ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Estruturas de dados 2022/2023

Trabalho Prático nº 2

Registo de Compras (Continuação)

Elaborado em: 29/06/2023

Guilherme Afonso Ferreira Gonçalves a2022156457

Índice

1. Introdução

Trabalho realizado no âmbito da unidade curricular de Estruturas de Dados, lecionada pela professora Celia Pereira.

Para este segundo trabalho prático, foi-nos proposto a partir do primeiro trabalho alterar alguns requisitos. Por exemplo, as compras e os vales tinham de ser armazenados em listas ligadas.

De seguida apresentarei o programa e explicarei um pouco daquilo que fiz ao longo do código. Como a forma do trabalho não difere muito do primeiro para este segundo, acho que não é necessário fazer um "manual de utilizador", visto que é praticamente o mesmo do primeiro trabalho.

Para cada opção de escolha, estará associado o subprograma, ou seja, uma função para cada opção de escolha. Cada função terá um nome com relação à opção em si.

No final do trabalho apresentarei as dificuldades que tive para a realização deste trabalho.

2. Apresentação do Programa

A primeira coisa que fiz para este trabalho foi criar as listas ligadas. Para isso defini 3 estruturas, uma para os vales, compras e a outra que já tinha para o outro trabalho dos clientes. Como só era necessário ter listas ligadas para as compras e vales, decidi manter o array com os clientes e apenas colocar as listas ligadas dentro de cada cliente.

```
typedef struct
{
    int valor;
    struct Vales * proxVale;
} Vales;

typedef struct
{
    float valor;
    int num_cliente, loja, dia, mes, ano;
    struct Compra * proxCompra;
} Compra;

typedef struct
{
    char nome[50], contacto[10], email[50], nif[10];
    int num_cliente, num_compras, num_vales, vales_usados, ativo;
    double valorTotalCompras;
    struct Vales * vale;
    struct Compra * compras;
} Cliente;
```

Figura 1 - Criação das listas ligadas

Após criar as estruturas para as listas ligadas, fiz a mesma função para adicionar um cliente que automaticamente tem cartão. Ou seja, o mesmo que tinha feito para o trabalho 1, que era perguntar ao utilizar os dados e guardá-los no array dos clientes no seu devido lugar. Para este segundo trabalho não criei nenhuma função para adicionar um cliente que não tenha cartão, visto que no primeiro trabalho tinha armazenado os dados deste num array de compras. E como este era numa lista ligada, não percebi muito bem como iria fazer visto que coloquei a lista ligada dentro o array dos clientes.

```
int registarNovoClienteComCartao(Cliente clientes[50], int ncliente)
{

    printf("**Pressione enter para adicionar cliente**\n");
    fflush(stdin);
    getchar();

    do
    {
        printf("Nome: ");
        fgets(clientes[ncliente].nome, 50, stdin);
        clientes[ncliente].nome[strlen(clientes[ncliente].nome) - 1] = 0;
    } while (strlen(clientes[ncliente].nome) == 0);

    do
    {
        printf("Contacto: ");
        scanf("%s", clientes[ncliente].contacto);
    } while (strlen(clientes[ncliente].contacto) < 9 || strlen(clientes[ncliente].contacto) > 9);

    fflush(stdin);

    do
    {
        printf("Email: ");
        fgets(clientes[ncliente].email, 50, stdin);
        clientes[ncliente].email[strlen(clientes[ncliente].email) - 1] = 0;
    } while (strlen(clientes[ncliente].email) == 0);

    do
```

Figura 2 - Função registar cliente (incompleta nesta imagem)

Depois mais três funções iguais ao trabalho 1, a de remover um cliente ou seja torna-lo inativo, outra para listar os clientes ativos e outra para editar os dados de um cliente.

```
void removerCliente(Cliente clientes[50], int num_clientes)
{
    int numCliente;
    printf("Introduza o numero do cliente que pretende remover: ");
    scanf("%d", &numCliente);

    for (int i = 0; i < num_clientes; i++)
    {
        if (clientes[i].num_cliente == numCliente && clientes[i].ativo)
        {
            clientes[i].ativo = 0;
            printf("Cliente removido com sucesso.\n");
            return;
        }
    }

    printf("Cliente nao encontrado. Verifique se inseriu o numero correto ou se o cliente existe.\n");
    printf("\n");
}
```

Figura 3 - Função Remover Cliente

```c
void listarClientes(Cliente clientes[50], int numCliente)
{
  int contador = 0;

    for (int i = 0; i < numCliente - 1; i++)
    {
        for (int j = 0; j < numCliente - i - 1; j++)
        {
            if (clientes[j].num_cliente > clientes[j + 1].num_cliente)
            {
                Cliente aux = clientes[j];
                clientes[j] = clientes[j + 1];
                clientes[j + 1] = aux;
            }
        }
    }

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

    for (int i = 0; i < numCliente; i++)
    {
        if (clientes[i].ativo == 1)
        {
            printf("Numero de cliente: %d\n", clientes[i].num_cliente);
            printf("Nome: %s\n", clientes[i].nome);
            printf("Contacto: %s\n", clientes[i].contacto);
            printf("Email: %s\n", clientes[i].email);
            printf("NIF: %s\n", clientes[i].nif);
            printf("\n");
            contador++;
        }
    }
```

Figura 4 - Função Listar Clientes

```c
void editarDadosCliente(Cliente clientes[50], int nClientes)
{
    int numCliente;

    printf("Introduza o numero do cliente a ser editado: ");
    scanf("%d", &numCliente);

    for (int i = 0; i < nClientes; i++)
    {
        if (clientes[i].num_cliente == numCliente && clientes[i].ativo == 1)
        {
            fflush(stdin);

            do
            {
                printf("Nome: ");
                fgets(clientes[i].nome, 50, stdin);
                clientes[i].nome[strlen(clientes[i].nome) - 1] = 0;
            } while (strlen(clientes[i].nome) == 0);

            do
            {
                printf("Contacto: ");
                scanf("%s", clientes[i].contacto);
            } while (strlen(clientes[i].contacto) < 9 || strlen(clientes[i].contacto) > 9);

            fflush(stdin);

            do
            {
                printf("Email: ");
```

Figura 5 - Função Editar Dados (incompleta na imagem)

Agora começam as funções com as listas ligadas. Primeiro a função para visualizar as compras de um determinado cliente. Em que para isso criei um ponteiro a apontar para as compras do cliente e caso a lista não fosse nula, ou seja, caso houvesse uma compra, este iria escrever os dados da compra.

```c
void visualizarComprasCliente(Cliente clientes[50], int num_clientes)
{
    int numCliente;
    printf("Introduza o numero do cliente que pretende exibir as compras: ");
    scanf("%d", &numCliente);

    for (int i = 0; i < num_clientes; i++)
    {
        if (clientes[i].num_cliente == numCliente && clientes[i].ativo == 1)
        {
            printf("\nCompras realizadas pelo cliente:\n");
            printf ("----------------------------------------\n");

            Compra* compra = clientes[i].compras;

            if (compra == NULL)
            {
                printf ("Nenhuma compra realizada");
                printf ("\n");
            }

            while (compra != NULL)
            {
                printf("Valor: %.2f euros\n", compra->valor);
                printf("Data: %02d/%02d/%04d\n", compra->dia, compra->mes, compra->ano);
                printf("Loja: %d\n", compra->loja);
                printf("\n");

                compra = compra->proxCompra;
            }
            return;
        }
```

Figura 6 - Função Visualizar Compras

Depois da função visualizar compras, temos a função para guardar as compras na lista ligada das compras de cada cliente. Para isso criei um ponteiro e aloquei memória utilizando a função malloc. Após isto guardei os dados introduzidos pelo utilizador. Para guardar os vales fiz a mesma coisa, mas ao longo da execução notei que por exemplo, após chegar aos primeiros 100 euros de compras, mesmo que o utilizador não passasse os outros 100 (visto que eu tenho que a cada 100 euros o utilizador ganha um vale) continuava a ganhar um vale.

```
void registarCompraCliente(Cliente* clientes, int num_clientes)
{
    int numCliente;
    printf("Introduza o numero do cliente que pretende registar a compra: ");
    scanf("%d", &numCliente);

    for (int i = 0; i < num_clientes; i++)
    {
        if (clientes[i].num_cliente == numCliente && clientes[i].ativo == 1)
        {
            Compra* novaCompra = (Compra*)malloc(sizeof(Compra));

            printf("Introduza o valor da compra: ");
            scanf("%f", &novaCompra->valor);

            fflush(stdin);

            printf("Introduza a data da compra (DD MM AAAA): ");
            scanf("%d %d %d", &novaCompra->dia, &novaCompra->mes, &novaCompra->ano);

            fflush(stdin);

            printf("Introduza o numero da loja: ");
            scanf("%d", &novaCompra->loja);

            novaCompra->proxCompra = clientes[i].compras;
            clientes[i].compras = novaCompra;

            novaCompra->valor -= (novaCompra->valor * 0.05);
            printf ("Desconto por ter cartao, aplicado na compra (5 porcento)!\n");

            clientes[i].valorTotalCompras += novaCompra->valor;
```

Figura 7 - Registar Compra (1)

```
    clientes[i].num_vales = numVales;

    Vales* novoVale = (Vales*)malloc(sizeof(Vales));

    if (clientes[i].num_vales > 0) {
        int escolhaVale;

        for (int j = 0; j < clientes[i].num_vales; j++) {
            novoVale->valor = 10;

            novoVale->proxVale = clientes[i].vale;
            clientes[i].vale = novoVale;
        }

        do
        {
            printf("Deseja utilizar um vale de desconto? (1 - Sim, 0 - Nao): ");
            scanf("%d", &escolhaVale);
        } while (escolhaVale < 0 || escolhaVale > 1);

        if (escolhaVale == 1 && clientes[i].vale != NULL) {
            Vales* primeiroVale = clientes[i].vale;
            clientes[i].vale = primeiroVale->proxVale;

            novaCompra->valor -= primeiroVale->valor;
            clientes[i].valorTotalCompras -= primeiroVale->valor;
            printf("Vale de 10 euros utilizado com sucesso.\n");

            free(primeiroVale);

            clientes[i].vales_usados++;
            clientes[i].num_vales--;
```

Figura 8 - Registar Compra (2)

O subprograma para mostrar as informações do cliente com o valor total de compras, etc, foi o mesmo que no trabalho 1.

```
void informacoesCliente(Cliente clientes[50], int num_clientes)
{
    int numCliente;
    printf("Introduza o numero do cliente: ");
    scanf("%d", &numCliente);

    for (int i = 0; i < num_clientes; i++)
    {
        if (clientes[i].num_cliente == numCliente && clientes[i].ativo == 1)
        {
            printf("\nValor total das compras realizadas: %.2lf euros\n", clientes[i].valorTotalCompras);
            if (clientes[i].num_compras > 0)
                printf("Media de cada compra realizada: %.2f euros\n", clientes[i].valorTotalCompras / clientes[i].num_compras);
            else
                printf("Media das compras realizadas: 0.00 euros\n");
            printf("Quantidade de vales usados: %d \n", clientes[i].vales_usados);
            if (clientes[i].num_vales > 0)
                printf("Existe(m) %d vale(s) disponiveis\n", clientes[i].num_vales);
            else
                printf("Nao existem vales disponiveis\n");
            return;
        }
    }

    printf("(Lista vazia ou cliente inativo ou invalido)\n");
}
```

Figura 9 - Informações Cliente

Para ordenar as compras dos clientes e após fazer várias pesquisas na internet e de ter usado várias funções, não consegui com que esta função funcionasse.

```
void ordenarComprasCliente(Cliente clientes[50], int num_clientes){
    int numCliente;
    printf("Introduza o numero do cliente: ");
    scanf("%d", &numCliente);

    for (int i = 0; i < num_clientes; i++) {
        if (clientes[i].num_cliente == numCliente && clientes[i].ativo == 1){

        Compra* compra = clientes[i].compras;
        Compra* lista = compra;
        Compra* maximo;
        Compra* anterior;

        if (compra == NULL){
            printf("Nenhuma compra realizada\n");
            return;
        }

        while (lista != NULL) {
            maximo = lista;
            Compra* atual = lista->proxCompra;
            anterior = lista;

            while (atual != NULL) {
                if (atual->valor > maximo->valor) {
                    maximo = atual;
                    anterior = lista;
                }
                atual = atual->proxCompra;
                lista = lista->proxCompra;
            }

            if (maximo != lista) {
```

Figura 10 - Ordenação de compras (incompleta na imagem)

No subprograma para criar um ficheiro csv foi relativamente rápido de fazer, visto que era parecido ao que fizemos no trabalho 1, mas apenas tendo que alterar a forma como iriamos percorrer cada compra de cada cliente e escrever.

```c
void criarCSVLojasIntervalo(Cliente clientes [50], int num_clientes) {
    char nomeFicheiro[50];
    int num_loja;
    int diaInicial, mesInicial, anoInicial, diaFinal, mesFinal, anoFinal;

    printf("Introduza o numero da loja: ");
    scanf("%d", &num_loja);

    printf ("Introduza a data inicial (DD MM AAAA): ");
    scanf ("%d %d %d", &diaInicial, &mesInicial, &anoInicial);

    printf ("Introduza a data final (DD MM AAAA): ");
    scanf ("%d %d %d", &diaFinal, &mesFinal, &anoFinal);

    sprintf(nomeFicheiro, "%d_%02d-%02d-%04d_%02d-%02d-%04d.csv",num_loja,diaInicial, mesInicial, anoInicial, diaFinal, mesFinal, anoFi

    FILE* ficheiro = fopen(nomeFicheiro, "w");
    if (ficheiro == NULL) {
        printf ("ERRO - O ficheiro para ler nao existe ou nao pode ser lido de momento");
        printf ("\n");
        return;
    }

    int cabecalho = 0;

    if (cabecalho == 0) {
    fprintf(ficheiro, "Data;Compra (euros);Numero Cliente\n");
    fprintf(ficheiro, "-----------------------------------------------\n");
    cabecalho = 1;
    }

    for (int i = 0; i < num_clientes; i++) {
```

Figura 11 - Ficheiro CSV (1)

```c
    if (cabecalho == 0) {
    fprintf(ficheiro, "Data;Compra (euros);Numero Cliente\n");
    fprintf(ficheiro, "-----------------------------------------------\n");
    cabecalho = 1;
    }

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

        if (clientes[i].compras == NULL)
            continue;

        Compra* compra = clientes[i].compras;

        while (compra != NULL) {
            if ((compra->dia >= diaInicial && compra->mes >= mesInicial && compra->ano >= anoInicial)&&(compra->mes <= mesFinal && comp
                if (compra->loja == num_loja) {
                    fprintf(ficheiro, "%04d%02d%02d;%.2f;%d\n", compra->dia, compra->mes, compra->ano, compra->valor, clientes[i].num_c
                }
            }
            compra = compra->proxCompra;
        }
    }

    fclose(ficheiro);
    printf("Ficheiro CSV criado com sucesso\n");
}
```

Figura 12 - Ficheiro CSV (2)

Por fim, faltavam guardar os dados e carregar os dados dos clientes e das compras. No entanto só consegui guardar os dados dos clientes e não das listas ligadas.

```c
void guardarDados (Cliente clientes [50], int totalClientes) {

    FILE* ficheiroClientes = fopen("dadosClientes.dat", "wb");
    if (ficheiroClientes == NULL) {
        printf("Erro ao guardar dados dos clientes.\n");
        return;
    }

    fwrite (&totalClientes, sizeof(int), 1, ficheiroClientes);
    fwrite(clientes, sizeof(Cliente), 50, ficheiroClientes);

    fclose(ficheiroClientes);

    printf("Dados guardados com sucesso\n");
}

int carregarDadosFicheiro (Cliente clientes [50]) {

    int totalClientes;

    FILE* ficheiroClientes = fopen("dadosClientes.dat", "rb");
    if (ficheiroClientes == NULL) {
        printf("Erro ao carregar dados dos clientes.\n");
        return 0;
    }

    fread (&totalClientes, sizeof(int), 1, ficheiroClientes);
    fread(clientes, sizeof(Cliente), 50, ficheiroClientes);


    fclose(ficheiroClientes);

    return totalClientes;
}
```

Figura 13 - Guardar Dados e Carregar

3. Conclusão

Ao longo deste trabalho encontrei vários problemas e tive várias dúvidas ao contrário do trabalho prático número 1. Acho que de uma maneira geral, a matéria sobre as listas ligadas ainda é muito confusa para mim, sendo mais fácil trabalhar com tabelas separadas.

Tal como disse ao longo do trabalho, não fiz um subprograma para adicionar as compras de uma pessoa que não seja cliente; a função para ordenar também não dá e a parte de guardar dados e carregar está incompleta, estes são de uma forma geral alguns dos problemas ao longo deste trabalho.

Concluindo, na minha opinião este trabalho está incompleto e necessitava um pouco mais de compreensão da matéria para o fazer, mesmo consultando alguns sites e vídeos disponibilizados na internet, fiquei algo confuso com algumas partes de código.

4. Referências

C program for deleting a node in a linked list (2022) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/c-program-for-deleting-a-node-in-a-linked-list/ (Accessed: 29 June 2023).

Taha HagarTaha Hagar 4944 bronze badges et al. (1968) Display a sorted linked list with every new element in C, Stack Overflow. Available at: https://stackoverflow.com/questions/71224915/display-a-sorted-linked-list-with-every-new-element-in-c (Accessed: 29 June 2023).