Programação Imperativa – EI (1º ano)

Teste final

15 de Junho de 2010 (11h00)

Dispõe de **2:00 horas** para realizar este teste.

Questão 1 (cálculo numérico)

Considere a seguinte definição em C para o tipo abstracto de dados *Fraccao*:

```
typedef struct sFraccao
  {
    int num;
    int den;
  } Fraccao;
```

Tendo como base esta definição, defina duas funções, tais que:

a) `simplifica` — recebe uma fracção, e devolve como resultado a fracção na sua forma irredutível:

```
Fraccao simplifica( Fraccao f )
{
  ...
}
```

b) `somafrac` — recebe duas fracções, e devolve como resultado a fracção irredutível resultante de somar as duas fracções que são passadas como argumento;

```
Fraccao somafrac( Fraccao f1, Fraccao f2 )
{
  ...
}
```

Questão 2 (arrays)

Como sabe uma string é um array especial. Desenvolva as seguintes alíneas:

a) Defina uma função `int subesp(char *s)` que recebe uma string (terminada com o caracter `'\0'`) e substitui os espaços por um hífen e devolve o número de substituições realizadas;

b) Defina uma função `int remCons (char *s)` que recebe uma string (terminada com o caracter `'\0'`) e apaga todos os caracteres iguais ao anterior. A função pretendida deverá fazer esta operação sem usar um array auxiliar e deve retornar o comprimento da string resultante.

Questão 3 (pergunta dada)

Considere uma árvore binária de procura definida pelo seguinte tipo abstracto de dados em C:

```
typedef struct sArvBin
  {
    int valor;
    struct sArvBin *esq, *dir;
  }
    *ArvBin, NodoArvBin;
```

Especifique uma função que faz uma listagem *inorder* dos nodos da árvore (na especificação desta função está proibida a utilização de recursividade quer na função principal quer em qualquer função auxiliar).

Questão 4 (estruturas dinâmicas)

Considere uma árvore genealógica ascendente (que relaciona uma pessoa com os seus pais) e que em termos mais formais pode ser descrita da seguinte maneira:

```
AG = Indivíduo * Pai * Mãe
          | desconhecido

Indivíduo = nome * BI * data-nascimento
Pai = Mãe = AG
```

Com base nesta especificação abstracta desenvolva as seguintes alíneas:

a) Especifique tipos abstractos de dados em C que permitam manipular árvores genealógicas. Sugestão: crie o tipo *Individuo* e o tipo *ArvGenoa*;

b) Especifique uma função de travessia que lista no monitor toda a informação guardada na árvore:

```
void ListArvGenoa( ArvGenoa a);
```

c) Defina uma outra função *IndivPorNasc* que, dada uma árvore genealógica, retorna uma lista de indivíduos ordenados por data de nascimento (defina o tipo *IndivList*):

```
IndivList IndivPorNasc(ArvGenoa a);
```

d) Defina uma função *Avos* que, dada uma árvore genealógica e o BI de um indivíduo, retorna a lista de indivíduos que são avós do detentor do BI (caso o elemento com esse BI não exista ou exista e não tenha avós deverá retornar uma lista vazia:

```
IndivList Avos( ArvGenoa a, BI b);
```

e) Defina uma função que calcula o número de gerações representadas na árvore:

```
int Geracoes( ArvGenoa a);
```

f) Defina uma função que calcula a lista de descendentes do indivíduo cujo BI é passado como argumento:

```
IndivList Descendentes( ArvGenoa a, BI b);
```