

Universidade de Brasília

Departamento de Ciência da Computação



Lista de Exercício 6 - OA

Autores:

Gabriel Bessa 16/0120811

Thiago Veras 16/0146682

Disciplina:

Organização de Arquivos

Turma:

A

Professor:

Oscar Gaidos

Brasília
5 de Junho de 2018

Exercícios:

1- O que é uma lista invertida? Quais são suas vantagens?

Listas invertidas é uma das maneiras que se têm de organizar um arquivo, mapeando os termos às ocorrências, em torno de um conjunto de documentos ou em um documento. De tal forma que cada registro contém uma chave secundária e um ponteiro para a lista de referência.

Vantagens:

- Há um rearranjo dos arquivos de índices secundários quando um novo compositor é adicionado ou o nome de um compositor é modificado.
- Adicionar ou excluir gravações para um compositor somente afeta o arquivo contendo a lista.
- Reutilização do espaço dos registros excluídos do arquivo contendo a lista, pois eles têm tamanho fixo.

Desvantagens:

- Chaves primárias associadas a uma certa chave secundária não estão adjacentes fisicamente no disco, sendo necessário vários seeks para recuperar a lista.
- Os labels das gravações que tem a mesma chave secundária não são contíguos no arquivo da lista.
- Talvez algumas das referências apontadas pela lista de referência não são mais válidas.

2- Por que se usa chaves secundárias?

O uso das chaves secundárias é normalmente atrelado para recuperar um conjunto especial de registros em um arquivo de dados. Sua utilização melhora o acesso às informações buscadas.

3- Por que é possível eliminar um registro apenas do índice primário, e não do secundário?

É possível eliminar um registro apenas do índice primário porque o índice secundário é apenas uma referência ao primário, resultando no armazenamento de mais chaves primárias. Por exemplo, numa biblioteca temos vários

livros, os quais podem possuir ou não o mesmo assunto. Dado tal informação, é possível remover apenas o arquivo do índice primário e deixar sua referência no índice secundário, sem que se faça necessário a atualização de todos os índices ligados a uma chave primária. Um problema que pode se enfrentar em tal situação, são as referências inválidas no arquivo de índice secundário.

4- Faça um programa que faça os seguintes procedimentos:

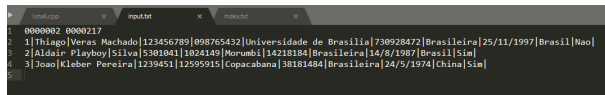
- Peça para o usuário preencher um arquivo o qual vai conter no mínimo 20 registros e cada registro deve ser composto no mínimo de 10 campos.
- A partir desse arquivo criado e usando dois desses campos crie a chave primaria.
- Crie um índice simples com essa chave primaria.
- Faça a busca de um registro nesse arquivo de índices simples.

Algoritmo:

Para o algoritmo, nós decidimos fazer o programa utilizando como registro as características convencionais do computador. Tanto os campos como os registros são separados por caracteres especiais. Para o campo nós utilizamos o caractere: '|'. Já para os registros, nós utilizamos o quebra de linha convencional: '\n'.

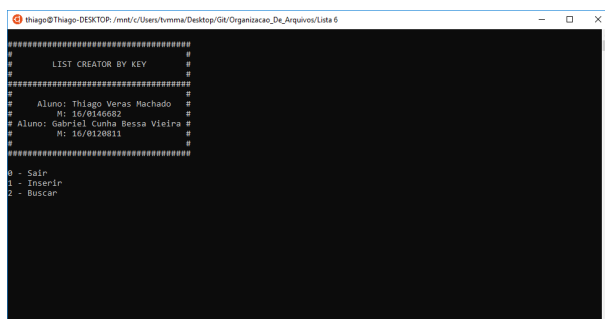
A cada inserção nova no arquivo, os registros(Figura 1), são atualizados adicionando quem foi inserido, posteriormente à quem já estava inserido. Caso o usuário forneça um id não existente, ele retorna que o id não é válido(Figura 5), já que ele não se encontra nos registros.

Outputs:



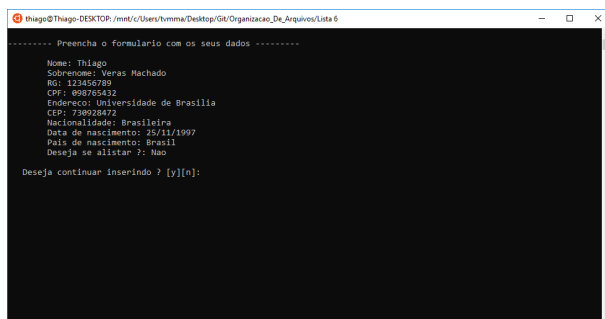
```
00000002 0000217
1|Thiago|Veras Machado|123456789|098765432|Universidade de Brasilia|730928472|Brasileira|25/11/1997|Brasil|Nao|
2|Aldair Playboy|Silva|5301841|1024149|Morumbi|14218184|Brasileira|14/8/1987|Brasil|Sim|
3|Joao|Kleber Pereira|1239451|12595915|Capatzena|38181484|Brasileira|24/5/1974|China|Sim|
```

Figura 1: Arquivo 'input.txt' que armazena a chave primária e suas informações



```
thiago@Thiago-DESKTOP: /mnt/c/Users/hmma/Desktop/Git/Organizacao_De_Arquivos/Lista 6
#####
# LIST CREATOR BY KEY #
#####
# Aluno: Thiago Veras Machado #
# R: 123456789 #
# Aluno: Gabriel Cunha Bessa Vieira #
# R: 10/0148811 #
#####
0 - Sair
1 - Inserir
2 - Buscar
```

Figura 2: Menu inicial do programa para a escolha da ação desejada



```
thiago@Thiago-DESKTOP: /mnt/c/Users/hmma/Desktop/Git/Organizacao_De_Arquivos/Lista 6
----- Preencha o formulario com os seus dados -----
Nome: Thiago
Sobrenome: Veras Machado
RG: 123456789
CPF: 098765432
Endereco: Universidade de Brasilia
CEP: 730928472
Nacionalidade: Brasileira
Data de nascimento: 25/11/1997
Pais de nascimento: Brasil
Deseja se alistar?: Nao
Deseja continuar inserindo? [y][n]:
```

Figura 3: Cadastrando uma nova pessoa na próxima posição nos registros .

```

@thiago@Thiago-DESKTOP: /mnt/c/Users/hmma/Desktop/Git/Organizacao_De_Arquivos/Lista 6
Digite o id do registro para realizar a busca: 3

-----
Dados do id 3
Nome : Joao
Sobrenome : Kleber Pereira
RG : 1238451
CPF : 12395915
Endereco : Copacabana
CEP : 38181484
Nacionalidade : Brasileira
Data de nascimento : 24/05/1974
País de nascimento : China
Deseja se alistar ? : Sim
-----

Deseja continuar buscando ? [y][n]:

```

Figura 4: Buscando o id fornecido dentro dos registros.

```

@thiago@Thiago-DESKTOP: /mnt/c/Users/hmma/Desktop/Git/Organizacao_De_Arquivos/Lista 6
Digite o id do registro para realizar a busca: 64

-----
Dados do id 64

ID NOT FOUND
-----

Deseja continuar buscando ? [y][n]:

```

Figura 5: Caso o id fornecido não exista.

```

1 00000002 00000027
2 1 16
3 2 128
4 3 217
5

```

Figura 6: Arquivo com os respectivos índices e offset de cada registro.

```

1 #include <bits/stdc++.h>
2
3 #ifdef WIN32
4     #define CLEAR "cls"
5 #else
6     #define CLEAR "clear"
7 #endif
8
9 using namespace std;
10
11 // Vetor com as perguntas a serem feitas
12 vector<string> questions = {"Nome", "Sobrenome", "RG", "
    CPF", "Endereco", "CEP", "Nacionalidade", "Data de

```

```

    nascimento", "País de nascimento", "Deseja se alistar
    ?"};

13
14 // Arquivo de registros
15 fstream arquivo("input.txt", ios::in | ios::out);
16
17 // Arquivo de índices
18 fstream indice("index.txt");
19
20 // Variável para pegar o offset de cada id no arquivo de
    registro
21 map<int,int> desloc;
22
23 // Variável que possui a quantidade de registros e o
    offset aonde está o final do arquivo de registros
24 int header, offset;
25
26 // Print menu on terminal
27 void intro(){
28     printf("\n");
29     printf("#####\n");
30     printf("#                               #\n");
31     printf("#           LIST CREATOR BY KEY           #\n");
32     printf("#                               #\n");
33     printf("#####\n");
34     printf("#                               #\n");
35     printf("#           Aluno: Thiago Veras Machado           #\n");
36     printf("#           M: 16/0146682                         #\n");
37     printf("# Aluno: Gabriel Cunha Bessa Vieira #\n");
38     printf("#           M: 16/0120811                         #\n");
39     printf("#                               #\n");
40     printf("#####\n");
41     printf("\n");
42 }
43
44 // Limpa o buffer do cin
45 void clear(){
46     cin.clear(); cin.ignore(INT_MAX, '\n');
47 }
48
49 // Le os dados para salvar no arquivo de registros
50 void read(){
51

```

```

52 // String para pegar os dados
53     string data;
54
55 // Iteracao com o usuario
56 cout << "\n----- Preencha o formulario com os seus
    dados ----- \n" << endl;
57
58 // Salva o registro e seu offset no mapa
59     desloc[header] = offset;
60
61 // Armazena o registro como id no arquivo
62     arquivo << header << "|";
63
64 // Caminha pelas perguntas
65     for(auto question : questions){
66
67         // Iteracao com o usuario
68             cout << "\t" << question << ": ";
69
70         // Le a resposta para a respectiva pergunta
71             getline(cin, data) ;
72
73         // Salva o dado no arquivo de registro
74             arquivo << data << "|";
75     }
76
77 // Incrementa a quantidade de registro
78     header++;
79
80 // Adiciona uma quebra de linha
81     arquivo << '\n';
82
83 // Atualiza o offset com a ultima posicao do arquivo
84     offset = arquivo.tellg();
85
86 }
87
88 // Funcao para pegar a quantidade de registros e offset
    do arquivo passado como parametro
89 pair<int,int> get_file_header(fstream &file){
90
91     // Variavel auxiliar para pegar a posicao que estava o
        arquivo

```

```

92     int aux = file.tellg();
93
94     // Variavel de retorno da quantidade de registro e de
        offset
95     int idx,off;
96
97     // Descola o ponteiro do arquivo para o comeco
98     file.seekp (0 ,ios::beg);
99
100    // Le os 2 dados
101    file >> idx >> off;
102
103    // Descola o ponteiro do arquivo para aonde estava
104    file.seekp(aux,ios::beg);
105
106    // Retorna os 2 dados
107    return make_pair(++idx,off);
108 }
109
110
111
112 // Funcao que retorna a quantidade de digitos daquele
        numero
113 int qnt_digits(int n){
114     return log10(n)+1;
115 }
116
117 // Gera o numero de 7 digitos com 0 a esquerda para o
        novo valor de header
118 string create_Header(int num){
119
120     // String de retorno
121     string ret = "";
122
123     // Quantidade de zeros que deve ser acrescentado a
        esquerda
124     int n = 7 - qnt_digits(num);
125
126     // Adicionando os zeros a esquerda
127     while(n--) ret += "0";
128
129     // Retorna o novo numero
130     return ret + to_string(num);

```



```

131 }
132
133 // Atualiza apenas o header do arquivo de registros
134 void update_datafile_header(){
135
136     // Gera o novo numero de acordo com a quantidade de
137     // registros
138     string newHeader = create_Header(header-1);
139
140     // Gera o novo numero de offset de acordo com a
141     // variavel offset
142     string newOffset = create_Header(offset);
143
144     // Volta para o comeco para salvar o novo header
145     arquivo.seekp (0 ,ios::beg);
146
147     // Salva o novo header
148     arquivo << newHeader + " " + newOffset;
149 }
150
151 // Atualiza o arquivo de indices
152 void update_indexfile(){
153
154     // Variavel para pegar o header e o offset do arquivo
155     // de indices
156     int indiceHeader, indiceOff;
157
158     // Pega os 2 valores
159     tie(indiceHeader, indiceOff) = get_file_header(indice)
160     ;
161
162     // Cria o novo header de acordo com o header do
163     // arquivo de registros
164     string newHeader = create_Header(header-1);
165
166     // Desloca o ponteiro para o final do arquivo de
167     // indices para a escrita no final
168     indice.seekp(indiceOff, ios::beg);
169
170     // Escreve os registros que faltam
171     for(int i = indiceHeader; i < header; i++)

```

```

168     indice << i << " " << desloc[i] << endl;
169
170     // Cria o novo dado de fim de arquivo de acordo com o
171     // final do arquivo de indices
172     string newOffset = create_Header(indice.tellg());
173
174     // Volta para o comeco do arquivo para escrever o novo
175     // header
176     indice.seekp(0, ios::beg);
177
178     // Escreve o novo header
179     indice << newHeader + " " + newOffset;
180
181     // Limpa o mapa
182     desloc.clear();
183 }
184
185 // Faz o split da string de acordo com o parametro
186 // passado pelo char c
187 const vector<string> split(const string& s, const char&
188 c){
189
190     // String base
191     string buff = "";
192
193     // Vetor que ira salvar os splits
194     vector<string> v;
195
196     // Andando por cada carcter
197     for(auto n:s){
198
199         // Se nao for o char de quebra
200         if(n != c)
201             buff += n;
202
203         // Char que deve ser quebrado, logo adicionar a
204         // string ao vetor
205         else if (n == c && buff != ""){
206             // Adicionando no vetor
207             v.push_back(buff);
208
209             // Resetando a string buff

```

```

206     buff = "";
207 }
208 }
209 // Fazendo a ultima adicao apos o char de quebra
210 if(buff != "")
211     v.push_back(buff);
212
213 // Retornando o vetor com os splits
214 return v;
215 }
216
217 // Menu principal
218 char menu(){
219
220     // Variavel de escolha da opcao do menu
221     char op;
222
223     // Limpa a tela
224     system(CLEAR);
225
226     // Iteracao com o usuario
227     intro();
228
229     // Display de opcoes
230     printf("0 - Sair\n");
231     printf("1 - Inserir\n");
232     printf("2 - Buscar\n");
233
234     // Pega a opcao do usuario e repete o mesmo passo caso
235     // seja invalido
236     while(cin >> op){
237
238         // Se a opcao for valida, encerra o loop
239         if (op >= '0' and op <= '2') break;
240
241         // Imprime para o usuario repetir caso a opcao
242         // escolhida seja invalida
243         printf("Digite 0, 1 ou 2: ");
244
245     }
246
247     return op;
248 }

```

```

247 // Pega a resposta do usuario
248 char get_op(){
249
250     // Retorno da escolha
251     char op;
252
253     // Enquanto le a escolha do usuario
254     while(cin >> op){
255
256         // Transforma para minusculo para facilitar a
257         // condicao
258         op = tolower(op);
259
260         // Checa se eh y ou n, pois sao as unicas 2 escolhas
261         // permitidas
262         if (op == 'y' or op == 'n') break;
263
264         // Pede para o usuario digitar corretamente
265         cout << "\n\tDgite apenas y, Y, n ou N: ";
266
267     }
268
269     // Retorna a escolha
270     return op;
271 }
272
273 // Funcao para inserir no arquivo de registros
274 void inserir(){
275
276     // Caracter para a escolha da operacao pelo usuario
277     char op;
278
279     // Limpa o buffer do cin
280     clear();
281
282     // Pega o header e o offset do arquivo
283     tie(header,offset) = get_file_header(arquivo);
284
285     // Desloca o ponteiro para o final do arquivo, para
286     // inserir novo elemento
287     arquivo.seekp(offset,ios::beg);
288
289     // Loop principal

```

```

287 while(true){
288
289     // Le os dados e insere no arquivo
290     read();
291
292     // Iteracao com o usuario
293     cout << "\n    Deseja continuar inserindo ? [y][n]: "
        ;
294
295     // Pega a operacao escolhida pelo usuario
296     op = get_op();
297
298     // Limpa a tela
299     system(CLEAR);
300
301     // Se a escolha for a de nao continuar
302     if(op == 'n'){
303
304         // Atualiza o header do arquivo de registros
305         update_datafile_header();
306
307         // Atualiza o arquivo de indices
308         update_indexfile();
309
310         // Encerra o loop
311         break;
312     }
313
314     // Limpa o buffer do cin
315     clear();
316
317 }
318
319
320 }
321
322 // Funcao para chegar se a string eh um numero
323 bool is_number(const string& s){
324     return( strspn( s.c_str(), "-.0123456789" ) == s.
        size() );
325 }
326
327 int get_id(){

```

```

328
329 // Iteracao com o usuario
330 cout << "Digite o id do registro para realizar a busca
      : ";
331
332 // String para ler a entrada do usuario
333 string id;
334
335 // Le o numero digitado
336 cin >> id;
337
338 // Checa se o numero digitado eh realmente um numero
339 while(!is_number(id)){
340
341     // Iteracao com o usuario
342     cout << "Por favor, digite um numero valido: ";
343
344     // Le novamente
345     cin >> id;
346 }
347
348 // Retorna o numero que era um string, transformado
      para inteiro
349 return stoi(id);
350 }
351
352
353 // Imprime os dados do registro do respectivo id
354 void show_message(int id){
355
356     // Iteracao com o usuario
357     cout << "\n-----" << endl;
358     cout << "\n\tDados do id " << id << endl << endl;
359
360
361     // Caso o id nao exista
362     if(!desloc.count(id)){
363
364         // Imprime que o id nao existe
365         cout << "\n\tID NOT FOUND" << endl;
366         cout << "\n-----\n" <<
              endl;
367

```

```

368     // Encerra a funcao
369     return;
370 }
371
372 // Desloca o ponteiro do arquivo ate o offset do id
    desejado
373 arquivo.seekp(desloc[id], ios::beg);
374
375 // String para ler a linha de dados do registo
376 string line;
377
378 // Le a linha no respectivo offset
379 getline(arquivo, line);
380
381 // Qubra o registo em blocos
382 vector<string> msg = split(line, '|');
383
384 // Imprime os dados do registo
385 for(int i = 1; i < msg.size(); i++)
386     cout << "\t" << questions[i-1] << " : " << msg[i] <<
        endl;
387
388 // Iteracao com o usuario
389 cout << "\n-----\n" <<
    endl;
390 }
391
392
393 // Busca os dados do registo pelo id
394 void buscar(){
395
396     // Redireciona para o comeco do arquivo
397     indice.seekp(0, ios::beg);
398
399     // Variaveis para pegar o header e o offset do arquivo
        de indices
400     int indiceHeader, indiceOff;
401
402     // Pega os 2 dados do header
403     tie(indiceHeader, indiceOff) = get_file_header(indice)
        ;
404
405     // Variaveis para pegar o id e o offset do arquivo de

```

```

    indices e armazenar em um mapa para consultar em
    tempo  $O(\log(n))$ 
406 int id, off; char op;
407
408 // Armazenando o id e o offset no mapa
409 while(indice >> id >> off)
410     desloc[id] = off;
411
412
413 // Loop principal das perguntas
414 while(true){
415
416     // Busca o id escolhido pelo usuario
417     id = get_id();
418
419     // Imprime os dados do registro correspondente
420     show_message(id);
421
422     // Iteracao com o usuario
423     cout << "\n    Deseja continuar buscando ? [y][n]: ";
424
425     // Pega a escolha do usuario
426     op = get_op();
427
428     // Limpa a tela
429     system(CLEAR);
430
431     // Se a escolha do usuario for n, entao encerrar
432     // loop
433     if(op == 'n') break;
434
435     // Limpar buffer do cin
436     clear();
437 }
438
439 }
440
441 // Verifica se o arquivo esta vazio para preencher com a
442 // quantidade de registros e deslocamento em bytes
443 void check(){
444     // Pega o final do arquivo de registros

```



```

445     arquivo.seekp(0,ios::end);
446
447     // Pega o final do arquivo de indices
448     indice.seekp(0, ios::end);
449
450     // Se estiver vazio ele escreve o header
451     if(!arquivo.tellg())
452         arquivo << "0000000 0000016" << endl;
453
454     // Se estiver vazio ele escreve o header
455     if(!indice.tellg())
456         indice << "0000000 0000016" << endl;
457 }
458
459 int main(){
460
461     check();
462
463     // Loop principal do programa
464     while(true){
465
466         // Pega o opcao com o usuario
467         char op = menu();
468
469         // Limpa a tela
470         system(CLEAR);
471
472         // Verifica qual opcao o usuario ira escolher
473         if(op == '0') break;
474
475         else if(op == '1') inserir();
476
477         else buscar();
478
479     }
480 }
481

```