

Engenharia Informática

Estruturas de Dados

2019/2020

Relatório

Estruturas

António Pereira Nº2019141051

David Ferreira Nº2019141281

Data: 30/04/2020

Índice

1	INTRODUÇÃO.....	3
2	TRABALHO REALIZADO	4
2.1	COMPLETE O SEGUINTE PROGRAMA TENDO EM CONTA OS COMENTÁRIOS	4
2.2	COMPLETE O PROGRAMA SEGUINDO AS INDICAÇÕES DOS COMENTÁRIOS:	8
a)	Indique o que faz o subprograma <i>funcao</i>	8
b)	Crie um subprograma <i>escreve_ponto</i> que apenas escreve as coordenadas de um ponto na forma usual. (Exemplo: o ponto com coordenadas 1 e 2 é apresentado como (1,2).).....	8
c)	Complete o <i>main</i> como indicado nos comentários.....	9
2.3	NO PROGRAMA CONSIDERE UMA NOVA ESTRUTURA <i>RECTÂNGULO</i> :	10
a)	Defina a estrutura <i>rectangulo</i> que tem 4 campos do tipo inteiro: <i>xmin</i> , <i>xmax</i> , <i>ymin</i> e <i>ymax</i>	10
b)	Crie um subprograma <i>area</i> que, dado um rectangulo devolva a sua área.....	10
c)	Crie um subprograma <i>esta_dentro</i> que, dado um ponto e um retângulo, escreve se o ponto está dentro ou fora do retângulo. Considere que os pontos do perímetro estão dentro do rectângulo.....	11
2.4	DEFINA UMA ESTRUTURA QUE PERMITA REPRESENTAR E EFETUAR OPERAÇÕES COM NÚMEROS RACIONAIS. ESTA ESTRUTURA DEVERÁ TER DOIS CAMPOS: UM PARA O NUMERADOR E OUTRO PARA O DENOMINADOR DO NÚMERO.....	13
a)	Ler uma fracção;.....	13
b)	Somar duas fracções;.....	14
c)	Subtrair duas fracções;.....	15
d)	Multiplicar duas fracções;.....	15
e)	Dividir duas fracções;.....	16
f)	Determinar a potência (com expoente inteiro) de uma fracção.	17
2.5	UMA PEQUENA EMPRESA FAMILIAR PRETENDE ORGANIZAR OS DADOS DOS SEUS FUNCIONÁRIOS: NÚMERO, NOME, TAREFA, SALÁRIO.....	19
a)	Defina uma estrutura de dados que permita representar cada funcionário.	19
b)	Declare uma tabela que armazene informação pretendida.	19
c)	Elabore subprogramas para:	20
3	CONCLUSÃO	28
4	REFERÊNCIAS	29
5	ANEXOS	ERRO! MARCADOR NÃO DEFINIDO.

1 Introdução

Este trabalho é uma introdução às estruturas e como elas funcionam. Vamos trabalhar com coisas básicas como por exemplo declaração de estruturas de modo a cimentarmos conhecimentos essenciais acerca desta matéria.

2 Trabalho realizado

2.1 Complete o seguinte programa tendo em conta os comentários

Neste primeiro exercício a professora pediu para completarmos o programa que foi fornecido pela professora. Foi-nos pedido para criar quatro funções. A primeira função é para o utilizador escrever a data e é a seguinte:

```
data le_data()
{
    data dat;
    printf("Insira a data: ");
    scanf("%d", &dat.dia);
    while(dat.dia>31 || dat.dia<=0)
    {
        printf("\ndia invalido, tente novamente: ");
        scanf("%d", &dat.dia);
    }

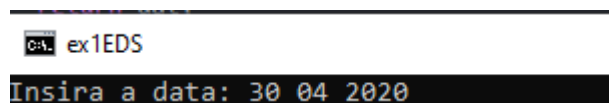
    scanf("%d", &dat.mes);
    while(dat.mes<=0 || dat.mes>12)
    {
        printf("\nmes invalido, tente novamente: ");
        scanf("%d", &dat.mes);
    }

    scanf("%d", &dat.ano);
    return dat;
}
```

Figura 1- Função para ler a data que o utilizador inserir

No início da função declaramos a variável dat do tipo estrutura. Pedimos ao utilizador para inserir a data e no fim a função retorna a data inserida.

O output desta função é o seguinte:



```
C:\ ex1EDS
Insira a data: 30 04 2020
```

Figura 2- Interface da função le_data

A segunda função do nosso programa é para apresentar a data que o utilizador inseriu. Para isso fizemos um único printf como podemos ver na figura abaixo:

```
void escreve_data(data d)
{
    printf("Data: %d-%d-%d", d.dia, d.mes, d.ano);
}
```

Figura 3- Função para apresentar data

O output da função é o seguinte:

```
Insira a data: 30 04 2020
Insira a data: 29 04 2020
Data: 30-4-2020
Data: 29-4-2020
```

Figura 4- Exemplo de output da função escreve_data

A próxima função é para comparar duas datas, ver qual é a maior, a menor e se são iguais. Se a primeira data for maior que a segunda retorna ">", caso seja menor retorna "<" e se for igual retorna "=". O código é o seguinte:

```
char compara_datas(data dat, data datt)
{
    if(dat.ano>datt.ano)
        return '>';
    else
        if(dat.ano == datt.ano)
        {
            if(dat.mes>datt.mes)
                return '>';
            else
                if(dat.mes == datt.mes)
                {
                    if(dat.dia>datt.dia)
                        return '>';
                    else
                        return '=';
                }
        }
    else
        return '<';
}
```

Figura 5- Código para comparar datas

Chamamento da função do main:

```
compara_datas(dat, datt);  
if(compara_datas(dat, datt) == '>')  
    printf("\nPrimeira data e maior");  
else  
    if(compara_datas(dat, datt) == '=')  
        printf("\nDatas iguais");  
    else  
        printf("\nSegunda data e maior");
```

Figura 6- Chamamento da função

Output da função compara_datas

```
Insira a data: 30 04 2020  
Insira a data: 29 04 2020  
Data: 30-4-2020  
Data: 29-4-2020  
Primeira data e maior
```

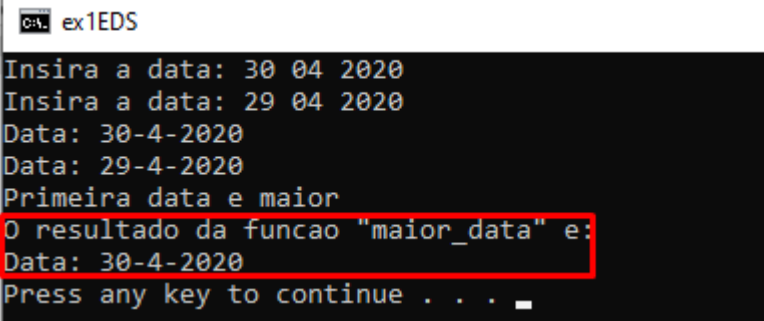
Figura 7- Exemplo de output

A última função é para ver qual é a maior data. Se a primeira data que o utilizador inseriu for maior que a segunda, também introduzida pelo o utilizador, retorna a primeira data. Caso a segunda seja maior que a primeira retorna a segunda data e se forem iguais retorna a primeira, como foi pedido pela professora. A função é a seguinte:

```
data maior_data(data dat, data datt)
{
    if(dat.ano>datt.ano)
        return dat;
    else
        if(dat.ano == datt.ano)
        {
            if(dat.mes>datt.mes)
                return dat;
            else
                if(dat.mes == datt.mes)
                {
                    if(dat.dia>datt.dia)
                        return dat;
                    else
                        return datt;
                }
            }
        }
    else
        return datt;
}
```

Figura 8- Função para ver qual é a maior data

O output desta função pode ser visto na figura abaixo:



```
ex1EDS
Insira a data: 30 04 2020
Insira a data: 29 04 2020
Data: 30-4-2020
Data: 29-4-2020
Primeira data e maior
O resultado da funcao "maior_data" e:
Data: 30-4-2020
Press any key to continue . . .
```

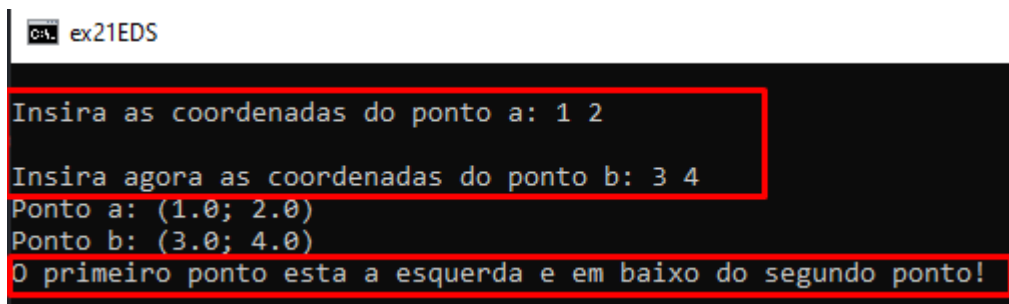
Figura 9- Interface da função maior_data

2.2 Complete o programa seguindo as indicações dos comentários:

- a) Indique o que faz o subprograma *funcao*.

O subprograma função o que faz é, imaginando um gráfico cartesiano, verifica se o ponto x ta acima, abaixo, à esquerda ou a direita do ponto y.

Output do subprograma funcao:



```
ex21EDS
Insira as coordenadas do ponto a: 1 2
Insira agora as coordenadas do ponto b: 3 4
Ponto a: (1.0; 2.0)
Ponto b: (3.0; 4.0)
O primeiro ponto esta a esquerda e em baixo do segundo ponto!
```

Figura 10- Exemplo de output do subprograma funcao

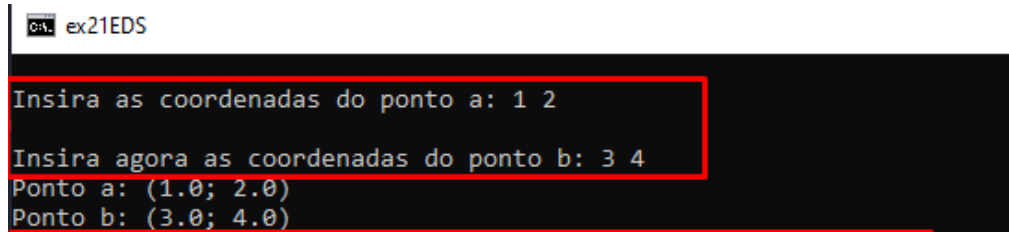
- b) Crie um subprograma *escreve_ponto* que apenas escreve as coordenadas de um ponto na forma usual. (Exemplo: o ponto com coordenadas 1 e 2 é apresentado como (1,2).)

Nesta alínea é pedido para criar uma função que escreva as coordenadas de um ponto. Para tal, fizemos um printf para cada ponto. A função é a seguinte:

```
void escreve_ponto(ponto a, ponto b)
{
    printf("Ponto a: (%.1f; %.1f)", a.x, a.y);
    printf("\nPonto b: (%.1f; %.1f)", b.x, b.y);
}
```

Figura 11- Função para escrever as coordenadas de um ponto

Output da função `escreve_ponto`:



```
ex21EDS
Insira as coordenadas do ponto a: 1 2
Insira agora as coordenadas do ponto b: 3 4
Ponto a: (1.0; 2.0)
Ponto b: (3.0; 4.0)
```

Figura 12- Exemplo de output

c) Complete o *main* como indicado nos comentários.

Na figura abaixo encontra-se o main completado como pedido pela professora:

```
void main()
{
    ponto a, b;
    printf("\nInsira as coordenadas do ponto a: ");
    scanf("%f", &a.x);
    scanf("%f", &a.y);
    printf("\nInsira agora as coordenadas do ponto b: ");
    scanf("%f", &b.x);
    scanf("%f", &b.y);
    escreve_ponto(a,b);
    funcao(a,b);
}
```

Figura 13- Função main

2.3 No programa considere uma nova estrutura *rectângulo*:

- a) Defina a estrutura *rectangulo* que tem 4 campos do tipo inteiro: *xmin*, *xmax*, *ymin* e *ymax*.

Para esta alínea fizemos uma estrutura retangulo e dentro desta declaramos as 4 variáveis do tipo inteiro. A estrutura é a seguinte:

```
struct retangulo{
    int xmin;
    int xmax;
    int ymin;
    int ymax;
};

typedef struct retangulo ret;
```

Figura 14- Definição da Estrutura retangulo

- b) Crie um subprograma *area* que, dado um rectangulo devolva a sua área.

Nesta alínea criamos a função *area* tal como é pedido e o que a função faz é calcular área do retângulo com os pontos que o utilizador inseriu e retornar essa area. Este é o subprograma que nos desenvolvemos:

```
int area(ret a)
{
    int areaa;
    int x, y;
    x = (a.xmax - a.xmin);
    y = (a.ymax - a.ymin);
    areaa = x * y;
    return areaa;
}
```

Figura 15- Função para calcular a área do retangulo

Output desta função:

```
CA: ex22EDS
Insira o x minimo: 2
Insira o x maximo: 4
Insira o y minimo: 5
Insira o y maximo: 8
A area do retangulo e: 6
```

Figura 16- Valor da área calculada através dos pontos

- c) Crie um subprograma *esta_dentro* que, dado um ponto e um retângulo, escreva se o ponto está dentro ou fora do retângulo. Considere que os pontos do perímetro *estão dentro* do rectângulo.

Na última alínea deste exercício foi-nos pedido para criar uma função que verifique se um ponto introduzido pelo utilizador esta dentro ou fora do retangulo introduzido pelo mesmo. A função é a seguinte:

```
void esta_dentro(ret a, ret b)
{
    {
        if (b.xmin >= a.xmin && b.xmax <= a.xmax && b.ymin >= a.ymin && b.ymax <= a.ymax)
            printf("\nO ponto pertence!");
        else
            printf("\nO ponto nao pertence!");
    }
}
```

Figura 17- Função que verifica se o ponto esta dentro ou fora do retangulo

O resultado desta função pode ver-se na figura abaixo:

```
ex22EDS
Insira o x minimo: 4
Insira o x maximo: 6
Insira o y minimo: 2
Insira o y maximo: 8
A area do retangulo e: 12
Insira o x do ponto: 4
Insira o y do ponto: 2
```

Figura 18- Interface da função esta_dentro

2.4 Defina uma estrutura que permita representar e efetuar operações com números racionais. Esta estrutura deverá ter dois campos: um para o numerador e outro para o denominador do número.

a) Ler uma fracção;

Nesta alínea, criamos a função `le_fracao` do tipo `frac`. Esta função serve para o utilizador inserir o numerador e o denominador da fracção. A função e o output desta função são os seguintes:

```
frac le_fracao()
{
    frac a;
    printf("Insira o numerador: ");
    scanf("%d", &a.numerador);
    printf("\nInsira agora o denominador: ");
    scanf("%d", &a.denominador);
    while(a.denominador == 0)
    {
        printf("\nO denominador nao pode ter 0. Insira novamente: ");
        scanf("%d", &a.denominador);
    }
    return a;
}
```

Figura 19- Função para inserir os valores da fracção

ex3EDS

```
Insira o numerador: 2
Insira agora o denominador: 3
Insira o numerador: 2
Insira agora o denominador: 3
```

Figura 20- Output da função `le_fracao`

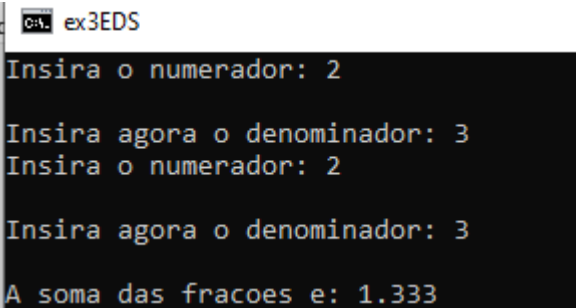
b) Somar duas frações;

Para esta alínea criamos uma função tal como é pedido. Nesta função primeiro dividimos o numerador pelo denominador da primeira e segunda fração para obtermos o valor decimal e depois para obter a soma de duas frações somamos o resultado da divisão da primeira fração pelo resultado da divisão da segunda fração e vai dar a soma na forma decimal. A função é a seguinte:

```
void soma_fracao(frac a, frac b)
{
    float soma, fraca, fracb;
    fraca = a.numerador / a.denominador;
    fracb = b.numerador / b.denominador;
    soma = fraca + fracb;
    printf("\nA soma das fracoes e: %.3f", soma);
}
```

Figura 21- Função para somar duas frações

O output desta função é a seguinte:



```
C:\ ex3EDS
Insira o numerador: 2
Insira agora o denominador: 3
Insira o numerador: 2
Insira agora o denominador: 3
A soma das fracoes e: 1.333
```

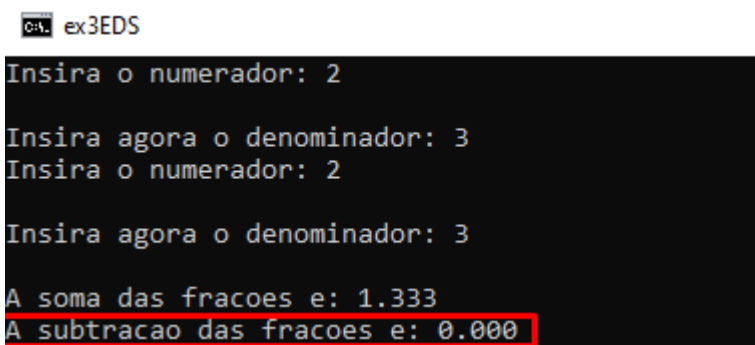
Figura 22- Interface da função soma_fracao

c) Subtrair duas fracções;

Para esta alínea o raciocínio é o mesmo da função soma_fracao, mas em vez de somar, subtraímos o resultado da primeira divisão pelo resultado da segunda divisão. A função e o output desta alínea são os seguintes:

```
void subtrai_fracao(frac a, frac b)
{
    float subtracao, fraca, fracb;
    fraca = a.numerador / a.denominador;
    fracb = b.numerador / b.denominador;
    subtracao = fraca - fracb;
    printf("\nA subtracao das fracoes e: %.3f", subtracao);
}
```

Figura 23- Função para subtrair duas fracções



```
ex3EDS
Insira o numerador: 2
Insira agora o denominador: 3
Insira o numerador: 2
Insira agora o denominador: 3
A soma das fracoes e: 1.333
A subtracao das fracoes e: 0.000
```

Figura 24- Output da função subtrai_fracao

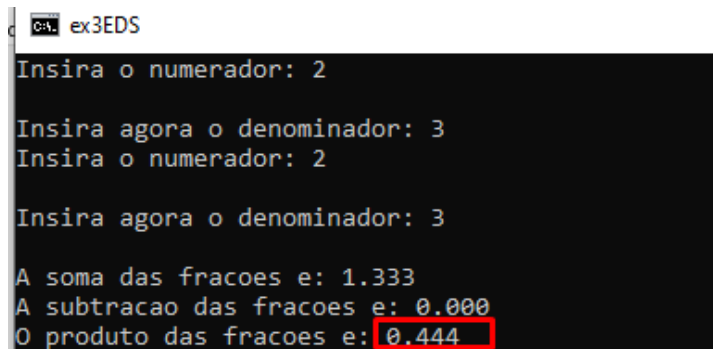
d) Multiplicar duas fracções;

Nesta alínea, fizemos a mesma coisa que fizemos na alínea anterior só que em vez de subtrair o resultado das duas divisões multiplicamos o resultado dessas duas divisões. A função é a seguinte:

```
void multiplica_fracao(frac a, frac b)
{
    float multiplicacao, fraca, fracb;
    fraca = a.numerador / a.denominador;
    fracb = b.numerador / b.denominador;
    multiplicacao = fraca * fracb;
    printf("\nO produto das fracoes e: %.3f", multiplicacao);
}
```

Figura 25- Função para multiplicar duas fracções

Resultado da função multiplica_fracao:



```

ex3EDS
Insira o numerador: 2
Insira agora o denominador: 3
Insira o numerador: 2
Insira agora o denominador: 3
A soma das fracoes e: 1.333
A subtracao das fracoes e: 0.000
O produto das fracoes e: 0.444
  
```

Figura 26- Exemplo de output

e) Dividir duas fracções;

Aqui criamos a função divide_fracao e nesta função primeiro dividimos o numerador pelo denominador da primeira função, fazendo o mesmo para a segunda fração. A seguir, para fazer a divisão de duas frações dividimos o resultado da divisão da primeira fração pelo resultado da divisão da segunda fração, caso o resultado da segunda divisão for igual a zero não possível fazer a divisão de duas frações. A função é a seguinte:

```

void divide_fracao(frac a, frac b)
{
    float divisao, fraca, fracb;
    fraca = a.numerador / a.denominador;
    fracb = b.numerador / b.denominador;
    if(fracb==0)
        printf("\nNao e possivel dividir nenhum valor por 0.");
    divisao = fraca / fracb;
    printf("\nO resultado da divisao das fracoes e: %.3f", divisao);
}
  
```

Figura 27- Função para dividir duas frações

O output da função é o seguinte:

```

C:\ex3EDS
Insira o numerador: 2
Insira agora o denominador: 3
Insira o numerador: 2
Insira agora o denominador: 3
A soma das fracoes e: 1.333
A subtracao das fracoes e: 0.000
O produto das fracoes e: 0.444
O resultado da divisao das fracoes e: 1.000

```

Figura 28- Interface da função divide_fracao

f) Determinar a potência (com expoente inteiro) de uma fracção.

Na última alínea, fizemos na mesma a divisão do numerador pelo denominador da primeira e segunda fração para determinarmos o valor decimal das duas frações. Para determinar a potência das duas frações utilizamos a função pow que tem dois parâmetros, sendo o primeiro parâmetro o valor da divisão da fração e o segundo parâmetro o valor do expoente que é inserido pelo utilizador e o que esta função faz é o primeiro parâmetro é elevado ao segundo parâmetro. A função e o output são os seguintes:

```

void expoente_fracao(frac a, frac b)
{
    double expoentea, expoenteb, fraca, fracb;
    int expoente;
    printf("\nInsira o expoente ao qual quer elevar as fracoes: ");
    scanf("%d", &expoente);
    fraca = a.numerador / a.denominador;
    fracb = b.numerador / b.denominador;
    expoentea = pow(fraca, expoente);
    expoenteb = pow(fracb, expoente);
    printf("\nO valor do expoente da fracao a e: %lf", expoentea);
    printf("\nO valor do expoente da fracao b e: %lf", expoenteb);
}

```

Figura 29- Fração para determinar a potência de uma fração

```
ex3EDS
Insira o numerador: 2
Insira agora o denominador: 3
Insira o numerador: 2
Insira agora o denominador: 3
A soma das fracoes e: 1.333
A subtracao das fracoes e: 0.000
O produto das fracoes e: 0.444
O resultado da divisao das fracoes e: 1.000
Insira o expoente ao qual quer elevar as fracoes: 2
O valor do expoente da fracao a e: 0.444444
O valor do expoente da fracao b e: 0.444444
Press any key to continue . . .
```

Figura 30- Interface da função expoente_fracao

2.5 Uma pequena empresa familiar pretende organizar os dados dos seus funcionários: número, nome, tarefa, salário.

- Defina uma estrutura de dados que permita representar cada funcionário.
- Declare uma tabela que armazene informação pretendida.

A estrutura que definimos foi a seguinte:

```
#include <stdio.h>
#include <string.h>
struct funcionarios{
    int numero;
    char nome[100];
    char tarefa[100];
    float salario;
};
typedef struct funcionarios func;
```

Figura 31- Estrutura definida para armazenar informações sobre funcionários

Já a tabela foi chamada no main:

```
printf("\nInsira o numero de funcionarios: ");
scanf("%d", &nfunc);
func dados[nfunc];
```

Figura 32- Tabela criada para armazenar informação sobre vários funcionários

c) Elabore subprogramas para:

i) Introduzir os dados de um funcionário na tabela:

Para tal, simplesmente fizemos operações básicas de leitura, como o gets e o scanf:

```
void IntroduzirDados(int nmrfunc, func dados[nmrfunc])
{
    for(int i=0;i<nmrfunc;i++)
    {
        printf("\nNome: ");
        fflush(stdin);
        gets(dados[i].nome);
        printf("\nNumero: ");
        scanf("%d", &dados[i].numero);
        printf("\nTarefa: ");
        fflush(stdin);
        gets(dados[i].tarefa);
        printf("\nSalario: ");
        scanf("%f", &dados[i].salario);
    }
}
```

Figura 33- Função para leitura e armazenamento de dados dos funcionários

Exemplo de output para 2 funcionários:

```
Nome: Antonio Mendes
Numero: 2000
Tarefa: java dev
Salario: 2000
Nome: Josué Silva
Numero: 1500
Tarefa: Marketing
Salario: 1500
```

Figura 34- Interface da função IntroduzirDados

ii) Listar todos os funcionários e respetivos dados:

À luz da função anterior, para esta função simplesmente utilizámos printf's dentro de um ciclo for:

```
void ApresentarDados(int nmrfunc, func dados[nmrfunc])
{
    for(int i=0;i<nmrfunc;i++)
    {
        printf("Nome: %s\n", dados[i].nome);
        printf("Numero: %d\n", dados[i].numero);
        printf("Tarefa: %s\n", dados[i].tarefa);
        printf("Salario: %.1f\n\n", dados[i].salario);
    }
}
```

Figura 35- Função para exibir dados de funcionários

Output de acordo com os dados inseridos em cima:

```
Nome: Antonio Mendes
Numero: 2000
Tarefa: Java dev
Salario: 2000.0

Nome: Josué Silva
Numero: 1500
Tarefa: Marketing
Salario: 1500.0
```

Figura 36- Exemplo de output

iii) Listar os funcionários com salário superior a 500€:

Para listar os funcionários com salário superior a 500€, bastou fazermos um ciclo para percorrer a lista de salários e caso fosse maior que 500, apresentaria os dados dos respetivos funcionários:

```
void ListarAcima500(int nmrfunc, func dados[nmrfunc])
{
    for(int i=0;i<nmrfunc;i++)
    {
        if(dados[i].salario > 500)
        {
            printf("Nome: %s\n", dados[i].nome);
            printf("Numero: %d\n", dados[i].numero);
            printf("Tarefa: %s\n", dados[i].tarefa);
            printf("Salario: %.1f\n\n", dados[i].salario);
        }
    }
}
```

Figura 37- Função para apresentar funcionários com salário superior a 500€

Output de acordo com os dados de input:

```
Funcionarios com salario acima de 500 euros:

Nome: Antonio Mendes
Numero: 2000
Tarefa: Java dev
Salario: 2000.0

Nome: Josué Silva
Numero: 1500
Tarefa: Marketing
Salario: 1500.0
```

Figura 38- Interface da função ListarAcima500

iv) Procurar e listar todos os dados de um funcionário, usando o seu nome (caso o funcionário não exista, deverá ser devolvida uma “estrutura vazia”: números a zero e strings vazias):

Para tal, tivemos que criar uma estrutura vazia dentro da função para retorná-la caso não encontre nenhum funcionário. Para encontrar os funcionários, fizemos um ciclo for para percorrer a informação e usámos um strcmp para verificar se o nome coincide:

```
func ProcurarFuncionario(int nmrfunc, func dados[nmrfunc])
{
    char procurar[100];
    func b;
    strcpy(b.nome, "");
    strcpy(b.tarefa, "");
    b.numero = 0;
    b.salario = 0;
    printf("\nIntroduza o nome do funcionario que deseja encontrar: ");
    fflush(stdin);
    gets(procurar);
    for(int i=0;i<nmrfunc;i++)
    {
        if(strcmp(dados[i].nome, procurar)==0)
            return dados[i];
    }
    return b;
}
```

Figura 39- Função para procurar funcionário

Output (funcionário encontrado):

```
Introduza o nome do funcionario que deseja encontrar:Antonio Mendes
Nome: Antonio Mendes
Numero: 2000
Tarefa: Java dev
Salario: 2000.0
```

Figura 40- Exemplo de output

v) Atualizar os dados de um funcionário (utilizando o seu número)

Para fazer esta função, simplesmente fizemos um ciclo para percorrer os dados dos números dos funcionários e, caso existisse o funcionário com o número correspondente, pedimos ao utilizador para meter todos os dados de novo, de modo a alterar:

```
void AtualizarDados(int nmrfunc, func dados[nmrfunc])
{
    int numerofunc;
    printf("\nInsira o numero do funcionario cujos dados quer atualizar: ");
    scanf("%d", &numerofunc);
    for(int i=0; i<nmrfunc; i++) {
        if (numerofunc == dados[i].numero) {
            printf("\nInsira de novo os dados: ");
            printf("\nNome: ");
            fflush(stdin);
            gets(dados[i].nome);
            printf("\nNumero: ");
            scanf("%d", &dados[i].numero);
            printf("\nTarefa: ");
            fflush(stdin);
            gets(dados[i].tarefa);
            printf("\nSalario: ");
            scanf("%f", &dados[i].salario);
        }
    }
}
```

Figura 41- Função para atualizar dados de funcionários.

Output:

```
Insira o numero do funcionario cujos dados quer atualizar:2000

Insira de novo os dados:
Nome:Antonio Jose Mendes

Numero:2100

Tarefa:Java dev

Salario:2100
```

Figura 42- Interface da função AtualizarDados

Novos dados do funcionário:

```
Introduza o nome do funcionario que deseja encontrar:Antonio Jose Mendes
Nome: Antonio Jose Mendes
Numero: 2100
Tarefa: Java dev
Salario: 2100.0
```

Figura 43- Dados Atualizados

vi) Ordene a tabela por ordem crescente dos números dos funcionários

Para esta função, usámos um algoritmo de ordenação da seguinte maneira:

```
void OrdenNumero(int nmrfunc, func dados[nmrfunc])
{
    func aux;
    for(int i=0;i<nmrfunc;i++)
    {
        for(int x=0;x<nmrfunc;x++)
            if (dados[x + 1].numero < dados[x].numero)
            {
                aux = dados[x];
                dados[x] = dados[x + 1];
                dados[x + 1] = aux;
            }
    }
}
```

Figura 44- Função para ordenar os funcionários por número

Output:

```

Numeros ordenados:
Nome: Josué Silva
Numero: 1500
Tarefa: Marketing
Salario: 1500.0

Nome: Antonio Jose Mendes
Numero: 2100
Tarefa: Java dev
Salario: 2100.0
```

Figura 45- Exemplo de output

vii) Ordenar os funcionários por ordem alfabética

Para conseguirmos fazer esta função, aproveitámos a funcionalidade da função “strcmp”, que se encontra na biblioteca “string.h”. O que fará é basicamente comparar a ordem alfabética dos nomes e caso o primeiro nome seja primeiro, irá retornar um valor maior que 0, caso seja em segundo, retorna um valor negativo e caso sejam iguais, retorna 0. Depois foi só adaptar com o algoritmo normal de ordenação:

```
void OrdemNome(int nmrfunc, func dados[nmrfunc])
{
    func aux;
    for(int i=0;i<nmrfunc;i++)
    {
        for(int x=0;x<nmrfunc;x++)
            if (strcmp(dados[x].nome, dados[x+1].nome)>0)
            {
                aux = dados[x];
                dados[x] = dados[x + 1];
                dados[x + 1] = aux;
            }
    }
}
```

Figura 46- Função para ordenar os clientes por ordem alfabética

Output:

```
Nomes ordenados:
Nome: Antonio Jose Mendes
Numero: 2100
Tarefa: Java dev
Salario: 2100.0

Nome: Josué Silva
Numero: 1500
Tarefa: Marketing
Salario: 1500.0
```

Figura 47- Interface da função OrdemNome

3 Conclusão

Concluindo, este trabalho foi bastante interessante e útil para nos familiarizarmos com as estruturas. Também, foi uma mais valia para nós, pois, aprendemos novos conceitos neste caso as estruturas. Assim, podemos dizer que foi um bom trabalho e esperamos que estes conhecimentos que aprendemos sejam úteis para o futuro.

4 Referências



ficha4-estruturas.p
df