

Folha de Exercícios II

Programação e Algoritmos Complementos de Informática Programação II

Universidade da Beira Interior
Departamento de Informática

3 Apontadores e gestão dinâmica da memória

Exercício 17:

Dadas as seguintes declarações e inicializações

```
struct a
{
    int b;
    float c;
} d, e, *p_a;
int *p_i1;
float *p_fl1;
d.b=2;
d.c=3.14;
```

indique se os seguintes blocos de código contêm erros ou não, justificando.

1. `p_a=malloc(sizeof(a));`
2. `d.b=p_a.b;`
3. `p_i1=malloc(5*sizeof(float));`
4. `p_i1=calloc(5,5);`
5. `p_a=&d;`

6. `realloc(p_a, 2 * sizeof(struct a));`
 7. `*p_f11 = d.c;`
 8. `free(p_f11);`
-

Exercício 18:

Escreva um programa que crie um array de floats dinâmico, com capacidade que deve ser indicada pelo utilizador. Após a introdução dos números no array, o seu tamanho deve ser alterado para metade. O programa deve mostrar o conteúdo do array após a introdução dos números e também após a alteração do tamanho (neste caso, percorrer apenas até metade do tamanho original!). O programa deve ser feito

1. usando `malloc`;
 2. usando `calloc`.
-

Exercício 19:

Escreva um programa que crie um array de caracteres dinâmico, com capacidade que deve ser indicada pelo utilizador. Todas as posições do array devem ser preenchidas sequencialmente com as letras maiúsculas do código ASCII (se a capacidade exceder 26 letras, na posição 26 deve estar a letra A, na posição 27 deve estar a letra B e assim sucessivamente). Após preencher e mostrar o array, deve ser indicada pelo utilizador uma nova capacidade (maior, menor ou igual à anterior!). Se necessário o programa deve preencher o conteúdo do array com nova capacidade e depois apresentá-lo no écran.

Exercício 20:

Escreva um programa que lê dois valores para duas variáveis de tipo `double` e troca os seus conteúdos. Deverá utilizar a seguinte função:

```
void trocar_double(double *d1, double *d2);
```

Exercício 21:

Baseando-se no tipo seguinte:

```
typedef struct { int x; int y;} POINT;
```

Escreva um programa que lê valores para duas variáveis de tipo POINT e troca os conteúdos. Deverá utilizar as três funções seguintes:

```
void trocar_pontos(POINT *x,POINT *y);
```

```
void mostrar_ponto(POINT x);
```

```
void ler_ponto(POINT *x);
```

Exercício 22:

O objectivo deste exercício é escrever uma função que simplesmente some dois inteiros.

1. Implemente primeiro os dois casos simples:

```
int soma1 ( int a, int b);  
int soma2 (int *a, int *b);
```

2. Escreva a seguinte função soma3 (errada !), e explique o seu comportamento indesejado utilizando, se necessário, exemplos de execução.

```
int * soma3 (int a, int b)  
{  
    int temp;  
    temp = a+b;  
    return &temp;  
}
```

3. Utilizando malloc, escreva

```
int * soma4(int a, int b);  
int * soma5(int *a, int *b);
```

Exercício 23:

Escreva um programa que leia do teclado um vector de valores inteiros, de dimensão desconhecida (termine com <ctrl+D> em sistemas Unix ou <ctrl+Z> em DOS/WIN), inverta o vector e apresente-o no écran depois de invertido.

4 Ficheiros e passagem de argumentos a main

Exercício 24:

Escreva um programa que receba uma série de números da linha de comando e devolva a sua soma. Os números são `floats`.

Exercício 25:

Escreva um programa que devolva a palavra mais extensa (e a indicação do seu tamanho) dentre os argumentos passados via linha de comando.

Exercício 26:

Escreva um programa que conte o número de linhas de um ficheiro.

1. usando `fgetc`;
2. usando `fgets`;
3. comente os resultados obtidos com os dois programas anteriores;

Altere o programa de forma a que o nome do ficheiro seja passado na linha de comando.

Exercício 27:

Escreva um programa que leia uma frase do teclado e a escreva num ficheiro cujo nome é passado via linha de comando.

Exercício 28:

Escreva um programa que procure num ficheiro uma palavra lida do teclado. Deve dizer o número de vezes que a palavra existe no ficheiro.

Exercício 29: (*Manipulação de Ficheiros*)

Escreva:

1. Uma função que permita copiar um ficheiro para um outro.
 2. Uma função que anexe um ficheiro ao final de outro.
-

Exercício 30:

Considere que o écran tem 25 linhas.

Escreva um programa que leia um ficheiro de texto (nome passado via linha de comandos) usando `fgets` e o escreva no terminal de forma a que o texto do ficheiro seja visto um écran de cada vez.

O programa deve esperar que o utilizador prima `<enter>` para mostrar o écran seguinte. Esta funcionalidade deve ser implementada sob a forma de função.

Exercício 31: (*Word Count*)

Escreva um programa C que simule o funcionamento do comando `wc` do sistema Unix com a restrição que as opções aceites são `-l` `-w` `-c` para respectivamente “linhas”, “palavras” e “caracteres”. Assume-se que a entrada é a entrada standard se nenhum ficheiro for indicado na linha de comando.

Exercício 32: (*Diz-me como me chamas, dir-te-ei o que faço...*)

Escreva um programa que:

- Se se chamar `ordem` devolve a maior palavra (usando a ordem alfabética “rato” é maior que “elefante”) dos argumentos fornecidos. Tome esta opção como sendo a opção por omissão;
- Se se chamar `conta` devolve o número de argumentos com que foi invocado;
- Se se chamar `lista` manda para a saída standard os seus argumentos.

Exercício 33: (**Processamento de texto*)

Escreva um programa que permita construir uma lista de todas as palavras contidas num ficheiro, assim como o número de ocorrências de cada uma delas.

Exercício 34: (*Inteiros em ficheiro de texto*)

Escreva um programa que solicite um valor inteiro ao utilizador e apresente os múltiplos desse valor que estiverem presentes no ficheiro de texto `dados1.txt` (uma sua cópia pode ser encontrada na página Web da disciplina).

Exercício 35: (*Acesso a ficheiro binário*)

Considere o ficheiro binário `dados2.dat` (uma sua cópia pode ser encontrada na página Web da disciplina).

Os dados guardados neste ficheiro são estruturas do seguinte tipo:

```
struct {  
  int BI;  
  char nome[100];  
  float peso, altura;  
};
```

1. Utilizando funções codifique um programa que:
 - (a) Liste o conteúdo do ficheiro.
 - (b) Solicite ao utilizador uma letra e copie todos os registos cujo campo nome comece por essa letra para o ficheiro `$_inicial.dat` (onde `$` representa a letra escolhida pelo utilizador).
 - (c) Apresente o campo nome para o registo com o valor de peso mais elevado.
 - (d) Acrescente um novo registo (campos introduzidos pelo utilizador) ao ficheiro `dados2.dat`.
2. Altere o programa anterior de forma a que o programa permaneça em ciclo e, através dum menu, o utilizador possa escolher qual a tarefa a realizar.

Exercício 36: (**Agenda electrónica*)

1. Escreva um programa que use uma estrutura que permita guardar o nome, a morada e o número de telefone duma pessoa. Deve permitir, através dum menu, ler dados para uma estrutura desse tipo que sejam automaticamente guardados num ficheiro. O programa deve permitir mostrar o conteúdo do ficheiro da seguinte forma

```
*****  
Registo 1  
Nome :  
Morada :  
Telefone :  
*****  
Registo 2  
Nome :  
Morada :  
Telefone :  
...
```

Assim, o menu deve ter as seguintes opções: inserir dados, mostrar dados, terminar. O menu, a inserção de dados e a exibição dos dados devem ser implementados usando funções.

2. Altere o programa anterior de forma a que seja possível:
 - (a) Alterar dados dum registo, indicando o número do registo a alterar;
 - (b) Apagar um registo, indicando o número do registo a apagar (tem de acrescentar um campo à estrutura original que lhe indique se o registo é válido ou foi apagado);
 - (c) Ver apenas um registo do ficheiro, indicando o seu número. Implemente estas funcionalidades sob a forma de funções. Devem estar disponíveis através do menu do programa.