



Universidade Federal de Sergipe - UFS
Departamento de Sistemas de Informação - Itabaiana - DSI/Ita

Programação II - SINF0064
Exercício 07 - Programando com Arquivos e
Análise Descritiva

Prof. Dr. Alcides Xavier Benicasa

Exercício **INDIVIDUAL AVALIATIVO**

Envio: encaminhar arquivos via *SIGAA*

Assunto: EX07 - Programando com Arquivos - Análise Descritiva



QUESTÕES:

1. Escreva um programa que receba do usuário **5 números inteiros** e o **nome do arquivo** no qual eles devem ser armazenados. Em seguida, ler do arquivo estes valores armazenados, copiando-os para um vetor de inteiros e imprimindo na tela.
2. *Arquivo Caixa Alta* - Escreva um programa que **lê e modifica** um arquivo texto lido, trocando cada letra pela sua correspondente **maiúscula**. Veja o exemplo abaixo, que mostra como o arquivo de entrada e o de saída:

```
The process of creating a stream linked  
to a disk file is called opening the  
file. When you open a file, it becomes  
available for (1) reading (meaning that  
data is input from the file to the  
program), (2) writing (meaning that  
data from the program is saved in the  
file), or (3) both.
```

```
THE PROCESS OF CREATING A STREAM LINKED  
TO A DISK FILE IS CALLED OPENING THE  
FILE. WHEN YOU OPEN A FILE, IT BECOMES  
AVAILABLE FOR (1) READING (MEANING THAT  
DATA IS INPUT FROM THE FILE TO THE  
PROGRAM), (2) WRITING (MEANING THAT  
DATA FROM THE PROGRAM IS SAVED IN THE  
FILE), OR (3) BOTH.
```

3. *Tabela de notas* - Escreva um programa que gerencie uma tabela de notas. Ao executar, o usuário terá as seguintes opções: **1 - Adicionar mais um aluno** (Nome e nota); **2 - Exibir todas as notas**; **3 - Calcular a média**. O programa deve usar um arquivo texto para armazenar e consultar as informações.

4. O exemplo NA02_EXEMPLO_07 utiliza um arquivo de acesso aleatório para obter acesso “instantâneo” aos registros de um arquivo, simulando contas de clientes em um banco. A opção número 1 deve ser utilizada somente uma única vez, já que ela cria o arquivo com 100 registros “em branco” ou sobrepõe os 100 registros existentes com valores que consideramos como registros vazios. Note que, embora interpretemos os registros como vazios, eles contêm dados, ou seja, zero para o número de conta e saldo e um vetor de caracteres de tamanho zero para o nome. Considere o projeto completo disponível para *download* no SIGAA e seu código fonte aqui impresso. Com suas palavras e de maneira organizada, faça um parecer descritivo para cada uma das funções encontradas no programa.

CÓDIGO-FONTE (main.cpp)

```
1  #include <locale.h>
2  #include <iostream>
3  #include <fstream>
4  #include <iomanip>
5  #include <cstdlib>
6  #include <string>
7  using namespace std;
8  #include "cliente.h"
9  enum Escolhas {CRIAR=1, TELA, ARQUIVOTEXTO, ATUALIZAR, NOVO, APAGAR, FIM};
10
11  //-----
12
13  Escolhas enterChoice()
14  {
15      int menuChoice;
16      cout << "\nMenu:" << endl
17           << "1 - Cria registros vazios no arquivo\n"
18           << "2 - Lista os dados na tela\n"
19           << "3 - Armazena os dados no arquivo texto \"print.txt\"\n"
20           << "4 - Atualiza uma conta que ja contenha informações\n"
21           << "5 - Insere informações em uma nova conta\n"
22           << "6 - Apaga informações de uma conta\n"
23           << "7 - Fim do programa\n"
24           << "Opção: ";
25      cin >> menuChoice;
26      return (Escolhas) menuChoice;
27  }
28
29  //-----
30
31  void create(fstream &f)
32  {
33      cliente clienteVazio = {0, "", 0.0};
34
35      f.seekp(0);
36      for (int i = 0; i < 100; i++)
37          f.write((const char *)&clienteVazio, sizeof(cliente));
38  }
39
40  //-----
41
42  void outputLine(ostream &output, const cliente &c)
43  {
44      output << setiosflags(ios::left)
45             << setw(10) << c.numero
46             << setw(30) << c.nome
47             << setw(10) << setprecision(2) << resetiosflags(ios::left)
48             << setiosflags(ios::fixed | ios::showpoint) << c.saldo << '\n';
49  }
50
51  //-----
52
53  void screen(fstream &f)
54  {
55      cliente c;
56
57      cout << setiosflags(ios::left)
58           << setw(10) << "Conta"
59           << setw(30) << "Nome"
60           << resetiosflags(ios::left) << setw(10) << "Saldo" << endl;
61
62      f.seekp(0);
63      f.read((char *)&c, sizeof(cliente));
64      while(! f.eof())
65      {
66          if(c.numero != 0)
67              outputLine(cout, c);
68          f.read((char *)&c, sizeof(cliente));
69      }
70  }
71
72  //-----
73
74  void textFile(fstream &f)
75  {
76      cliente c;
77      ofstream outPrintFile("print.txt", ios::out);
78
79      if(! outPrintFile)
```

```

80     {
81         cerr << "Arquivo print.txt não pode ser aberto." << endl;
82         exit(1);
83     }
84
85     outPrintFile << setiosflags(ios::left)
86         << setw(10) << "Conta"
87         << setw(30) << "Nome"
88         << resetiosflags(ios::left) << setw(10) << "Saldo" << endl;
89
90     f.seekp(0);
91     f.read((char *)&c, sizeof(cliente));
92     while(! f.eof())
93     {
94         if(c.numero != 0)
95             outputLine(outPrintFile, c);
96         f.read((char *)&c, sizeof(cliente));
97     }
98     outPrintFile.close();
99 }
100
101 //-----
102
103 int getAccount(string msg)
104 {
105     int conta;
106
107     do
108     {
109         cout << msg << " (1 - 100): ";
110         cin >> conta;
111     }
112     while (conta < 1 || conta > 100);
113
114     return conta;
115 }
116
117 //-----
118
119 void updateRecord(fstream &f)
120 {
121     int conta;
122     cliente c;
123     float transacao;
124
125     conta = getAccount("Conta a ser atualizada");
126     f.seekp((conta-1)*sizeof(cliente)); // posicionar na conta desejada
127     f.read((char *)&c, sizeof(cliente)); // ler dados da conta
128     if(c.numero != 0) // conta contem informacao?
129     {
130         outputLine(cout, c);
131         cout << "\nEntre deposito (+) ou retirada (-): ";
132         cin >> transacao;
133         c.saldo += transacao;
134         outputLine(cout, c);
135         f.seekp((conta - 1) * sizeof(cliente)); // posicionar na conta desejada
136         f.write((const char *)&c, sizeof(cliente)); // atualizar
137     }
138     else
139         cerr << "Conta #" << conta << " não possui informação." << endl;
140 }
141
142 //-----
143
144 void newRecord(fstream &f)
145 {
146     int conta;
147     cliente c;
148
149     conta = getAccount("Número da nova conta");
150     f.seekp((conta-1) * sizeof(cliente)); // posicionar na conta desejada
151     f.read((char *)&c, sizeof(cliente)); // ler dados da conta
152     if(c.numero == 0)
153     {
154         cout << "Entre nome, saldo\n? ";
155         cin >> c.nome >> c.saldo;
156         c.numero = conta;
157         f.seekp((conta - 1) * sizeof(cliente)); // posicionar na conta desejada
158         f.write((const char *)&c, sizeof(cliente)); // atualizar
159     }
160     else
161         cerr << "Conta #" << conta << " já possui informação." << endl;
162 }
163
164 //-----
165
166 void deleteRecord(fstream &f)
167 {
168     int conta;
169     cliente c, clienteVazio = {0, "", 0.0};
170
171     conta = getAccount("Conta a ser apagada");
172     f.seekp((conta-1) * sizeof(cliente));
173     f.read((char *)&c, sizeof(cliente));
174     if(c.numero != 0)
175     {
176         f.seekp((conta - 1) * sizeof(cliente));
177         f.write((const char *)&clienteVazio, sizeof(cliente));
178         cout << "Conta #" << conta << " apagada." << endl;
179     }
180     else
181         cerr << "Conta #" << conta << " já esta apagada." << endl;
182 }

```

```

183
184 //-----
185
186 int main()
187 {
188     setlocale(LC_ALL, "portuguese");
189
190     Escolhas opcao;
191     fstream inOutCredito("credito.dat", ios::in | ios::out);
192
193     if(! inOutCredito)
194     {
195         cerr << "Arquivo credito.dat não pode ser aberto." << endl;
196         exit (1);
197     }
198
199     while ((opcao = enterChoice()) != FIM)
200     {
201         switch (opcao)
202         {
203             case CRIAR:
204                 create(inOutCredito);
205                 break;
206             case TELA:
207                 screen(inOutCredito);
208                 break;
209             case ARQUIVOTEXTO:
210                 textFile(inOutCredito);
211                 break;
212             case ATUALIZAR:
213                 updateRecord(inOutCredito);
214                 break;
215             case NOVO:
216                 newRecord(inOutCredito);
217                 break;
218             case APAGAR:
219                 deleteRecord(inOutCredito);
220                 break;
221             default:
222                 cerr << "Opção incorreta\n";
223                 break;
224         }
225         inOutCredito.clear();
226     }
227     return 0;
228 }

```

CÓDIGO-FONTE (cliente.h)

```

1 #ifndef CLIENTE_H_INCLUDED
2 #define CLIENTE_H_INCLUDED
3 struct cliente
4 {
5     int numero;
6     char nome[30];
7     float saldo;
8 };
9 #endif

```