utad Universidade de Trás-os-Montes e Alto Douro

Escola de Ciências e Tecnologia. Departamento de Engenharias

Eng. Informática

unidade curricular: **Algoritmia** ano lectivo: **2017 / 2018**

Teste Prático 02. **30.Mai.2018.** duração: **60 min** (+5 min tolerância)

(sem consulta)

Número: _____ Nome: _____

Considere o seguinte programa em linguagem C:

```
typedef struct _DEPT {
   char     desig[20];
   float    ganhos,despesas;
   int      n_pessoas;
}DEPT;

typedef struct _BTREE_NODE {
  void * data;
  struct _BTREE_NODE * left;
  struct _BTREE_NODE * right;
} BTREE_NODE;

#define DATA(node)    ((node)->data)
#define LEFT(node)    ((node)->left)
#define RIGHT(node)   ((node)->right)
typedef BTREE_NODE * BTREE;
typedef enum _BOOLEAN {FALSE = 0,TRUE = 1} BOOLEAN;
typedef enum _STATUS {ERROR = 0,OK = 1} STATUS;

int main(int argc, char *argv[])
{
   BTREE btree;
   void *   depts[15];
   char  file_name[20];

   printf("Nome do ficheiro: ");
   scanf("%s", file_name);
   if(ReadFile(depts,file_name)) {
      btree = CreateBtree(depts,0,15);









      BtreeFree(btree);
   }
   else printf("ERRO na leitura do ficheiro\n");
   return 1;
}
```

Invoque as funções necessárias ao funcionamento do programa, a partir do `main`

```
BTREE_NODE * NewBtreeNode(void * data)
{
    BTREE_NODE * tmp_pt;
    if ((tmp_pt = (BTREE_NODE *)malloc(sizeof(BTREE_NODE)))!=NULL)   {
          DATA(tmp_pt) = data;
          LEFT(tmp_pt) = RIGHT(tmp_pt) = NULL;   }
    return tmp_pt;
}
```

```
void BtreeFree(BTREE btree)
{
    if (btree != NULL)   {
        BtreeFree(LEFT(btree));
        BtreeFree(RIGHT(btree));
        Free(DATA(btree));
        free(btree);   }
}
BTREE_NODE *InitNode(void * ptr_data,BTREE_NODE * node1,BTREE_NODE * node2)
{
    BTREE_NODE * tmp_pt = NULL;
    tmp_pt = NewBtreeNode(ptr_data);
    LEFT(tmp_pt) = node1;
    RIGHT(tmp_pt) = node2;
    return(tmp_pt);
}
STATUS ReadFile(void ** depts, char * file_name)
{
    FILE * fp;
    Int j, i = 0;
    void * ptr_data;

    if ((fp = fopen(file_name, "r")) != NULL)   {
        while (!feof(fp)) {
            if ((ptr_data = malloc(sizeof(DEPT))) != NULL)   {
                fscanf(fp, "%[^;];%d;%f;%f\n", ((DEPT *)ptr_data)->desig, &((DEPT *)ptr_data)->n_pessoas,
                        &((DEPT *)ptr_data)->ganhos, &((DEPT *)ptr_data)->despesas);
                depts[i] = ptr_data;
                i++;   }
            else {
                for (j = i; j >= 0; j--)
                    free(depts[j]);
                return(ERROR);   }
        }
        fclose(fp);
        return(OK);   }
    else
        return(ERROR);
}
BTREE_NODE *CreateBtree(void ** v, int i, int size)
{
    if( i >= size)    return(NULL);
    else              return(InitNode(*(v+i),CreateBtree(v,2*i+1,size),CreateBtree(v,2*i+2,size)));
}
```

Suponha um programa que, através de uma árvore binária genérica, representa de forma hierárquica os diversos departamentos de uma empresa (16 departamentos). Os dados de cada departamento são: designação, nº de pessoas que nele trabalham, volume de despesas e volume de ganhos.

01 Desenvolva o código necessário para apresentar no ecrã os lucros da empresa: volume de ganhos - volume de despesas.

02 Desenvolva o código necessário para mostrar no ecrã os dados do departamento com o maior número de pessoas.

03 Desenvolva o código necessário para eliminar todos os sub-departamentos de um determinado departamento, cuja designação deve ser lida através do teclado, transferindo todos os trabalhadores dos sub-departamentos para este.