Programação Imperativa – EI (1º ano)

Mini-Teste 4C

Data: 2 de Maio de 2012
Hora: TP3

Dispõe de **40 minutos** para realizar este mini-teste.
Nome: .....   Número: .....

Questão 1 (produto cartesiano)

Foste encarregado de criar uma pequena aplicação para gerir uma agência matrimonial (aquelas onde uma pessoa à procura de um parceiro se dirige). O sistema de informação que irá suportar a aplicação é basicamente constituído por 3 entidades: a informação sobre a agência (nome, morada, telef, etc), a base de dados de pessoas registadas e uma lista de pedidos de *"compatibilidade/matching"*. Foi-te pedido que desenvolvesses algumas funcionalidades sobre o modelo de dados do protótipo que se apresenta a seguir:

```
#define SIM 'S'
#define NAO 'N'

typedef struct sData
  {
    int ano, mes, dia;
  } Data;

typedef struct sAtributo
  {
    char *designacao;     /* "fumador", "desportista", "música", ... */
    char valor;    /* SIM ou NAO */
  } Atributo;

typedef struct sLAtrib    /* a lista de atributos de uma pessoa */
  {
    Atributo a;
    struct sLAtrib *seg;
  } *LAtrib, NLAtrib;

typedef struct sPessoa    /* a Pessoa */
  {
    char* nome;
    Data nascimento;
    char sexo;
    char* cidade;
    LAtrib atributos;
  } Pessoa;

 typedef struct sLPessoa    /* a Lista de Pessoas */
  {
    Pessoa p;
    struct sLPessoa *seg;
  } *LPessoa, NLPessoa;
```

```
typedef struct sPedido
  {
    char* nome;   /* quem fez o pedido */
    char sexo;
    LAtrib atributos;
  }  Pedido;

typedef struct sLPedido    /* a Lista de Pedidos de Match */
  {
    Pedido ped;
    struct sLPedido *seg;
  } *LPedido, NLPedido;

typedef struct sAgencia
  {
    char *nomeag;
    char *morada;
    char *telef;
    LPessoa pessoas;
    LPedido pedidos;
  } Agencia;
```

Que poderão ser usados num programa da seguinte forma:

```
int main()
{
   Agencia matrimonius;
   Pessoa p1 = {"Ana", {1982, 2, 7}, 'F', "Porto", NULL }, p2;
   LPessoa lpessoas = NULL;
   Data dfran = {1981, 7, 11};
   LAtrib franatrib = NULL;
   Pedido ped1;
   ...
   p2 = consPessoa( "Francisco", dfran, 'M', "Braga", franatrib );
   ...
   ped1 = InsAtribPedido( ped1, "Música", SIM );
   ped1 = InsAtribPedido( ped1, "Fumador", NAO );
   ...
   matrimonius = InsereCandidato( matrimonius, p2 );
   ...
   ListarNaoFumadores( matrimonius );
   ...
   lpessoas = Match( matrimonius, 4 );
   ...
}
```

Especifique as funções utilizadas no exemplo:

Pessoa consPessoa(char* n, Data d, char s, char* cidade, LAtrib la) - que constrói uma *Pessoa* com os valores que lhe são passados;

Pedido InsAtribPedido(Pedido p, char* id, char v) - que acrescenta o atributo com nome *id* e valor *v* à lista de atributos do pedido *p* e devolve o novo pedido;

Agencia InsereCandidato(Agencia a, Pessoa p) - que dá como resultado uma nova agência resultante da inserção na nova pessoa;

void ListarNaoFumadores(Agencia a) - que lista no monitor as pessoas (nome, data de nascimento e sexo) que têm "Fumador"com o valor NAO na sua lista de atributos (a ausência do atributo pode ser assumida como NAO);

LPessoa Match(Agencia a, int npedido) - que dá como resultado a lista de pessoas que têm um *match* superior a 70% com a pessoa que consta do pedido na posição *npedido* da lista de pedidos. O *match* é calculado pelo quociente entre o número de atributos em comum e o total de atributos de cada pessoa registada na agência;