Escola Superior de Tecnologia e Gestão

Instituto Politécnico de Viana do Castelo

Programação 1

Gestão de Alugueres de Recintos Desportivos

Engenharia Informática

João Pedro Alves Vaz Vieira,21651

2018/2019

Índice

Enquadramento do Trabalho

Neste trabalho prático, foi proposta a realização de uma aplicação capaz de gerir, não só recintos desportivos, como também clientes e alugueres feitos. O ideia adotada para esta aplicação foi fazer um menu principal e depois 3 secundários, um para cada gestão. Tendo em conta que não existiam limites de alugueres, nem de clientes armazenou-se essa informação em listas ligadas. Já os recintos armazenaram-se no array, visto que apresentavam um limite.

Na gestão dos recintos a aplicação é capaz de:

- ✓ Acrescentar informação sobre um recinto;
- ✓ Alterar a informação sobre o recinto, nomeadamente o preço e hora de abertura e de fecho;
- ✓ Inativar/ativar um recinto;
- ✓ Listar os recintos por ordem decrescente do preço;
- ✓ Listar os recintos de um determinado tipo;
- ✓ Listar recintos de um concelho;
- ✓ Listar recintos existente num determinado concelho de um determinado tipo;
- ✓ Listar recintos com o preço dentro de um intervalo de preços;
- ✓ Guardar a informação dos recintos em ficheiro binário e texto.

No que à gestão dos clientes diz respeito, a aplicação é capaz de:

- ✓ Acrescentar a informação sobre um novo cliente;
- ✓ Alterar os dados de um cliente;
- ✓ Listar os clientes por ordem alfabética do nome;
- ✓ Apresentar os dados de um cliente, dado o NIF;
- ✓ Listar todos os clientes de uma determinada localidade;
- ✓ Guardar a informação do recintos em ficheiros binários e de texto.

No que toca à gestão dos alugueres a aplicação é capaz de:

- ✓ Fazer uma reserva de um recinto;
- ✓ Ativar um reservar e efetuar o pagamento;
- ✓ Listar todos os recintos disponíveis para alugar num período indicado;
- ✓ Listar recintos disponíveis para alugar num determinado concelho;
- ✓ Listar recintos disponíveis de um determinado tipo para alugar num determinado concelho;
- ✓ Listar todos os alugueres feitos por um determinado cliente (através do número), ao longo dos tempos;
- ✓ Listar todos os dados dos alugueres de um determinado recinto;
- ✓ Listar todos as reservas ativos (ainda não realizadas);
- ✓ Fecho do dia;
- ✓ Fecho do mês;
- ✓ Fecho do ano;

- ✓ Calcular o IVA;
- ✓ Guardar a informação dos recintos em ficheiro binário e texto.

Codificação

Estruturas

- Estrutura CLIENTE: esta estrutura tem a informação dos clientes e está guardada em listas ligadas, visto que não há limite máximo de clientes.
- Estrutura RECINTO: esta estrutura tem a informação dos recintos guardada num array de tamanho 80.
- Estrutura ALUGUER: esta estrutura tem a informação dos alugueres guardada em listas ligada, visto que não há limite máximo de alugueres
- Estrutura TEMPO: esta estrutura serve para armazenar uma data em apenas uma variável, guardando o ano, a mês, o dia, a hora e os minutos.

Funções Mais Importantes

Funções relacionadas com a gestão de clientes, estão no ficheiro "cliente.c":

- ***void infoCliente(ELEMENTOC *iniLista,CLIENTE *newInfo,int *total,int *totalR)***

Esta função foi feita para pedir informação do cliente e armazená-la. A função verifica se já existe no sistema o email, NIF, o número telefónico e o número de cartão de cidadão igual ao que foi introduzido e caso haja da return para o menu cliente, não verifica o nome, pois existem várias pessoas com o mesmo nome. O mesmo acontece com a morada, pois várias pessoas da mesma casa podem-se registar, nem a localidade pois também pode ser a mesma para várias pessoas. Depois de passar todos os campos de introdução de informação e de verificação é incrementado uma unidade a variável (*total), que irá servir para dar o código interno e a variável (*totalR) que serve para saber quantos clientes estão registados no momento.

- ***int inserirCliente(ELEMENTOC **iniLista,CLIENTE newInfo)***

Esta função é uma função básica de armazenamento de uma lista ligada no final da lista.

- ***void listarOrdemAlfabetica(ELEMENTOC **iniLista,int total)***

Esta função ordena os nomes por ordem alfabética. Para isso, utilizou-se um algoritmo de ordenação (buble sort) adaptando-o às listas ligadas. Moveu-se apenas a informação dos clientes, mantendo-se os endereços. Para listar, percorreu-se as listas ligadas através de um ciclo for e mostrou-se a informação ao longo do ciclo.

- ***void listarPorNif(ELEMENTOC *iniLista)***

Esta função lista todos os clientes com um determinado NIF, sendo apenas um para cada, visto que cada pessoa tem um NIF diferente. Para isso, utilizou-se um ciclo for para percorrer toda a lista ligada, e dentro do for colocou-se uma condição com o if, condição esta de o NIF introduzido ser igual ao NIF de um dos clientes registados. Caso seja, mostra a informação e coloca na variável y o valor de "1", caso contrário sai do ciclo sem mostrar nada. Depois se o "y" for "0",ou seja, não passou pela condição do if dentro do ciclo for, mostra uma mensagem a dizer que não existe nenhum cliente com esse NIF.

- ***void listaPorLocalidade(ELEMENTOC *iniLista)***

Esta função lista todos os clientes de uma localidade e segue o mesmo raciocínio logico da função anterior.

- ***int veriCli(ELEMENTOC *iniLista,int num)***

Esta função serve para verificar se existe um cliente com um determinado número de cliente. É utilizada nos alugueres e segue o mesmo raciocínio lógico das funções anteriores em termos de pesquisa. Retorna "0" se existir, e "-1" se não existir.

As restantes funções deste ficheiro são funções básicas de listas ligadas abordadas na aula.

Funções relacionadas com a gestão de recintos, estão no ficheiro "recinto.c":

- ***void listarIntPre(RECINTO recintos[],int total)***

Esta função serve para listar os recintos que estejam num determinado intervalo de tempo. Para isso, percorreu-se o array através de um ciclo for e dentro deste, através do if colocou-se a condição de que o preço mínimo tem de ser maior ou igual ao preço do recinto e o preço máximo menor ou igual a este preço. Se se verificar esta condicação, mostra o recinto e coloca a variável "y" com valor "1". Depois de sair do ciclo for, tem a condição que permite saber se existe algum recinto dentro desses preços ou não. Essa condição foi feita com um if e se o valor da variável "y" for igual a 0,isto é, o valor inicial, quer dizer que não existe nenhum recito dentro desses preços e mostra uma mensagem.

- ***int veriReci(RECINTO recintos[],int num,int total)***

Esta função serve para saber se existe um recinto com o código interno recebido e se esse recinto está interdito ou não. A função é usada nos alugueres. A função percorre o array com um ciclo for e tem duas condições dentro do ciclo. A primeira verifica se existe algum recinto com o código interno igual ao valor recebido pela função. Caso seja igual tem a segunda condição, que verifica se o recinto está livre (não interdito), e se estiver retorna "0", caso contrário retorna "2". Se não encontrar nenhum recinto com o código interno igual ao número recebido retorna "-1". Estes valores de return vão ser usados na gestão de alugueres.

- ***void stats(RECINTO recintos[],int total)***

Esta função serve para ativar e inativar um recinto. Esta pesquisa o recinto com o código interno igual ao que for introduzido através de um ciclo for, que percorre todo o array, e uma condição com if. Caso encontre o recinto com o código interno igual, coloca o valor "1" na variável "y" e verifica através de uma condição com if, se está livre. Se estiver pergunta se quer inativar ou não, sendo que caso queira inativa atribuindo o valor 1 à variável "status" do recinto. Caso não queira, dá return para o menu dos recintos. Se estiver inativo pergunta se quer ativar, caso contrário segue o raciocínio anterior. No fim de sair do ciclo, há uma condição que permite saber se existe algum recinto com o valor introduzido ou não e se o valor de "y" for igual a "0" significa que não passou na condição de verificar se existe um recinto com um código interno igual.

Todas as outras funções do ficheiro "recinto.c", são funções que seguem algoritmos básico de arrays.

Funções relacionadas com a gestão de tempo, estão no ficheiro "tempo.c":

- ***void tempoAtual(TEMPO *temp)***

Esta função server para colocar o tempo atual numa variável.

Funções relacionadas com a gestão de alugueres, estão no ficheiro "aluguer.c":

- ***int reservarRecinto(ELEMENTOA **iniALista,RECINTO recintos[],int totalR,ELEMENTOC *iniCLista)***

Esta função serve para a reserva de um recinto. A função utiliza a função veriCli() para confirmar se existe um cliente atribuído ao valor retornado pela função a variável "t". Caso "t" seja diferente de "0" satisfaz a condição do if, dá return a "2" e volta para o menu alugueres. Caso contrário pede informação do recinto e usa a função veriReci() para verificar se o mesmo existe e está livre (não interdido), atribuindo o valor retornado pela função à variável "t". Caso "t" seja igual a "-1" satisfaz a condição do if, entra e retorna a "2". Caso contrário continua até à próxima condição, ou seja, "t" ser igual a "2". Se for igual significa que está interdito e retorna a 2. Caso contrário continua a pedir informação.

Depois de pedir informação sobre a data que pretende alugar, através da função verificarAluguer() averigua se a data é passível de ser alugada. Foi atribuído o valor retornado pela função à variável "t" e com um switch dependente do valor de "t", apresenta a mensagem dizendo o porquê da data não ser possível e retorna em todos os casos a "2". Com recurso à função tempoAtual(), atribuiu-se o tempo atual à variável "info.dataReserva". Por fim, coloca-se o valor "0" na variável "info.modoPagamento", significando que não foi pago, e executa-se a função reserva(), uma função básica de adicionar elementos numa lista ligada. Para além disso, atribuiu-se à variável "t" o valor retornado pela função. Se o "t" for "0" significa que a reserva foi feita com sucesso, entrando assim na condição if.

- ***int verificarAluguer(ELEMENTOA *iniALista,ALUGUER x,RECINTO recintos[])***

Esta função serve para verificar se é possível fazer o aluguer. A primeira condição é se o valor retornado pela função VerificarValidade é igual a “-1”, significando que a data de aluguer é inferior à data atual. Caso seja, entra no if e retorna “-1”. Caso contrário, continua até à próxima condição, que verifica se o mês de aluguer é “2” (fevereiro) e o dia de aluguer é superior a “29”, visto que fevereiro apenas tem 29 dias. Caso seja, entra no if e retorna “-1”. Se não verificar essa condição, continua até à próxima que verifica se o ano é bissexto. Os anos bissextos são divisores de quatro, e/ou são divisores de 400 ou não são divisores de 1000. Caso o ano seja efetivamente bissexto, significa que fevereiro pode ter 29 dias e continua a executar. Caso contrário verifica se o dia é maior que 28 e o mês é fevereiro “2”. Se for, entra no if e retorna “-1”, caso contrário sai.

As próximas duas condições servem para saber se o mês introduzido é um mês com 30 ou 31 dias. Caso se esteja perante um mês de 30 dias e na data introduzida tenha dia superior a 30, ou seja 31 dias ou mais, ela retorna “-1”. A condição seguinte verifica se a data e o tempo são possíveis tendo em conta a hora de fecho do recinto. Se a hora pretendida for superior ou igual à hora de fecho ou à hora de fim de alguer, a variável “fimx”, isto é, a hora de aluguer mais o tempo de aluguer for superior à hora de fecho, a função retorna a “-2”.

A próxima condição verifica se o recinto está aberto na hora pretendida para o aluguer. Se a hora de aluguer for menor que a hora de abertura significa que está fechado e retorna a “-2”. A condição seguinte verifica se existem alugueres. Se não existirem, significa que o recinto pode ser alugado e retorna a “0”. Depois, através de um ciclo for, percorre-se a lista e na primeira condição verifica se existe algum aluguer na mesma data para o mesmo recinto. Se sim, atribui o valor à variável “fimAux”, ou seja, a soma da hora de aluguer do recinto já alugado mais o tempo de aluguer. Ainda dentro do ciclo for, e da primeira condição, é feita uma segunda condição que verifica se a hora de aluguer pretendida está entre a hora de inicio e a hora do fim do aluguer encontrado. Se sim, é impossível alugar e retorna a “2”. Caso contrário segue para a próxima condição, que se encontra também dentro do ciclo for, e da primeira condição. Esta verifica se o fim do aluguer pretendido está entre o inicio de um aluguer e o fim do mesmo. Caso esteja, retorna a “2” significando que não dá para alugar. Caso contrário, sai do ciclo for e retorna a “0”, mostrando que o aluguer é possível**.**

- ***void ativarReservar(ELEMENTOA **iniALista,RECINTO recintos[],int totalR,ELEMENTOC *iniCLista)***

Esta função serve para ativar uma reserva já feita e efetuar o pagamento. Depois de introduzir o número do cliente, verifica se este existe através da função veriCli(), e atribui o valor retornado pela função à variável “t”. Caso seja diferente de “0” satisfaz a condição e da return para o menu alugueres. Caso contrário precede a execução e entra no ciclo while. Assim, percorre a lista ligada de alugueres através de um ciclo for. Caso satisfaça a condição de existir um aluguer para aquele cliente e o modo de pagamento ser igual a “0”, significa que não foi pago

logo está efetivamente ativo. Deste modo, entra no ciclo if e pergunta se esta é a reserva que o cliente quer ativar. Caso não seja irá procurar por outra até sair, se for efetua o pagamento através da função funcaoPagamentos.Caso se cancele a reserva, a variável “mdp” fica com valor “0” e entra na condição seguinte, dando return. Caso não seja “0”, atribui o valor da variável à variável “aux->info.modoPagemento” e atribui a data de pagamento através da função tempoAtual(), dando, por fim, return. Depois de sair do ciclo, se a variável “w” tiver valor de “2” significa que não existem mais reservas feitas por esse cliente, entrando na condição e dando return. Caso contrário significa que o cliente não tem nenhuma reserva feita, e ,então, pergunta se deseja fazer uma reserva. Neste caso, se houver intenção de fazer a reserva, entra no ciclo if e executa a função reservarRecintos(), sendo que caso o valor retornado pela função seja 2, não é possível fazer a reserva, retornando para o menu aluguer. Caso seja feita a reserva, o valor retornado será 1, ou seja, não sairá do ciclo while, e irá, assim, direto para o efetuamento do pagamento.

- ***int funcaoPagamentos()***

Esta função seve para efetuar o pagamento, retornando o modo de pagamento feito que depois é atribuído ao modo de pagamento do aluguer.

- ***int reservaInativaPassada(ELEMENTOA **iniALista)***

Esta função serve para apagar as reservas que já passaram do tempo e que não foram ativas, através da pesquisa por reservar com modo de pagamento “’0” e data de aluguer anteriores à data atual.

- ***float calcularIVA(ELEMENTOA *iniALista,int x)***

Esta função calcula o IVA de um determinado ano, retornando esse valor.

- ***int fechoDiario(ELEMENTOA *iniALista)***

Esta função serve para fazer o fecho diário e guardá-lo em ficheiro. O nome do ficheiro é do tipo “RelatorioDiaxxMesxxAnoxx.txt”, sendo que para isso se teve que usar o comando “sprintf” de forma a colocar as variáveis numa string, e depois utilizando essa string para dar o nome ao ficheiro.

Todas as restantes funções deste ficheiro utilizam raciocínios lógicos semelhas a funções já aqui explicadas

Testes feitos

Clientes

- Adicionar recintos, sucesso
- Alterar informação**,** sucesso
- Listagem, sucesso

Recintos

- Adicionar recintos, sucesso
- Inativar recintos, sucesso
- Listagem, sucesso

Alugueres

- Reservar recinto, sucesso.
- Recinto alugado na mesma data/hora não da para reservar, sucesso
- Hora de aluguer superior a hora de fecho, sucesso
- Hora de aluguer inferior a hora de abertura, sucesso
- Recintos interditos não dão para alugar, sucesso
- Ativar reserva, sucesso.
- Caso não tenha reserva, faz uma reserva, sucesso.
- Listagens, sucesso.
- Fazer fechos, sucesso.

Dados Introduzidos:

Deixou-se um cliente já guardado e dois alugueres feitos de modo a facilitar alguns testes.

- Aluguer do recinto 1 no dia 02/06/2019 às16h: esta reserva não vai ser ativa, sendo que quando o programa for executado após esta data, irá aparecer uma mensagem relativa à eliminação (1 reserva não ativa passada);
- Aluguer do recinto 1 para o dia 05/12/2020 às 14h com um tempo de duração de 3 horas: reserva não ativa, sendo que neste caso, se se quiser alugar para o mesmo dia nas mesmas horas ou, por exemplo, às 12h com tempo de duração maior de 2h, não irá ser possível.

Dificuldades

Neste trabalho prático, apresentaram-se algumas barreiras que foram ultrapassadas com sucesso. Uma dessas, foi a questão dos alugueres superiores a 1 hora. Inicialmente, impôs-se a condição de apenas se poder fazer alugueres de uma hora, mas depois de algum tempo foi possível a resolução deste problema. Este não advinha em saber como implementar, mas sim das condições que teriam de ser feitas, posteriormente, para verificar se o recinto se encontrava livre.

Conclusão

Em suma, através deste trabalho prático foi possível atingir todos os objetivos propostos. A aplicação está operacional, bem como todas as suas funcionalidades. Para além disso, foi possível aplicar novos conceitos, não só aprendidos nas aulas de programação 1, como também outros fora desse contexto. De uma forma geral, foi importante para estimular a criatividade e o raciocínio lógico, bem como a organização de código.

Anexo

Menu-de-utilizador.docx