

Ficha 1

Programação Imperativa

1 Estado e atribuições

Diga, justificando, qual o output de cada um dos seguintes excertos de código C. Pode comprovar a sua resposta copiando o código de cada uma das alíneas para <https://tinyurl.com/vaiycnt5>.

1.

```
int x, y;  
x = 3;    y = x+1;  
x = x*y; y = x + y;  
printf("%d %d\n", x, y);
```

2.

```
int x, y;  
x = 0;  
printf ("%d %d\n", x, y);
```

3. (assuma que os códigos ASCII dos caracteres 'A', '0', ' ' e 'a' são respectivamente 65, 48, 32 e 97)

```
char a, b, c;  
a = 'A'; b = ' '; c = '0';  
printf ("%c %d\n", a, a);  
a = a+1; c = c+2;  
printf ("%c %d %c %d\n", a, a, c, c);  
c = a + b;  
printf ("%c %d\n", c, c);
```

4.

```
int x, y;  
x = 200; y = 100;  
x = x+y; y = x-y; x = x-y;  
printf ("%d %d\n", x, y);
```

2 Estruturas de controlo

1. Diga, justificando, qual o output de cada um dos seguintes excertos de código C. Mais uma vez, pode usar o C Tutor para validar a sua resposta.

(a)

```
int x, y;
x = 3; y = 5;
if (x > y)
    y = 6;
printf ("%d %d\n", x, y);
```

(b)

```
int x, y;
x = y = 0;
while (x != 11) {
    x = x+1; y += x;
}
printf ("%d %d\n", x, y);
```

(c)

```
int i;
for (i=0; (i<20) ; i++)
    if (i%2 == 0) putchar ('_');
    else putchar ('#');
```

(d)

```
void f (int n) {
    while (n>0) {
        if (n%2 == 0) putchar ('0');
        else putchar ('1');
        n = n/2;
    }
    putchar ('\n');
}

int main () {
    int i;
    for (i=0;(i<16);i++)
        f (i);
    return 0;
}
```

3 Programas iterativos

Escreva programas que desenhem as seguintes figuras no ecran. Para cada um deles faça a sua solução baseada numa função que recebe como argumento a dimensão da figura. Pode

usar como plataforma de escrita/teste dos seus programas o seguinte endereço <https://codeboard.io/projects/217668>.

1. Escreva um programa que desenhe no ecran (usando o caracter #) um quadrado de dimensão 5. O resultado da invocação da função com um argumento 5 deverá ser

```
#####
#####
#####
#####
#####
```

2. Escreva um programa que desenhe no ecran (usando os caracteres # e _) um tabuleiro de xadrez. O resultado da invocação dessa função com um argumento 5 deverá ser

```
#_#_#
_#_#_
#_#_#
_#_#_
#_#_#
```

3. Escreva duas funções que desenhem triangulos (usando o caracter #). O resultado da invocação dessas funções com um argumento 5 deverá ser

```
#
##
###          #
####         ###
#####       #####
#####      #####
#####     #####
###        #####
##
#
```

Defina cada uma dessas funções (com o nome `triangulo`), num ficheiro separado (`vertical.c` e `horizontal.c`). Compile esses dois ficheiros (usando o comando `gcc -c`) separadamente.

Considere agora o seguinte programa `triangulo.c`

```
#include<stdio.h>

void triangulo (int n);

main () {
    triangulo (5);
    return 0;
}
```

Compile este programa (com o comando `gcc -c triangulo.c`). Construa (e use) agora dois executáveis, usando os comandos

- gcc -o t1 triangulo.o vertical.o
- gcc -o t2 triangulo.o horizontal.o

4. Escreva um programa que desenhe no ecran (usando o caracter #) um circulo. O programa deverá começar por ler o raio do círculo pretendido. O resultado para um valor de raio 4 deverá ser

```

      #
    #####
  #####
#####
#####
#####
#####
#####
      #

```

Faça com que a função que desenha o círculo retorne o número de '#' impressos, e inspeccione os resultados obtidos em algumas experiências.