Programação Imperativa – CC (1º ano)

Teste de Avaliação Final

Data: 22 de Junho de 2009
Hora: 14:30h

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

Questão 1 (cálculo numérico)

O tamanho de papel ISO, de A10 a A1, são calculados com base uns nos outros. Por exemplo, o comprimento de uma folha A4 é a largura de uma folha A3. Do mesmo modo, a largura de uma folha A4 é o comprimento de uma folha A5[1].

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

- `succ` — recebe a largura e comprimento de uma folha, e **imprime** a largura e comprimento da folha de tamanho superior;
- `prec` — recebe a largura e comprimento de uma folha, e **imprime** a largura e comprimento da folha de tamanho inferior;

Questão 2 (arrays)

Pretende-se uma função que receba um array de valores (100 valores) e que os imprima, um por linha, de forma aleatória. Para obter valores aleatórios considere que existe a função `aleatorio(int n)` que retorna um valor x tal que $0 \leq x < n$.

- Implemente em C uma função `aleatoriza(int v[100])` com o comportamento acima descrito.
- Indique que alterações seriam necessárias para que a função pudesse funcionar com arrays com qualquer tamanho.

Questão 3 (recursividade)

Considere a seguinte assinatura de uma função em C:

```
int progarit( int A[], int nelems )
  {
    ...
  }
```

A função *progarit* verifica se uma determinada sequência passada como parâmetro é ou não uma progressão aritmética. A função deverá retornar o valor 0 caso a sequência não seja uma progressão aritmética e deverá retornar n caso contrário, em que n é o termo da progressão.

Por exemplo, considere a seguinte sequência: 0, 3, 6, 9, 12, 15. Neste caso, a função deverá retornar 3 porque é o valor que é somado ao elemento corrente para se obter o elemento seguinte da sequência que é uma progressão aritmética.

Apresente duas especificações para a função *progarit*: uma recursiva e outra iterativa.

Questão 4 (estruturas dinâmicas)

Considere as seguintes estruturas de dados:

[1] Note que as medidas não são valores inteiros.

```
#define OPERADOR 1
#define OPERANDO 2

typedef struct _operador {
  char op;
  struct _expressao *esq;
  struct _expressao *dir;
} operador;

typedef struct _operando {
  float valor;
}

typedef struct _expressao {
  int tipo;
  union {
    operador *o;
    operando *v
  } dados
} expressao;
```

Esta definição de dados especifica uma árvore binária de expressões. Cada nodo pode ter um operador (identificado pela estrutura `operador`) ou um operando (identificado pela estrutura `operando`).
A variável `tipo` na estrutura `expressao` controla o conteudo da `union`, de acordo com as definições `OPERADOR` e `OPERANDO`

- **implemente as funções** `val` e `op` que permitam a construção de uma árvore de expressões do seguinte modo:

  ```
  expressao *arv = op('+', val(3), op('-', val(2), val(4)));
  ```

  A estrutura resultante será a árvore para a expressão $3 + (2 - 4)$

- **implemente a função** `eval` que permita receber uma árvore de expressões, e retorne o resultado de a avaliar (considere apenas os quatro operadores algébricos básicos):

  ```
  float resultado = avalia(arv);
  ```

- defina um novo **tipo de dados** para que esta árvore também suporte a invocação de funções trignométricas como seno, coseno ou tangente.

Questão 5 (a função Y)

Considere as seguintes estruturas de dados definidas em C:

```
typedef struct sListAutor {
    char *autor;
    struct sListAutor *seg;
  } NodoListautor, *ListAutor;

typedef struct sPub {
    char *titulo;
    char* data;
    ListAutor al;
  } Pub;

typedef struct sListPub {
        Pub p;
        struct sListPub *seg;
} NodoListPub, *ListPub;
```

Em determinada aplicação de gestão bibliográfica temos numa variável *pubs* a lista de publicações de uma instituição (como pode calcular das definições apresentadas, cada uma das publicações pode ter vários autores). Pretende-se que desenvolva em C, uma função Y que dada uma lista de publicações calcula uma lista de autores em que cada autor tem a ele associado a lista de titulos das publicações em que participou. A função deverá ter a seguinte assinatura:

```
ListAutorPub Y( ListPub lp )
{
...
}
```

Não se esqueça de definir o novo tipo de dados: *ListAutorPub*.