```c
#include <stdio.h>
#include <stdlib.h>
/*
 * Este programa lê diversos caracteres do ecrã e interpreta-os do modo como
 * indicamos à função scanf para o fazer.
 * Também são escritos diversos caracteres e o modo como é feito é aquele
 * indicado pela função printf.
 *   O teclado é reconhecido pelo compilador de C como o ficheiro de entrada
 * standard STDIN, e o ecrã é reconhecido como o ficheiro de saída STDOUT.
 *   O ficheiro de erro standard STDERR também está redireccionado para o ecrã.
*/

int main(int argc, char *argv[])
{
int num;          /* modo de declarar uma variável do tipo inteiro */
char texto[256];
/*
   modo de declarar uma string com capacidade para 255 caracteres.
  o valor de 256 não é engano, é preciso ter espaço para o caracter de fim
  de string, o caracter NUL (zero 0)
    nos exemplos que se seguem não é feito qualquer tipo de verificação para
  assegurar que o utilizador não escreve mais do que esse número de caracteres.
    Experimente, para "ver o que acontece" quando coloca texto com a dimensão de
  4 caracteres e depois escreve uma string (linha de texto) com cerca de 120 caracteres
*/

  /* Observe bem o que encontra escrito no ecrã e aquilo que está aqui escrito */
   printf("Este programa le do teclado e escreve no ecran.\n");
   printf("Utiliza a funcao printf para escrever e a funcao gets para ler");
   printf("Observe bem\n o resultado da execucao deste programa\n\n    para");
   printf("poder compreender melhor ");
   printf("o que faz o \\n e tambem\to que faz o \\t");
   printf("\n\tvamos entao comecar:\n\n");

  /*      Tal como foi explicado no programa primeiro.c apresenta-se aqui o uso
     do argc e do argv.
       Experimente dar ao programa diversos argumentos e acrescente aqui a
     escrita no ecrã desses argumentos.
  */
   printf("Este programa chama-se \n\t<%s>\n", argv[0]);
   printf("Foi executado com o numero de %d argumentos\n", argc);

  /*      O uso do gets para ler uma string.
       Repare que como num printf acima, os caracteres <> e || neste caso apenas
     servem para delimitar o texto que foi lido para se "ver" melhor.
       Experimente começar o texto e terminá-lo com e s p a ç o s   |   lol  |
  */
   printf("\nEscreva um texto:              ");
   gets(&texto[0]);
   printf("\nO texto lido foi: |%s|\n", &texto[0]);

   printf("\nEscreva novamente outro texto: ");
   gets(texto);
   printf("\nO texto lido foi: |%s|\n", texto);
```

```
 /*  Observe bem as linhas acima. Se escrever o mesmo texto, verificará que o
   resultado é idêntico. Contudo num caso utiliza-se  &texto[0]
                          e no outro simplesmente        texto
     Ao escrever &texto[0], estamos a indicar o seguinte:
        o endereço da posição 0 no array texto

     Ao escrever texto, estamos a indicar o seguinte:
        o endereço do início do array texto


     Experimente escrever outro valor sem ser o 0, por exemplo   &texto[2]

    e escreva no teclado a frase:     Agora já sei o que significa &texto[2]!

     Se quisesse fazer o mesmo com a segunda forma, como o faria?
 */

  printf("\n\nAgora chegou a vez dos numeros ....\n");

  printf("\tEscreva um numero:                ");
  gets(texto);

  printf("\tEscreva outra vez o mesmo numero: ");
  scanf("%d", &num);

  printf("\nO primeiro numero escrito foi |%s| e o segundo foi |%d|\n",
                  texto, num);

 /*   Que diferenças encontra ?
      Experimente substituir gets(texto); por scanf("%s", texto);
 */

 /*   O printf também pode ser utilizado para representar um número de diversas formas  */

  printf("\nagora em diversas formas:\n");
  printf("\tnum= %d    %07d     %u\n", num, num, num);
  printf("\tnum= %d (base 10)      num= %o (base 8)     num= %x (base 16)\n",
          num, num, num);

 /*     Experimente com vários números ... e já agora com negativos também!
     O que acontece se em vez de %x utilizar %X ?
 */

 /* vamos observar novamente ....*/
  printf("\n\nnum= %d  &num= %d\n\n", num, &num);
  printf("&texto= %d    &texto[0]= %d    &texto[2]= %d\n\n",
           &texto, &texto[0], &texto[2]);
  printf(" texto= %d     texto+0 = %d     texto+2 = %d\n",
           texto, texto+0, texto+2);
  printf("\n\n");

  system("PAUSE");
  return 0;
}
```