C

```
/*
 *  Esta série de programas pretende formar um conjunto didáctico de exercícios
 * para a aprendizagem da linguagem C
 *
*/

/* (início de comentário)
 *  Este texto que está aqui escrito não é interpretado pelo compilador porque
 * está inserido entre o início e fim de comentário.
 *  O compilador não aceita níveis de comentário, ou seja,
 *   um comentário dentro de comentário
 (fim de comentário) */


/*
 *   As próximas duas linhas permitem a inclusão dos ficheiros stdio.h  e stdlib.h
 *   Esses ficheiros estão numa directoria do compilador chamada de include
 *   Os ficheiros de include podem ser lidos, e contêm habitualmente declarações
 * de constantes e definições de estruturas de dados. Os ficheiros de include
 * que vêm com o compilador não devem ser alterados, a não ser que o programador
 * saiba exactamente o que está a fazer.
 *   Os ficheiros de include também podem ser construídos pelo próprio utilizador.
 *   A sintaxe a utilizar nestes casos deverá ser   #include "meu_include.h"
 *   É necessário acrescentar ao compilador as directorias onde podem ser encontrados
 * os ficheiros de include. Habitualmente é nas opções do compilador.
 *   A extensão dos ficheiros de include é habitualmente a .h embora qualquer
 * outra também sirva.
 *
*/
#include <stdio.h>  /* Funções de entradas e saída */
#include <stdlib.h> /* Funções que são muito utilizadas */

/*
 *    A linguagem C baseia-se em funções, e a primeira das funções a ser chamada,
 * (dita de programa principal) é a que tem o nome de main.
 *    A linguagem C é sensível às letras minúsculas e maiúsculas, de modo que é
 * necessário manter um mesmo estilo de escrita em todo o programa.
 *   Observar os ficheiros de include e programas de exemplo que vêm com o
 * compilador é um bom exercício de adaptação à forma da escrita.
 *   Sempre que possível, (ou em dúvida) devemos utilizar as letras minúsculas.
 *
 *   int main(int argc, char *argv[])
 *
 *   Esta declaração da função indica que a função main recebe dois argumentos e
 * devolve um inteiro.
 *   O primeiro argumento argc, é um inteiro que contém o número total de
 * argumentos com que é evocado o programa na linha de comando. O próprio nome
```

```
 * do programa conta também como argumento.
 *   Exemplo:  Um programa com o nome primeiro.exe vai ser executado.
 *
 *       c:\> primeiro ola 12034 buu!
 *
 *       A função main é chamada com argc=4
 *
 *   O segundo, argv, contém uma tabela (array, matriz) em que cada elemento
 * dessa tabela é a sequência dos argumentos que foram escritos. Utilizando
 * o mesmo exemplo, argv contém argc elementos (4)
 *
 *     argv[0] = "primeiro"
 *     argv[1] = "ola"
 *     argv[2] = "12034"
 *     argv[3] = "buu!"
 *
 *   Na linguagem C, o primeiro elementos dos arrays (matriz, tabela) é sempre
 * o zero [0].
 *
 *   A definição de argv indica que é um array de apontadores para caracter
 *   Continuando com o mesmo exemplo, isso significa que argv[2] é uma string,
 * um conjunto de caracteres seguidos uns aos outros e terminados pelo caracter
 * nulo (caracter 0). Isto significa que "12034" é texto e não é o inteiro 12034
 *   O que está mesmo escrito em argv[2] é o seguinte: '1' '2' '0' '3' '4' 0
 * ao todo 6 caracteres. ´0´ é o caracter que representa o dígito zero.
 *  O caracter 0 é o caracter nulo.
 *   Observando a Tabela ASCII, encontramos que o valor do caracter '0' é 30h ou
 * seja 48 em decimal, e o valor do caracter NUL é 0h em hexadecimal e 0 em decimal.
 *
 *   O que está então em argv[2]=[ 31h, 32h, 30h, 33h, 34h, 00h ];
*/
int main(int argc, char *argv[])
{
/*
 *   Para se escrever no ecrã utiliza-se a função printf.
 *   A função printf tem muitas formas predefinidas para facilitar a escrita.
 * Como são muitos os modos, eles estão escritos num documento apropriado.
 *
*/
  printf("Este é o primeiro programa");

/*
 *   Habitualmente utilizam-se sistemas de desenvolvimento com janelas, e estas
 * fecham após a conclusão do programa. Para evitar essa situação como é neste
 * utiliza-se a função system("PAUSE"); que vai evocar o comando PAUSE do sistema
 * operativo, e esse comando fica à espera que se pressione uma tecla para continuar.
 *   Nos primeiros programas é aceitável esta forma de terminar um programa ...
*/
  system("PAUSE");
```

```
/*
 *   A função main necessita de devolver um valor que indica o sucesso ou não
 * da evocação do programa.
 *   É habitual considerar o valor 0 (zero) como aquele que indica que tudo
 * correu como o previsto, ou seja, não houve erro.
 *   Qualquer outro valor diferente de 0 indica que houve algum tipo de erro.
 * É necessário saber os valores a devolver, e podem ser encontrados nos
 * ficheiros apropriados, nomeadamente errno.h e stderr.h
 *
*/
  return 0;
}


/*
```

É EXTREMAMENTE IMPORTANTE CRIAR O HÁBITO
DE COMENTAR OS PROGRAMAS
E TAMBÉM DE OS IDENTAR CONVENIENTEMENTE.

O programa que se segue é o mesmo que está escrito acima, mas sem qualquer tipo de identação. O compilador não dá erro, pois a identação é para que os humanos possam ler e compreender.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){printf("  Este é o primeiro
programa  ");system("PAUSE");return 0;}
```

Apresenta-se novamente o mesmo programa apenas com os comentários necessários.

```
*/
```

```
#include <stdio.h>

/*
 * Este é o primeiro programa recomendado
 *
*/

int main(int argc, char *argv[])
{
    printf("  Este é o primeiro programa  ");
    system("PAUSE");          /* evita o fechar da janela */
    return 0;
}
```

Observe a execução do programa.
Que diferenças encontrou relativamente ao que esperava?