Departamento de Matemática da Universidade de Coimbra		
2007/2008	**Métodos de Programação II**	**Folha 0**

Cada um deste módulos deve ser testado num programa em C. Tenha atenção em tentar fazer os seus programas robustos e livres de erros, validando o mais possível as entradas de dados quando necessárias.

1. Escreva um módulo em C para carregar uma linha inteira do *sdtin* para uma *string*, ignorando a mudança de linha.
2. Escreva um módulo em C para concatenar 2 dadas *strings*, devolvendo o resultado na primeira delas.
3. Elabore um programa que:
   (a) Recolha do *sdtin* um número inteiro;
   (b) Escreva no *sdtout* o número invertido.

   Nota: use a função *itoa*.
4. Use as funções anteriores para alterar o seu programa de modo a determinar se o número dado é um palíndromo (vulgo, capicua).
5. Escreva uma função análoga à de *itoa* mas para converter um dado número para guardar os caracteres da sua representação binária, denominada *itob*.
6. Modifique a versão *itoa* para receber 3 argumentos: o número a converter; a *string* para guardar a conversão e um argumento *min* para *tamanho mínimo para a string*, ou seja, se a conversão tiver menos que *min* caracteres, deve-se colocar espaços em branco até atingir o tamanho pretendido.
7. Construa uma função para recolher do *sdtin* um número (real ou inteiro), que limpe a string (a deixe sem espaços em branco entre os dígitos ou o número e o sinal). Esta função deve devolver um (1) se conseguiu construir a *string* que representa um número e zero (0), caso não tenha conseguido recolher um número.
8. Escreva um programa que verifique se:
   (a) um dado número n é perfeito (isto é, se a soma dos seus divisores própios é igual a n; exemplo: 6 é um número perfeito);
   (b) dois dados números n e m são amigos (isto é, a soma dos divisores próprios de um deles é igual ao outro; exemplo: 220 e 284 são números amigos).
9. Seja "v"um vector de n inteiros. Escreva um módulo em C para determinar:
   (a) a soma e o mínimo dos elementos contidos no vector;
   (b) a soma e o mínimo dos elementos pares contidos no vector;
10. Considere o seguinte programa:

```
#include <stdio.h>
int main()
{ double x, passo;

  x = 0; passo = 0.1;
  while (x != 1)
  { printf("x = %f < 1.0\n", x);
    x += passo;
  }
  return 0;
}
```

    Quantas vezes prevê que seja realizado o ciclo "while"? Justifique.

11. Considere o seguinte programa:

```
#include <stdio.h>
int main()
{ float x, y, passo;
  int i;
  x = 0; passo = 0.1;
  for (i = 0; i < 100; i++)
    x += passo;
  y = passo*100;
  printf("x = %f, y = %f \n", x, y);
  return 0;
}
```

(a) Qual é o output deste programa?

(b) Repita o exercício substituindo as variáveis do tipo "float"para "double".

(c) Repita o exercício substituindo "100"por "100000". Comente os resultados obtidos.

12. Indique qual a mensagem que será escrita no ecrã depois de executar o seguinte conjunto de instruções:

```
int i, j, k;
i = 3; j = i++; k = ++i;
printf("i = %d, j = %d, k = %d\n", i, j, k);
```

13. O seguinte programa contém alguns erros. Corrija-o por forma a estar de acordo com as intruções de escritas.

```
#include <stdio.h>
int main()
{ int opcao, i, v[6];

  opcao = 1;
  switch (opcao)
  { case '1': printf("Escolheu a opcao 1");
    case  2:  printf("Escolheu a opcao 2\n");
    default:  printf("Opcao errada\n");
  }
  v[0] = 2; v[1] = 1; v[2] = 0; v[3] = 9; v[4] = 0; v[5] = 7;
  for (i = 0; (v[i] % 2) && (i < 6); );
    i++;

  if (i = 10)
    printf("v nao tem numeros impares\n");
  else
    printf("Primeiro numero impar em v: v[%d] = %d\n", i, v[i]);
  return 0;
}
```

14. Em criptografia, o código de César é uma das mais simples e conhecidas técnicas de encriptação. É um tipo de cifra de substituição em que cada letra do texto é substituída por outra, que se apresenta deslocada no alfabeto um número fixo de unidades (a chave). Note-se que o deslocamento é circular. Por exemplo, com a chave 3, "PROGRAMAR"seria substituído por "SURJUDPDU". Escreva um programa que permita ler uma sequência de letras maiúsculas e um inteiro (a chave) e devolva a sequência de letras maiúsculas codificada.

Departamento de Matemática da Universidade de Coimbra		
2007/2008	**Métodos de Programação II**	**Folha 1**

Cada um deste módulos deve ser testado num programa em C. Tenha atenção em tentar fazer os seus programas robustos e livres de erros, validando o mais possível as entradas de dados quando necessárias.

1. Implemente o seguinte módulo substituindo o ciclo **for** por um ciclo **while** mantendo o "terminador"(*break*) e o "repetidor"*continue:*

```
for (i = 0; 1; i++)
{ printf("Escreva um numero positivo par: ");
  scanf("%d", &n);
  if (n <= 0)
    continue;
  if (! (n % 2))
    break;
}
printf("Errou %d vezes...\n", i);
```

   Indique, caso existam, as principais diferenças resultantes da aplicação das instruções *break* e *continue* nos ciclos **for** e **while**.

2. Indique o resultado das seguintes instruções sem utilizar o computador:

```
char c;
c = 'A';
printf("%c %c %c %d %p\n", 'B', 'c', c, c, &c);
c++;
printf("%c %c %c %d %p\n", 'B', 'c', c, c, &c);
```

   Confirme o resultado executando o respectivo programa em C.

3. Considere a declaração "`int a[10], b[10][10]`".

   (a) A variável "a"é de que tipo? O que representa "a[3]"?

   (b) A variável "b"é de que tipo? O que representa "b[3]"? E "b[3][6]"?

4. Escreva um módulo em C que permita calcular:

   (a) a norma máxima de um vector v, isto é, $||v||_\infty = \max\{|v_i| : 1 \leq i \leq n\}$.

   (b) a norma infinita de uma matriz A, isto é, $||A||_\infty = \max\{\sum_{j=1}^{n} a_{i,j} : 1 \leq i \leq n\}$.

5. Indique o resultado das seguintes instruções sem utilizar o computador:

```
int x, *ptrX, y, *ptrY;
ptrX = &x; ptrY = ptrX; x = 5; y = 17;
printf("x = %d, y = %d, ptrX -> %d,
        ptrY -> %d\n", x, y, *ptrX, *ptrY);
x++; *ptrX = 9; (*ptrY)--; y = 21;
printf("x = %d, y = %d, ptrX -> %d,
        ptrY -> %d\n", x, y, *ptrX, *ptrY);
```

   Confirme o resultado executando o respectivo programa em C.

6. Indique o resultado das seguintes instruções sem utilizar o computador:

```
int v[4] = {1, 6, 3, 8}, x, *ptrInt;
x = 17; ptrInt = &x;
printf("v = {%d, %d, %d, %d}; x = %d, endereco de v = %p,
        endereco de x = %p, p = %p, p -> %d \n",
        v[0], v[1], v[2], v[3], x, v, &x, ptrInt, *ptrInt);
v[0] = x++; v[1] = ++x; *ptrInt = 23;
printf("v = {%d, %d, %d, %d}; x = %d, endereco de v = %p,
        endereco de x = %p, p = %p, p -> %d \n",
        v[0], v[1], v[2], v[3], x, v, &x, ptrInt, *ptrInt);
v[2] = ++(*ptrInt); ptrInt = v; v[3] = *(ptrInt++);
*ptrInt = 23;
printf("v = {%d, %d, %d, %d}; x = %d, endereco de v = %p,
        endereco de x = %p, p = %p, p -> %d \n",
        v[0], v[1], v[2], v[3], x, v, &x, ptrInt, *ptrInt);
v[0] = (x == 17); v[0] = (x = 17); ptrInt += 2;
printf("v = {%d, %d, %d, %d}; x = %d, endereco de v = %p,
        endereco de x = %p, p = %p, p -> %d \n",
        v[0], v[1], v[2], v[3], x, v, &x, ptrInt, *ptrInt);
```

Confirme o resultado executando o respectivo programa em C.

7. No exercício anterior, qual seria o efeito da seguinte instrução?

```
v = &x;
```

8. Considere as seguintes funções em C:

```
void troca1(int a, int b)
{ int c;
  c = a; a = b; b = c;
} /* troca1 */

void troca2(int *a, int *b)
{ int *c;
  c = a; a = b; b = c;
} /* troca2 */

void troca3(int *a, int *b)
{ int c;
  c = *a; *a = *b; *b = c;
} /* troca3 */
```

Faça a simulação das funções sem utilizar o computador. Confirme o resultado executando o respectivo programa em C.

Departamento de Matemática da Universidade de Coimbra		
2007/2008	**Métodos de Programação II**	**Folha 2**

Cada um deste módulos deve ser testado num programa em C. Tenha atenção em tentar fazer os seus programas robustos e livres de erros, validando o mais possível as entradas de dados quando necessárias.

1. São dados os seguintes tipos:

```
struct Planeta {
  char Nome[8];
  short Visivel; /* 1- sim; 0 - nao */
  float RaioOrbital;
};

struct Planeta Sistema[8];
```

   que descrevem o nosso Sistema Solar. Escreva um sub-programa que leia o seguinte ficheiro,

Mercúrio	s	0.39	Júpiter	s	5.2
Vénus	s	0.72	Saturno	s	9.5
Terra	s	1.0	Úrano	n	19.2
Marte	s	1.5	Neptuno	n	30.1

   e que escreva o nome e o raio orbital dos planetas visíveis, da Terra, a olho nu.

2. Considere as seguintes declarações:

```
struct data {
  int dia, mes, ano;
} dia_casamento;

struct identidade {
  char nome[35];
  struct data data_nas;
  short sexo, /* 1=feminino, 2=masculino */
        estado_civil; /* 1=solt, 2= cas, 3=viu, 4=div */
} candidato1, candidato2;
```

   (a) Faça um sub-programa para saber se um candidato para se casar tem 16 anos no dia do previsto casamento.

   (b) Faça um programa para saber se `candidato1` e `candidato2` podem casar, isto é, se são de sexos opostos, se têm ambos mais de 16 anos no dia do previsto casamento e se não são casados.

   (c) O que necessitaria de alterar nas declarações acima de modo a conseguir determinar se os candidadtos são irmãos? Escreva o novo tipo de dados mais adequado a acomodar esta alteração.

3. Considere os tipos do exercício anterior e, ainda, as declarações seguintes:

```
#define max 100
struct identidade lista[max];
```

   (a) Escreva um programa que leia um ficheiro no seguinte formato:

```
 1 / 2 / 1970 1 2 Isabel Ara\'ujo
31 / 1 / 1970 2 1 Paulo Leal
...
```

   e escreva, no monitor, a lista lida mas **ordenada**:

   i. por ordem alfabética;

   ii. por idades.

   (b) Admita que a lista já se encontra ordenada alfabeticamente e escreva um sub-programa para a consultar, isto é, dado um nome, saber se esse nome consta da lista. Em caso afirmativo, indique a respectiva data de nascimento.

4. (a) Escreva sub-programas para operar com fracções inteiras (ler, escrever, simplificar, somar, multiplicar, dividir, subtrair, calcular potências de fracções), declarando as fracções como estruturas do tipo `struct` cujos campos são do tipo `int`:

```
struct fraccao {
  int num, den;
};
```

   (b) Elabore um programa para escrever os primeiros n termos de uma sucessão associada à *série harmónica*:

$$H = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

   sob a forma de fracção. Por exemplo, para $n = 4$, a saída do programa deverá ser:

```
1  3/2  11/6  25/12
```

   Deverá estruturar devidamente o seu programa, incluindo sub-programas (da alínea anterior) para:

   — escrever uma fracção;

   — somar duas fracções;

   — simplificar uma fracção.

5. Considere as seguintes declaração :

```
struct Modalidade {
  char nome_mod[30], campeao[30], nacionalidade[30];
  float pontuacao;
} guiness[1000];

char atleta[30], pais[30], esta_modalidade[30];
float pontos;
```

   onde o vector `guiness` está ordenado por ordem alfabética do campo `nome_mod`. Em face dos `pontos` obtidos por um `atleta` do `pais` em `esta_modalidade`, pretende-se actualizar (se necessário) o `guiness` existente.

   Elabore um sub-programa para o efeito, tendo em conta que se `esta_modalidade` ainda não faz parte do `guiness`, deve ser introduzida na ordem respectiva.

6. Sendo dados os seguintes tipos:

```
#define Codigo 1000
struct Peca {
  char Nome[20];
  short Disponivel;
  float PrecoUnitario;
} Armazem[Codigo];
```

que descrevem um armazém de peças. Escreva um sub-programa de facturação que, recebendo os códigos e quantidades das peças encomendadas escreva, em papel, uma factura cujas *linhas de detalhe*, organizadas por coluna, contêm:

(a) para cada peça *encomendada e disponível*, o código, nome, preço unitário, quantidade encomendada e preço total da quantidade encomendada;

(b) para cada peça *encomendada e não disponível*, o código, nome, preço unitário, quantidade encomendada e a mensagem "não disponível" na coluna correspondente ao preço total;

e cuja *linha de total*, posicionada após todas as *linhas de detalhe*, contém, na coluna correspondente ao preço total, a soma dos preços totais das quantidades encomendadas de todas as peças disponíveis. Declare os tipos e variáveis de que necessitar para elaborar o sub-programa pedido. Indique como invocaria o dito sub-programa.

7. Escreva um programa que resolva uma equação do 2º grau, calcule as suas soluções e as escreva. Para esse efeito, considere os seguintes tipos:

```
struct equacao {
  float a, b, c;
};

struct complexo {
  float re, im;
};

union raiz {
  float r;
  struct complexo c;
};

short tipo_raizes; /* 0 - reais; 1- complexas */
```

Departamento de Matemática da Universidade de Coimbra		
2007/2008	**Métodos de Programação II**	**Folha 3**

Cada um deste módulos deve ser testado num programa em C. Tenha atenção em tentar fazer os seus programas robustos e livres de erros, validando o mais possível as entradas de dados quando necessárias.

1. Indique e corrija os erros da implementação do seguinte código:

```
typedef struct conjunto {
  char elementos[100];
  int dim
} CONJUNTO X, Y;

/* Verifica se os conjuntos X e Y são iguais
 * Devolve 1 em caso afirmativo e 0 em caso contrário */
int iguais(conjunto X, Y)
{ if (X.dim = Y.dim)
  { if (X.elementos = Y.elementos) //X e Y tem os mesmos elementos
      return 1
    else
      return 0
  }
  else
    return 0;
}
```

2. Preveja e explique a saída da implementação do seguinte código:

```
struct lixo {
  int n, *ptr;
} x, y;

main()

{ x.n = 15; x.ptr = &(x.n);
  y.n =  3; y.ptr = x.ptr;
  printf("%d %d %d %d\n", x.n, *(x.ptr), y.n, *(y.ptr));
  (x.n) += 2; *(x.ptr) *= 3;
  (y.n)++; *(y.ptr)++; y.ptr = &(y.n);
  printf("%d %d %d %d\n", x.n, *(x.ptr), y.n, *(y.ptr));
  return 0;
}
```

3. Considere as seguintes declarações:

```
typedef struct pessoa {
  char nome[10];
  struct pessoa *prox;
} PESSOA, *Seta;
```

   (a) Que informação pretendem representar computacionalmente estes tipos?

   (b) Leia os seguintes sub-programas e explique qual o efeito de cada um deles, supondo que `ponta` é uma estrutura, já construída, do tipo `Seta`.

```
void X1(Seta ponta)
{ Seta p;

  for (p = ponta; p; p = p->prox)
    printf("%s\n", p->nome);
}

int X2(Seta ponta)
{ Seta p;
  int c = 0;

  for (p = ponta; p; p = p->prox)
    c++;
  return c;
}

void X3(char n[10], Seta ponta)
{ Seta ant, este;

  ant = NULL;
  for (este = ponta; strcmp(este->nome, n); este = este->prox)
    ant = este;
  if (ant == NULL)
    ponta = este->prox;
  else
    ant->prox = este->prox;
  free(este);
}
```

4. Considerando as declaração de tipos do exercício 3 da folha 3, escreva sub-programas para:
   (a) Criar uma lista vazia.
   (b) Verificar se uma lista é vazia.
   (c) Devolver o
      i. primeiro elemento de uma lista.
      ii. último elemento de uma lista.
      iii. n-ésimo elemento de uma lista, $n \in \mathbb{N}$.
   (d) Inserir um elemento numa lista de modo a que fique
      i. no ínicio da lista.
      ii. no fim da lista.
      iii. na n-ésima posição da lista, $n \in \mathbb{N}$.
   (e) Remover o
      i. primeiro elemento de uma lista.
      ii. último elemento de uma lista.
      iii. n-ésimo elemento de uma lista, $n \in \mathbb{N}$.
   (f) Duplicar uma lista.
   (g) Destruir uma lista.

5. Considere as seguintes declarações:

```
typedef struct elemento {
  int numero;
  struct elemento *seguinte;
} ELEMENTO, *Seta;

Seta lista, p, nova;
```

   Dada a lista ligada, `lista`, já construída em memória, escreva sub-programas para:

   (a) Retirar de `lista` o elemento seguinte ao apontado por `p`.

   (b) Retirar de `lista` o elemento apontado por `p`.

   (c) Inserir o elemento apontado por `novo`, entre o elemento apontado por `p` e o seguinte.

   (d) Inserir o elemento apontado por `novo`, entre o elemento apontado por `p` e o anterior.

   (e) Procurar o inteiro `n` em `lista` e devolver um ponteiro para o elemento anterior ao primeiro elemento que contém `n`. Note que, caso não o encontre, deve devolver `NULL` mas este valor também deverá ser o valor devolvido caso `n` esteja na primeira posição da lista.

6. Continue a considerar as declarações do exercício anterior, admitindo agora que a lista se encontra **ordenada** por ordem não decrescente do campo `numero`. Elabore sub-programas para, *mantendo essa ordem*:

   (a) Inserir em `lista` um inteiro `n`.

   (b) Procurar o inteiro `n` em `lista` e devolver `NULL` caso não o encontre.

   (c) Remover *todas* as ocorrências do inteiro `n` de `lista`.

7. Dados os tipos do exercício 5 (folha 3), qual será o resultado da execução do procedimento `Beta`:

```
void Beta(Seta *x)

{ Seta a, b, c;

  a = NULL; b = *x;
  while (b != NULL)
  { c = b->seguinte;
    b->seguinte = a;
    a = b;
    b = c;
  }
  *x = a;
}
```

8. Pelo Teorema Fundamental da Aritmética qualquer número inteiro positivo pode ser decomposto como produto de potências não negativas de números primos, $n = p_1^{n_1} \times p_2^{n_2} \times \ldots \times p_k^{n_k}$. Por exemplo,

$$2268 = 2^2 \times 3^4 \times 7^1.$$

   Considere que a representação canónica de um inteiro positivo é definida pelos tipos de dados:

```
typedef struct canonico {
  int primo, expoente;
  struct canonico *prox;
} CANONICO;
```

   e elabore sub-programas para:

(a) Obter a representação canónica de um dado inteiro positivo.

(b) Verificar se um dado inteiro na forma canónica é, ou não, um número primo.

(c) Calcular o produto de dois números inteiros na forma canónica.

(d) Calcular o máximo divisor comum de dois inteiros na forma canónica.

(e) Calcular o mínimo múltiplo comum de dois inteiros na forma canónica.

9. Um polinómio em x de coeficientes reais pode ser representado por uma sequência de pares ordenados (coeficientes, grau), de preferência ordenados por ordem decrescente das potências de x. Por exemplo, o polinómio

$$x^8 + 5x^6 - 7x^5 + 6x + 1$$

pode ser representado por

$$(1, 8)(5, 6)(-7, 5)(6, 1)(1, 0)$$

(a) Utilizando ponteiros defina um tipo que represente a sequência anterior como uma lista ligada.

(b) Escreva um sub-programa `lerpolinomio`, que leia uma sequência de pares de números (coeficientes, grau), dado por ordem decrescente das potências de x.

Por exemplo, o polinómio anterior seria dado por:

$$1\ 8\ \ 5\ 6\ \ -7\ 5\ \ 6\ 1\ \ 1\ 0$$

O sub-programa deverá construir a lista ligada representativa do polinómio.

(c) Escreva um sub-programa `soma` para somar dois polinómios e um sub-programa `produto` para multiplicar um número real por um polinómio.

(d) Usando as alíneas anteriores, elabore um programa que permita manipular (estes) polinómios, desenvolvendo mais algumas operações entre polinómios.

10. Escreva sub-programas para:

(a) Retirar da lista e devolver um ponteiro para o "maior" elemento de uma dada lista.

(b) Ordenar uma dada lista por ordem não decrescente,

i. usando selecção linear;

ii. usando inserção linear.

(c) Fazer a fusão ordenada de duas listas previamente ordenadas:

i. devolvendo uma nova lista;

ii. devolvendo uma das listas dadas depois de actualizada.

11. Repita o exercício 4 da folha 3, usando listas ligadas simples circulares. Indique as vantagens/desvantagens desta nova representação.

12. Repita o exercício anterior, considerando agora listas duplamente ligadas.