Por que aprender C?

- **Principal característica:** Eficiência e controle de memória
- Portabilidade: pode ser compilado em uma grande variedade de plataformas
- Maior interface com o hardware (consegue lidar com atividades de baixo nível);
- Essencial para entender como computadores funcionam (Infra de Software);
- Essencial para melhor entender e desenvolver algoritmos e estruturas de dados (listas, árvores, tabelas hash, etc)
- *Língua Franca* entre programadores
- **Muito importante nos recrutamentos de seleção**

Fundamentos de C

- **Atributos da variável**
    - **Nome da variável**
    - **Endereço na memória**
    - **Tamanho da memória**

Hello World em C

Um **"Hello, World!"** é um programa simples que apenas exibe a mensagem **"Hello, World!"** na tela, uma tradição dos programadores ao iniciar o estudo em qualquer linguagem de programação, dizem que aqueles que não fizerem um hello world quando começam em uma nova linguagem nunca conseguiram sair do começo dela até cumprir a tradição.

```
#include <stdio.h>

int main()
{
        printf("Hellp World!");
        return 0;
}
```

Mas o que cada coisa significa neste código? segue a função de cada estrutura:

Estrutura	Descrição
#include <stdio.h>	Incluir a biblioteca stdio dentro do arquivo stdio.h (pré-processador)
int main()	Função principal -> Inicia a execução do programa
printf()	Função que imprime informação na tela (interface de saída)
return 0	Termina a execução do programa. Se a função main retorna o valor 0 o programa terminou de maneira correta

/.../	Comentários (ignorado pelo compilador)
Chaves { ... }	Indicam o escopo de uma função
Ponto e virgula (;)	Indicam o fim de uma instrução (nunca esqueçam de adiciona-lo)
Identificadores	C é case-sensitive.

Tokens e Notações

Estrutura	Descrição
Ponto e virgula (;)	Indicam o fim de uma instrução (nunca esqueçam de adiciona-lo)
Chaves { ... }	Indicam o escopo de uma função
/.../	Comentários (ignorado pelo compilador)
\n	Quebra de linha
Identificadores	Nomes usados para identificar variáveis, funções, ou qualquer outra estrutura definida pelo programador Identificadores podem começar com letras de A a Z (minúsculo ou maiúsculo) ou underscore '_' seguido de número ou letras.C e case-sensitive.
Palavras-Chaves	Palavras reservadas em C que não podem ser usadas para nomear variáveis ou funções. Alguns exemplos Auto | if | else | break | return | int | void | Char | for | struct | while | double | do | static

Tipos Primitivos

Tipos primitivos na programação são os tipos de dados mais básicos que uma linguagem oferece. Eles representam valores simples e fundamentais, como:

- Inteiros (int) – números inteiros, como 10, -3
- Ponto flutuante (float/double) – números com casas decimais, como 3.14, -0.5
- Caractere (char) – um único caractere, como 'a', '%'
- Booleano (bool) – valores lógicos: true ou false

Eles servem como base para construir estruturas mais complexas, como listas, objetos ou classes. Na linguagem C eles são:

Tipo	Descrição	Espaço	Especificação de Formatação
Char	Caracteres	1 byte	%c
Int	Inteiros	2 ou 4 bytes	%d
float	Reais	4 bytes	%f
double	Reais	8 bytes	%f
String	Lista de caracteres	n*(1 byte)	%s
Array	lista de números	N*(2,4 ou 8 bytes)	%d ou %f

Refs: C data types

Variáveis

```
tipo nomeVariavel = valor;

//exemplos:
int myNum = 15;
float myFloat = 0.5;
```

- Nomes de variáveis correspondem a locais na memória do computador
- Todas as variáveis possuem um nome, um tipo e um valor
    - Atributos da variável
        - Nome da variável
        - Endereço na memória
        - Tamanho da memória

- O valor digitado pelo usuário é colocado no local da memória ao qual o nome foi atribuído
- Sempre que um valor é colocado em um local da memória, o novo valor invalida o anterior naquele local.

```
// Declare uma variável
int myNum = 10;

// Atribua outro valor a variável
myNum = 15;


printf(myNum);// a saída será 15
```

Refs: w3schools

ScanF

O scanf na linguagem C é uma função usada para ler dados da entrada padrão (geralmente o teclado) e armazená-los em variáveis.

```
scanf("formato", &variavel);
```

- O formato indica o tipo de dado esperado (ex: %d para inteiro, %f para float, %s para string).
- O & (e comercial) é usado para passar o endereço da variável, onde o valor será armazenado.
- Exemplo:

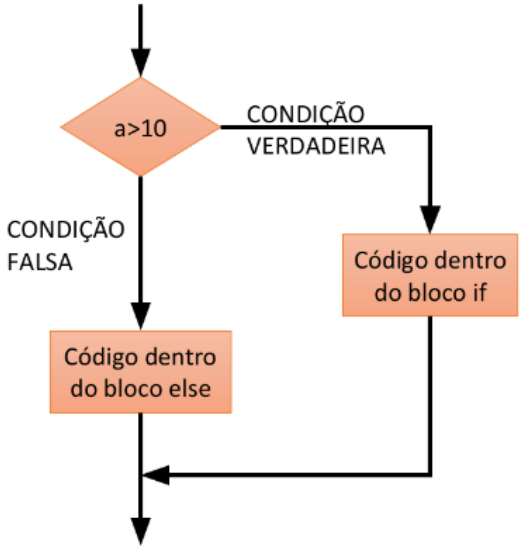
```
int numero = 0;

scanf("%d",&n);

printf(numero);
```

Esse código lê um número inteiro digitado pelo usuário e armazena na variável número.

Estrutura de Decisão

Em C, a estrutura de decisão permite executar diferentes blocos de código com base em condições. As principais são:

- if: Executa um bloco se a condição for verdadeira.
- else if: testa nova condição se o if for falso.
- else: Executa um bloco se nenhuma das condições anteriores for verdadeira.




```
#include <stdio.h>

int main(){
        int a;
        printf("Digite um numero de 0 a 10: ");
        scanf("%d", &a);

        if(a == 5) {
                printf("parabens, vc acertou\\n");
                }else{

                        prinf("Tente novamente\\n");
                }

                return 0;
}
```

Expressões Lógicas

A || B

A	B	A OR B
V	V	V
V	F	V
F	V	V
F	F	F

A && B

A	B	A AND B
V	V	V
V	F	F
F	V	F
F	F	F

!A

A	NOT A
V	F
F	V

Operadores Lógicos ou Relacionais

OPERADORES LÓGICOS E RELACIONAIS	
COMPARAÇÃO	>, >= , < <=,
IGUALDADE	==, !=
LÓGICA	&&, ||, !

Por definição, o valor numérico de uma expressão relacional ou lógica é 1 se a relação for verdade ou 0 se for falsa.

OBS - não existe tipo boolean em C

Operadores Relacionais (comparam valores):

- == igual a
- ! = diferente de
- > maior que
- < menor que
- >= maior ou igual a
- <= menor ou igual a

Operadores Lógicos (combinam condições):

- && **E** (verdadeiro se ambas as condições forem verdadeiras)
- | | **OU** (verdadeiro se pelo menos uma for verdadeira)
- ! **NÃO** (inverte o valor lógico)