Ficha de Trabalho n.º 0 – 1.ª Ficha de Revisão
Proposta de Resolução

Objectivos: Revisão sobre ponteiros, memória dinâmica, usando exercícios de provas de AP.

1. Desenvolva uma função que dada uma matriz bidimensional MxN de inteiros indique se é esparsa. Se a matriz for esparsa é retornado 1 senão é retornado 0. Uma matriz é esparsa, se a percentagem de elementos nulos estiverem acima de um determinado limite (percentagem).

```
int Esparsa(int *Matriz, int M, int N, float percentagem)
```

```
int Esparsa(int *Matriz, int M, int N, float percentagem)
{
   int i, j, nElemZeros=0;
   for (i=0; i<M; i++)
   {
      for (j=0; j<N; j++)
      {
         if(*(Matriz+i*N+j)==0)
            nElemZeros++;
      }
   }
   if (((float)nElemZeros/(M*N))*100 < percentagem)
      return 0;
   else
      return 1;
}
```

2. Pretende-se analisar um conjunto de músicas. Cada música caracteriza-se pelo título, intérprete, ano, single (1:sim ou 0:não) e número de downloads nas plataformas aderentes.

Obs. 1. n, em todos os cabeçalhos das funções, representa o número de músicas no vetor lista;
2. utilize ponteiros em todas as alíneas para aceder ao valor dos atributos dos elementos do vetor.

a) Defina o tipo de dados **Musica** e crie um vetor de nome lista, com um número de músicas definido pelo utilizador (valor disponível na variável nMusicas).

```
typedef struct musica
{
    char titulo[50];
    char interprete[50];
    short ano;
    short single;
    int nDownloads;
} Musica;

Musica *lista = (Musica *) malloc(nMusicas * sizeof(Musica));
```

b) Implemente uma função que escreva no ecrã, as características das músicas com mais do que um determinado número de downloads (parâmetro numD).

```
void musicasDownload(Musica *lista, int n, int numD)
```

```
void MusicasDownLoad(Musica *lista, int n, int numD)
{
    int i=0, j=0;
    char tpD [4];
    for (i=0; i<n; i++)
    {
        if((lista+1)->nDownloads > numD)
        {
            j++;
            if((lista+i)->single == 1)
                strcpy(tpD, "Sim");
            else
                strcpy(tpD, "Não");
            printf("\nMúsica %d -->", j);
            printf("\nTítulo: %s, Intérprete: %s, Ano: %d, É Single? %s,
                                                    Número de Dowloads: %d",
                  (lista+i)->titulo, (lista+i)->interprete, (lista+i)->ano, tpD,
                                          (lista+i)->nDownloads);
        }
    }
}
```

c) Desenvolva uma função que receba a listagem de músicas e devolva quantas têm um dado intérprete.

```
int interpIdade(Musica *lista, int n, char *cantor)
{
    int i=0, nMusCantor=0;
    for (i=0; i<n; i++)
    {
        if(strcmp((lista+i)->interprete, cantor) == 0)
            nMusCantor++;
    }
    return nMusCantor;
}
```

d) Desenvolva uma função que guarde num ficheiro binário, as músicas singles, não singles ou ambas (parâmetro tpMeio='s', 'n' ou 'a´, respetivamente), posteriores a um determinado ano. A função deve ainda devolver o número médio de downloads dessas músicas.

```
float gravaFichMusicas (Musica *lista, int n, int ano, char *nFich, char tpMeio)
```

```
Obs. Sintaxes:    FILE* fopen(const char* fileName, const char* mode);
                  size_t fwrite (const void* ptr, size_t size, size_t nElements, FILE* file);
```

```
float gravaFichMusicas(Musica *lista, int n, int ano, char *nFichNome, char tpMeio)
{
    float totDwnL=0;
    int i, nMusicas=0;
    FILE *f = fopen(nFichNome, "w+b");
    if (f==NULL)
    {
        printf("Impossível abrir o ficheiro!!");
        return -1.0;
    }
    for (i=0; i<n; i++)
    {
        if((lista+i)->ano > ano)
```

```
        {
            switch (tpMeio)
            {
            case 'a':
                fwrite((lista+i), sizeof(Musica), 1, f);
                totDwnL += (lista+i)->nDownloads;
                nMusicas++;
                break;
            case 's':
                if((lista+i)->single == 1)
                    fwrite((lista+i), sizeof(Musica), 1, f);
                    totDwnL += (lista+i)->nDownloads;
                    nMusicas++;
                    break;
            case 'n':
                if((lista+i)->single == 0)
                    fwrite((lista+i), sizeof(Musica), 1, f);
                    totDwnL += (lista+i)->nDownloads;
                    nMusicas++;
                    break;
            }
        }
    }
    fclose(f);
    return (totDwnL/nMusicas);
}
```

3. Pretende-se efetuar um estudo sobre os resultados de avaliação de um determinado número de alunos. Para cada aluno, regista-se o seu número mecanográfico, o seu nome e os resultados de avaliação de 5 UCs. Cada UC caracteriza-se pelo nome e respetiva avaliação.

Obs. 1. **n**, em todos os cabeçalhos das funções, representa o número de alunos no vetor lista;
2. utilize **ponteiros** em todas as alíneas para aceder ao valor dos atributos dos elementos do vetor.

a) Defina o tipo de dados Aluno, UC e crie um vetor de nome ListAlun para guardar a informação necessária para este estudo, supondo que se pretendem guardar nReg elementos, sendo nReg especificado pelo utilizador.

```
typedef struct uc{
        char nome[20];
        float aval;
}UC;

typedef struct aluno {
        char nome[30];
        int num;
        UC notas[5];
}ALUNO;
```

```
ALUNO *listAlun;
listAlun = (ALUNO*)malloc(sizeof(ALUNO)*nReg);
```

b) Para cada aluno, calcula-se a média das notas. Considera-se que um aluno transita se a média, arredondada à unidade, for igual ou superior a 10 valores. Defina a função qtosPassam que recebe a lista de alunos e calcula e devolve quantos alunos transitam.

Solução 1

```
// Exercício 4 - b)
int qtosPassam(ALUNO *lista, int n)
{
   int i, cont, j;
   float soma, media;
   cont = 0;
   for (i = 0; i < n; i++)
   {
      soma = 0;
      for (j = 0; j < 5; j++)
      {
         soma = soma + lista->notas[j].aval;
      }
      media = soma / 5;
      if (media > 9.5)
         cont++;
      lista++;
   }
   return cont;
}
```

Solução 2 (todos ponteiros)

```
// Exercício 4 – b)
int qtosPassam(ALUNO *lista, int n)
{
    int i, cont,j;
    float soma, media;
    cont = 0;
    UC *uc1;
    for (i = 0; i < n; i++)
    {
        soma = 0;
        uc1=&lista->notas[0];
        for (j = 0; j < 5; j++)
        {
            soma = soma + uc1->aval;

            //soma = soma + (*lista).notas[j].aval;
            printf("Nota da UC=%.1f\n",uc1->aval);
            uc1++;
        }
        media = soma / 5;
        if (media > 9.5)
            cont++;
        lista++;
    }
    return cont;
}
```

c) Determine o número médio de UCs, com notas inferiores a lim (e.g., 8) valores (considere esta média de UCs, como a soma total do número de UCs com classificação inferior a lim, a dividir pelo número de alunos).

```
float numUcInfLim(ALUNO *lista, int n, int lim)
{
   int i,j;
   int soma;
   soma = 0;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < 5; j++)
      {
         if (lista->notas[j].aval < lim)
            soma=soma+1;
      }
      lista++;
   }
   return soma*1.0 / n;}
```

d) Elabore uma função que permita listar, num ficheiro de texto, os alunos com as UCs onde obtiveram (val) ou mais valores. No fich., deve haver uma linha com o nome do aluno e depois novas linhas para o nome e nota das UC que corresponderem à condição estabelecida. Depois, uma linha tipo "----------" como separador, seguindo-se um novo aluno e notas respetivas e assim sucessivamente.

```
void listaNotas(Aluno *lista, int n, int val, char *nomeFich)
```

Obs.: Em nomeFich está disponível o nome do ficheiro a usar.

```
Obs. Sintaxes: FILE* fopen(const char* fileName, const char* mode);
                                    fprintf (const void* ptr, …);
```

```
void listaNotas(ALUNO *lista, int n, int val, char *nomeFich)
{
    int i,j;
    FILE* f;
    f=fopen(nomeFich, "w");
    for (i = 0; i < n; i++)
    {
        fprintf(f, "%s\n", lista->nome);
        for (j = 0; j < 5; j++)
        {
            if (lista->notas[j].aval >= val)
                fprintf(f, "%s -> %f\n",
                    lista->notas[j].nome, lista->notas[j].aval);
        }
        fprintf(f, "---------\n");
    }
}
```

4. Pretende-se efetuar um estudo sobre a saúde de um determinado número de indivíduos, num dado ano. Para cada indivíduo, regista-se o seu número do Cartão de Cidadão, a sua altura, a sua massa, o género (M/F) e a data de nascimento (dia, mês, ano).

Obs. 1. **n**, em todos os cabeçalhos das funções, representa o número de indivíduos no vetor lista;
2. utilize **ponteiros** em todas as alíneas para aceder ao valor dos atributos dos elementos do vetor.

a) Defina o tipo de dados Indiv, Data e crie um vetor de nome ListIndiv para guardar a informação necessária para este estudo, supondo que se pretendem guardar nReg elementos, sendo nReg especificado pelo utilizador;

```
typedef struct data
{
      int dia, mes, ano;
} DATA;

typedef struct indiv
{
      int CC;
      DATA nasc;
      float massa, alt;
      char genero;
} Indiv;
```

```
Indiv *ListIndiv;
ListIndiv = (Indiv*)malloc(sizeof(Indiv)*nReg);
```

b) Para cada indivíduo calcula-se o Índice de Massa Corporal (IMC) como IMC = massa / altura^2. Considera-se que um indivíduo é saudável, se o seu IMC estiver compreendido entre 18,5 e 24,9. Defina a função indSaudaveis que recebe a lista de indivíduos e calcula e devolve quantos indivíduos são saudáveis.

```
int indSaudaveis(Indiv *vet_ind, int n)
{
       int num, i;
       float imc;
       num = 0;
       for (i = 0; i < n; i++)
       {
        imc = vet_ind->massa / ((vet_ind->alt)*(vet_ind->alt));
        if ((imc >= 18,5)&&(imc <= 24,9))
               num++;
        vet_ind++;
       }
       return num;
}
```

c) Determine a média da massa corporal dos indivíduos nascidos entre dois anos especificados como parâmetros.

```
float mediaMassCorp(Indiv *vet_ind, int n, int anoInf, int anoSup)
{
       int num, i;
       float soma;
       num = 0; soma = 0;
       for (i = 0; i < n; i++)
       {
        if ((vet_ind->nasc.ano >= anoInf)&&(vet_ind->nasc.ano <= anoSup))
        {
               soma += vet_ind->massa / ((vet_ind->alt)*(vet_ind->alt));
               printf("soma=%f\n", soma);
               num++;
        }
        vet_ind++;
       }
       if (num > 0)
        return soma / num;
       else
        return 0;
  }
```

d) Elabore uma função que permita remover do vetor um determinado indivíduo, devendo o vetor ficar ajustado, ou seja, não conter elementos vazios.

```
void removeIndiv(Indiv *vet_ind, int n, int nCC)
```

```
void removeIndiv(Indiv *vet_ind, int *n, int nCC)
{
       int i,j;
       i = 0;
       while ((vet_ind->CC != nCC)&&(i < *n))
       {
        vet_ind++; i++;
       }
       if (i < *n)
```

```
        {
         for (j = i; j < *n - 1; j++)
                 *(vet_ind + j) = *(vet_ind + j + 1);
         *n = *n - 1;
        }
}
```

5. Crie um main() que permita testar todas as funções criadas.

```
//
//  main.c
//  Ficha0
//

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
// Ex. 2
typedef struct musica
{
    char titulo[50];
    char interprete[50];
    short ano;
    short single;
    int nDownloads;
} Musica;

// Ex. 3
typedef struct uc{
    char nome[20];
    float aval;
}UC;

typedef struct aluno {
    char nome[30];
    int num;
    UC notas[5];
}ALUNO;

// Ex. 4
typedef struct data
{
    int dia, mes, ano;
} DATA;

typedef struct indiv
{
    int CC;
    DATA nasc;
    float massa, alt;
    char genero;
} Indiv;

// Ex. 1
int Esparsa(int *Matriz, int M, int N, float percentagem);

// Ex. 2
int interpIdade(Musica *lista, int n, char *cantor);
void MusicasDownLoad(Musica *lista, int n, int numD);
float gravaFichMusicas(Musica *lista, int n, int ano, char *nFichNome, char tpMeio);

// Ex. 3
```

```
int qtosPassam (ALUNO *lista, int n);
float numUcInfLim(ALUNO *lista, int n, int lim);
void listaNotas(ALUNO *lista, int n, int val, char *nomeFich);

// Ex. 4
int indSaudaveis(Indiv *vet_ind, int n);
float mediaMassCorp(Indiv *vet_ind, int n, int anoInf, int anoSup);
void removeIndiv(Indiv *vet_ind, int *n, int nCC);

int main(int argc, const char * argv[]) {
// Ex. 1
    printf("\n\n Exercício 1....\n");

    float perc=0.0;
    int mat1[9]={1,2,0,0,0,0,9,4,5};
,
    // ou com memória dinâmica
    /*
    int * mat;
    mat=(int *) malloc(9*sizeof(int));
    *(mat+0)=1;
    *(mat+1)=2;
    *(mat+2)=0;
    *(mat+3)=0;
    *(mat+4)=0;
    *(mat+5)=0;
    *(mat+6)=9;
    *(mat+7)=4;
    *(mat+8)=5;
    */
    printf("\nQual a percentagem a considerar apara considerar a matriz esparsa? ");
    scanf("%f.2", &perc);
    if (Esparsa(mat1, 3, 3, perc))
     printf("\nA matriz é esparsa!");
    else
     printf("\nA matriz não é esparsa!");

    // ou com memória dinâmica
    // free (mat);

    // Ex. 2
    printf("\n\n Exercício 2....\n");
    int nMusicas=4;
    // 2.a)
    Musica *musicas = (Musica *) malloc(nMusicas * sizeof(Musica));

    // 2.b)
    MusicasDownLoad(musicas, nMusicas, 1000);

    // 2.c)
    char cantor[50]="Pink Floyd";
    strcpy(musicas->titulo, "Hello");
    strcpy(musicas->interprete, "Lionel Ritchie");
    musicas->ano=1985;
    musicas->single=1;
    musicas->nDownloads=290;

    strcpy((musicas+1)->titulo, "Hotel California");
    strcpy((musicas+1)->interprete, "Eagles");
    (musicas+1)->ano=1980;
    (musicas+1)->single=0;
    (musicas+1)->nDownloads=429;
```

```
        strcpy((musicas+2)->titulo, "Confortably Numb");
        strcpy((musicas+2)->interprete, "Pink Floyd");
        (musicas+2)->ano=1987;
        (musicas+2)->single=1;
        (musicas+2)->nDownloads=1390;

        strcpy((musicas+3)->titulo, "Money");
        strcpy((musicas+3)->interprete, "Pink Floyd");
        (musicas+3)->ano=1975;
        (musicas+1)->single=0;
        (musicas+3)->nDownloads=1429;

        printf("\nO número de músicas dos Pink Floyd é %d",
                        interpIdade(musicas, nMusicas, cantor));

        // 2.d)
        int nAno=1980;
        char nFich[20]="FichMusicas.dat";
        printf("\nO número médio de downloads das músicas de ano > %d foi %.2f", nAno,
                        gravaFichMusicas(musicas, nMusicas, nAno, nFich, 's'));
        // Libertar a memória
        free (musicas);

        // Ex. 3
        printf("\n\n Exercício 3....\n");
        // 3.a)
        int nReg,i,j;
        char *fich = "Fich";
        ALUNO *listAlun;
        printf("\nQuantos alunos? ");
        scanf("%d", &nReg);
        listAlun = (ALUNO*)malloc(sizeof(ALUNO)*nReg);

        //3.b)
        char s[30];
        for (i = 0; i < nReg; i++)
        {
            strcpy((listAlun+i)->nome, "abel");
            (listAlun+i)->num = i + 5010;
            for (j = 0; j < 5; j++)
            {
                if(i%2)
                    (listAlun+i)->notas[j].aval = 8 + j;
                else
                    (listAlun+i)->notas[j].aval = 6 + j;
                sprintf(s, "UC %d", j+1);
                strcpy((listAlun+i)->notas[j].nome, s);
            }
        }
        printf("Passam %d \n", qtosPassam(listAlun, nReg));

        // 3.c)
        printf("media de ucs: %f\n", numUcInfLim(listAlun, nReg, 9));

        // 3.d)
        listaNotas(listAlun, nReg, 10, fich);

        // Libertar a memória
        free (listAlun);

        // Ex. 4
        printf("\n\n Exercício 4....\n");
        // 4.a)
        int nReg1;
```

```
        Indiv *ListIndiv;
        printf("\nQuantos individuos? ");
        scanf("%d", &nReg1);
        ListIndiv = (Indiv*)malloc(sizeof(Indiv)*nReg1);
        for (i = 0; i < nReg1; i++)
        {
            printf("\nCC, alt, massa? ");
            scanf("%d%f%f", &ListIndiv->CC, &ListIndiv->alt, &ListIndiv->massa);
            printf("\ndia, mes, ano: ");
            scanf("%d%d%d", &ListIndiv->nasc.dia, &ListIndiv->nasc.mes,
                                &ListIndiv->nasc.ano);
            fseek(stdin, 0, 2);
            printf("gen: ");
            scanf(" %c", &ListIndiv->genero);
        }
        // 4.b)
        printf("\nSaudáveis: %d\n", indSaudaveis(ListIndiv, nReg1));
        // 4.c)
        printf("\nMédia imc: %.2f", mediaMassCorp(ListIndiv, nReg1, 1981, 2000));
        // 4.d)
        removeIndiv(ListIndiv, &nReg1, 22);
        printf("\n%d\n", nReg1);
        // Libertar a memória
        free (ListInviv);


        return 0;
    }
```