como o abaixo:

   Para: estruturas.dados.inf.ufes@gmail.com
   Assunto: [SUBSCRIBE] <nome>

   Substitua <nome> pelo nome completo do representante do grupo. Não use letras latinas (é, í, ç, etc.) no nome. Por exemplo:

   Para: estruturas.dados.inf.ufes@gmail.com
   Assunto: [SUBSCRIBE] Claudine Santos Badue Goncalves

b) Para credenciar o grupo, envie um e-mail como o abaixo:

   Para: estruturas.dados.inf.ufes@gmail.com
   Assunto: [WORKING GROUP] <nome 1>:<nome 2>

Substitua <nome 1> pelo nome completo do primeiro membro do grupo, e <nome 2> pelo nome completo do segundo membro do grupo. Por exemplo:

Para: estruturas.dados.inf.ufes@gmail.com
Assunto: [WORKING GROUP] Claudine Santos Badue Goncalves:Sergio Paulo Tavares Goncalves

c) Para enviar o trabalho, envie um e-mail como o abaixo até às 23:59 horas da data limite de entrega:

Para: estruturas.dados.inf.ufes@gmail.com
Assunto: [SUBMIT] <codigo da atividade>

Substitua <codigo da atividade> por EDIIEC171T2P1 e EDIIEC171T2P2 para as Partes 1 e 2, respectivamente. Por exemplo, para a Parte 1:

Para: estruturas.dados.inf.ufes@gmail.com
Assunto: [SUBMIT] EDIIEC171T2P1

Anexe ao e-mail: (i) o arquivo Makefile; (ii) os arquivos *.c e *.h do código; (iii) os arquivos de entrada; e (iv) o arquivo PDF do relatório.

Não compacte os arquivos, anexe-os diretamente ao e-mail!

d) A fórmula para desconto por atraso na entrega do trabalho é:

$$\frac{2^{d-1}}{0{,}32}\%$$

onde d é o atraso em dias. Note que após cinco dias, o trabalho não poderá ser mais entregue.

e) Se você enviar o trabalho múltiplas vezes, apenas a última versão enviada será considerada, inclusive para efeito de desconto por atraso.

5. Maiores Detalhes

Maiores detalhes serão discutidos em sala de aula. Considerações feitas em sala terão valor superior ao daquelas contidas nesta especificação.

Referência

T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Algoritmos: Teoria e Prática*. Rio de Janeiro: Elsevier, Campus, 2002.

N. Ziviani. *Projeto de Algoritmos: com Implementações em PASCAL e C*. 3a. edição revista e ampliada. São Paulo: CENGAGE Learning, 2011.