

```
/* e-fólio A
   Compilação 2024/25
   Linguagem: MOC (My Own C)
   Compilador da linguagem: MOCC
*/

/* Os comentários são o habitual do C */

/* A sintaxe é a habitual do C, com as restrições que se seguem */
/* não há #include nem qualquer diretiva # */

/* as variáveis podem ser int ou double, podendo ser simples ou vetores */
/* o valor inteiro pode ser visto como um carácter */
/* um vetor de inteiros pode ser visto como uma string de caracteres */
/* as variáveis podem ser declaradas de três formas */
/* sem inicialização (valor 0 é atribuído por omissão), uma ou mais variáveis */
int m,n,v[10];
double x,y,z;
```

```
/* iniciadas com uma expressão aritmética (operadores +,-,*, / e %) */  
int m=1,n=2*m;  
double x=3.14,y=x/2;  
  
/* iniciadas com um valor introduzido pelo utilizador */  
/* usamos a função read(), readc() ou reads() para o efeito */  
int m=read(); /* lê um inteiro */  
double n=read(); /* lê um double */  
int c=readc(); /* lê um carácter, guardando o código ASCII do mesmo */  
int s[] = reads(); /* lê uma string, guardando os códigos ASCII terminando em 0 */  
  
/* podemos ter uma mistura de tudo e os vetores estarão entre chavetas */  
int a, b= read(), c=2*b, v[]={1,2,3}; /* v fica com o tamanho 3 automaticamente */  
  
/* NOTA: se a variável ainda não tiver sido declarada anteriormente, deve dar erro */  
  
/* não existem estruturas */
```

```
/* as conversões entre int e double seguem as regras do C */  
/* pode haver casting (double) e (int) */  
/* os protótipos das funções devem ser declarados antes de qualquer função ou variável */  
/* a função main é a primeira a ser executada */  
  
/* todos os blocos devem estar entre chavetas, mesmo quando têm uma só instrução */  
/* as condições são simplificadas, só podendo ser na forma Expr ou Expr OpCond Expr,  
   além dos operadores lógicos &&, || e ! */  
  
/* os if podem ou não ter else */  
if(x>y) {y=x;} else {x=y;}  
if(x>y) {y=0;}  
  
/* os ciclos podem ser do tipo while */  
while (x>0) {x = x-10;}  
  
/* ou for */  
for(i=0;i<10;i=i+1) {x=x+i; y=y+x;}
```

```
/* não é possível usar ++, --, +=, etc. */  
/* a função read() lê apenas um valor de cada vez */  
/* para ler um vetor de int ou double tem de se fazer um ciclo for */  
/* para ler uma string ou carácter, tem de usar reads ou readc() */  
c=readc(); /* se for um inteiro, lê apenas um carácter */  
s=reads(); /* se for um vetor lê a string toda */  
/* os valores lidos são convertidos no código ASCII, o enter é transformado em 0 */  
/* para escrever no ecrã existem as funções write() e writec(), para variáveis simples,  
    writev() para vetores e writes() para strings */  
/* Exemplo v={97,89,99,0} */  
write(v[0]); /* escreve: 97 */  
writec(v[0]); /* escreve: a */  
writev(v); /* escreve: {97,98,99,0} */  
writes(v); /* escreve: abc */  
  
/* writes também pode ser usado para escrever uma string literal */  
writes("Hello, World!"); /* muda de linha após a string */
```

```
/* exemplo 1
   fatorial versão recursiva*/
int fact(int);
void main(void);

int fact (int k) {
    if (k<=1)
        return 1;
    else
        return k * fact (k-1);
}

void main(void){
    int n;
    writes("Introduza inteiro: ")
    n = read();
    write(fact(n));
}
```

```
/* exemplo 2
   fatorial versão recursiva*/
int fact(int);
void main(void);

int fact (int k) {
    int i,n=1;
    for(i=2;i<=k;i=i+1) { n = n*i;}
    return n;
}

void main(void){
    int n;
    writes("Introduza inteiro: ")
    n = read();
    write(fact(n));
}
```

```
/* exemplo 3
   média de uma lista de valores positivos */
double avg(double[]);
void main(void);
double avg(double v[], int size){
    int i;
    double sum = 0;
    for (i=0;i<size;i=i+1) { sum = sum + i;}
    return sum/size;
}

void main(void){
    int i,n;
    double v[100];
    writes("Introduza tamanho do vetor, seguido dos respetivos valores: ")
    n = read();
    for(i=0;i<n;i=i+1) { v[i]=read(); }
    write(avg(v,n));
}
```