FUNÇÕES

```
#include <stdio.h>
#include <stdlib.h>

#define SAIR         0
#define BINARIO      2
#define OCTAL        8
#define HEXADECIMAL 16
#define DECIMAL     10

/*
 * ler
 *
 *     Esta função mostra um menu de comandos e lê a escolha  do utilizador.
 *  Enquanto o utilizador não der um comando certo a função insiste com o menu.
*/
int ler()
{
int resultado;
char valor;

    resultado=-1;
    while (resultado == -1 ) {
        putchar('\n');
        printf("\nMenu\ns- sair\n");
        printf("b- escrever valor em binário\n");
        printf("h- escrever valor em hexadecimal\n");
        printf("o- escrever valor em octal\n");
        printf("d- ler número em decimal para converter\n\n");
        scanf(" %c", &valor);
        switch (valor){
            case 'S':
            case 's':    resultado=SAIR;
                break;
            case 'B':
            case 'b':    resultado=BINARIO;
                break;
            case 'H':
            case 'h':    resultado=HEXADECIMAL;
                break;
            case 'O': case 'o': resultado=OCTAL;
                break;
            case 'D': case 'd': resultado=DECIMAL;
                break;
        }
    }
    return(resultado);        /*É assim que se devolve o valor da Função*/
}
```

```
/*
 * Dec_Bin
 *
 *    Esta função tem como entrada um valor em Decimal e
 *  converte esse número para Binário.
 *    A conversão é feita através das divisões sucessivas.
 *    A função é recursiva (chama-se a si mesma), porque é
 *  necessário dividir primeiro e só depois é que se podem
 *  escrever os valores na ordem inversa;
 *    A recursividade termina quando o valor a dividir for 0.
 *  Nessa altura a função retorna ao local de onde foi chamada
 *  e escreve o valor do resto;
*/

void Dec_Bin(int valor)
{
int quociente, resto;

    if(valor != 0){
        quociente=valor / 2;
        resto=valor-quociente * 2;
        Dec_Bin(quociente);
        printf("%d", resto);
    }
}

/*
 * Dec_Oct
 *
 *   Esta função tem como entrada um valor em Decimal e converte esse
 * número para Octal.
 *   A conversão é feita através das divisões sucessivas.
 *   A função é recursiva (chama-se a si mesma), porque é necessário
 * dividir primeiro e só depois é que se podem escrever os valores na ordem inversa;
 *   A recursividade termina quando o valor a dividir for 0. Nessa
 * altura a função retorna ao local de onde foi chamada e escreve o valor do resto;
 *   O uso do atributo register faz com que a variável quociente
 * fique associada a um registo e como consequência uma rapidez no cálculo.
 *    Ao utilizar o shift << e >> de 3, faz com que as operações de
 * divisão e de muiltiplicação se tornem muito mais rápidas.
 *   Isto acontece porque 8 = 2^3
*/

int Dec_Oct(int valor)
{
register int quociente;

  if(quociente=(valor >> 3))        /* if ( (quociente = ( valor / 8)) != 0 ) */
  printf("%d", Dec_Oct(quociente));    /*  a função devolve um inteiro  */
  return(valor-(quociente << 3));     /* return( valor - (quociente * 8) ) */
}
```

```
/*
 * Dec_Hex
 *
 *    Esta função tem como entrada um valor em Decimal e  converte esse número
 * para Hexadecimal.   A conversão é feita através das divisões sucessivas.
 *    A função é recursiva (chama-se a si mesma), porque é necessário dividir
 * primeiro e só depois é que se podem escrever os valores na ordem inversa;
 *    A recursividade termina quando o valor a dividir for 0.
 *  Nessa altura a função retorna ao local de onde foi chamada  e escreve
 * o valor do resto;
*/
int Dec_Hex(int valor)
{
int quociente, resto;

    if(valor){
        quociente=valor / 16;
        resto=valor-quociente * 16;
        Dec_Hex(quociente);

        switch (resto){            /* O resto é um número hexadecimal */
            case 10: putchar('A'); break;       /* por isso é preciso */
            case 11: putchar('B'); break;              /* escrever os */
            case 12: putchar('C'); break;     /* símbolos correctos. */
            case 13: putchar('D'); break;
            case 14: putchar('E'); break;
            case 15: putchar('F'); break;
            default: printf("%d", resto);   /* ou putchar('0'+resto); */
        }
         /* O switch acima poderia ser substituído pelas seguintes instruções: */
            /*      resto += ((resto <= 9) ? '0' : 'A'-10);  */
            /*      putchar(resto);          */
    }
}

int main(int argc, char *argv[])
{
int oper=DECIMAL, numero;

    printf("  Este programa permite converter um número \n");
    printf(" decimal (base 10) para outra base à escolha.\n");
    printf("Escreva o número: ");
    while (oper != SAIR){
        switch (oper){
            case BINARIO: Dec_Bin(numero); break;
            case OCTAL: printf("%d", Dec_Oct(numero)); break;
            case HEXADECIMAL: Dec_Hex(numero); break;
            case DECIMAL: scanf("%d", &numero); break;
        }
        oper=ler();
    }
    return 0;
}
```

Este programa tem diversas Funções. Escreve-o e põe-o a correr.

Procura compreender o que faz na globalidade e o que é que cada Função faz em particular.

Se olhares com atenção apercebes-te de que uma Função é um programa, ou melhor, o Programa (`main`) é uma Função, a Função principal.

Por vezes, para simplificar os programas, é preciso chamar uma Função em que essa Função se chama a si mesma. Ou seja, é recursiva. Parece complicado mas não é; é como se descesses uma escada, e em cada degrau colocas as operações que fazes dentro da função. Se chamas novamente a função, desces mais um degrau e começas tudo de novo. E vais descendo, descendo, descendo … Depois, quando a Função sai pela primeira vez, sobes o degrau em que estás com o resultado dessa Função, e utiliza-lo aí, no degrau de cima. É provável que de seguida voltes a ter novo resultado e subas mais um degrau. E vais subindo, subindo, subindo …

Até que chegas outra vez ao lugar de partida!

Vês, afinal programar é bem divertido ;)