

Aula 1.1 -

APRENDA A USAR O DEV C++

* Onde usa?

- Sistemas Operacionais
- " Desktop
- " embarcados (arduino)

* Características

- Código para multiplataformas
- Mais desempenho
- Próxima ao recursos de hardware
- Sintaxe rebuscada
 - ↳ mais fácil para migrar para outra linguagem
- Não é muito utilizada para sistemas web e mobile

* Qual o ambiente de programação? (IDE)

↳ DEV C++

↳ Integrated Development Environment

* Precisa incluir bibliotecas

* Todo programa em linguagem C fica dentro da função "main"

```
1 #include <stdio.h>
2
3 int main () {
4     printf ("Hello, world! \n");
5 }
6
7 }
```

↳ inclui biblioteca → digendo pro mág. "vamos usar isso!"
↳ vai estar em todos os programas
↳ corpo principal do programa
↳ todos que estiver entre as chaves vai ser executado pela mág.
↳ pula de linha
↳ mostra a mensagem (ESCREVA)
↳ prestar atenção nisso!

executável

* Para transformar o arquivo em .exe preciso compilar (F9)

* Run (F10)

Aula 1.2 - Aprenda a DEBUGAR em C

- * Debugar é executar o código passo a passo
 - ↳ mostrar onde está o erro
- * Tem que ser projeto
- * Break point → seleciona a linha onde quer que pare
 - ↳ F5 (DEBUG)
 - face o passo a passo

Aula 2.1 - Saída de dados

Sintaxe do comando ESCREVA:

```
printf(<texto>, <v1>, <v2>);
```

→ Quando quero imprimir na tela um conteúdo que é diferente de um texto literal: utilizar especificadores de formato

Sequências de escape: printf()

Escape	Descrição
\a	Toca um bipe, <u>alarme sonoro</u> padrão do sistema
\b	Backspace
\n	Quebra de linha → pula linha
\t	Tabulação horizontal (TAB)
\r	Retorna ao início da linha
\0	Caractere nulo → indica que acabou a frase
\v	Tabulação vertical
\\\	Caractere \
\'	Caractere '
\"	Caractere "
\?	Caractere ?
\123	Caractere relacionado ao código 123 em octal (ASCII)
\X12	Caractere relacionado ao código 12 em hexadecimal (ASCII)
%%	Caractere %

→ as vezes o caractere não está presente no teclado, dai usar o código dele da tabela ASCII

Ex.:

`printf ("Valor inteiro : %d.\n", 10);`

↳ o %d vai ser substituído por 10. Ele formata o 10.

Especificadores de formato: printf()

Dígito	Descrição
d ou i	Números inteiros em base decimal
x	Números inteiros em base hexadecimal
f	Números em ponto flutuante (com casas decimais)
e	Números em notação científica (com casas decimais)
c	Caracteres alfanuméricos (texto)
s	Sequência de caracteres alfanuméricos (texto)
.<num>	Especifica quantos dígitos serão impressos após a vírgula

Cura 2.2 - Entrada de dados

Sintaxe do comando `leia`:

`scanf (<form> &<v1>, &<v2>);` → recomendado: fazer um scanf para cada variável

↓
especificador
de formato
(similar ao printf)

Especificadores de formato: scanf()

Dígito	Descrição
d ou i	Números inteiros em base decimal
x	Números inteiros em base hexadecimal
f	Números em ponto flutuante (com casas decimais)
e	Números em notação científica (com casas decimais)
c	Caracteres alfanuméricos (texto)
s	Sequência de caracteres alfanuméricos (texto)
[^chars]	Lê todos os dados digitados, exceto os especificados em "chars"

↳ exclui caracteres não desejados

Exemplo:

```
#include<stdio.h>
int main(){
    int idade=0;
    printf("Valor inicial da idade: %d.\n", idade);
    printf("Digite uma idade: \n");
    scanf("%d", &idade);
    printf("Idade informada: %d.\n", idade)
}
```

* Precisa sempre colocar especificador de formato?
↳ Não, mas é bom

