



Universidade Federal de Sergipe

PROGRAMAÇÃO II

PROJETO 2

SISTEMAS DE INFORMAÇÃO

GRUPO I

CADASTRO DE CLIENTES

COMPONENTES: EDUARDO AFONSO SOBRASANTOS
CARLOS EDUARDO PEREIRA SILVA
FELIPE MENDONÇA SACRAMENTO
JOSÉ GEILSON DOS SANTOS
SAMUEL SILVA DOS ANJOS



BIBLIOTECAS UTILIZADAS

- `#include <iostream>`
- `#include <locale.h>`
- `#include <fstream>`
- `#include <windows.h>`
- `#include <conio2.h>`
- `#include <stdio.h>`
- `#include <sstream>`
- `#include <time.h>`
- `#include "cpf.h"`
- `#include "clientes.h"`



ARQUIVO SEQUENCIAL

- `void atualizarCliente(clienteStr *clienteEntrada) {`
- `fstream clientesARQ;`
- `ofstream clientesARQSaida;`
- `clienteStr retorno;`
- `string linha;`
- `clientesARQ.open("clientes.dat",ios::in);`
- `clientesARQSaida.open("clientesCopia.dat",ios::out | ios::trunc);`

FUNÇÃO "CADASTAR CLIENTES": PONTEIRO, CRIAÇÃO DE ARQUIVO.



```
void cadastrarCliente(cliente *cliente) {  
    fstream clientesARQ;  
    time_t rawtime;  
    struct tm *info;  
    time(&rawtime);  
    info = gmtime(&rawtime );  
    cliente->diaEntrada = info->tm_mday;  
    cliente->mesEntrada = info->tm_mon+1;  
    cliente->anoEntrada = 1900+info->tm_year;  
    clientesARQ.open("clientes.dat",ios::app);  
    clientesARQ    << cliente->nome << ","  
                    << cliente->cpf << ","  
                    << cliente->senha << ","  
                    << cliente->idade << ","  
                    << cliente->profissao << ","... << endl;  
    clientesARQ.close();}
```

TELA PRINCIPAL



TELA DE CADASTRO DE CLIENTES

NOME :

CPF :

NACIONALIDADE :

PROFISSAO:

ENDERECO:

IDADE :

SEXO [M|F|O]:

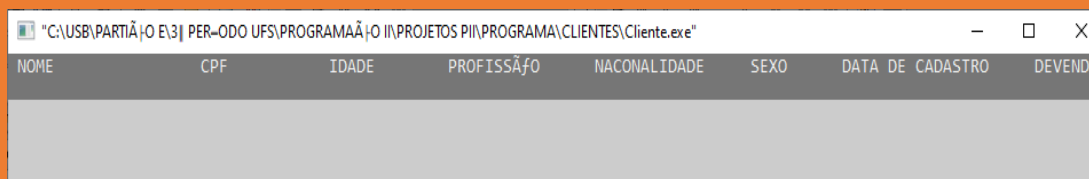
FUNÇÃO "LISTAR CLIENTES":

ABERTURA DE ARQUIVO PARA LEITURA

```
void listarClientes() {  
    system("cls");  
    fstream clientesARQ;  
    clienteStr retorno;  
    string linha = "";  
    int verificacao, linhaMostrar = 2;
```

```
    clientesARQ.open("clientes.dat",ios::in);
```

NOME	CPF	IDADE	PROFISSÃO	NACIONALIDADE	SEXO	DATA DE CADASTRO
carlos	12345678909	61	Estudante	Brasileiro	masculino	6/4/2020



NOME	CPF	IDADE	PROFISSÃO	NACIONALIDADE	SEXO	DATA DE CADASTRO	DEVENDO
------	-----	-------	-----------	---------------	------	------------------	---------

FUNÇÃO MODIFICAR DADOS DO CLIENTE

```
• void atualizarClienteController() {  
  • string cpf;  
  • clienteStr clienteAtualizar, novoCliente;  
  
  • textbackground(WHITE);  
  • textcolor(LIGHTBLUE);  
  • system("cls");  
  • gotoxy(45,15);  
  • cout << "INFORME O CPF DO CLIENTE: ";  
  • textbackground(LIGHTGRAY);  
  • cout << "          ";  
  • textcolor(BLACK);  
  • gotoxy(72,15);  
  • cin >> cpf;  
  • fflush(stdin);  
  • textbackground(WHITE);  
  
  • clienteAtualizar = buscarCliente(cpf);  
  • system("cls");  
}
```

FUNÇÃO
MODIFICAR
DADOS DO
CLIENTE

TELA -> INFORME O CPF DO CLIENTE: 12345678909



NOME:	carlos		
CPF:	12345678909	NACIONALIDADE:	Brasileiro
PROFISSAO:	Estudante		
ENDERECO:	Itabaiana		
IDADE:	61	SEXO [M F O]:	M

NOME:	Carlos		
CPF:	12345678909	NACIONALIDADE:	Uruguaio
PROFISSAO:	Estudante		
ENDERECO:	Itabaiana		
IDADE:	61	SEXO [M F O]:	M



TELA-> USUÁRIO ATUALIZADO

FUNÇÃO "MODIFICAR DADOS DO CLIENTE"

Atualizar Cliente

```
void atualizarCliente(clienteStr *clienteEntrada)
{
    fstream clientesARQ;
    ofstream clientesARQSaida;
    clienteStr retorno;
    string linha;

    clientesARQ.open("clientes.dat", ios::in);

    clientesARQSaida.open("clientesCopia.dat", ios::out
| ios::trunc);

    while (getline(clientesARQ, linha))
    {
        separarValoresClientes(linha, retorno);
        if (clienteEntrada-> cpf != retorno.cpf)
        {
            clientesARQSaida << retorno.nome << ","
                << retorno.cpf << ","
                << retorno.senha << ","
                << retorno.idade << ","
                << retorno.profissao << ","
                << retorno.nacionalidade << ","
                << retorno.sexo << ","
                << retorno.endereco << ","
                << retorno.diaEntrada << ","
                << retorno.mesEntrada << ","
                << retorno.anoEntrada << ","
                << retorno.consumo << ";" << endl;
        }
    }
}
```

Continuação:

Atualizar Cliente

•if (clienteEntrada->cpf == retorno.cpf)

• {
•
• clientesARQSaida << clienteEntrada->nome << ","
• << clienteEntrada->cpf << ","
• << clienteEntrada->senha << ","
• << clienteEntrada->idade << ","
• << clienteEntrada->profissao << ","
• << clienteEntrada->nacionalidade << ","
• << clienteEntrada->sexo << ","
• << clienteEntrada->endereco << ","
• << clienteEntrada->diaEntrada << ","
• << clienteEntrada->mesEntrada << ","
• << clienteEntrada->anoEntrada << ","
• << clienteEntrada->consumo << ";" << endl;
• }
• }

FUNÇÃO "EXCLUIR CLIENTE"

TELA -> INFORME O CPF DO CLIENTE: 12345678909



A screenshot of a graphical user interface for a client management system. The form is titled 'INFORME O CPF DO CLIENTE: 12345678909'. It contains several input fields: 'NOME:' with the value 'carlos', 'CPF:' with '12345678909', 'NACIONALIDADE:' with 'Brasileiro', 'PROFISSAO:' with 'Estudante', 'ENDERECO:' with 'Itabaiana', 'IDADE:' with '61', and 'SEXO [M|F|O]:' with 'M'. The form has a blue border and a light gray background.



Pressione qualquer tecla para continuar. . .

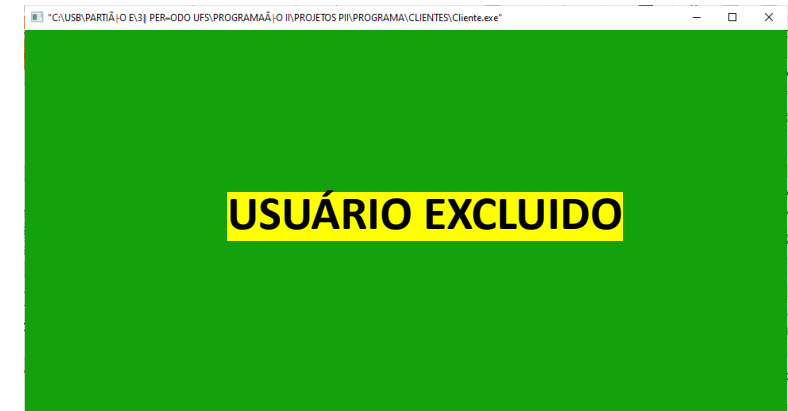
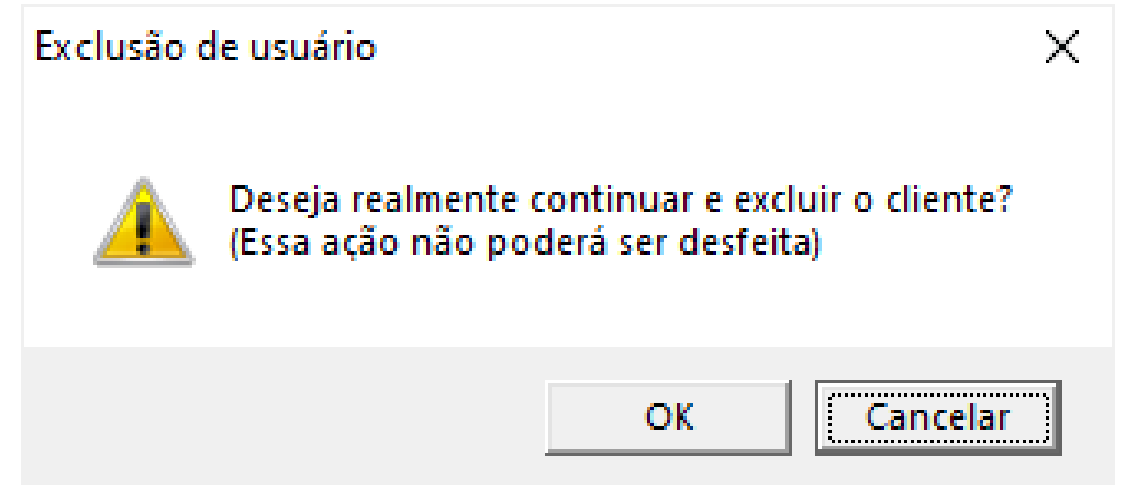
```
void excluirClienteController() {  
    string cpf;  
    clienteStr clienteExcluir;  
  
    setlocale(LC_ALL, "portuguese");  
  
    textbackground(WHITE);  
    textcolor(LIGHTBLUE);  
    system("cls");  
    gotoxy(45,15);  
    cout << "INFORME O CPF DO CLIENTE:"  
";  
    textbackground(LIGHTGRAY);  
    cout << "  
";  
    textcolor(BLACK);  
    gotoxy(72,15);  
    cin >> cpf;  
    fflush(stdin);  
    textbackground(WHITE);
```

FUNÇÃO "EXCLUIR CLIENTE"

```
clienteExcluir = buscarCliente(cpf);  
system("cls");  
imprimeCampoPadrao(32,12);
```

TELA -> INFORME O CPF DO CLIENTE: 12345678909

```
int opc = MessageBox(  
    NULL,  
    "Deseja realmente continuar e excluir o cliente?\n(Essa ação não  
poderá ser desfeita)",  
    "Exclusão de usuário",  
    MB_ICONWARNING | MB_OKCANCEL | MB_DEFBUTTON2  
);
```



```
#include <iostream>
#include <locale.h>
#include <fstream>
#include <stdio.h>
#include <sstream>
#include <time.h>
using namespace std;
struct cliente{
    char cpf[10];
    string senha;
    string nome;
    int idade;
    string profissao;
    string nacionalidade;
    string sexo;
    string endereco;
    int diaEntrada,
    mesEntrada, anoEntrada;
    float consumo;
};
```

ARQUIVO DE CABEÇALHO DE CLASSE

#include "clientes.h"

```
struct clienteStr{
    string cpf;
    string senha;
    string nome;
    int idade;
    string profissao;
    string nacionalidade;
    string sexo;
    string endereco;
    int diaEntrada,
    mesEntrada, anoEntrada;
    float consumo;
    int quarto_aux;
};
```

ESTRUTURA



ARQUIVO DE CABEÇALHO DE CLASSE #include "clientes.h"

```
•int position = 1;
• setlocale(LC_ALL, "portuguese"); //Define a linguagem padrão

• system("MODE con cols=120 lines=30 "); //Define o tamanho da CMD)

• while(!GetAsyncKeyState(VK_ESCAPE)) {//Aqui fica toda a navegação por tec
• interfacepadraoClientes(position);
• while(!(GetAsyncKeyState(VK_RETURN) &&
  GetAsyncKeyState(VK_CONTROL))) {

•   if(GetAsyncKeyState(VK_LEFT) && position >1){
•     position--;
•     interfacepadraoClientes(position);
•   }
•   if(GetAsyncKeyState(VK_RIGHT) && position <4){
•     position++;
•     interfacepadraoClientes(position);
•   }
•   if(GetAsyncKeyState(VK_ESCAPE)){
•     exit(1);
•   }
•   Sleep(100);
• }
```


Continuação: ARQUIVOS DE CABEÇALHOS DE CLASSE.

#include "cliente.h"

```
•if(position == 1) {  
•    interfaceECadastroCliente();  
•    }else if(position == 2) {  
•        listarClientes();  
•    }else if(position == 3) {  
•        atualizarClienteController();  
•    }else if(position == 4) {  
•        excluirClienteController();  
•    }else if(position == 5) {  
•        break;  
•    }  
•    Sleep(500);  
• }  
• }
```

ARQUIVOS DE CABEÇALHOS DE CLASSE.

#include "cpf.h"

```
•include <stdio.h>
```

```
•#include <iostream>
```

```
•#include <string.h>
```

```
•using namespace std;
```

```
•///RECEBE UM ARRAY DE 11 ESPAÇOS QUE CONTEM O CPF SEM CARACTERES ESPECIAIS
```

```
•///RETORNA UM VALOR DO TIPO BOOL QUE INFORMA SE O CPF E VALIDO OU NÃO
```

```
•bool verificarCpf(char cpf[10]) {
```

```
•    bool verificador = true;
```

```
•    int soma = 0;
```

```
•    int numero = 0;
```

```
•    int j = 0;
```

```
•    for (int i = 10; i >= 2; i--)
```

```
•    {
```

```
•        numero = cpf[j] - '0';
```

```
•        soma = soma + (i * numero);
```

```
•        j++;
```

```
•    }
```

```
• }
```

```
• //FIM DA PRIMEIRA CAMADA DE VERIFICAÇÃO
```

Continuação : ARQUIVO DE CABEÇALHO DE CLASSE

#include "cpf.h"

```
soma = soma*10%11;
    if(soma==10){soma = 0;}
    if(soma != (cpf[10] - '0')) {
        verificador = false;
    } //FIM DA SEGUNDA CAMADA DE VERIFICAÇÃO

    if(strlen(cpf) != 11)
        verificador = false; //FIM DA TERCEIRA VERIFICAÇÃO
    return verificador; //FIM DA VERIFICAÇÃO DE CPF

//INICIO DA TRANSFORMAÇÃO DE STRING PARA UM VETOR DE CHAR

void stringToCharVector(string& vetorEntrada, char* vetorSaida) {
    strcpy(vetorSaida,vetorEntrada.c_str());
}
```



NOSSOS AGRADECIMENTOS A TODOS !