João Gaspar – 2021221276
Tiago Almeida – 2021221615

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

31 de maio de 2023

PRINCÍPIOS DE PROGRAMAÇÃO PROCEDIMENTAL

OBJETIVOS DO PROJETO

Este projeto teve como objetivo desenvolver uma plataforma de organização de reservas de uma oficina de modo a auxiliar os funcionários. A aplicação armazena em dois ficheiros de texto (reservas.txt e prereservas.txt) todas as informações sobre os seus clientes e as suas respetivas reservas/pré-reservas.

Na oficina há dois tipos de serviços: limpeza (que tem uma duração de 30 minutos) e manutenção (que tem uma duração de 60 min).

O programa permite ao funcionário fazer uma reserva/pré-reserva, sem ter de verificar se tem disponibilidade ou não. Isto é possível uma vez que a aplicação automaticamente corre o ficheiro de reservas, verificando se a oficina tem disponibilidade, ou não, de realizar o serviço requisitado. Na eventualidade de não haver tempo suficiente para a realização do serviço, o programa automaticamente coloca o cliente na fila de pré-reservas, priorizando a "ordem de chagada". Por outras palavras, fica com prioridade o cliente que realizou a tentativa de reserva em primeiro lugar.

Para além disso, o código, na eventualidade de haver um cancelamento de reserva, automaticamente aloca um cliente com uma pré-reserva compatível à lista de reservas, evitando com que o funcionário tenha que percorrer a lista de pré-reservas, procurando se algum cliente tem disponibilidade.

Tendo explicitado os objetivos do projeto, iremos passar a explicar, de forma sucinta, como o código foi desenvolvido.

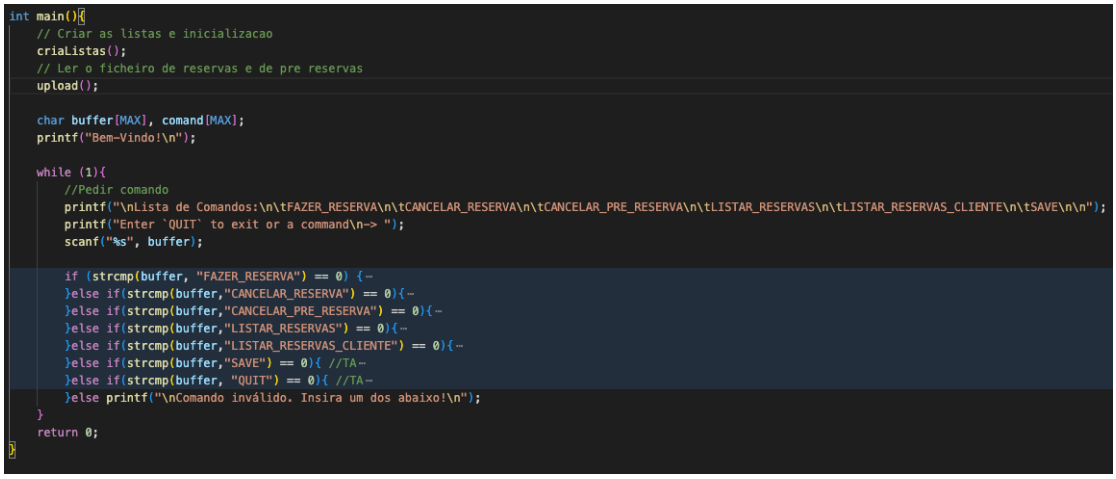
EXPLICAÇÃO DO CÓDIGO

Nos temos 3 ficheiros no total. Dois .c e um .h. O Projeto.c é o ficheiro que contem o main(). De seguida, temos o funcs.c que é onde estão presentes as funções a serem utilizadas. Por fim, no funcs.h tem as declarações das funções e as "struct's".

- Projeto.c

  Neste ficheiro está presente a função main().

  Sucintamente explicando as funções desta função, é nela onde o "input" do utilizador é interpretado e, dependendo desse mesmo "input", realizadas as operações pretendidas. Nela foi também inserida uma pragmática interface para facilitar a interpretação do utilizador.



```c
int main(){
    // Criar as listas e inicializacao
    criaListas();
    // Ler o ficheiro de reservas e de pre reservas
    upload();

    char buffer[MAX], comand[MAX];
    printf("Bem-Vindo!\n");

    while (1){
        //Pedir comando
        printf("\nLista de Comandos:\n\tFAZER_RESERVA\n\tCANCELAR_RESERVA\n\tCANCELAR_PRE_RESERVA\n\tLISTAR_RESERVAS\n\tLISTAR_RESERVAS_CLIENTE\n\tSAVE\n\n");
        printf("Enter `QUIT` to exit or a command\n-> ");
        scanf("%s", buffer);

        if (strcmp(buffer, "FAZER_RESERVA") == 0) {…
        }else if(strcmp(buffer,"CANCELAR_RESERVA") == 0){…
        }else if(strcmp(buffer,"CANCELAR_PRE_RESERVA") == 0){…
        }else if(strcmp(buffer,"LISTAR_RESERVAS") == 0){…
        }else if(strcmp(buffer,"LISTAR_RESERVAS_CLIENTE") == 0){…
        }else if(strcmp(buffer,"SAVE") == 0){ //TA…
        }else if(strcmp(buffer, "QUIT") == 0){ //TA…
        }else printf("\nComando inválido. Insira um dos abaixo!\n");
    }
    return 0;
}
```

No caso do funcionário inserir na plataforma "FAZER_RESERVA", a aplicação vai requisitar 5 informações:

- Nome do cliente;
- Tipo de serviço pretendido;
- Data do serviço;
- Horário do serviço.

Se as horas inseridas pelo utilizador estiverem fora do intervalo de funcionamento da oficina, o programa vai alertar o utilizador para o mesmo e dizer-lhe que não é possível realizar a reserva, questionando se pretende inserir uma outra data.

Por fim, quando as informações foram recolhidas com sucesso a plataforma adiciona-as à lista.

```c
if (strcmp(buffer, "FAZER_RESERVA") == 0) {
    char nome[MAX], tipo[MAX], date[MAX], time[MAX];
    int dia = 0, mes = 0, ano = 0, hora = 0, min = 0;
    strcpy(comand, buffer);
    printf("Nome (CAPS) -> ");
    scanf(" %[^\n]", nome);

    printf("Tipo de Reserva (LAVAGEM / MANUTENCAO) -> ");
    scanf("%s", tipo);
    printf("Data (dd/mm/aaaa) -> ");
    scanf("%s", date);

    while (1) {
        printf("Horario (hh/mm) -> ");
        scanf("%s", time);
        if (sscanf(time, "%d/%d", &hora, &min) != 2) {
            printf("Horario invalido. Insira um horario no formato hh/mm.\n");
            continue;
        }
        if (hora >= 8 && hora < 18 && (hora != 18 || min == 0)) {
            break;
        } else {
            printf("Horario invalido. Insira um horario entre as 08:00 e as 18:00.\n");
        }
    }

    sscanf(date, "%d/%d/%d", &dia, &mes, &ano);

    insereReserva(1, hora, min, nome, tipo, dia, mes, ano);
```

Para um utilizador cancelar uma reserva, ele tem de inserir de duas uma, ou "CANCELAR_RESERVA", ou "CANCELAR_PRE_RESERVA", dependendo da intenção do mesmo. Para realizar tal ação, apenas são necessárias duas informações:

- Nome do cliente;
- Data do serviço.

A diferença entre as duas funcionalidades é que, na eventualidade de se pretender cancelar uma pré-reserva, apenas é preciso apagá-la da lista das pré-reservas. Caso se cancele uma reserva, não só ela tem que ser apagada a partir da função "delete()" como também verificar se na vaga que foi libertada, alguém tem uma pré-reserva compatível, sendo, nesse caso, executada a função "substituirReserva()", responsável por essa operação.

```
}else if(strcmp(buffer,"CANCELAR_RESERVA") == 0){
    noListaReservas* head;
    char nome[MAX], date[MAX];
    int dia = 0, mes = 0, ano = 0;
    printf("Nome (CAPS) -> ");
    scanf(" %[^\n]", nome);

    printf("Data (dd/mm/aaaa) -> ");
    scanf("%s", date);

    sscanf(date, "%d/%d/%d", &dia, &mes, &ano);

    delete(nome, dia, mes, ano, 1);
    substituirReserva();

    strcpy(comand, buffer);

}else if(strcmp(buffer,"CANCELAR_PRE_RESERVA") == 0){
    noListaPreReservas* head;
    char nome[MAX], date[MAX];
    int dia = 0, mes = 0, ano = 0;
    printf("Nome (CAPS) -> ");
    scanf(" %[^\n]", nome);

    printf("Data (dd/mm/aaaa) -> ");
    scanf("%s", date);

    sscanf(date, "%d/%d/%d", &dia, &mes, &ano);

    delete(nome, dia, mes, ano, 2);

    strcpy(comand, buffer);
```

```
}else if(strcmp(buffer,"LISTAR_RESERVAS") == 0){
    listarReservas();

    strcpy(comand, buffer);

}else if(strcmp(buffer,"LISTAR_RESERVAS_CLIENTE") == 0){
    char nome[MAX];
    printf("Nome (CAPS) -> ");
    scanf(" %[^\n]", nome);

    listarCliente(nome);
    strcpy(comand, buffer);

}else if(strcmp(buffer,"SAVE") == 0){ //TA
    printf("A guardar alteracoes...\n");
    // Gravar as listas de reservas e pre-reservas
    download();
    strcpy(comand, buffer);

}else if(strcmp(buffer,"ATUALIZAR") == 0){ //TA
    substituirReserva();
    strcpy(comand, buffer);

}else if(strcmp(buffer, "QUIT") == 0){ //TA
    if(strcmp(comand, "SAVE") != 0){
        char resposta;
        printf("Alteracoes nao gravadas\n");
        printf("Deseja salvar? (S/N) -> ");
        scanf(" %c", &resposta);
        if(resposta == 'S' || resposta == 's'){
            printf("A guardar alteracoes...\n");
            download();
        }
    }
printf("A fechar aplicacao...\n");
break;
}else printf("\nComando inválido. Insira um dos abaixo!\n");
```

Por fim, há mais cinco operações possíveis:

- "LISTAR_RESERVAS";
- "LISTAR_RESERVAS_CLIENTES";
- "SAVE";
- "ATUALIZAR";
- "QUIT";

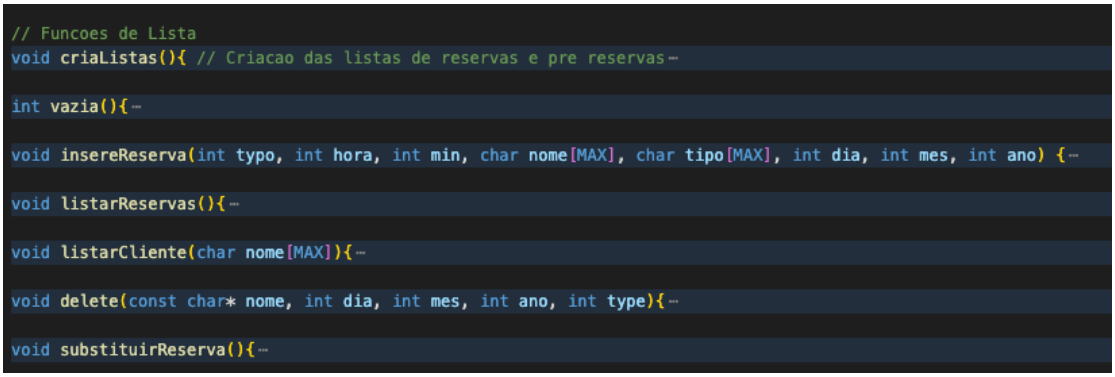
Como os próprios comandos sugerem, estes comandos servem apenas para mostrar ao utilizador a tabela de reservas e pré-reservas (total ou de um cliente em concreto), guardar as alterações nos ficheiros de texto, atualizar as listas e, por fim, sair da aplicação. Na eventualidade de o utilizador querer sair do programa sem antes ter salvo os processos, o programa irá questionar se pretende ou não sair sem salvas as atualizações, aguardando, deste modo, pelo "input" do utilizador de "S" (para se pretende salvar) ou "N" (caso não queira guardar as alterações).

- funcs.c

Este ficheiro está dividido em três partes distintas.

Inicialmente estão presentes três funções com finalidades apenas utilitárias, aritméticas e interpretação do "input" do utilizador para o programa.

De seguida temos as funções de modelação das listas de reservas e pré-reservas.



Primeiramente, temos a função "criaListas()". Esta função tem como objetivo inicializar o ponteiro de inicío da lista.

De seguida, a função "vazia()" verifica se uma lista está ou não, como o próprio nome indica, vazia.

A função "insereReserva()" foi a mais complicada de realizar. Esta função está responsável por vários procedimentos. A partir do "input" feito pelo funcionário, ela está responsável por inserir na lista de reservas, o agendamento do cliente. Porém, na eventualidade de não haver disponibilidade, o requisito do cliente vai passar para a lista de pré-reservas. Para além do mais, sempre que esta função é executada, as listas são organizadas da forma como é requisitado no enunciado do projeto.

Posteriormente, temos duas funções semelhantes. A "listarReservas()", responsável por listar todas as reservas e pré-reservas presentes na plataforma (tendo sido estas guardadas no ficheiro ou não), e a "listarClientes()", encarregada por mostrar ao funcionário todas as reservas referentes a um cliente em específico.

Como mencionado anteriormente, as funções "delete()" e "substituirReserva()" têm duas funcionalidades que, apesar de serem distintas, estão interligadas na eventualidade de uma reserva ser cancelada. Para o cancelamento de uma reserva e/ou pré-reserva, o código presente na função "delete()" está responsável por, como o próprio nome indica, apagar a reserva inserida pelo utilizador da lista. Apenas quando uma reserva é cancelada, a função "substituirReserva()" é executada. Esta função tem como objetivo verificar se, para a hora e data da reserva cancelada, existe algum cliente com uma pré-reserva que esteja apto a ocupar a vaga liberada.

- funcs.c

Relativamente a este ficheiro não há muito a apontar uma vez que nele apenas estão presentes as declarações das funções e as "struct's" utilizadas.

CONCLUSÃO

Concluindo, na nossa opinião, este projeto teve um grande impacto no que toca ao desenvolvimento das nossas capacidades em C, nomeadamente na compreensão e utilização de ponteiros, como também na nossa aptidão de planeamento e estrutura de um programa. Por outras palavras, este projeto vez com que nós, antes de iniciarmos a "escrever" código, realizássemos um planeamento prévio de como o código devia ser elaborado.