**Códigos Programação C e C+:**

#include **(Diz ao compiler para juntar os conteúdos de um ficheiro)**

<stdio.h> **(Para ficheiros que estejas relacionados ou precisem de inputs e outputs)**

Int main(){ … } **(Principal função/ Ponto de partida do ficheiro)**

Return 0; **(Return 0 statement/ Elemento de fecho do ficheiro) (Transmite um 0 se o programa correr sem erros) (Se houver um erro no programa ele produz um 1)**

; **(Serve para dar fim a um conteúdo/ afirmação)**

Printf(“…”); **(Serve para exibir algo como output)**

\n **(Fazer um parágrafo dentro de um uma função printf(“”)). (Ex: printf(“Program with C.\n”);. (Colocar \n em, por exemplo, 2 funções printf para realizar o parágrafo entre as duas).**

\t **(Fazer um espaço como se fosse com a tecla tab, na mesma linha);**

// **(Serve para fazer uma nota ou comentário que não seja lido pelo compiler);**

**OU**

/\* … \*/ **(Tudo o que estiver dentro destas barras será considerado uma nota ou comentário pelo compiler);**

“\”…\” – Author” **(Serve para fazer referências sem que o código de erro)**

**OU**

“\’…\’ – Author”;

**Como compilar e correr um documento em C na linha de controlos:**

1. Verificar se já temos um gcc compilier **(gcc –version);**
2. Alterar a localização do ficheiro que está em C **(cd (Localização do ficheiro));**
3. Compilar o ficheiro **(gcc (Nome do Ficheiro (Ex:FirstFile.c)));**
4. Correr o ficheiro **(a.exe e ENTER).**

**Escape Sequences:**

**Uma imagem com texto, captura de ecrã, número, Tipo de letra

Descrição gerada automaticamente**

**Variables (Tabela: Data Types):**

**Uma imagem com texto, captura de ecrã, número, Tipo de letra

Descrição gerada automaticamente**

**How to work with variables**: Declarar uma variável para alojar na na memória para gerar um valor.

**How to display the variables:**

- **Array of characters**: printf(“Hello **%s**”, name);

- **Integer**: printf(“You are **%d** years old”, age);

- **Single character**: printf(“Your grade was **%c**, well done”, grade);

- **Floating Point Number**: printf(“Your gpa is **%f**, above the common gpa’s”, gpa);

**Format Specifiers:**

**%.” number of digits”:** printf(“Item1 = %.2f\n”, item1); (It would display a number with 2 decimal numbers, as 5.75).

**%” number of spaces between the text and the number”:** printf(“Item1 = %5f\n”, item1); (It would display a number with a space of 5, Item1: 5,75).

**%-:** printf(“Item1: %-f\n”); (To left align the number displayed).

It’s possible to use all the three together like this:

Printf(“Item1: %-5.2f\n”, item1); (It would be displayed like this: Item1: 5.75 .)

**Constants:**

- Is a fixed value that cannot or may not be changed by the program.

**Const** float pi = 3.14159; (If you try to change a number that has been attributed with a “const” command, when you run the code, it will give you an error).

**Data Types:**

**Floats:** Use 4 bytes (32 bits of precision) 6 – 7 digits. **(%f)**

**Double:** Use 8 bytes (64 bits of precision) 15 – 16 digits. **(%lf)**

**It’s possible to choose how many digits you want to appear when it’s displayed:**

**Printf(“… %0.15f or %0.15lf …”);**

**Example of the difference between precision:**

Printf(“… %0.15**f** ..”); = 3.141592741012573 (32 bits)

Printf(“… %0.15**lf** …”); = 3.141592653589793 (64 bits

Ambas as funções apresentam 15 dígitos, mas a função %f, apresenta um número menos preciso que o número da função %lf.

Normalmente a função mais usada é a double porque tem mais precisão.

**Bool:** Use 1 byte (True or False) **(%d)** (To use this function you need to insert on the beginning of the C document “**#include <stdbool.h>**”. (bool employed = true;); printf(“Are you employed? %d, employed);

**Char/ Single Character:** Can be used to display a character using a number:

Ex: char “letter” = 120; printf(“The letter is **%s**”); (This would display a x) (1 byte (-128 to +127)).

Uma imagem com texto, menu, captura de ecrã, número

Descrição gerada automaticamente

**Unsigned char:** Can be used to display a character using a number:

Ex: Unsigned char “letter” = 255; printf (“The letter is %d”, letter); (1 byte (0 to 255)) (Se for ultrapassado o limite é reposto para 0). (%c or %d).

**Short:** To display a large number:

Ex: Short int “largenumber” = 32767; printf(“The large number I’m talking about is %d”, largenumber); (2 bytes (-32,768 to +32,767)). (%d)

**Unsigned short:** To display a large positive number:

Ex: Unsigned short int “largenumber2” = 65535; printf(“The larger number is %d”, largenumber2); (2 bytes (o to +65,535). (%d)

**Int:** To display a huge number:

Ex: Int “hugenumber” = 2147483647; printf(“The huge number is %d”, hugenumber); (4 bytes (-2,147,483,648 to +2,147,483,648)). (%d).

**Unsigned int:** To display a huge positive number:

Ex: Unsigned int “hugenumber2” = 4294967295; printf(“The hugest number is %u, hugenumber2); (4 bytes (0 to +4294967295). (%u).

**Long Long int:** To display a massive number:

Ex: Long long int “massive number” = 9223372036854775807; printf(“ The massive number is %lld”, massivenumber); (8 bytes (- 9 quintillion to +9 quintillion)). (%lld).

**Unsigned long long int:** To display a positive massive number:

Ex: Long long int “massivenumber2” = 18446744073709551615; printf(“The massive positive number is %llu”, massivenumber2); (8 bytes (0 to +18446744073709551615)). (%llu). (Se der erro ao rodar o ficheiro meter um “U” no final do nº).

**Arithmetic Operators:**

**+: Addition:**

Ex: int x = 5; int y = 2; int z = x + y; printf(“%d”, z); (The result would be 7, because x is 5 and y is 2).

**-: Subtraction:**

Ex: int x = 5; int y = 2; int z = x - y; printf(“%d”, z); (The result would be 3, because x is 5 and y is 2).

**\*: Multiplication:**

Ex: int x = 5; int y = 2; int z = x \* y; printf(“%d”, z); (The result would be 10, because x is 5 and y is 2).

**/: Division:**

Ex: int x = 5; float y = 2; float z = x / y; printf(“%f”, z); (The result would be 2,5, because x is 5 and y is 2). (We need to store the result with a float or a double in order to give us a decimal number) and (Change one of the main integers to a float or a double to give us a non-rounded number).

**%: Modulus: (Gives with the remaining value of any division):**

Ex: int x = 5; int y = 2; int z = x % y; printf(“%d”, z); (The result would be 1, because x is 5 and y is 2).

**++: Increment:**

Ex: Int x = 5; x++; printf(“%d”, x); (The result would be 6).

**--: Decrement:**

Ex: Int y = 2; y--; printf(“%d”, y); (The result would be 1).

**OR (Shortcut)**

Uma imagem com Software de multimédia, captura de ecrã, software, texto

Descrição gerada automaticamente

**User Inputs:**

- If it’s in vscode you need to change from output to terminal and turn on code runner (Run in Terminal).

**Scanf: Is used to read the input value (The opposite of the printf).**

Ex: int age; scanf(“%d”, **&**age); (You have to precede the input you want to run with a “&”).

**Fgets: Used to read and run white spaces too) (It’s basically a scanf function but it will read the white spaces, useful for names).**

Ex: int name; fgets(name, 25, stdin); printf(“Nice to meet you %s”, name);

(Name of the Input, Length of the output, Code to make it run)

**Name[strlen(name)-1] = ‘\0’;** (To run the input without giving 1 character to the next line).

**Math Functions:**

- Para utilizar math functions no ficheiro em C é necessário inserir no início do ficheiro (#include <math.h>.

**Raiz Quadrada: (sqrt(x));**

Ex: double A = sqrt(9); printf(“%lf”, A); This would be 3.0000.

**Número Elevado a x: (pow(x, y));**

Ex: double B = pow(2, 4); printf(“%lf”, B); This would be 16.0000.

**Arredondar: round(x)**

Ex: int C = round(3.14); printf(“%d”, C); This would be 3.

**Arredondar para o núemro acima: ceil(x);**

Ex: int D = ceil(3.14); printf(“%d”, D); This would be 4.

**Arredondar para o número abaixo: floor(x);**

Ex: int E = floor(3.99); printf(“%d”, E); This would be 4.

**Encontrar o número absolute (Diferença para chegar ao zero): fabs(x);**

Ex: double F = fabs(-100); printf(“%lf”, F); This would be 100.

**Saber a tan, cos e sin de x: (tan(x)); (cos(x)); (sin(x)); (Trignometria).**

**Cálculos de área, raio e circunferência no ficheiro em C. (VSCode)**

**Cálculo da hipotenusa no ficheiro em C. (VSCode)**

**If Statements and Switch Statements: (Consultar file IfState.c e SwitchState.c).**

**String Functions: (IMPORTANTE!!!)**

**- Passar Uppercase para Lowercase: (STRLWR)**

**Ex:** char string1[ ] = “Pedro”; **strlwr(string1);** printf(“%s”, string1); Result = pedro.

**- Passar Lowercase para Uppercase: (STRUPR)**

**Ex:** char string1[ ] = “Pedro”; **strupr(string1);** printf(“%s”, string1); Result = PEDRO.

**- Para colar a string1 à string2: (STRCAT)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; **strcat(string1, string2);** printf(“%s”, string1); Result = PedroCardoso.

**- Para passar n letras da string2 para a string1: (STRNCAT)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; **strncat(string1, string2, 1);** printf(“%s”, string1); Result = PedroC.

**- Para copiar a string2 para o lugar da string1: (STRCPY)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; **strcpy(string1, string2)**; printf(“%s”, string1); Result = Cardoso.

**- Para copiar n letras da string2 para o lugar da string1: (STRNCPY)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; **strncpy(string1, string2, 3)**; printf(“%s”, string1); Result = Carro. Porque não chega a completer as 5 letras do nome Pedro.

**- Para transformar todas as letras de uma string numa letra, simbolo etc que seja indicado na função: (STRSET)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; **strset(string1, ‘?’);** printf(“%s”, string1); Result =?????.

**- Para transformar n letras (a contar da primeira) de uma string numa letra, simbolo etc que seja indicado na função: (STRNSET)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; **strnset(string1, ‘?’, 2);** printf(“%s”, string1); Result =??dro.

**- Para mudar a ordem da palavra de trás para a frente: (STRREV)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; **strrev(string1);** printf(“%s”, string1); Result = ordeP.

**- Para mostrar quantos caracteres estão na string: (STRLEN)**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; int result = **strlen(string1);** printf(“%d”, result); Result = 5.

**- Para comparer se as strings tem o mesmo número de caracteres ou não: (STRCMP).**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; int result = **strcmp(string1, string2);** if(result == 0){ printf(“Thes strings are the same”) } else{ printf(“These strings are not the same”); Result = These strings are not the same.

**- Para comparar se as strings tem n números de caracteres iguais ou não: (STRCMP).**

**Ex:** char string1[ ] = “Pedro”; char string2[ ] = “Cardoso”; int result = **strcnmp(string1, string2, 1);** if(result == 0){ printf(“Thes strings are the same”) } else{ printf(“These strings are not the same”); Result = These strings are not the same.