UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Ciência, Tecnologia e Ensino Superior

p-Fólio

U.C. 21090

Programação

07 de Fevereiro de 2011

-- INSTRUÇÕES --

- O tempo de duração da prova de p-fólio é de 90 minutos.
- O estudante deverá responder no enunciado e preencher o cabeçalho e todos os espaços reservados à sua identificação, com letra legível, pelo que o mesmo deverá ser entregue ao vigilante, **não sendo permitido ao estudante levar o enunciado.**
- Verifique no momento da entrega do enunciado se todas as páginas estão rubricadas pelo vigilante.
- Em hipótese alguma serão aceites folhas de ponto dobradas ou danificadas.
- Exclui-se, para efeitos de classificação, toda e qualquer resposta apresentada em folhas de rascunho.
- Os telemóveis deverão ser desligados durante toda a prova e os objectos pessoais deixados em local próprio da sala de exame.
- A prova é constituída por 8 páginas e termina com a palavra **FIM**. Verifique o seu exemplar e, caso encontre alguma anomalia, dirija-se ao professor vigilante nos primeiros 15 minutos da mesma, pois qualquer reclamação sobre defeito(s) de formatação e/ou de impressão que dificultem a leitura não será aceite depois deste período.
- Utilize unicamente tinta azul ou preta.
- O p-fólio é constituído por quatro Grupos, com a cotação de 3 valores cada.
- A resposta a cada grupo deve ser dada no espaço de resolução do grupo respectivo, e ocupando apenas o espaço destinado para o efeito. No grupo I apenas deve ser preenchida a tabela com a execução passo-a-passo, e os restantes grupos devem respeitar a tabela com as 40 linhas de código.
- Ao resolver os grupos III e IV, pode e deve utilizar as funções definidas nos grupos anteriores. Se não tiver espaço suficiente no grupo IV, pode utilizar o espaço vazio dos grupos II e III para implementar funções necessárias apenas no grupo IV.
- Os programas devem ser escritos em **linguagem C** podendo utilizar funções da biblioteca *standard*. Em anexo está uma lista com as funções da biblioteca *standard* mais utilizadas, não sendo necessário utilizar a primitiva *#include*.

Nome: .....

Nº de Estudante: ..... **B. I. nº** .....

Turma: ..... **Assinatura do Vigilante:** .....

Grupo I (3 valores)

Considere o programa e execução de exemplo seguintes, e efectue a execução passo-a-passo com a entrada de dados utilizada na execução de exemplo. No caso de serem necessários mais de 25 passos, deve parar no passo 25.

Programa:

```
1  #include <stdio.h>
2
3  /* programa que calcula N^N  */
4  int main()
5  {
6      int i, N, resultado=1;
7      printf("Indique N: ");
8      scanf("%d",&N);
9      for(i=0;i<N;i++)
10         resultado*=N;
11     printf("Resultado %d^%d: %d", N, N, resultado);
12 }
```

Execução de exemplo:

```
C:\>normal1011g1
Indique N: 4
Resultado 4^4: 256
```

Grupo II (3 valores)

Implemente uma função *NumeroPalavras* que recebe uma string e retorna o número de palavras que essa string tem, sendo uma palavra qualquer conjunto de um ou mais caracteres seguidos que não sejam um espaço, tab ou mudança de linha. Em baixo está um programa que utiliza a função e uma execução de exemplo.

Programa:

```
/* programa que calcula o número de palavras de uma string */
int main()
{
    char str[MAX_STR];
    printf("Escreva uma string:\n");
    gets(str);
    printf("Numero de palavras: %d", NumeroPalavras(str));
}
```

Execução de Exemplo:

```
C:\>normal1011g2
Escreva uma string:
abcd    dddd   cccc 1
Numero de palavras: 4
```

Nome: .....

Nº de Estudante: ..... **B. I. nº** .....

Turma: ..... **Assinatura do Vigilante:** .....

Grupo III (3 valores)

Implemente um procedimento *RemoveTags* que recebe uma string com tags de XML, e altera a string, removendo as tags de XML. Uma tag é um conjunto de caracteres que começa com o caracter '<' e acaba com o caracter '>'. Em baixo está um programa que utiliza a função e uma execução de exemplo.

Programa:

```
#define MAX_STR 255
/* programa que remove tags de uma string */
int main()
{
    char str[MAX_STR];
    printf("Escreva uma string com tags de XML:\n");
    gets(str);
    RemoveTags(str);
    printf("String sem tags: %s",str);
}
```

Execução de Exemplo:

```
C:\>normal1011g3
Escreva uma string com tags de XML:
<html><body><p>Paragrafo numa pagina de html</p></body></html>
String sem tags: Paragrafo numa pagina de html
```

Grupo IV (3 valores)

Faça um programa que receba um ficheiro xml como argumento, e indique quantas palavras fora das tags contém, considerando a definição de palavra e tag do grupo II e III respectivamente. Assuma que uma tag apenas pode estar numa só linha do ficheiro, e que os argumentos são passados correctamente para o programa. O programa deve assumir definida uma macro MAXSTR, com o valor máximo para uma string. O programa deve também imprimir uma linha por cada 10 linhas lidas do ficheiro, colocando um asterisco '*' por cada 10 palavras. Em baixo está uma execução de exemplo do programa pretendido.

Execução de Exemplo:

```
C:\>normal1011g4 teste.xml

***
*****
****
*******
****
****
Numero linhas: 63
Numero palavras: 301
```

Nome: .....

Nº de Estudante: ..... **B. I. nº** .....

Turma: ..... **Assinatura do Vigilante:** .....

Resolução Grupo I:

Passo	Linha	Instrução	Resultado				
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							

- Instrução: colocar o nome da função, e no ciclo “for” indicar a componente em execução
- Resultado: colocar input / output do programa, e resultado de expressões lógicas
- Utilize as colunas extra para colocar uma variável por coluna

Nome: .....

Nº de Estudante: ..... **B. I. nº** .....

Turma: ..... **Assinatura do Vigilante:** .....

Resolução Grupo II:

1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									
38									
39									
40									

Nome: .....

Nº de Estudante: ..... **B. I. nº** .....

Turma: ..... **Assinatura do Vigilante:** .....

Resolução Grupo III:

1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									
38									
39									
40									

Nome: .....

Nº de Estudante: ..... **B. I. nº** .....

Turma: ..... **Assinatura do Vigilante:** .....

Resolução Grupo IV:

1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									
38									
39									
40									

Nome: .....

Nº de Estudante: ..... **B. I. nº** .....

Turma: ..... **Assinatura do Vigilante:** .....

Anexo

Funções standard mais utilizadas

Exemplos de chamadas:

- `printf("texto %d %g %s %c", varInt, varDouble, varStr, varChar);`
  Imprime no ecran uma string formatada, em que é substituído o **%d** pela variável inteira seguinte na lista, o **%g** pela variável real na lista, o **%s** pela variável string na lista, o **%c** pela variável caracter na lista.
- `scanf("%d", &varInt); gets(str);`
  scanf é a função inversa do **printf**, lê um inteiro e coloca o seu resultado em **varInt**, cujo endereço é fornecido. A função **gets** lê uma string para **str**.

Protótipos:

- `int atoi(char *str); float atof(char *str);`
  Converte uma string num número inteiro/real respectivamente
- `int strlen(char *str);`
  Retorna o número de caracteres da string **str**
- `strcpy(char *dest, char *str); [strcat]`
  Copia **str** para **dest**, ou junta **str** no final de **dest**, respectivamente
- `char *strstr(char *str, char *find); char *strchr(char *str, char find);`
  Retorna a primeira ocorrência de **find** em **str**, ou NULL se não existe. Na versão **strchr find** é um caracter.
- `char *strtok(char *string, char *sep); char *strtok(NULL, char *sep);`
  Retorna um apontador para uma token, delimitada por **sep**. A segunda chamada retorna a token seguinte, na mesma string, podendo-se continuar a chamar a função até que retorne NULL, o que significa que a string inicial não tem mais tokens para serem processadas.
- `sprintf(char *str, …); sscanf(char *str,…);`
  Estas funções têm o mesmo funcionamento de **printf/scanf**, mas os dados são colocados (ou lidos) em **str**.
- `int strcmp(char *str1, char *str2);`
  Retorna 0 se **str1** é igual a **str2**, retornando um valor negativo/positivo se uma string é maior/menor que a outra
- `int isalpha(int c); [isdigit,isalnum,islower,isupper,isprint]`
  Retorna true se **c** é uma letra / dígito numérico / letra ou dígito / minúscula / maiúscula / imprimivel.
- `void *malloc(size_t); free(void *pt);`
  malloc retorna um apontador para um bloco de memória de determinada dimensão, ou NULL se não há memória suficiente, e a função **free** liberta o espaço de memória apontado por **pt** e alocado por **malloc**
- `FILE *fopen(char *fich, char *mode); fclose(FILE *f);`
  fopen abre o ficheiro com nome **fich**, no modo **mode** ("rt" – leitura em modo texto, "wt" – escrita em modo texto), e **fclose** fecha um ficheiro aberto por **fopen**
- `fprintf(f,…); fscanf(f,…); fgets(char *str, int maxstr, FILE *f);`
  idênticos ao **printf/scanf** mas direccionados para o ficheiro, e **fgets** é uma versão do **gets** mas com limite máximo da string indicado em **maxstr**.
- `int feof(FILE *f);`
  feof retorna true se o ficheiro **f** está no fim, e false c.c.
- `fseek(f,posicao,SEEK_SET); fwrite/fread(registo,sizeof(estrutura),1,f);`
  funções de leitura binária (abrir em modo "rb" e "wb"). **fseek** posiciona o ficheiro numa dada posição, **fwrite/fread** escrevem/lêm um bloco do tipo estrutura para o endereço de memória registo.
- `int rand(); srand(int seed);`
  rand retorna um número pseudo-aleatório e **srand** inicializar a sequência pseudo-aleatória
- `time_t time(NULL); clock_t clock();`
  time retorna um número segundos que passaram desde uma determinada data, e **clock** o número de instantes (há **CLOCKS_PER_SEC** instantes por segundo)
- `double sin(double x); [cos,log,log10,sqrt] double pow(double x,double y);`
  Funções matemáticas mais usuais, com argumentos e valores retornados a double

FIM