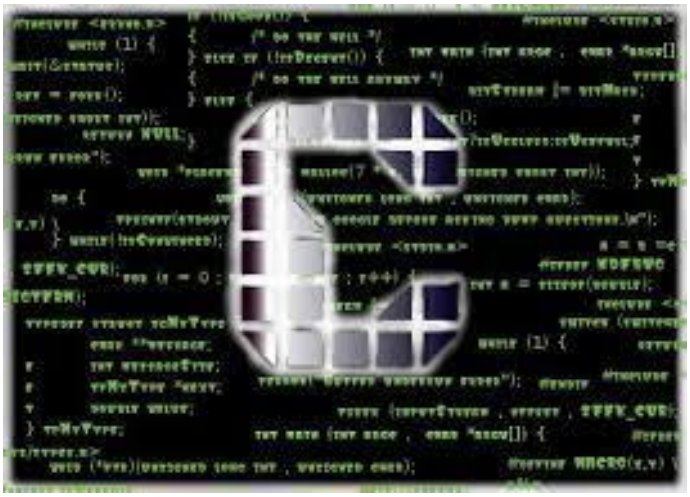
1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

dei departamento de engenharia informática

Relatório projeto

Princípios de Programação Procedimental

2021/2022



TOMÁS BERNARDO MARTINS DIAS

2020215701

1. INDICE

2. VISÃO GERAL SOBRE O PROGRAMA

O programa desenvolvido tem como objetivo gerir as contas do bar dos alunos de uma escola secundária. Para esse efeito é necessário guardar as informações sobre os mesmos.

As informações sobre os alunos e sobre as possíveis despesas são guardadas em listas simplesmente ligada.

É possível realizar varias operações tais como: verificar a data atual, mudar a data para um dia posterior á data atual (é impossível andar para trás no tempo), inserir e remover clientes, listar os clientes com saldo inferior a um determinado valor, sendo esta lista ordenada por ordem decrescente do valor do saldo, mostrar as informações sobre um cliente, efetuar uma despesa e por ultima pode carregar a conta de um cliente.

O cliente também pode fechar o programa e ao fazê-lo será verificado se existe informações sobre os clientes e caso existam estas serão guardadas. Estas informações são lidas no inicio do programa caso existam.

3. ESTRUTURAS DE DADOS

Para a implementação do programa foi necessário definir algumas estruturas:

- Data: usada para guardar o momento de quando uma despesa foi efetuada
- Student: usada para guardar os dados do cliente (usada nesta estrutura)
- Expense: usada para criar despesas
- Expense_list: lista ligada usada para guardar despesas, o seu no apresenta uma despesa e um ponteiro para a despesa seguinte. A lista alem do no que aponta para o inicio apresenta também o seu tamanho
- Client: estrutura principal que guarda os dados do cliente, o valor da sua conta e as suas despesas efetuadas
- Client_list: lista ligada usada para guardar os clientes, o seu no apresenta um cliente e um ponteiro para o cliente seguinte. A lista alem do no que aponta para o inicio apresenta também o seu tamanho

4. PRINCIPAIS FUNÇÕES

O programa apresenta variadas funções umas auxiliares e outras principais.

O programa está dividido em 4 ficheiros.c, o principal, um com as funções associadas as estruturas do programa, outro associado as funções principais do programa e finalmente um que apresenta as funções para leitura e escrita de informação num ficheiro binário.

Existem diversas funções auxiliares que permitem verificar o tipo de determinada string, por exemplo double, inteiro e char. Funções que permitem verificar se a data é anterior ou posterior á atual e se a data é valida.

O programa apresenta funções para inicializar, adicionar e remover elementos de uma lista. Também permite mostrar informações sobre um determinado no dessas listas. Como referido anteriormente existem 3 tipos de listas, 2 listas para clientes e 1 para despesas. As listas de clientes são ordenadas por ordem alfabética e por ordem descendente de saldo respetivamente. A lista de despesas segue a ordem First-In-First-Out, as primeiras a chegar são as primeiras da lista.

Todas as funções á exceção do inserir cliente, mostrar a lista de clientes e os clientes ordenados por saldo abaixo de um determinado valor, pedem o nome do cliente e verificam se o cliente existe, caso contrario mostram ao utilizador uma mensagem de erro. O nome pode ser inserido em minúsculas ou maiúsculas, e com espaços tanto no inicio como no final, pois estes são removidos e os caracteres passados a maiúsculas. É preciso ter em atenção aos espaços presente entre nomes, pois estes não são alterados.

```
/***
 * method to make a expense that receive a list where try to find client, and a expense list with expense.
 * if client dont exists or ID of product dont exists the operation is aborted.
 * @param list client list
 * @param explist expense list
 * @param todayDate date
 */
void makeExpense(client_list *list,expense_list *explist,data todayDate){
```

Figura 1- função para efetuar uma despesa

A função principal do programa é a que permite efetuar uma compra, como referido anteriormente esta pede o nome do cliente em questão, verifica se este existe, caso não existe a operação é abortada. Após o cliente ser validado, se a lista de despesas possuir elementos é mostrada ao utilizador e é lhe pedido um ID de uma despesa caso esse ID não existe a compra é abortada, caso contrario se o saldo do cliente for suficiente a compra é finalizada e a despesas adicionada às informações do cliente.

```c
while(1){
    if(showExpenselist( list: explist)!=-1){
        printf( format: "ENTER A EXPENSE ID:");
        ID = int_input( max: MAX_BUFFER);
        no_expense *aux = explist->begin;
        //search if expense exists
        if(aux->expense.ID != ID){
            while(aux!=NULL && aux->expense.ID!=ID){
                aux=aux->pseg;
            }
            if(aux == NULL) {
                printf( format: "ID DONT EXITS\n");
                break;
            }
        }
        if(temp->client.balance >= aux->expense.value){
            temp->client.balance -= aux->expense.value;
            expense new = aux->expense;
            new.buydate = todayDate;
            temp->client.expenses[temp->client.numberExpenses] = new;
            temp->client.numberExpenses+=1;
            printf( format: "THANK YOU\n");
            break;
        }
        else{
            printf( format: "DONT HAVE ENOUGH MONEY\n");
        }
```

Figura 2- verificação do ID/Compra

```c
no_client *temp = list->begin;
printf( format: "ENTER A NAME:");
fgets( Buf: name_aux, MaxCount: MAX_BUFFER, File: stdin);
if(verify_char( s: name_aux)==-1) {
    printf( format: "INVALID NAME\n");
    return;
}
strupr( Str: name_aux);
strcpy( Dest: name, Source: removespaces( s: name_aux));

//search if client exists
if(temp!=NULL && strcmp(temp->client.student.name,name) != 0){
    while(temp!=NULL && strcmp(temp->client.student.name,name)!=0){
        temp=temp->pseg;
    }
    if(temp == NULL) {
        printf( format: "CLIENT DONT EXITS\n");
        return;
    }
}
```

Figura 3- Receção do nome /Verificação do cliente

Outra função importante é a função importante é a função que mostra os clientes com saldo abaixo de um determinado valor por ordem decrescente de saldo. Nesta é usada uma lista auxiliar, que será libertada após a sua visualização. É percorrida a lista principal e verificado se o saldo do cliente está abaixo do valor estipulado, caso esteja é efetuada a despesa.

```c
/***
 * method that make and show a order by descend balance client list wich balance is over certain value
 * @param list client list
 */
void descbalance_list(client_list *list){
    double value = 0.0;
    if(list == NULL) return;
    printf( format: "ENTER A VALUE:");
    value = double_input( max: MAX_BUFFER);
    if(list->length == 0) return;
    client_list *aux = malloc( Size: sizeof(client_list));
    if (aux == NULL) return;
    initClist( list: aux);
    no_client *no = list->begin;
    while(no!=NULL){
        if(no->client.balance < value){
            addClientBalance( list: aux, client: no->client);
        }
        no = no -> pseg;
    }
    if(aux->length == 0){
        free( Memory: aux);
    }
    else{
        show_balance( list: aux);
        free( Memory: aux);
    }
```

Figura 4- função para mostrar lista com saldo por ordem decrescente

5. FICHEIROS

Os ficheiros escolhidos para este projeto foram os binários, nele é escrito a data, o tamanho da lista e um array com os clientes presentes na lista. No caso da leitura é passado por referencia a data e o tamanho e é retornado um array com clientes que posteriormente é introduzido na lista ligada.

O ficheiro é lido sempre quando o programa inicia, sendo que se não existir dados este passo é passado.

6. MANUAL DO UTILIZADOR

Após a compilação do programa será apresentado um menu com varias opções, se ainda não existem dados guardados, estes não serão lido e o utilizador pode começar a efetuar operações, mas para isso é necessário ter pelo menos 1 cliente.

Será pedido ao utilizador para inserir um número e enquanto isso não for satisfeito o programa não avançara. De seguida se o número não for o de uma das opções possíveis voltará a ser apresentado o menu.

Para mudar a data é necessário inserir dia, mês e ano. Para inserir um cliente é necessário introduzir todos os dados corretamente, incluído uma data de nascimento valida. Todos os dados são inteiros á exceção

do nome do cliente. Para remover um cliente e mostrar os seus detalhar é necessário introduzir o nome deste. Para mostrar a lista o saldo abaixo de um valor é necessário introduzir um valor valido. Para fazer uma despesa é necessário inserir um nome e um ID da respetiva despesa, para carregar a conta é necessário introduzir um nome e o valor a ser carregado.

A operação de mostrar a lista apenas exige ser escolhida.