

5954001 - Introdução à Computação I (Optativa para Química)

Tipos e Operadores Booleanos

Prof. Dr. Denis M. L. Martins
DCM | FFCLRP | USP

Objetivos de aprendizagem

- Representar estados com `bool` e prever a conversão de inteiros.
- Montar tabelas-verdade para condições com `!`, `&&` e `||`.
- Explicar quando C deixa de avaliar parte de uma condição.
- Traduzir uma regra em código e testá-la com casos representativos.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Uma condição é uma pergunta

Para decidir se alguém pode se inscrever, o programa pergunta se a idade é de pelo menos 18 anos.

```c
#include <stdio.h>
int main(void) {
    int idade = 17;
    bool maior_de_idade = idade >= 18;

    if (maior_de_idade) {
        printf("Inscricao permitida");
    }
   return 0;
}
```

A comparação **idade >= 18** é falsa para 17 e verdadeira a partir de 18. A variável **maior_de_idade** dá um nome à resposta.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

O tipo `bool` de C99 a C23

Padrão	Tipo	Valores escritos no código
C99 a C17	`_Bool` é o tipo nativo. `bool` é uma macro de `<stdbool.h>`.	O cabeçalho define `false` e `true` como macros para `0` e `1`.
C23	`bool` é palavra-chave. `_Bool` é uma grafia alternativa.	`false` e `true` são constantes predefinidas do tipo `bool`.

O mateoal usa **C17** com `#include <stdbool.h>`. Em C23, esses nomes estão disponíveis sem o cabeçalho.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Conversão para booleano

Na conversão de um inteiro para **bool**, **zero vira 0** e **qualquer valor diferente de zero vira 1**, inclusive negativos.

```c
#include <stdbool.h>
#include <stdio.h>

int main(void) {
    bool desligado = 0;
    bool ligado = -7;
    printf("%d %d\n", desligado, ligado);
    return 0;
}
```

A saída é **0 1**. Ao passar para **printf**, os valores de **bool** são promovidos a **int**.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Comparações em condições

- `<`, `<=`, `>`, `>=`, `==` e `!=` produzem um `int` com valor `0` ou `1`.
- Com uma expressão inteira, o `if` entra no bloco se o valor for diferente de zero.

```c
#include <stdio.h>

int main(void) {
    int saldo = -3;

    if (saldo) { // Equivale a testar saldo != 0
        printf("Saldo diferente de zero");
    }
    if (saldo > 0) {
        printf("Saldo positivo");
    }

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Negação lógica com `!`

O operador NÃO inverte o valor lógico do operando.

a	!a
0	1
1	0

- `!0` vale `1` e `!(-4)` vale `0`.
- `!!(-4)` vale `1`. A dupla negação converte a interpretação lógica em um resultado `int` de valor `0` ou `1`.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Conjunção lógica com `&&`

O operador E produz `1` quando **as duas** condições são verdadeiras.

a	b	a && b
0	0	0
0	1	0
1	0	0
1	1	1

```
bool pode_entrar = tem_ingresso && portao_aberto;
```

Um ingresso válido só basta quando o portão também está aberto.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Disjunção lógica com `||`

O operador OU produz `1` quando **pelo menos uma** condição é verdadeira.

a	b	a || b
0	0	0
0	1	1
1	0	1
1	1	1

```
bool pode_entrar = tem_ingresso || esta_na_lista;
```

O OU é **inclusivo**. Quem tem ingresso e está na lista também pode entrar.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Tabela-verdade de uma expressão composta

Cada linha representa uma combinação. Com **n** entradas booleanas, há **2^n** linhas. As colunas intermediárias mostram o valor de cada subexpressão.

a	b	!b	a && !b	(a && !b) || b
0	0	1	0	0
0	1	0	0	1
1	0	1	1	1
1	1	0	0	1

A última coluna coincide com **a || b** em todas as linhas. As duas expressões têm o mesmo valor lógico para essas entradas.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Precedência de operadores booleanos em C

Da maior para a menor, a ordem é **!** (NÃO), **&&** (E), **||** (OU).
A precedência define o agrupamento. Os parênteses podem mudá-lo.

```c
// Em uma função, com #include <stdbool.h> no arquivo.
bool convite = true, inscrito = true, lotado = true;

bool a = convite || inscrito && !lotado;    // 1
bool b = (convite || inscrito) && !lotado;  // 0
```

- Em **a**, o agrupamento é **convite || (inscrito && !lotado)**. O convite basta.
- Em **b**, **!lotado** precisa ser verdadeiro para qualquer entrada.

Teste rápido. Se apenas **convite** mudar para **false**, quanto valem **a** e **b**?

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício

Considere o trecho abaixo, com `<stdbool.h>` incluído.

```c
bool a = false;
bool b = true;
bool c = false;

int x = !a || b && c;
int y = (!a || b) && c;
```

1. Reescreva a expressão de `x` com parênteses que explicitem o agrupamento adotado por C.
2. Determine os valores de `x` e `y`. Explique o que os parênteses mudam.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

OU exclusivo e leis de De Morgan

OU exclusivo (XOR) vale `1` quando exatamente uma entrada é verdadeira. Com entradas do tipo `bool`, escreva `a != b`. Para inteiros gerais, use `(a != 0) != (b != 0)`.

Leis de De Morgan

```
!(a && b)  // Equivale a !a || !b
!(a || b)  // Equivale a !a && !b
```

Negar “tem ingresso OU está na lista” significa exigir que a pessoa esteja sem ingresso **e** fora da lista.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Armadilhas com atribuição e intervalos

- `x = 0` atribui zero a `x`. Em `if (x = 0)`, a expressão vale zero e o bloco é ignorado. Para comparar, use `if (x == 0)`.
- `0 <= x <= 10` agrupa como `(0 <= x) <= 10`. A primeira comparação vale `0` ou `1`, e ambos são menores ou iguais a 10.

```
bool dentro = x >= 0 && x <= 10;
```

Para `x = 99`, a comparação encadeada dá `1`. A condição com `&&` dá `0`, como exige a regra do intervalo.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercícios

Preveja o resultado no papel e depois use o compilador para conferir suas hipóteses.

1. Conversão e operadores. Quais são os valores de `(bool)-3`, `!-3` e `-3 && 8`? Qual é o tipo de cada resultado?

2. Tabela-verdade. Construa a tabela de `!(a || b)` com uma coluna intermediária para `a || b`. Compare com `!a && !b`.

3. Avaliação. Suponha `x = 0`. Qual é o valor de `x != 0 && 100 / x > 3`? O que acontece se a comparação com a divisão vier primeiro?

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício Prático 1: Validação de intervalo

Implemente **bool nota_valida(int nota)**. A nota válida está entre 0 e 100, inclusive. Faça o programa exibir o resultado para cada caso abaixo.

Nota	-1	0	42	100	101
Resultado esperado	0	1	1	1	0

As notas 0 e 100 testam os limites. Os valores -1 e 101 testam os vizinhos externos.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício Prático 2: Tabela gerada em C

Use dois laços `for` para percorrer `a` e `b`, cada um de 0 a 1. Mostre os valores das expressões `!a`, `a && b`, `a || b` e `a != b`.

- Imprima o cabeçalho e as quatro linhas na ordem `00`, `01`, `10`, `11`.
- Compare o resultado com as tabelas da aula.
- Na linha `11`, o OU inclusivo deve dar `1` e o exclusivo deve dar `0`.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício Prático 3: Permissões de acesso

Uma conta pode entrar se for administradora ou tiver convite. Uma conta bloqueada tem o acesso negado em qualquer situação.

1. Escreva a função **bool pode_entrar(bool admin, bool convite, bool bloqueado)**.
2. Monte a tabela com as oito combinações e marque os casos permitidos.
3. Teste explicitamente o caso de uma conta administradora bloqueada.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Dúvidas e Discussão

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br