Livro Estrutura de Dados (EDDA2)

Professor: Jose Oscar Machado Alexandre

Aluno: Matheus Savoia Resende – SP3097781

**Sumário**

1. Estudos na linguagem C

2. Algoritmos de Ordenação

2.1 Bubble Sort

2.2 Insertion Sort

2.3 Select Sort

***1. Estudos na linguagem C***

Para se fazer os estudos na linguagem C, estou utilizando o livro: C Completo e Total, Hebert Schildt. E a apostila “Projeto de Algoritmos (em C)” por Paulo Feofiloff.

Abaixo segue o projeto que acredito ter sido o mais avançado nos meus conhecimentos sobre a linguagem C. O código dele é referente a um sistema de contas bancárias, onde pode fazer diversas ações como: criar uma conta, listar titular, depositar e sacar.

#include <stdio.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#define max\_cadastros 200

typedef struct{

char nomeC[50], nomeP[50];

int ativoC, ativoP,titularesC[4], titularesP[4], saldoC[50], saldoP[50];

}conta;

conta cadastros[max\_cadastros];

void menu();

void criarConta();

void listarTitular();

void depositar();

void sacar();

int main()

{

menu();

return 0;

}

void menu()

{

int opcao;

do

{

system("cls");

printf("MENU");

printf("\n1 - Criar conta");

printf("\n2 - Alterar o tipo da conta");

printf("\n3 - Depositar");

printf("\n4 - Sacar");

printf("\n5 - Lista de titulares");

printf("\nDigite o numero da operacao desejada: ");

scanf("\n%d", &opcao);

getchar();

switch(opcao)

{

case 1:

criarConta();

break;

case 2:

break;

case 3:

depositar();

break;

case 4:

sacar();

break;

case 5:

listarTitular();

break;

}

}while(opcao != 0);

}

void criarConta()

{

char nomeC[50], nomeP[50];

int ativoC, ativoP, opConta, opcao, titularesC[4], titularesP[4], saldoC[50], saldoP[50];

do

{

system("cls");

printf("CRIACAO CONTA");

printf("\n1 - Conta Corrente");

printf("\n2 - Conta Poupanca\n");

printf("\nDigite a opcao: ");

scanf("%d", &opConta);

getchar();

if(opConta == 1)

{

system("cls");

printf("Digite o seu nome completo: ");

fgets(nomeC, sizeof(nomeC), stdin);

printf("Deseja ter titulares? \n1 - Sim \n2 - Nao");

printf("\nDigite a opcao: ");

scanf("%d", &opcao);

if(opcao == 1)

{

printf("Digite o CPF do primeiro titular: ");

scanf("%d", &titularesC[0]);

printf("Digite o CPF do segundo titular: ");

scanf("%d", &titularesC[1]);

printf("Digite o CPF do terceiro titular: ");

scanf("%d", &titularesC[2]);

printf("Digite o CPF do quarto titular: ");

scanf("%d", &titularesC[3]);

}

for(int i = 0; i < max\_cadastros; i++)

{

if(cadastros[i].ativoC == 0)

{

strcpy(cadastros[i].nomeC, nomeC);

cadastros[i].titularesC[0] = titularesC[0];

cadastros[i].titularesC[1] = titularesC[1];

cadastros[i].titularesC[2] = titularesC[2];

cadastros[i].titularesC[3] = titularesC[3];

cadastros[i].ativoC = 1;

cadastros[i].saldoC[i] = 0;

break;

}

}

}

else

{

system("cls");

printf("Digite o seu nome completo: ");

fgets(nomeP, sizeof(nomeP), stdin);

printf("Deseja ter titulares? \n1 - Sim \n2 - Nao");

printf("\nDigite a opcao: ");

scanf("%d", &opcao);

if(opcao == 1)

{

printf("\nDigite o CPF do primeiro titular: ");

scanf("%d", &titularesP[0]);

printf("Digite o CPF do segundo titular: ");

scanf("%d", &titularesP[1]);

printf("Digite o CPF do terceiro titular: ");

scanf("%d", &titularesP[2]);

printf("Digite o CPF do quarto titular: ");

scanf("%d", &titularesP[3]);

}

for(int i = 0; i < max\_cadastros; i++)

{

if(cadastros[i].ativoP == 0)

{

strcpy(cadastros[i].nomeP, nomeP);

cadastros[i].titularesP[0] = titularesP[0];

cadastros[i].titularesP[1] = titularesP[1];

cadastros[i].titularesP[2] = titularesP[2];

cadastros[i].titularesP[3] = titularesP[3];

cadastros[i].ativoP = 1;

cadastros[i].saldoP[i] = 0;

break;

}

}

}

printf("\nCliente cadastrado com sucesso!\n");

printf("\n1 - Continuar\n0 - Sair");

printf("\nDigite a opcao: ");

scanf("%d", &opcao);

getchar();

}while (opcao != 0);

}

void alterar()

{

}

void sacar()

{

char nomeC[50], nomeP[50];

int ativoC, ativoP, opcao, opConta, saldoC[50], saldoP[50], saque;

do

{

system("cls"); //windows

printf("Digite o seu tipo de conta: \n1 - Conta Corrente \n2 - Conta Poupanca\n");

scanf("%d", &opConta);

getchar();

if(opConta == 1)

{

printf("\nDigite o nome completo do cliente para sacar: ");

fgets(nomeC, sizeof(nomeC), stdin);

printf("Digite o valor do deposito: ");

scanf("%d", &saque);

for (int i = 0; i < max\_cadastros; i++)

{

if (strstr(cadastros[i].nomeC, nomeC) != NULL)

{

cadastros[i].saldoC[i] = cadastros[i].saldoC[i] - saque;

printf("Saldo da conta: %d", cadastros[i].saldoC[i]);

}

}

}

else

{

printf("\nDigite o nome completo do cliente para depositar: ");

fgets(nomeP, sizeof(nomeP), stdin);

printf("Digite o valor do deposito: ");

scanf("%d", &saque);

for (int i = 0; i < max\_cadastros; i++)

{

if (strstr(cadastros[i].nomeP, nomeP) != NULL)

{

cadastros[i].saldoP[i] = cadastros[i].saldoP[i] - saque;

printf("Saldo da conta: %d", cadastros[i].saldoP[i]);

}

}

}

printf("\nDigite 0 para sair ou 1 para outra lista: ");

scanf("%d", &opcao);

getchar();

}while(opcao != 0);

}

void depositar()

{

char nomeC[50], nomeP[50];

int ativoC, ativoP, opcao, opConta, saldoC[50], saldoP[50], deposito;

do

{

system("cls"); //windows

printf("Digite o seu tipo de conta: \n1 - Conta Corrente \n2 - Conta Poupanca\n");

scanf("%d", &opConta);

getchar();

if(opConta == 1)

{

printf("\nDigite o nome completo do cliente para depositar: ");

fgets(nomeC, sizeof(nomeC), stdin);

printf("Digite o valor do deposito: ");

scanf("%d", &deposito);

for (int i = 0; i < max\_cadastros; i++)

{

if (strstr(cadastros[i].nomeC, nomeC) != NULL)

{

cadastros[i].saldoC[i] = cadastros[i].saldoC[i] + deposito;

printf("Saldo da conta: %d", cadastros[i].saldoC[i]);

}

}

}

else

{

printf("\nDigite o nome completo do cliente para depositar: ");

fgets(nomeP, sizeof(nomeP), stdin);

printf("Digite o valor do deposito: ");

scanf("%d", &deposito);

for (int i = 0; i < max\_cadastros; i++)

{

if (strstr(cadastros[i].nomeP, nomeP) != NULL)

{

cadastros[i].saldoP[i] = cadastros[i].saldoP[i] + deposito;

printf("Saldo da conta: %d", cadastros[i].saldoP[i]);

}

}

}

printf("\nDigite 0 para sair ou 1 para outra lista: ");

scanf("%d", &opcao);

getchar();

}while(opcao != 0);

}

void listarTitular()

{

char nomeC[50], nomeP[50];

int ativoC, ativoP, opcao,opConta,titularesC[4], titularesP[4];

do

{

system("cls");

printf("Digite o seu tipo de conta: \n1 - Conta Corrente \n2 - Conta Poupanca\n");

scanf("%d", &opConta);

getchar();

if(opConta == 1)

{

printf("\nDigite o nome completo do cliente para pesquisar: ");

fgets(nomeC, sizeof(nomeC), stdin);

for (int i = 0; i < max\_cadastros; i++)

{

if (strstr(cadastros[i].nomeC, nomeC) != NULL)

{

printf("\nNome do dono: %s\n", cadastros[i].nomeC);

printf("CPF do primeiro titular: %d\n", cadastros[i].titularesC[0]);

printf("CPF do segundo titular: %d\n", cadastros[i].titularesC[1]);

printf("CPF do terceiro titular: %d\n", cadastros[i].titularesC[2]);

printf("CPF do quarto titular: %d\n", cadastros[i].titularesC[3]);

printf("-----------\n");

}

}

}

else

{

printf("\nDigite o nome completo do cliente para pesquisar: ");

fgets(nomeP, sizeof(nomeP), stdin);

for (int i = 0; i < max\_cadastros; i++)

{

if (strstr(cadastros[i].nomeP, nomeP) != NULL)

{

printf("\nNome do dono: %s\n", cadastros[i].nomeP);

printf("CPF do primeiro titular: %d\n", cadastros[i].titularesP[0]);

printf("CPF do segundo titular: %d\n", cadastros[i].titularesP[1]);

printf("CPF do terceiro titular: %d\n", cadastros[i].titularesP[2]);

printf("CPF do quarto titular: %d\n", cadastros[i].titularesP[3]);

printf("-----------\n");

}

}

}

printf("\nDigite 0 para sair ou 1 para outra lista: ");

scanf("%d", &opcao);

getchar();

} while (opcao != 0);

}

***2. Ordenação***

**2.1 Bubble Sort**

O código abaixo faz uma aplicação do algoritmo de ordenação bubble sort, o objetivo do programa é pegar 100 elementos aleatoriamente que estão entre 0 à 1000, e ordená-los de forma ascendente e descendente:

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#define TAM 100

void BubbleSort(int x[TAM], int temp);

void Ascendente(int x[TAM], int temp);

void Descendente(int x[TAM], int temp);

int main()

{

setlocale(LC\_ALL, "PORTUGUESE");

int x[TAM];

int temp,p;

printf("| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | BUBBLE SORT | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |\n");

printf("\nGeração aleatória de 100 números(valores entre 0-1000):\n");

srand(time(NULL));

for(p = 0; p < TAM; p++){

int result = rand() % 1000; //gerando valores aleatórios entre zero e 1000

printf(" %d ", result);

x[p] = result;

}

BubbleSort(x, temp);

return 0;

}

void BubbleSort(int x[TAM], int temp)

{

Ascendente(x, temp);

Descendente(x, temp);

}

void Ascendente(int x[TAM], int temp)

{

for(int j = 0; j < TAM; j++){

for(int i = 0; i < TAM - 1; i++){

if(x[i] > x[i+1]){

temp = x[i];

x[i] = x[i+1];

x[i+1] = temp;

}

}

}

printf("\n");

printf("\nOrdenação ascendente:\n");

for(int i = 0; i < TAM; i++){

printf(" %d ", x[i]);

}

printf("\n");

}

void Descendente(int x[TAM], int temp)

{

for(int j = 0; j < TAM; j++){

for(int i = 0; i < TAM; i++){

if(x[i] < x[i+1]){

temp = x[i];

x[i] = x[i+1];

x[i+1] = temp;

}

}

}

printf("\n");

printf("Ordenação descendente:\n");

for(int i = 0; i < TAM; i++){

printf(" %d ", x[i]);

}

printf("\n");

}

**2.2 Insertion Sort**

**2.3 Select Sort**