

Universidade de Brasília

Departamento de Ciência da Computação



## Lista de Exercícios 6

### Organização de Arquivos

#### **Autores:**

Giovanni M Guidini 16/0122660

Vitor F Dullens 16/0148260

Brasília  
2 de Junho de 2018

## Lista de Exercícios 6

Alunos: Giovanni Guidini 16/0122660; Vitor Dullens 16/0148260.

Prof. Oscar Gaidos

## Organização de Arquivos

CIC 116327

Data: 2 de Junho de 2018

### Exercício 1

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

#### Resposta

Listas invertidas são uma maneira de organizar o arquivo mapeando termos às suas ocorrências em um documento ou conjunto de documentos, de maneira que cada registro contenha a chave secundária e um ponteiro para uma lista de referência.

Vantagens:

- Rearranjar o arquivo de índices secundário é mais rápido, pois ele está menor;
- Buscas mais precisas e rápidas;
- Evita cópias de chaves secundárias.

Desvantagens:

- A lista de referência terá um limite de tamanho, e podemos ter mais registros para incluir na lista de referência;
- Podemos ter fragmentação interna após muitas inclusões, deleções e adaptações;
- Talvez algumas das referências apontadas pela lista de referência não são mais válidas.

### Exercício 2

Por que se usa chaves secundárias?

#### Resposta

As chaves secundárias são utilizadas a fim de melhorar o acesso as informações buscadas. Um exemplo dessa situação é a biblioteca, cada livro tem um código, porém normalmente não é pesquisado o código do livro, e sim seu assunto, título, etc (i.e. uma chave secundária). A partir da chave secundária se encontra o código e por fim o livro.

### Exercício 3

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

## Resposta

Pois o índice secundário é apenas uma referência ao primário, fazendo com que ele possa armazenar mais chaves primárias. Seguindo o exemplo da biblioteca, vários livros podem possuir o mesmo assunto. Com isso, é possível apenas remover o arquivo do índice primário e deixar sua referência no índice secundário, ao invés de ter que atualizar todos os índices ligados a mesma chave primária. Ao tentar se acessar a chave primária inválida não a acharemos no arquivo índice principal, simplesmente.

Um problema disso é que podemos ter muitas referências inválidas no arquivo índice secundário. Essas referências precisam ser tratadas.

## Exercício 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 primária.
- Crie um índice simples com essa chave primária.
- Faça a busca de um registro nesse arquivo de índices simples.
- Opcional : crie chaves secundárias e usando essas chaves buscar a chave primária.

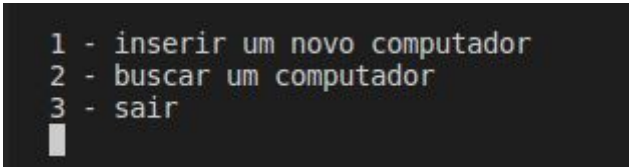
## Código

Optamos por fazer o programa utilizando como registro as características de um computador convencional. Os campos utilizados podem ser vistos na figura 4. Os campos e os registros tem tamanho variável e são separados por caractere especial. Os campos são separados pelo caractere '|' e os registros pelo caractere '\n'.

O arquivo que armazena os dados possui um cabeçalho que é atualizado a cada nova inserção, para garantir integridade dos dados. Tal arquivo está mostrado na figura 2. A partir desse cabeçalho é verificado se o arquivo de índices está atualizado ou não, pois os cabeçalhos tem que ser idênticos. Se não estiver, ele é atualizado. O arquivo de índices está na figura 3.

Feito isso o usuário pode fazer a busca dos dados, inclusive os inseridos mais recentemente. Um exemplo de recuperação de dados está na figura 5. O código para o programa se encontra no apêndice A.

## Saídas



```
1 - inserir um novo computador
2 - buscar um computador
3 - sair
█
```

Figura 1: Menu do programa - Usuário escolhe a ação

```
computer.txt x ex4.cpp .index.txt
1 4-2018-06-02.16:12:00
2 1|Aspire v3|Acer|Intel|Intel Core i7-3537U|Mesa DRI Intel(R) Ivybridge Mobile|1366x768|6Gb|2013|500Gb|1500$|
3 2|Icore7|Acer|Acer|Intel® Core™ i5-5200U|NVIDIA® GeForce® 820M|1366 x 768|4 GB|2016|1 TB|1950$|
4 3|Inspiron|Dell|Dell|Intel Core i3 6006U|On-board|HD|4 GB|2017|1 TB|2082$|
5 4|IdeiaPad|Lenovo|Intel|Intel Core i3 6006U|On-board|Full HD|4 GB|2017|1 TB|1708$|
```

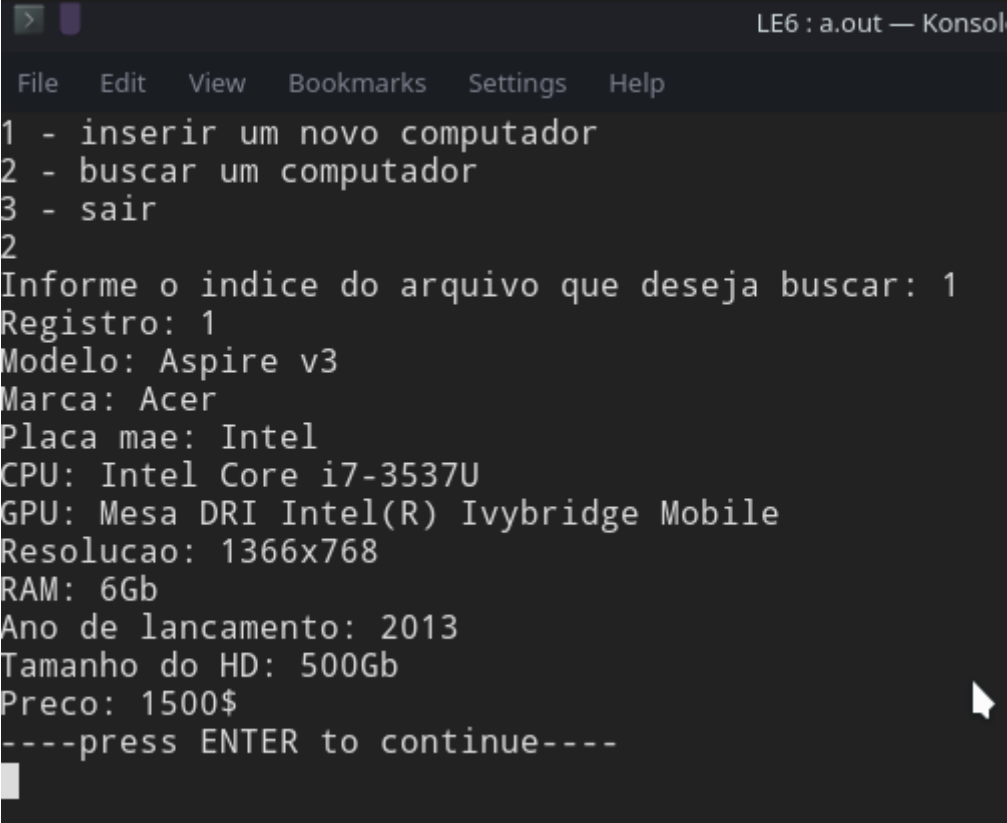
Figura 2: Arquivo que armazena a chave primaria e as informações

```
computer.txt ex4.cpp .index.txt x
1 4-2018-06-02.16:12:00
2 1 31 2 140 3 241 4 316 4 316 4 316 4 316 4 316
```

Figura 3: Arquivo de indices para a busca

```
1 - inserir um novo computador
2 - buscar um computador
3 - sair
1
## Preencha o formulario com base nas informacoes do seu Computador ##
Modelo? inspiron
Marca? dell
Placa mae? dell
CPU? i5
GPU? 1080
Resolucao? full hd
RAM? 8 gb
Ano de lancamento? 2018
Tamanho do HD? 1 tb
Preco? 4300
quer digitar outro computador? [y/n] █
```

Figura 4: Adicionando um novo computador



The image shows a terminal window titled "LE6 : a.out — Konsol". The window has a menu bar with "File", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal content displays a menu with three options: "1 - inserir um novo computador", "2 - buscar um computador", and "3 - sair". Option "2" is selected. The program then prompts "Informe o indice do arquivo que deseja buscar: 1". It displays the following details for "Registro: 1":  
Modelo: Aspire v3  
Marca: Acer  
Placa mae: Intel  
CPU: Intel Core i7-3537U  
GPU: Mesa DRI Intel(R) Ivybridge Mobile  
Resolucao: 1366x768  
RAM: 6Gb  
Ano de lancamento: 2013  
Tamanho do HD: 500Gb  
Preco: 1500\$  
The prompt "----press ENTER to continue----" is shown at the bottom, followed by a cursor.

```
>
File Edit View Bookmarks Settings Help
1 - inserir um novo computador
2 - buscar um computador
3 - sair
2
Informe o indice do arquivo que deseja buscar: 1
Registro: 1
Modelo: Aspire v3
Marca: Acer
Placa mae: Intel
CPU: Intel Core i7-3537U
GPU: Mesa DRI Intel(R) Ivybridge Mobile
Resolucao: 1366x768
RAM: 6Gb
Ano de lancamento: 2013
Tamanho do HD: 500Gb
Preco: 1500$
----press ENTER to continue----
```

Figura 5: Buscando e mostrando esse computador na tela

## A Exercício 4

```
1 #include <bits/stdc++.h>
3 #ifdef WIN32
4     #define CLEAR "cls"
5 #else
6     #define CLEAR "clear"
7 #endif
9 using namespace std;
11
12 void read(int id, fstream& arquivo){
13     vector<string> data(10);
14     vector<string> question = {"Modelo?", "Marca?", "Placa mae?", "CPU?", "GPU?",
15                               "Resolucao?", "RAM?", "Ano de lancamento?", "Tamanho do HD?", "Preco?"};
16
17     cout << "## Preencha o formulario com base nas informacoes do seu Computador
18     ##" << endl;
19     arquivo << id << "|";
20     for(int i=0;i<10;i++){
21         cout << question[i] << " ";
22         getline(cin, data[i]);
23         arquivo << data[i] << "|";
24     }
25     arquivo << endl;
26 }
27
28 const string currentDateTime()
29 {
30     time_t now = time(0);
31     struct tm tstruct;
32     char buf[80];
33     tstruct = *localtime(&now);
34     // Visit http://en.cppreference.com/w/cpp/chrono/c/strftime
35     // for more information about date/time format
36     strftime(buf, sizeof(buf), "%Y-%m-%d.%X", &tstruct);
37
38     return buf;
39 }
40
41 void showResult(string str){
42     string now;
43     vector<string> parts = {"Registro", "Modelo", "Marca", "Placa mae", "CPU", "GPU", "Resolucao", "RAM", "Ano de lancamento", "Tamanho do HD", "Preco"};
44     int count = 0, it = 0;
45     while(count < 11){
46         now = "";
47         while(str[it] != ' '){
48             now += str[it];
49             it++;
50         }
51         cout << parts[count] << ": " << now << endl;
52         count++;
53         it++;
54     }
55 }
56
57 int binSearch(vector<pair<int,int>>& idx, int low, int high, int target){
58     if(low > high) return -1;
59     int mid = (low+high)/2;
```

```

57     if(idx[mid].first == target){
        return idx[mid].second;
    }
59     else if(idx[mid].first < target){
        return binSearch(idx, mid+1, high, target);
61     }
    else{
63         return binSearch(idx, low, mid-1, target);
    }
65 }
void search( ifstream& fd, vector<pair<int, int>>& idx, bool flag = false){
67     int pk;
    if (flag){
69         ifstream index(".index.txt");
        string trash;
71         int num, offset;
        getline(index, trash);
73         while(!index.eof()){
            index >> num;
75             index >> offset;
            idx.push_back(make_pair(num, offset));
77         }
    }
79     cout << "Informe o indice do arquivo que deseja buscar: ";
    cin >> pk;
81
    int off = binSearch(idx, 0, idx.size()-1, pk);
83     if(off == -1){
        cout << "Registro nao existe\n";
85     }
    else{
87         fd.seekg(off);
        string content;
89         getline(fd, content);
        showResult(content);
91         cin.clear(); cin.ignore(INT_MAX, '\n');
        cout << "——press ENTER to continue——" << endl;
93         getchar();
    }
95     return;
}
97 void index(){
    ifstream arquivo;
99     arquivo.open("computer.txt");
    fstream index;
101     index.open(".index.txt", ios::in);
    string header, indexHeader;
103     vector<pair<int, int>> idx;
    int offset, num, total;
105     arquivo >> total;
    arquivo.seekg(0);
107     getline(arquivo, header);
    getline(index, indexHeader);
109
    index.close();
111
    if(header == indexHeader){
113         search(arquivo, idx, true);
        return;
115     }
}

```

```

else
117     while(!arquivo.eof() && num < total){
119         offset = arquivo.tellg();
121         arquivo >> num;
123         idx.push_back(make_pair(num, offset));
125         getline(arquivo, indexHeader);
127     }
129
131     index.open(".index.txt", ios::out);
133     index << header << endl;
135     for(auto q : idx) {
137         index << q.first << " " << q.second << " ";
139     }
141     index.close();
143     arquivo.seekg(0);
145     search(arquivo, idx);
147 }
149 void intro(){
151     system(CLEAR);
153     printf("#####\n");
155     printf("#          LE6 Exercise 4          #\n");
157     printf("#####\n");
159     printf("#      Aluno: Vitor Dullens      #\n");
161     printf("#      Matricula: 16/0148260      #\n");
163     printf("#          #\n");
165     printf("#      Aluno: Giovanni Guidini    #\n");
167     printf("#      Matricula: 16/0122660      #\n");
169     printf("#####\n");
171     getchar();
173     system(CLEAR);
175 }
177 int menu(){
179     int op;
181     system(CLEAR);
183     cout << "1 - inserir um novo computador" << endl;
185     cout << "2 - buscar um computador" << endl;
187     cout << "3 - sair" << endl;
189     cin >> op;
191     getchar();
193     while(op<1 or op>3) menu();
195     return op;
197 }
199 int main(){
201     intro();
203     int header;
205     string time;
207     char traco;
209     string op;
211     bool go = true;
213     fstream arquivo;
215     vector<pair<int, int>> idx;
217
219     arquivo.open("computer.txt", ios::in | ios::out);
221     arquivo >> header >> traco >> time;
223     getline(arquivo, op); // chew trailing spaces
225     int opc;
227     opc = menu();
229     while(opc != 3){

```



```

177     if(opc == 1){
179         arquivo.seekp(0, arquivo.end);
181         while(go){
183             read(++header, arquivo);
185             cout << "quer digitar outro computador? [y/n] ";
187             while(true){
189                 getline(cin, op);
191                 if(op[0] == 'n' or op[0] == 'N'){
193                     go = false;
195                     break;
197                 }
199                 else if(op[0] == 'y' or op[0] == 'Y') break;
201                 cout << "## digite y ou n apenas: ";
203             }
177         }
179         time = currentDateTime();
181         arquivo.seekp(0, arquivo.beg);
183         arquivo << header << "-" << time << "
185                                     \n"; // extra spaces
187         to manage growing headers
189         arquivo.close(); // flush changes
191     }
193     else if(opc == 2){
195         index();
197     }
199     opc = menu();
201 }
203 }

```

Listing 1: "Code"