UNIVERSIDADE ABERTA
www.univ-ab.pt

p-Fólio

Nome:.....

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

Curso: .....

Turma: .....

Unidade Curricular:.....

Código: ..... **Data**: ____/__/__

Assinatura do Vigilante: .....

Classificação

( )

Assinatura do Docente:

.....

LEIA ATENTAMENTE as instruções para a resolução do p-fólio:

1. O tempo de resolução do p-fólio é de uma hora e trinta minutos.

2. Não é permitida a utilização de calculadora durante a execução do p-fólio.

3. O p-fólio é constituído por quatro Grupos e termina com a palavra **FIM**.

4. A cotação de cada grupo é de 3 valores.

5. 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.

6. A 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.

7. 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*.

8. Se o seu exemplar não estiver completo ou nele se verificar qualquer outra anomalia, por favor dirija-se ao professor vigilante.

Enunciado

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 o próximo ano bissexto */
 4 int main()
 5 {
 6     int i, ano;
 7     printf("Indique Ano: ");
 8     scanf("%d",&ano);
 9     for(i=ano+1; ; i++)
10         if(i%4==0 && i%100!=0 || i%400==0)
11         {
12             printf("Proximo ano bissexto: %d",i);
13             break;
14         }
15 }
```

Execução de exemplo:

```
C:\>recurso1011g1
Indique Ano: 2010
Proximo ano bissexto: 2012
```

Grupo II (3 valores)

Implemente uma função *Concatenar* que recebe duas strings e retorna a concatenação de ambas as strings, alocando memória suficiente para a string retornada, e libertando a memória alocada na primeira string. No entanto, se a primeira string for vazia (igual a NULL), a função deve retornar apenas uma cópia da primeira string. Em baixo está um programa que utiliza a função e uma execução de exemplo.

Programa:

```
/* Programa que junta várias strings introduzidas e as mostra no ecran */
int main()
{
    char str[MAX_STR], *pt=NULL;
    printf("Introduza strings (acabar com a string vazia): \n");
    do
    {
        gets(str);
        pt=Concatenar(pt,str);
    } while(pt!=NULL && strlen(str)!=0);
    if(pt!=NULL)
    {
        printf("\nTodas as strings introduzidas:\n%s", pt);
        free(pt);
    }
}
```

Execução de Exemplo:

```
C:\>recurso1011g2
Introduza strings (acabar com a string vazia):
ABC D
FFF SF
12345

Todas as strings introduzidas:
ABC DFFF SF12345
```

Grupo III (3 valores)

Implemente um procedimento *strnorm* que recebe uma string e a normaliza relativamente aos espaços da seguinte maneira:

- Substituir caracteres não imprimíveis e os caracteres '\n', '\r' e '\t' por espaços;
- Substituir dois ou mais espaços seguidos por um só espaço.

Em baixo está um programa que utiliza a função e uma execução de exemplo.

Programa:

```
/* programa que normaliza as strings introduzidas */
int main()
{
    char str[255];
    printf("Introduza uma string: ");
    gets(str);
    strnorm(str);
    printf("String normalizada: %s", str);
}
```

Execução de Exemplo:

```
C:\>recurso1011g3
Introduza uma string: ABC     123 !"#
String normalizada: ABC 123 !"#
```

Grupo IV (3 valores)

Faça um programa que receba um ficheiro como argumento, e o leia trocando todos os caracteres não imprimíveis e espaços seguidos por um só espaço, e mostre no ecrã todas as palavras em linhas com não mais de 60 caracteres, com um espaço entre palavras. Assuma que os argumentos são passados correctamente para o programa, e que uma linha do ficheiro tem menos de MAXSTR caracteres. No final deve retornar o número de linhas do ficheiro, mostrado desta forma. Em baixo está uma execução de exemplo do programa pretendido.

Execução de Exemplo:

```
C:\>recurso1011g4 recurso1011g1.c
#include <stdio.h> /* programa que calcula o próximo ano
bissexto */ int main() { int i, ano; printf("Indique Ano:
"); scanf("%d",&ano); for(i=ano+1; ; i++) if(i%4==0 &&
i%100!=0 || i%400==0) { printf("Proximo ano bissexto:
%d",i); break; } }
Numero de linhas: 5
```

cotação

Resolução Grupo I:

Passo	Linha	Instrução	Resultado	i	ano		
1	6	cnt		?	?		
2	7	printf	Indique Ano:	?	?		
3	8	scanf	2010	?	2010		
4	9	for(;;)		2011	2010		
5	10	if	i%4==0 - F	2011	2010		
6	9	for(;;)		2012	2010		
7	10	if	i%4==0 && i%100!=0 V	2012	2010		
8	12	printf	Proximo ano bissexto: 2012	2012	2010		
9	13	break					
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

Resolução Grupo II:

cotação

```
1  char * Concatenar ( char * str1, char * str2)
2  {
3      char *pt;
4      if (str1==NULL)
5      {
6          pt = (char*) malloc(sizeof(char) *(strlen(str2)+1);
7          if( pt!=NULL)
8              strcpy (pt, str2);
9      } else {
10         pt = (char*) malloc(sizeof(char) *
11             (strlen(str1) + strlen(str2) + 1));
12         if ( pt!=NULL)
13         {
14             strcpy( pt, str1);
15             strcat (pt , str2);
16         }
17         free(str1);
18     }
19     return pt;
20 }
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

cotação

Resolução Grupo III:

```
1  void strnorm(char str[])
2  {
3      int i, j;
4      for(i=0; str[i]!=0; i++)
5          if(Espaco(str[i]))
6              str[i]=' ';
7      for(i=0, j=0; str[i]!=0; i++, j++)
8      {
9          str[j]=str[i];
10         while(str[i]==' ' && str[i+1]==' ')
11             i++;
12     }
13     str[j]=0;
14 }
15 int Espaco(char c)
16 {
17     return strchr(" \r\n\t", c)!=NULL || !isprint(c);
18 }
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

Resolução Grupo IV:

cotação

```
1  void MostrarTexto(char str[])
2  {
3      int linhas=0, i;
4      char *pt=str;
5      while(strlen(pt)>60)
6      {
7          linhas++;
8          for(i=59; i>0 && pt[i]!=' '; i--);
9          pt[i]=0;
10         printf("\n%s", pt);
11         pt+=i+1;
12     }
13     printf("\n%s\nNumero de linhas: %d", pt, linhas);
14 }
15 int main(int argc, char ** argv)
16 {
17     char str[MAXSTR], *pt=NULL;
18     FILE *f;
19     f=fopen(argv[1], "rt");
20     if(f!=NULL)
21     {
22         while(fgets(str, MAXSTR, f)!=NULL)
23         {
24             strnorm(str);
25             pt=Concatenar(pt, str);
26         }
27         MostrarTexto(pt);
28         if(pt!=NULL)
29             free(pt);
30         fclose(f);
31     }
32 }
33
34
35
36
37
38
39
40
```

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