Programação Imperativa – EI e CC (1º ano)

Exame de recurso

15 de Julho de 2011 (09h30)

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

Questão 1 (cálculo numérico, 4v)

Codifique em C uma função que recebe um argumento do tipo inteiro, a, e imprime separadamente cada um dos seus dígitos (considerando, claro a base 10) por ordem inversa, isto é, da direita para a esquerda. Por exemplo, se o argumento fosse "123", a função em causa escreveria na saída o "3", depois o "2", e por fim o "1".

Apresente definições para duas versões desta função:

- `int decit( int a, int b )` — que realiza a decomposição de forma iterativa;
- `int decrec( int a, int b )` — que realiza a decomposição de forma recursiva.

Questão 2 (processamento de strings, 4v)

Q 2.1 (transformação)

Codifique em C uma função `void capitaliza( char f[] )` que recebe uma frase e a retorna com todas as letras em maiúsculas (não pode usar nenhuma função pré-definida.

Q 2.2 (pesquisa)

Codifique em C uma função `int mascara( char ch, char f[] )` que recebe uma frase e a retorna substituindo todas as ocorrências do caráter `ch` por `"*"`, devolvendo o número de conversões que fez .

Questão 3 (arrays, 4v)

Desenvolva um programa em C que irá ler as N (número inteiro) notas de cada um dos M (número inteiro) atletas numa dada prova (M e N devem ser perguntados ao utilizador). Por cada atleta (o seu nome é indicado pelo utilizador também antes de serem dadas as notas) o seu programa deve escrever num ficheiro externo ("`resultadost.txt`"o nome do atleta e a indicação "APROVADO"ou "EXCLUÍDO"conforme a média das notas seja (respetivamente) maior ou igual a 10, ou menor que 10.
Note que a média final é calculada excluindo das N notas lidas a *maior* e a *menor*.

Questão 4 (estruturas de dados, 5v)

Volte a considerar a estrutura de dados (semelhante à do exame anterior) descrita pelos tipos abaixo declarados, que servem para suportar a Agenda de 1 mês num telemóvel Nokia:

```
#define sMAX 10
#define DIAS 31

typedef struct sTarefa
  {
    tHora hora; //armazena horas e minutos
    char evento[sMAX]; //reunião, consulta, aula, exame, outros...
    char *descricao;
  } Tarefa;

typedef struct sLLTarefa
  {
```

```
    Tarefa t;
    struct sLLTarefa *seg;
  } *LLTarefa, NodoLLTarefa;

typedef struct sAgendaMensal
  {
    int mes;
    LLTarefa ag[DIAS];
  } AgendaMensal;
```

Com base nesta especificação responda às seguintes alíneas:

a) declare um tipo `tHora` apropriado a facilitar a operação aritmética com as horas e com os minutos;

b) declare um tipo `tHora` apropriado a economizar memória quando não quer operar algebricamente com os valores;

c) diga qual a diferença entre o tipo do campo `evento` e o tipo do campo *descrição* da estrutura `Tarefa` (use algum exemplo de programação que ilustre essa diferença);

d) diga como modificava/adaptava essa estrutura de dados para criar em memória uma Agenda Anual;

e) explique através de 1 desenho como armazenava 2 eventos no dia 12 do mês em causa, uma consulta de dentista às 09h00 e uma reunião com o Diretor às 11h30.

Questão 5 (mais estruturas de dados, 3v)

Considerar a estrutura de dados descrita pelos tipos abaixo declarados, que servem para suportar as receitas passadas por médicos a doentes e que foram recepcionadas numa determinada farmácia:

```
#define dMAX 8

typedef struct P
  {
    char *nome;
    char *código;
    char *morada;
  } Pessoa;

typedef struct L
  {
    char *medicamento; //nome
    int   embalags; //qt de embalagens
    char *posologia; //descrição das tomas diárias
    char  generico;  //sim ou nao
  } Linha;

typedef struct sLL
  {
    Linha lin;
    struct sLL *prox;
  } *LLLinha, NodoLL;

typedef struct R
  {
    char      data[dMAX];
    Pessoa    doente,
              medico;
    LLLinhas  linhas;
  } Receita;

typedef struct sLLReceita
  {
    Receita r;
    struct sLLReceita *seg;
```

```
  } *LLReceita, NodoLLReceita;

typedef struct F
  {
    char * nomefarmacia;
    LLReceita receitas;
 } Farmacia;
```

Com base nesta especificação responda às seguintes alíneas:

a) Codifique em C uma função `percGeneric( Farmacia f )` que recebe uma `Farmacia` e devolve como resultado um inteiro indicativo da percentagem de medicamentos genéricos receitados nessa farmácia;

b) Codifique em C uma função `LMedicos listMedicos( Farmacia f )` que recebe uma `Farmacia` e devolve uma lista ordenada alfabeticamente e sem repetições contendo os nomes dos médicos que prescreveram receitas nessa farmácia (defina o tipo *LMedicos*);

c) Codifique em C uma função `int totalDoentes( Farmacia f )` que recebe uma `Farmacia` e devolve um inteiro indicando o total de doentes diferentes abrangidos pelas receitas aviadas (note que um doente poderá ter várias receitas aviadas mas o mesmo só deverá entrar uma vez na contabilização deste total).