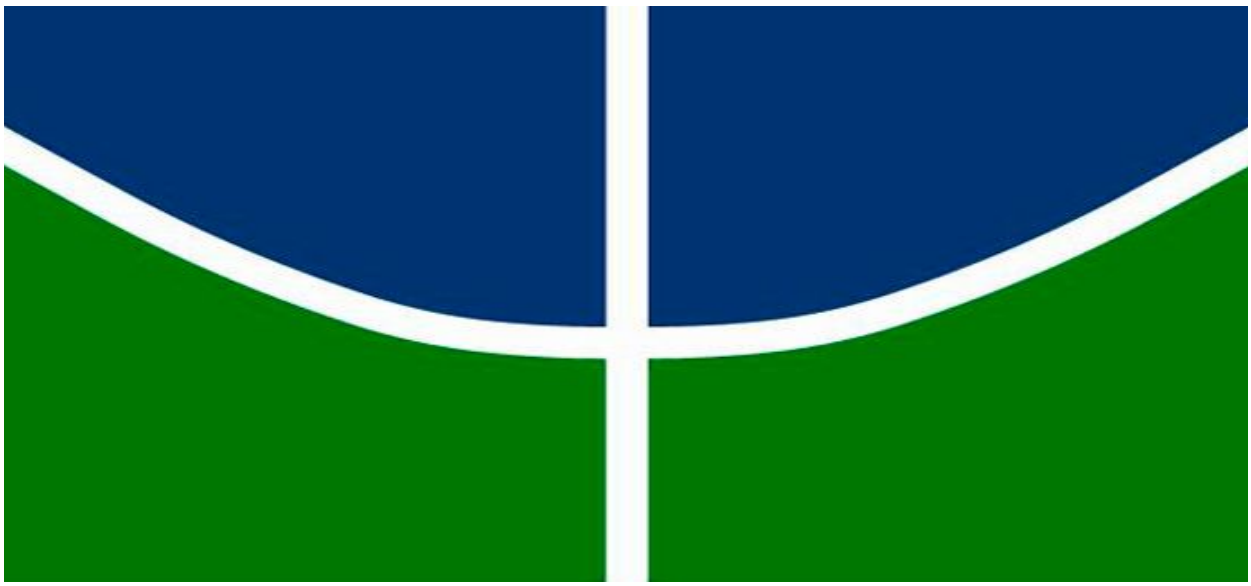


Lista de Exercícios 3 - OA

12/04/2018

Otto Kristian von Sperling
12/0131510

Gabriel Guimarães A. de Castro
15/0126425

Universidade de Brasília - UnB
Departamento de Ciência da Computação
Curso: Ciência da Computação - Bacharelado
Disciplina: Organização de Arquivos
Professor: Oscar Gaidos

Exercício - *CODE*

Faça um programa que leia os dados sobrenome, nome, endereço, CEP e telefone do teclado e grave esses dados segundo os métodos abaixo. Esse programa deve ser capaz de listar os registros após gravá-los

main() com passagem de parâmetros por terminal

```
int main(int argc, char** argv)
{
    int user_opt = 0;

    if(argc == 2)
    {
        user_opt = atoi(argv[1]);
        if(user_opt < 5)
            main_menu(user_opt);
        else
            argc = 3;
    }
    if(argc > 2)
    {
        cout << "Limited to 1 parameter.\n\t1- Static datastream\t\t2- Size in front of datastream\n";
        cout << "\t3- Separator in datastream\t4- Attribute in datastream\n";
    }
    else
    {
        user_opt = 0;
        main_menu(user_opt);
    }
    return 0;
}
```

Item 1

Campos de tamanho fixo

Função principal

A função abaixo usa várias vezes cin.getline que é uma função que obtém a entrada com espaços. Mas sempre que é usada também é necessário que o buffer seja limpo para evitar erro, por isso usa-se o cin.clear.

```
 9  void tamanhoFixo()
10  {
11      string nome, sobrenome, endereco, cep, telefone;
12
13      registro_1.open("registro1.txt", ofstream::out | ofstream::app);
14
15      cout << "Digite seu nome:"  << endl;
16      getline(std::cin,nome);
17      getline(std::cin,nome);
18      for(int i = nome.size(); i < TAM; ++i)
19          nome.append("#");
20      registro_1 << nome;
21
22      cout << "Digite seu sobrenome:" << endl;
23      cin.clear();
24      getline(std::cin,sobrenome);
25      for(int i = sobrenome.size(); i < TAM; ++i)
26          sobrenome.append("#");
27      registro_1 << sobrenome;
28
29      cout << "Digite seu endereço:"  << endl;
30      cin.clear();
31      getline(std::cin,endereco);
32      for(int i = endereco.size(); i < TAM; ++i)
33          endereco.append("#");
34      registro_1 << endereco;
35
36      cout << "Digite seu cep:" << endl;
37      cin.clear();
38      getline(std::cin,cep);
39      for(int i = cep.size(); i < TAM; ++i)
40          cep.append("#");
41      registro_1 << cep;
42
43      cout << "Digite seu telefone:" << endl;
44      cin.clear();
45      getline(std::cin,telefone);
46      for(int i = telefone.size(); i < TAM; ++i)
47          telefone.append("#");
48      registro_1 <<   telefone;
49      registro_1.close();
50      cout << "Registrado\n";
51
52  }
```

Procurar registro

A função encontrarTodasStr_1 procura todas ocorrências de uma string dada dentro de uma outra que contém o conteúdo do arquivo que guarda as informações.

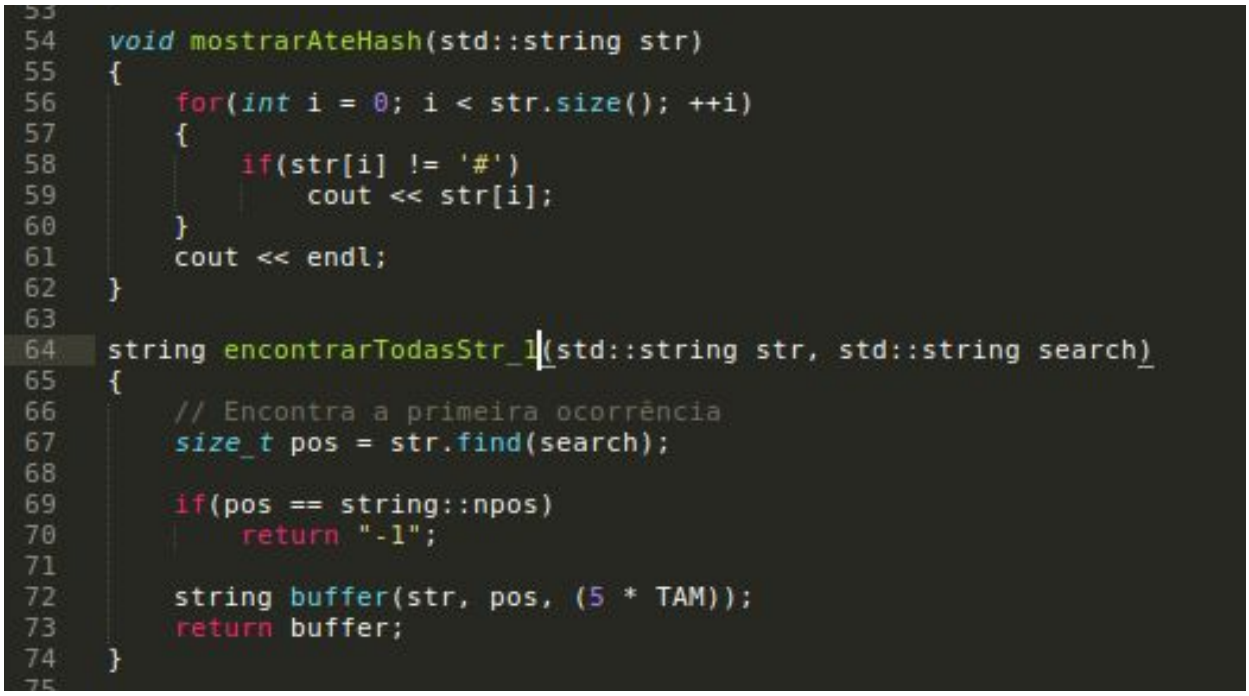
```
64  string encontrarTodasStr_1(std::string str, std::string search)
65  {
66      // Encontra a primeira ocorrência
67      size_t pos = str.find(search);
68
69      if(pos == string::npos)
70          return "-1";
71
72      string buffer(str, pos, (5 * TAM));
73      return buffer;
74  }
```

Já a função procurar_1 usa a função acima para encontrar as ocorrências e organiza para que seja impresso na tela de forma organizada e legível cada dado.

```
76  void procurar_1()
77  {
78      string search, str, found, buf;
79      string filename = "registro1.txt";
80
81      ifstream file(filename.c_str());
82      stringstream buffer;
83
84      buffer << file.rdbuf();
85      str = buffer.str();
86
87      cout << "Digite o nome da pessoa que deseja procurar_1:" << endl;
88      cin.clear();
89      getline(std::cin, search);
90      getline(std::cin, search);
91
92      found = encontrarTodasStr_1(str , search);
93      if(found == "-1")
94      {
95          cout << "Pessoa nao localizada." << endl;
96          file.close();
97      }
98      else
99      {
100         {
101             cout << "-----------------------------------------\n";
102             cout << "Nome: ";
103             buf = found.substr(0, TAM);
104             mostrarAteHash(buf);
105
106             cout << "Sobrenome: ";
107             buf = found.substr(TAM, TAM);
108             mostrarAteHash(buf);
109
110             cout << "Endereco: ";
111             buf = found.substr((2 * TAM), TAM);
112             mostrarAteHash(buf);
113
114             cout << "CEP: ";
115             buf = found.substr((3 * TAM), TAM);
116             mostrarAteHash(buf);
117
118             cout << "Telefone: ";
119             buf = found.substr((4 * TAM), TAM);
120             mostrarAteHash(buf);
121             cout << "-----------------------------------------\n";
122         }
123     }
124     cout << "Pressione Enter pra voltar ao menu\n";
125     getchar();
126 }
```

Mostrar na tela

Para mostrar na tela é usada a função mostrarAteHash que mostra tudo até encontrar pela primeira vez o símbolo “#”. Esse simbolo que preenche o resto do espaço que sobra em cada registro.



```
54  void mostrarAteHash(std::string str)
55  {
56      for(int i = 0; i < str.size(); ++i)
57      {
58          if(str[i] != '#')
59              cout << str[i];
60      }
61      cout << endl;
62  }
63
64  string encontrarTodasStr_1(std::string str, std::string search)
65  {
66      // Encontra a primeira ocorrência
67      size_t pos = str.find(search);
68
69      if(pos == string::npos)
70          return "-1";
71
72      string buffer(str, pos, (5 * TAM));
73      return buffer;
74  }
```

Menu

```
128
129  void menu_1()
130  {
131      int choice;
132      bool menu = true;
133      while (menu != false){
134          cout << "*********************************\n";
135          cout << " 1 - Registrar uma pessoa\n";
136          cout << " 2 - Procurar registro\n";
137          cout << " 3 - Exit.\n";
138          cout << " Digite uma opção: ";
139
140          cin >> choice;
141          //cin.clear();
142          //cin.ignore(256, '\n');
143
144          switch (choice)
145          {
146              case 1:
147              tamanhoFixo();
148              break;
149
150              case 2:
151              procurar_1();
152              break;
153
154              case 3:
155              menu = false;
156              break;
157
158              default:
159              cout << "Escolha não válida \n";
160              cout << "Escolha novamente\n";
161              cin >> choice;
162              //cin.clear();
163              //cin.ignore(256, '\n');
164              break;
165          }
166      }
167  }
```

Item 2

Campos com tamanho no início

Gravar

```
void burn_sizeInFront(fstream &file)
{
    string buffer;
    list<string> data_fields = {"Sobrenome", "Nome", "Endereco", "CEP", "Telefone"};
    //list<string> data_input = {"Sperling", "Otto", "SQNSQN", "77777777", "999999999"};
    string data_size;
    list<string>::iterator jt = data_input.begin(); // for the sake of debugging

    for(list<string>::iterator it = data_fields.begin(); it != data_fields.end(); ++it)
    {
        cout << (*it) << " = ";
        cin >> buffer;  // in the actual program
        //buffer = (*jt);   // for the sake of debugging

        if(buffer.size() < 10)                      // START_ standardize size field to 2 bytes
        {                                           //
            data_size = "0";                        //
            data_size.append(to_string(buffer.size())); //
        }                                           //
        else                                        //
        {                                           //
            data_size = to_string(buffer.size());       //
        }                                           // END_ standardize size field to 2 bytes

        buffer.insert(0, data_size);
        file << buffer;
        buffer.clear();
        data_size.clear();
    //  jt++;
    }

}
```

Ler

```
void read_sizeInFront(fstream &file)
{
    int data_length = 0;
    string buffer;
    list<string> data_fields = {"Sobrenome", "Nome", "Endereco", "CEP", "Telefone"};
    list<string>::iterator it = data_fields.begin();
    streambuf *pbuf = file.rdbuf();

    while(pbuf->sgetc() != EOF)
    {
        if(buffer.size() < 2)
        {
            buffer.append(1, pbuf->sbumpc());
        }
        else
        {
            data_length = stoi(buffer);
            buffer.clear();
            for(int i = 0; i < data_length; ++i)
            {
                buffer.append(1, pbuf->sbumpc());
            }
            cout << (*it) << ": " << buffer << endl;
            buffer.clear();
            it++;
        }
    }
}
```

Item 3

Campos com separadores

Gravar

```
void burn_separator(fstream &file)
{
    string buffer;
    list<string> data_fields = {"Sobrenome", "Nome", "Endereco", "CEP", "Telefone"};
//  list<string> data_input = {"Sperling", "Otto", "SQNSQN", "77777777", "999999999"};
    list<string>::iterator jt = data_input.begin(); // for the sake of debuggin


    for(list<string>::iterator it = data_fields.begin(); it != data_fields.end(); ++it)
    {
        cout << (*it) << " = ";
        cin >> buffer;  // in the actual program
        //buffer = (*jt);   // for the sake of debugging

        buffer.append("|");
        file << buffer;
        buffer.clear();
        //jt++;
    }

}
```

Ler

```
void read_separator(fstream &file)
{
    string buffer;
    list<string> data_fields = {"Sobrenome", "Nome", "Endereco", "CEP", "Telefone"};
    list<string>::iterator it = data_fields.begin();
    streambuf *pbuf = file.rdbuf();
    char trash;

    while(pbuf->sgetc() != EOF)
    {
        while(pbuf->sgetc() != '|')
            buffer += pbuf->sbumpc();
        trash = pbuf->sbumpc();
        cout << (*it) << ": " << buffer << endl;
        buffer.clear();
        it++;
    }
}
```

Item 4

Campos com atributo = valor

Função principal

```
15
16  void atributo()
17  {
18      registro.open("registro4.txt", ofstream::out | ofstream::app);
19      string str;
20
21      cout << "Digite seu nome:\n";
22      str = "Nome=";
23      guarda(str);
24
25      cout << "Digite seu sobrenome:\n";
26      str = "Sobrenome=";
27      guarda(str);
28
29      cout << "Digite seu endereço:\n";
30      str = "Endereço=";
31      guarda(str);
32
33      cout << "Digite seu cep:\n";
34      str = "CEP=";
35      guarda(str);
36
37      cout << "Digite seu telefone:\n";
38      str = "Telefone=";
39      guarda(str);
40
41      registro.close();
42
43  }
```

Mostrar Atributo

As informações no arquivo são gravadas no seguinte formato:

Sobrenome=Castro Guimarães|

Então a função abaixo mostra tudo entre o "=" e o "|":

```
int mostrarAtributo(std::string str,int pos)
{
    if(str[pos-1]!='=')
    {
        while (str[pos] != '=')
        {
            pos+=1;
        }
        pos+=1;
    }

    while(str[pos] != '|')
    {
        cout << str[pos];
        pos+=1;
    }
    cout << endl;
    pos+=1;
    return pos;
}
```

Procurar registro

Procurar todas ocorrências de uma string em outra

Essa função é muito parecida com a função mostrada no item 1:

```
66  void encontrarTodasStr(std::vector<size_t> & vec, std::string str, std::string search)
67  {
68      size_t pos = str.find(search);
69
70      while( pos != string::npos)
71      {
72          vec.push_back(pos);
73
74          pos =str.find(search, pos + search.size());
75      }
76  }
```

```
77
78  void procurar()
79  {
80      vector<size_t> vec;
81      string filename = "registro4.txt";
82      string search;
83      registro.open(filename.c_str(), ofstream::out | ofstream::app);
84
85      ifstream file(filename.c_str());
86      stringstream buffer;
87
88      buffer << file.rdbuf();
89      string str = buffer.str();
90
91      cout << "Digite o nome da pessoa que deseja procurar: \n";
92      cin >> search;
93      //cin.clear();
94      //cin.ignore(256, '\n');
95
96
97      encontrarTodasStr(vec, str , search);
98
99      for (size_t pos : vec)
100     {
101         cout << "-----------------------------------------\n";
102         cout << "Nome: ";
103         pos = mostrarAtributo(str,pos);
104         cout << "Sobrenome: ";
105         pos = mostrarAtributo(str, pos);
106         cout << "Endereço: ";
107         pos = mostrarAtributo(str, pos);
108         cout << "CEP: ";
109         pos = mostrarAtributo(str, pos);
110         cout << "Telefone: ";
111         mostrarAtributo(str, pos);
112         cout << "-----------------------------------------\n";
113     }
114
115     cout << "Pressione Enter pra voltar ao menu\n";
116     getchar();
117     registro.close();
118 }
119
```

Guarda registro:

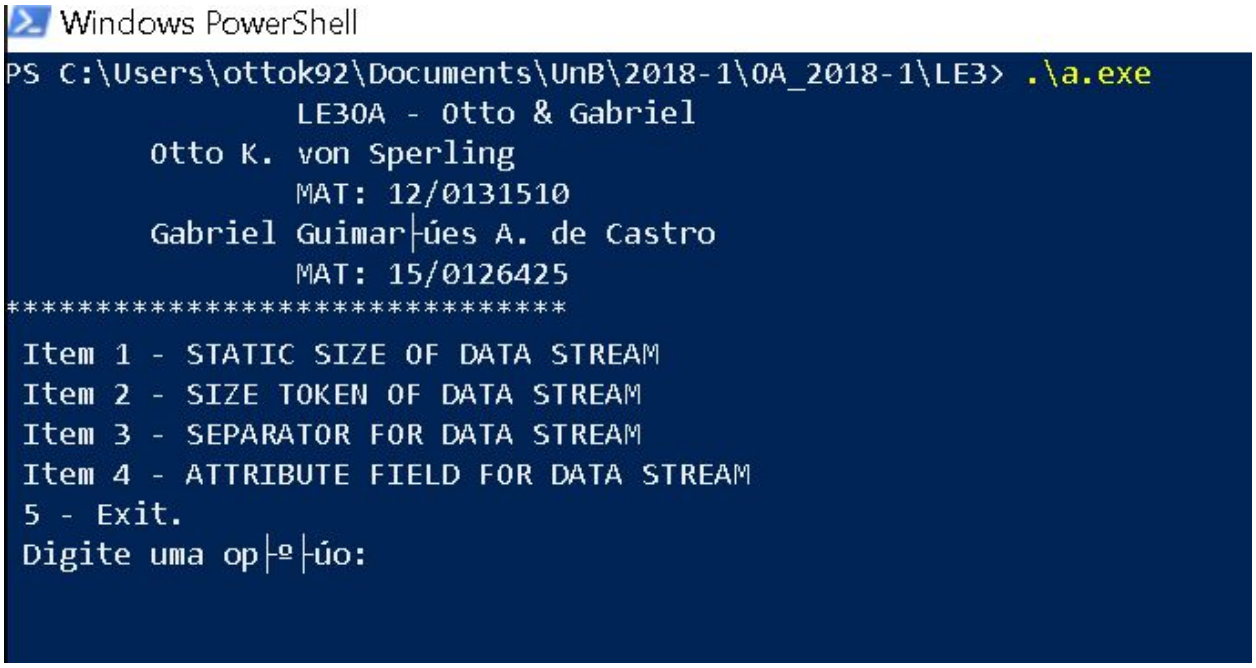
```
7   void guarda(std::string str)
8   {
9       string aux;
10      getline(std::cin, aux);
11      str += aux;
12      str += '|';
13      registro << str;
14  }
15
```

Menu

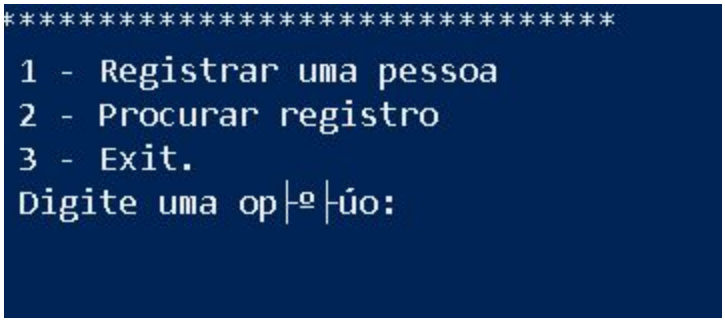
```
120
121 void menu_4()
122 {
123     int choice;
124     bool menu = true;
125     while (menu != false){
126         cout << "*********************************\n";
127         cout << " 1 - Registrar uma pessoa\n";
128         cout << " 2 - Procurar registro\n";
129         cout << " 3 - Exit.\n";
130         cout << " Digite uma opção: ";
131
132         cin >> choice;
133         cin.clear();
134         cin.ignore(256, '\n');
135
136         switch (choice)
137         {
138             case 1:
139             atributo();
140             break;
141
142             case 2:
143             procurar();
144             break;
145
146             case 3:
147             menu = false;
148             break;
149
150             default:
151             cout << "Escolha não válida \n";
152             cout << "Escolha novamente\n";
153             cin >> choice;
154             cin.clear();
155             cin.ignore(256, '\n');
156             break;
157         }
158     }
159 }
160
```

Exercício - *RUNTIME*

Main menu



Sub-menu para cada item.



Registrando um usuário

```
 1 - Registrar uma pessoa
 2 - Procurar registro
 3 - Exit.
 Digite uma op├º├úo: 1
Sobrenome = Sperling
Nome = Otto
Endereco = SQNSQN
CEP = 77777-777
Telefone = 99999-9999
```

Imprimindo dados na tela

```
 1 - Registrar uma pessoa
 2 - Procurar registro
 3 - Exit.
 Digite uma op├º├úo: 2


Sobrenome: Sperling
Nome: Otto
Endereco: SQNSQN
CEP: 77777-777
Telefone: 99999-9999
```