

Apresentação das diferenças de sintaxe e bibliotecas entre C# e C

Nota: Para a resolução dos exercícios de C deverão utilizar o compilador online disponível em <http://onlinegdb.com>

Os conceitos de C serão apresentados pela mesma ordem dos conceitos de C# apresentados no livro.

1. Estrutura de um programa

A estruturação de programas em C é similar à dos programas em C# existindo, no entanto, algumas diferenças:

- Tal como em C# é necessário no início do programa declarar as bibliotecas que vão ser utilizadas pelo mesmo. Por exemplo, para as funções de leitura e escrita de dados em C é preciso utilizar a biblioteca `stdio.h`. No C a declaração de inclusão de bibliotecas é feita da seguinte forma: **#include <stdio.h>**
- Como em C não existe o conceito de classes a estrutura do programa é mais simples existindo somente a declaração das funções, de forma sequencial. Ao contrário do C#, em que os métodos podem ser declarados em qualquer parte do programa, em C uma função, para ser utilizada por outra função, tem de ser declarada antes. Para evitar respeitar esta ordem também se pode colocar o cabeçalho de todas as funções no início do programa.
- À semelhança do C# o C também tem de ter sempre uma função principal, a função `Main`.

```
9  #include <stdio.h>
10
11 int main ()
12 {
13     printf ("Hello World");
14
15     return 0;
16 }
17
```

Figura 1. Exemplo de um programa em C

2. Tipos de dados

Os tipos de dados primitivos em C são similares aos do C#. Na seguinte tabela é possível ver um resumo dos tipos de dados disponíveis.

Tipo	Espaço ocupado	Intervalo de valores	
char	1 byte	-128 to 127 or 0 to 255	
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295	
short	2 bytes	-32,768 to 32,767	
unsigned short	2 bytes	0 to 65,535	
long	4 bytes	-2,147,483,648 to 2,147,483,647	
unsigned long	4 bytes	0 to 4,294,967,295	
Tipo	Espaço ocupado	Intervalo de valores	Precisão
float	4 byte	1.2E-38 to 3.4E+38	6 casas decimais
double	8 byte	2.3E-308 to 1.7E+308	15 casas decimais
long double	10 byte	3.4E-4932 to 1.1E+4932	19 casas decimais

Figura 2. tipos de dados inteiros e reais

A declaração de constantes e variáveis em C é similar ao C#, com a diferença de que em C as variáveis têm de ser declaradas no início do programa.

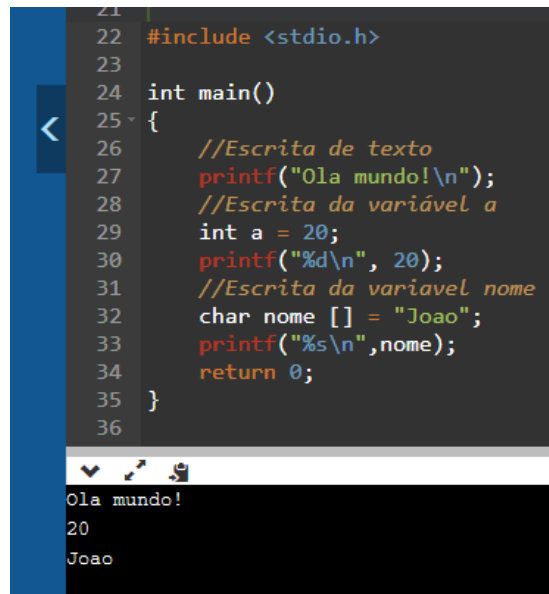
3. Leitura e escrita de dados

As funções de leitura e escrita de dados em C fazem parte da biblioteca <stdio.h> que tem de ser declarada no início do programa.

As duas funções mais utilizadas para escrita (saída) de dados são

- Função int putchar(int), que coloca um caractere no ecrã
- Função int printf(char *string_de_formato, lista_de_argumentos), que permite fazer a escrita formatada no ecrã, à semelhança das funções Write/WriteLine do C#

Em C os formatos são indicados no texto de forma similar ao C#, com diferença de em vez de se utilizarem as chavetas se utilizarem as percentagens. A figura seguinte mostra a escrita de três coisas diferentes, o texto “Ola mundo!”, a variável a do tipo inteiro e a variável nome do tipo vetor de caracteres.

A screenshot of a code editor showing a C program. The code includes <stdio.h> and defines a main function. Inside main, it prints "Ola mundo!\n", then an integer variable 'a' with the value 20 using the format "%d\n", and finally a character array 'nome' with the value "Joao" using the format "%s\n". The output window below shows the results: "Ola mundo!", "20", and "Joao".

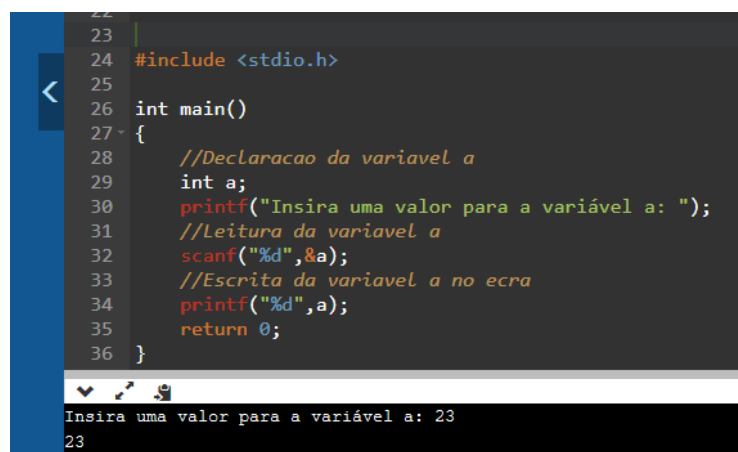
```
21 |
22 | #include <stdio.h>
23 |
24 | int main()
25 | {
26 |     //Escrita de texto
27 |     printf("Ola mundo!\n");
28 |     //Escrita da variável a
29 |     int a = 20;
30 |     printf("%d\n", 20);
31 |     //Escrita da variavel nome
32 |     char nome [] = "Joao";
33 |     printf("%s\n", nome);
34 |     return 0;
35 | }
36 |
```

Ola mundo!
20
Joao

Figura 3. Exemplo de escrita no ecrã

As duas funções mais utilizadas para leitura (entrada) de dados são:

- Função int getchar() que lê um caracter do teclado, semelhante à função Read() do C#
- Função int scanf (char *string_de_formato, lista_de_argumentos) que permite fazer a leitura formatada
- Enquanto que em C# é preciso fazer a leitura e colocar o resultado dessa leitura numa variável através de uma instrução de atribuição, fazendo a respetiva conversão, caso seja necessário, em C, com a instrução scanf basta indicar o formato e a variável onde se quer colocar o valor utilizando o operador “&”. A figura seguinte mostra o exemplo de leitura de um número inteiro e a escrita do mesmo no ecrã.

A screenshot of a code editor showing a C program. The code includes <stdio.h> and defines a main function. Inside main, it declares an integer variable 'a', prompts the user to enter a value, reads the input using scanf with the format "%d" and the address of 'a', and then prints the value using printf with the format "%d". The output window shows the prompt "Insira uma valor para a variável a: " and the user input "23".

```
22 |
23 |
24 | #include <stdio.h>
25 |
26 | int main()
27 | {
28 |     //Declaracao da variavel a
29 |     int a;
30 |     printf("Insira uma valor para a variável a: ");
31 |     //Leitura da variavel a
32 |     scanf("%d",&a);
33 |     //Escrita da variavel a no ecrã
34 |     printf("%d",a);
35 |     return 0;
36 | }
```

Insira uma valor para a variável a: 23
23

Figura 4. Exemplo de leitura e escrita de uma variável do tipo inteiro

A lista de formatos disponível em C é a seguinte:

Código	Significado
%c	Exibe um caractere
%d	Exibe um inteiro em formato decimal
%i	Exibe um inteiro
%e	Exibe um número em notação científica (com e minúsculo)
%E	Exibe um número em notação científica (com E maiúsculo)
%f	Exibe um ponto flutuante em formato decimal
%g	Usa %e ou %f, o que for menor
%G	O mesmo que %g, só que um E maiúsculo é usado se o formato %e for escolhido
%o	Exibe um número em notação octal
%s	Exibe uma string
%u	Exibe um decimal sem sinal
%x	Exibe um número em hexadecimal com letras minúsculas
%X	Exibe um número em hexadecimal com letras maiúsculas
%%	Exibe um sinal de %
%p	Exibe um ponteiro

Figura 5. Formatos disponíveis em C

a. Exercícios

- Escreva um programa que apresente no ecrã o resultado da expressão $3 + 4$.
- Escreva um programa que apresente no ecrã o resultado da expressão $5/2$.
- Escreva um programa que calcule a área e o perímetro de um quadrado.
- Desenvolva um programa que leia o valor do raio de uma circunferência e apresente a sua área e o seu perímetro.
- Escreva um programa que peça ao utilizador uma data no formato “dd-mm-aaaa” e a apresente no formato “dd/mm/aaaa”.

4. Instruções de decisão e repetição

As instruções de decisão e repetição são similares ao C#, com exceção da instrução **foreach**, que não existe em C.

a. Exercícios

- Escreva um programa que devolva o maior de dois valores.
- Desenvolva um programa que converta um valor real positivo para um inteiro por arredondamento.
- Escreva um programa que leia um número e indique se é positivo, negativo ou nulo.
- Escreva um programa que verifique se um caractere introduzido pelo utilizador é maiúsculo, minúsculo ou outro.
- Escreva um programa que peça a hora atual ao utilizador no formato 0-24 horas e escreva no ecrã a hora no formato AM - PM. Exemplo: Hora atual: 18:30 > 6:30 PM.
- Faça um programa que peça dois números, base e expoente, e mostre o primeiro número elevado ao segundo número. Não utilize a função de potência da linguagem.
- Escreva um programa que leia uma sequência de números inteiros a partir do teclado e acumule unicamente a soma dos inteiros positivos. O programa termina quando o número lido for zero.
- Desenvolva um programa que determine os divisores de um número inteiro introduzido pelo utilizador.
- Faça um programa que calcule o fatorial de um número inteiro fornecido pelo utilizador. Ex.: $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.
- Faça um programa que calcule o número médio de alunos por turma. Para isto, peça a quantidade de turmas e a quantidade de alunos de cada turma. As turmas não podem ter mais de 40 alunos.

5. Funções

Existem algumas diferenças na utilização de funções em C relativamente ao C#, sendo as mais importantes:

- Necessidade de declarar as funções antes da sua utilização.

```

23
24 #include <stdio.h>
25
26 int soma(int a, int b){
27     return a+b;
28 }
29 int main()
30 {
31     int num1, num2;
32     printf("Insira um valor para o numero 1: ");
33     scanf("%d",&num1);
34     printf("Insira um valor para o numero 2: ");
35     scanf("%d",&num2);
36     printf("A soma de %d com %d e %d.",num1,num2,soma(num1,num2));
37     return 0;
38 }
39

```

input

```

Insira um valor para o numero 1: 3
Insira um valor para o numero 2: 4
A soma de 3 com 4 e 7.

...Program finished with exit code 0
Press ENTER to exit console.

```

Figura 6. Exemplo de utilização de funções

- Outra diferença do C para o C# é que em C a passagem por referência é feita com recurso aos operadores “*” e “&” ao contrário do C# em que se utiliza a palavra chave “ref”.

```

23
24 #include <stdio.h>
25
26 void troca1(int a, int b){
27     int aux;
28     aux = a;
29     a = b;
30     b = aux;
31 }
32 void troca2(int *a, int *b){
33     int aux;
34     aux = *a;
35     *a = *b;
36     *b = aux;
37 }
38 int main()
39 {
40     int num1 = 5, num2 = 7;
41     printf("Valores originais:\n");
42     printf("a: %d, b: %d\n",num1,num2);
43     troca1(num1, num2);
44     printf("Resultado de troca1:\n");
45     printf("a: %d, b: %d\n",num1,num2);
46     troca2(&num1,&num2);
47     printf("Resultado de troca2:\n");
48     printf("a: %d, b: %d\n",num1,num2);
49
50     return 0;
51 }
52

```

```

Valores originais:
a: 5, b: 7
Resultado de troca1:
a: 5, b: 7
Resultado de troca2:
a: 7, b: 5

```

Figura 7. Exemplo de passagem por referência em C

a. Exercícios

- i. Implemente um programa que simule uma máquina de calcular, de números inteiros, em que cada uma das operações, soma, subtração, multiplicação e divisão são implementadas através de funções.
- ii. Escreva e teste uma função que devolva o máximo de dois valores.
- iii. Escreva e teste uma função que verifique se um determinado valor está dentro ou fora de um intervalo de valores.
- iv. Geração de números primos:
 1. Escreva uma função que verifique se um número inteiro positivo é primo.
 2. Implemente um programa que utilize a função da alínea anterior para imprimir os primeiros N números primos, em que N é um número inteiro inserido pelo utilizador.
- v. Escreva e teste uma função que calcule o fatorial de um número de forma recursiva.

6. Vetores e matrizes

A utilização de vetores e matrizes em C é similar a C#, possuindo, no entanto, algumas diferenças.

- A primeira diferença é na declaração dos vetores/matrizes. Os parêntesis retos vêm a seguir ao nome da variável e não antes.
- A segunda diferença, no caso das matrizes, é que não se utiliza a vírgula para indicar as diferentes dimensões, mas sim os parêntesis retos. Exemplo da declaração de uma matriz de dimensão 3x2:
 - (em C#) `int [3,2] mat`
 - (em C) `int mat [3][2]`

O mesmo se aplica à utilização das matrizes, ou seja, para ler ou escrever numa matriz em C utilizam-se os parentesis retos: `m[i][j]`.

- A terceira diferença é que não existem métodos deem o tamanho dos vetores/matrizes, pelo que tem de utilizar uma ou mais constantes para guardar o tamanho de cada dimensão das matrizes. No caso dos vetores é suficiente uma variável para guardar a dimensão do vetor, já no caso das matrizes bidimensionais são precisas duas constantes para guardar o número de linhas e o número de colunas. Também é comum utilizar a diretiva de pré-processador `#define` para definir estas constantes.

De seguida mostram-se dois exemplos de impressão no ecrã de um vetor e de uma matriz, o primeiro utilizando constantes definidas pelo utilizador e o segundo utilizando o `#define`.

```

24 #include <stdio.h>
25 void mostra_vetor(int v[], int dim){
26     int i;
27     for(i=0;i<dim;i++){
28         printf("%d ", v[i]);
29     }
30 }
31 void mostra_matriz(int m[3][2],int L, int C){
32     int i,j;
33     for(i=0;i<L;i++){
34         for(j=0;j<C;j++){
35             printf("%d ", m[i][j]);
36         }
37         printf("\n");
38     }
39 }
40 int main()
41 {
42     const int dim = 5;
43     int v [5] = {10,20,30,40,50};
44     const int L = 3, C = 2;
45     int m [3][2] = {{10,20},{30,40},{50,60}};
46     printf("Vetor:\n");
47     mostra_vetor(v,dim);
48     printf("\n");
49     printf("Matriz:\n");
50     mostra_matriz(m,L,C);
51     printf("\n");
52     return 0;
53 }
54

```

Vetor:
10 20 30 40 50
Matriz:
10 20
30 40
50 60

```

23 #include <stdio.h>
24
25 #define TAM 5
26 #define LIN 3
27 #define COL 2
28
29 void mostra_vetor(int v[], int dim){
30     int i;
31     for(i=0;i<dim;i++){
32         printf("%d ", v[i]);
33     }
34 }
35 void mostra_matriz(int m[3][2],int L, int C){
36     int i,j;
37     for(i=0;i<L;i++){
38         for(j=0;j<C;j++){
39             printf("%d ", m[i][j]);
40         }
41         printf("\n");
42     }
43 }
44 int main()
45 {
46     int v [TAM] = {10,20,30,40,50};
47     int m [LIN][COL] = {{10,20},{30,40},{50,60}};
48     printf("Vetor:\n");
49     mostra_vetor(v,TAM);
50     printf("\n");
51     printf("Matriz:\n");
52     mostra_matriz(m,LIN,COL);
53     printf("\n");
54     return 0;
55 }
56

```

Vetor:
10 20 30 40 50
Matriz:
10 20
30 40
50 60

Figura 8. Exemplos de definição/apresentação de matrizes utilizando constantes/#define

a. Exercícios

- Escreva um programa que peça 10 números ao utilizador e apresente a média dos mesmos.
- Escreva uma função que recebendo um vetor de números inteiros, inverta a posição dos seus elementos, ou seja, o que está na primeira vai para a última posição, o que está na segunda para a penúltima, etc.
- Crie uma função que receba dois vetores de inteiros e verifique se estes são iguais. Dois vetores são iguais, se na mesma posição, tiverem elementos com o mesmo valor.
- Escreva uma função que receba como argumentos dois vetores A e B de dimensão N, sendo o primeiro um vetor de caracteres e o segundo um vetor de inteiros. A função deve mostrar no ecrã cada elemento de A (um caracter), repetindo um número de vezes igual ao valor inteiro guardado no elemento correspondente de B. Cada elemento de A deve ser escrito numa nova linha.
- Escreva e teste uma função que recebe um vetor de inteiros e apresenta as frequências absolutas dos valores do vetor.
- Escreva e teste uma função que leia e apresente um conjunto de números inteiros para uma matriz bidimensional 4 X 3;
- Escreva e teste uma função que leia um conjunto de valores inteiros para uma matriz bidimensional 2 X 5 e apresente o valor e posição do maior número inteiro.
- Escreva e teste uma função que inicialize uma matriz 10 X 3 da seguinte forma: em cada uma das linhas, a primeira coluna deve ficar com um inteiro entre 1 e 100 introduzido pelo utilizador, a segunda coluna com o quadrado deste valor e a terceira com o cubo. Após a inicialização, o programa deve contar quantas posições da matriz que têm valores superiores a 1000.

7. Texto

Em C tal como em C# o texto corresponde a um conjunto de caracteres.

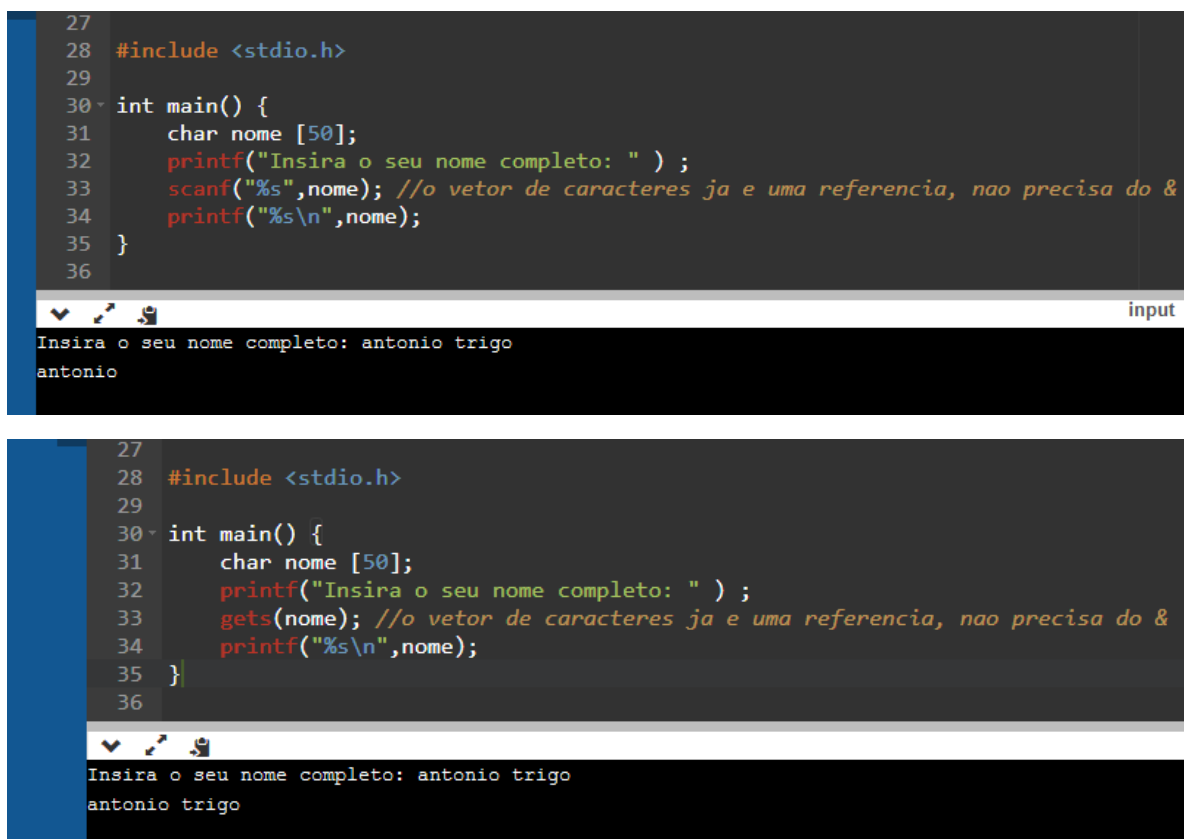
Em C# o texto tem um tipo de dados dedicado, o tipo de dados string.

Em C isto não acontece, utilizando-se para guardar texto um vetor do tipo de dados char, que tem a particularidade de ser terminado com o caractere ‘\0’.

Em C# uma string pode guardar vários caracteres ‘\0’ pois esta noção de terminação não existe.

Para leitura de texto (strings) do utilizador em C podemos utilizar a função scanf, recorrendo ao formato “%s”. No entanto esta função só vai ler o texto até ao primeiro caractere que for espaço, pelo que se quisermos ler um texto que tenha várias palavras temos de utilizar uma outra função, a função gets().

De seguida mostra-se um exemplo de leitura e escrita de texto, com a função scanf e gets.



The image contains two screenshots of a C program running in a terminal. Both screenshots show the same code, which prompts the user to enter their full name and then prints it back. The first screenshot uses the `scanf` function, and the second uses the `gets` function. In both cases, the user input 'antonio trigo' is shown, but `scanf` only captures 'antonio' because it stops at the first space character.

```
27
28 #include <stdio.h>
29
30 int main() {
31     char nome [50];
32     printf("Insira o seu nome completo: " ) ;
33     scanf("%s",nome); //o vetor de caracteres ja e uma referencia, nao precisa do &
34     printf("%s\n",nome);
35 }
36
```

input

Insira o seu nome completo: antonio trigo
antonio

```
27
28 #include <stdio.h>
29
30 int main() {
31     char nome [50];
32     printf("Insira o seu nome completo: " ) ;
33     gets(nome); //o vetor de caracteres ja e uma referencia, nao precisa do &
34     printf("%s\n",nome);
35 }
36
```

Insira o seu nome completo: antonio trigo
antonio trigo

Figura 9. Exemplos de leitura de texto

a. Exercícios

- Escreva e teste uma função que leia o seu nome completo e o apresente no ecrã.
- Escreva e teste uma função que conte o número de caracteres existentes numa String.
- Escreva e teste uma função que devolva a primeira posição numa String de um caractere introduzido pelo utilizador.
- Escreva e teste uma função que devolva quantas vezes um determinado caractere aparece numa String.
- Escreva e teste uma função que concatene duas Strings inseridas pelo utilizador.
- Escreva e teste uma função que coloque um caractere introduzido pelo utilizador numa posição *i* da String.
- Escreva e teste uma função que coloque um caractere introduzido pelo utilizador entre as posições *i* e *j* da String.
- Escreva e teste uma função que coloque asteriscos a seguir a cada um dos caracteres de uma String.
- Escreva e teste uma função que converta todos os caracteres de uma String em maiúsculas.