

MINE



CRAFTANDO ALGORITMOS



Maria Clara Nascimento Silva

INTRODUÇÃO

Bem-vindo, jovem programador, ao universo empolgante da programação! Neste ebook, vamos explorar a lógica de programação através do mundo fascinante de Minecraft, transformando blocos de código em estruturas incríveis. Prepare-se para craftear algoritmos e dominar uma das linguagens de programação mais poderosas e versáteis!

Pegue sua
espada e
abra o seu
compilador,
vamos partir
numa grande
aventura!



01

CONCEITOS BÁSICOS DE PROGRAMAÇÃO



LÓGICA DE PROGRAMAÇÃO E ALGORITMOS

A **lógica de programação** é a base para resolver problemas através da criação de **algoritmos**, que são sequências de passos finitos para atingir um objetivo.

Por exemplo, um algoritmo para somar dois números pode ser descrito assim:

1. Ler o primeiro número.
2. Ler o segundo número.
3. Somar os dois números.
4. Mostrar o resultado.

Aqui está um exemplo simples em C:



```
#include <stdio.h>

int main() {
    int num1, num2, soma;

    printf("Digite o primeiro número: ");
    scanf("%d", &num1);
    printf("Digite o segundo número: ");
    scanf("%d", &num2);

    soma = num1 + num2;

    printf("A soma é: %d\n", soma);

    return 0;
}
```

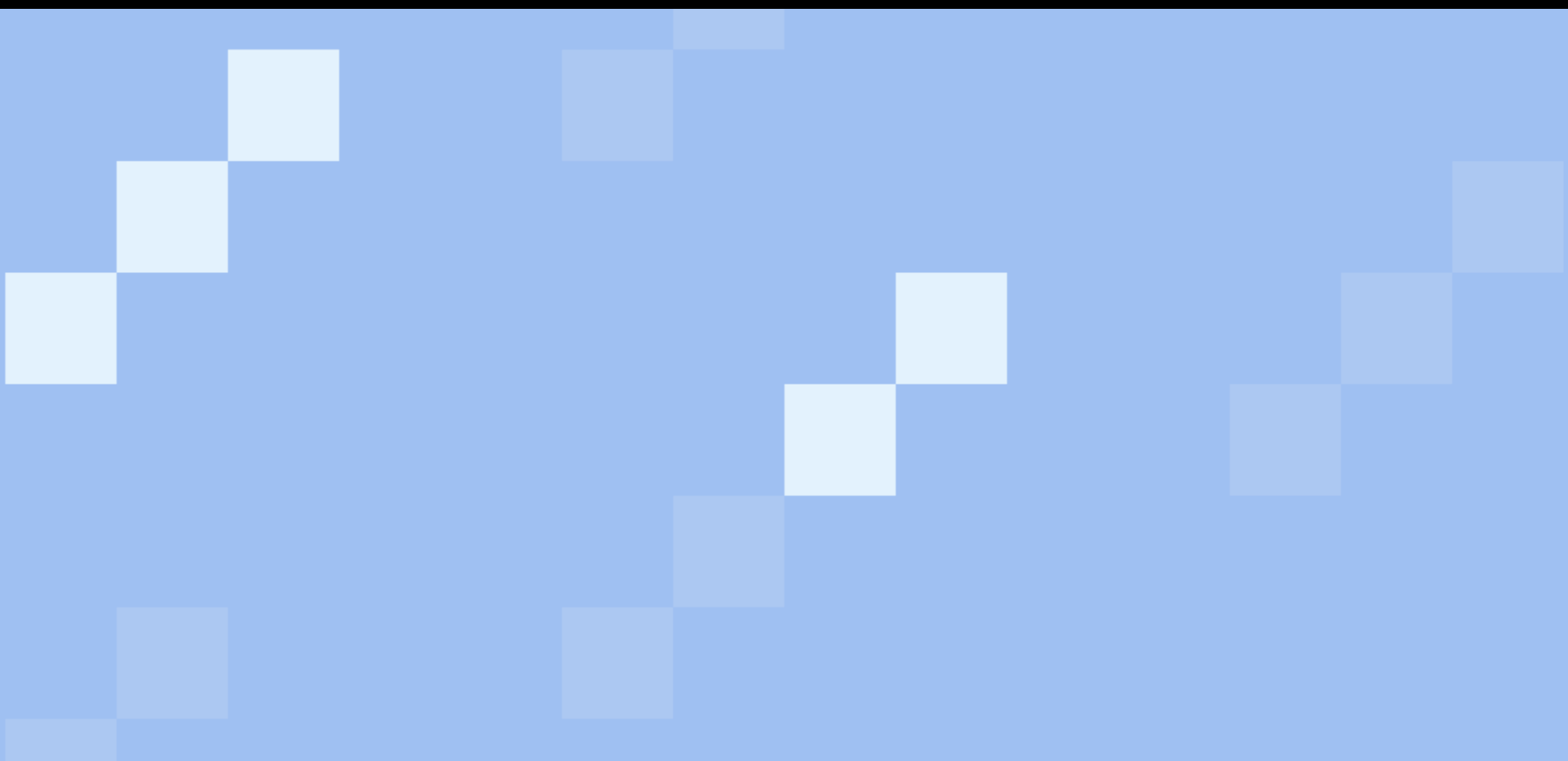
HISTÓRIA DA LINGUAGEM C

A linguagem C foi criada nos anos 1970 por Dennis Ritchie no Bell Labs. Ela foi desenvolvida para o sistema operacional Unix e se tornou uma das linguagens de programação mais influentes, servindo como base para muitas outras linguagens, como C++, Java e Python.

02

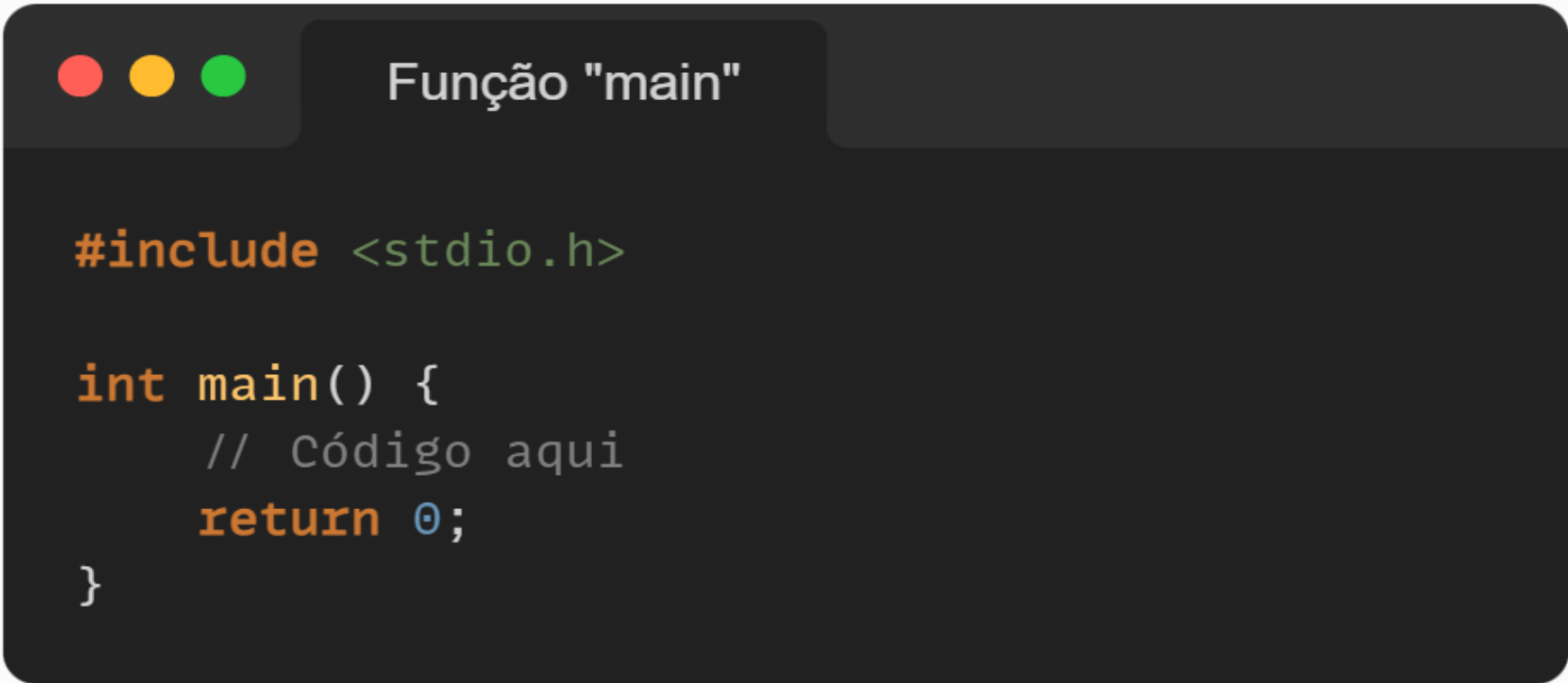


SINTAXE E ESTRUTURA BÁSICA



ESTRUTURA DE UM PROGRAMA EM C

Um programa em C geralmente começa com a inclusão de bibliotecas, seguida pela definição da função *main*:

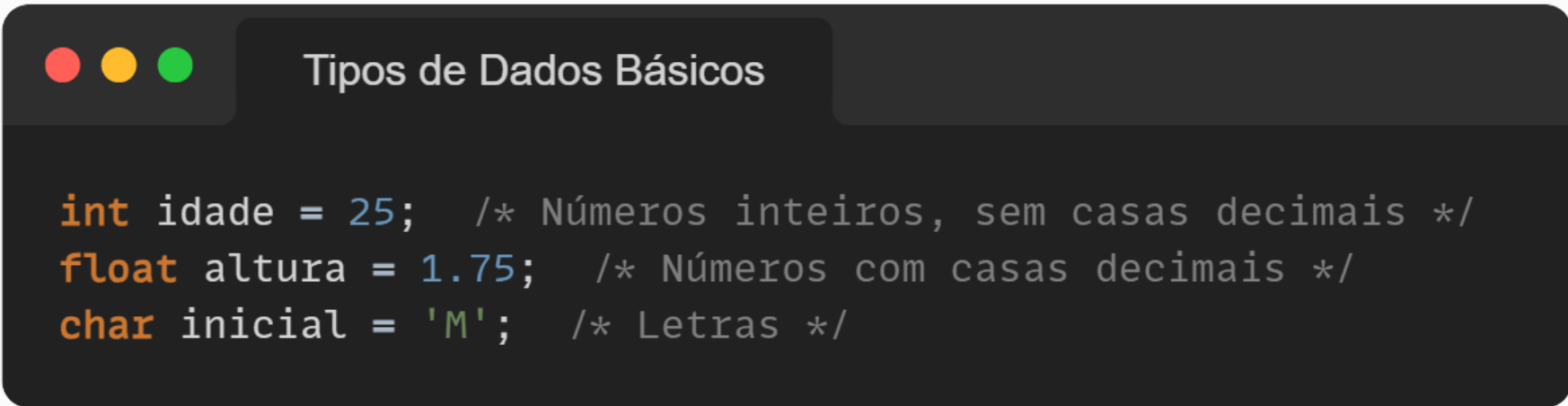


```
#include <stdio.h>

int main() {
    // Código aqui
    return 0;
}
```

TIPOS DE DADOS BÁSICOS

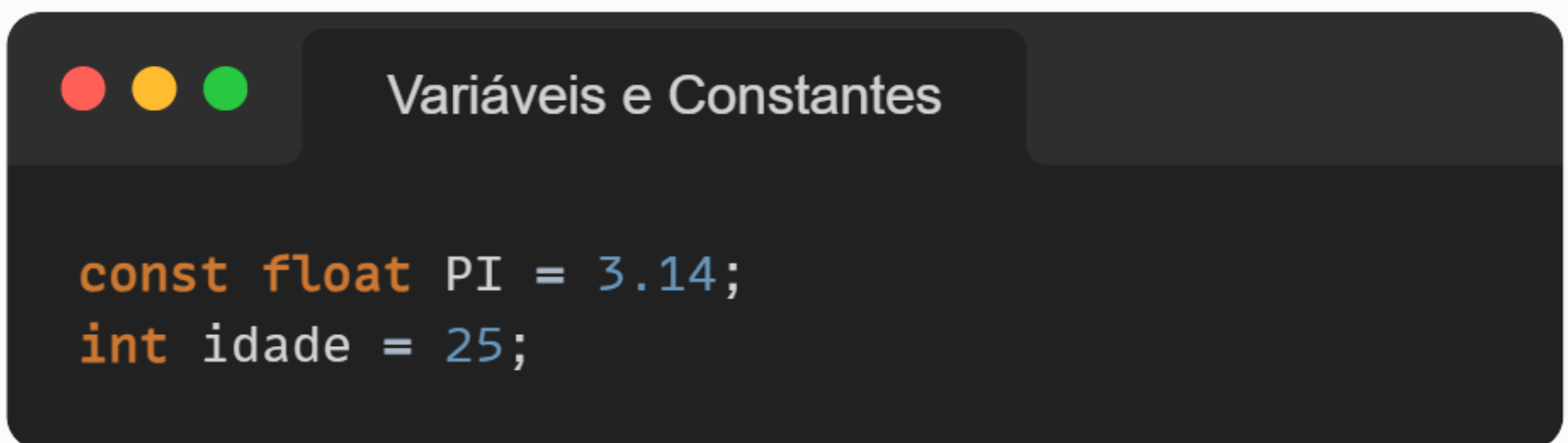
Os tipos de dados básicos incluem *int*, *float*, *double* e *char*:



```
int idade = 25;    /* Números inteiros, sem casas decimais */
float altura = 1.75; /* Números com casas decimais */
char inicial = 'M'; /* Letras */
```

VARIÁVEIS E CONSTANTES

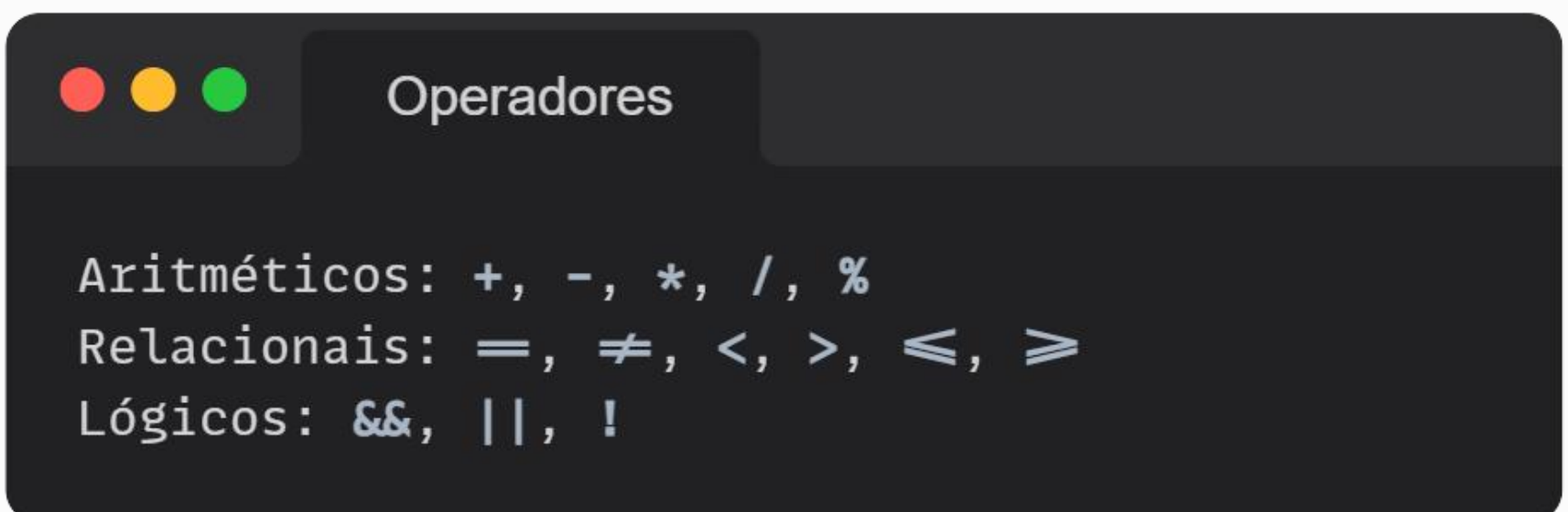
Variáveis armazenam valores que podem mudar, enquanto *constantes* mantêm valores fixos:



```
const float PI = 3.14;  
int idade = 25;
```

OPERADORES

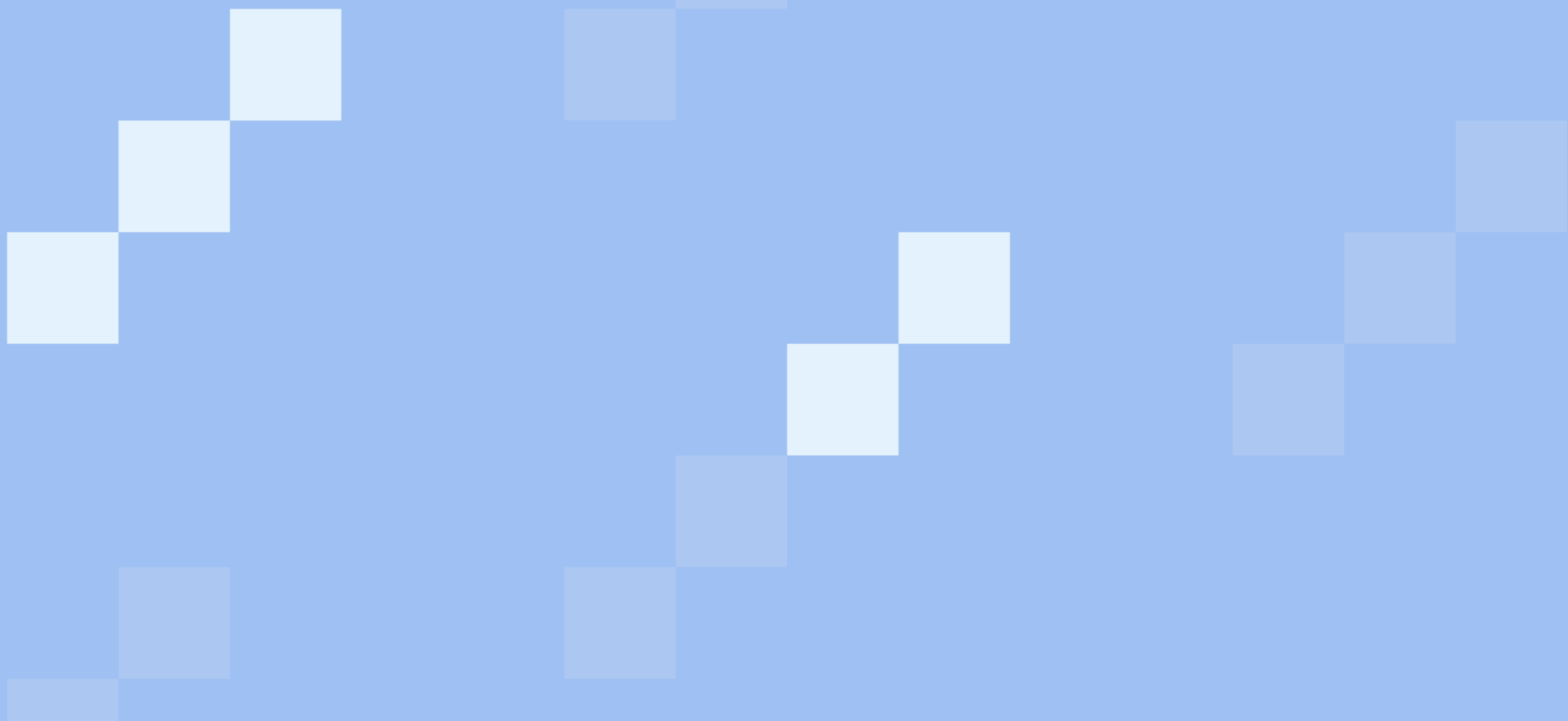
Operadores realizam operações com variáveis:



```
Aritméticos: +, -, *, /, %  
Relacionais: =, !=, <, >, <=, >=  
Lógicos: &&, ||, !
```


03

CONTROLE DE FLUXO

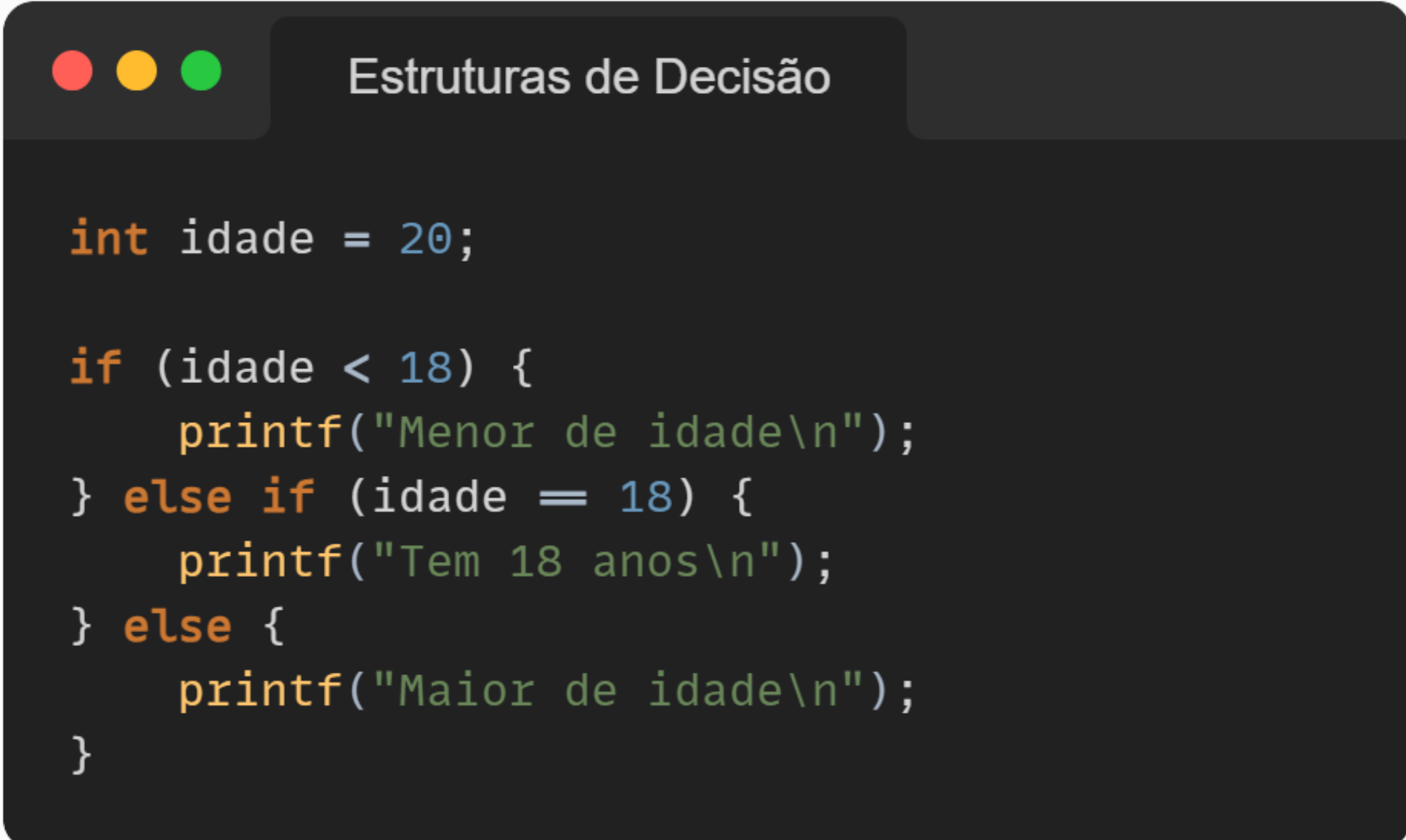


ESTRUTURAS DE DECISÃO

IF, ELSE E ELSE-IF

As estruturas de decisão controlam o fluxo do programa:

Se primeira a condição é atendida, o bloco *if* é executado; Se a segunda condição é atendida, o bloco *else if* é executado; Se nenhuma condição for atendida, o bloco *else* é executado:



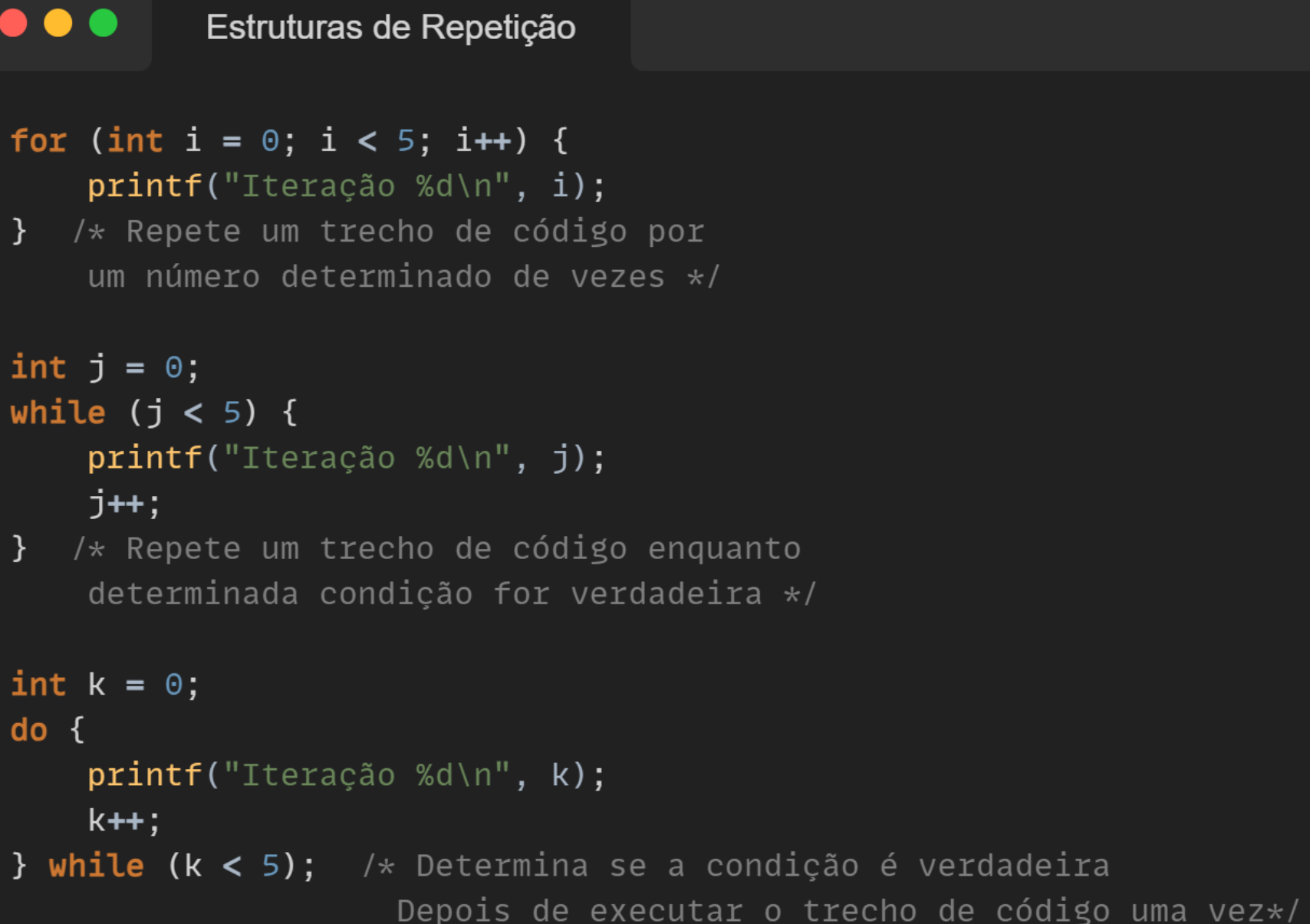
```
int idade = 20;

if (idade < 18) {
    printf("Menor de idade\n");
} else if (idade == 18) {
    printf("Tem 18 anos\n");
} else {
    printf("Maior de idade\n");
}
```

ESTRUTURAS DE REPETIÇÃO

FOR, WHILE, DO-WHILE

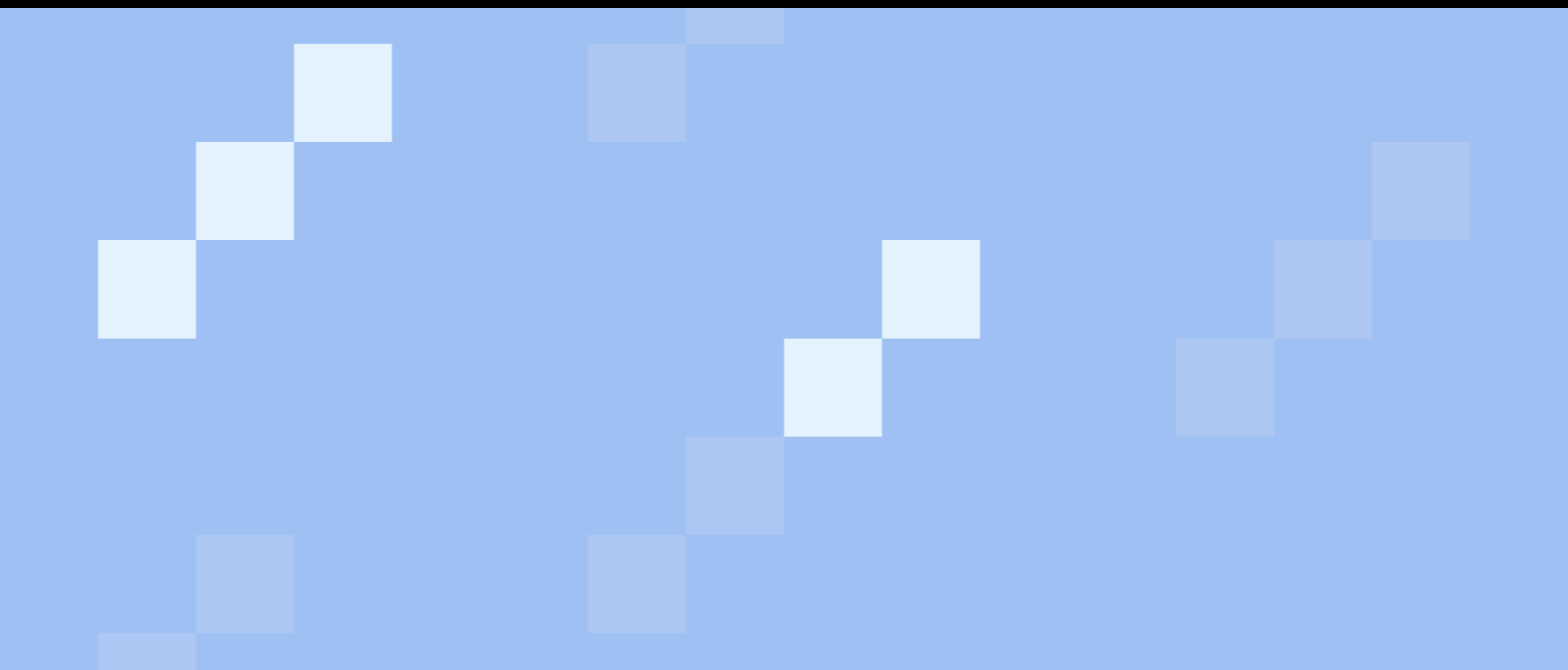
As estruturas de repetição executam blocos de código múltiplas vezes:



```
for (int i = 0; i < 5; i++) {  
    printf("Iteração %d\n", i);  
} /* Repete um trecho de código por  
   um número determinado de vezes */  
  
int j = 0;  
while (j < 5) {  
    printf("Iteração %d\n", j);  
    j++;  
} /* Repete um trecho de código enquanto  
   determinada condição for verdadeira */  
  
int k = 0;  
do {  
    printf("Iteração %d\n", k);  
    k++;  
} while (k < 5); /* Determina se a condição é verdadeira  
                  Depois de executar o trecho de código uma vez*/
```

04

ENTRADA E SAÍDA (I/O)



ENTRADA E SAÍDA PADRÃO

Leia e escreva dados com funções padrão:



```
#include <stdio.h>

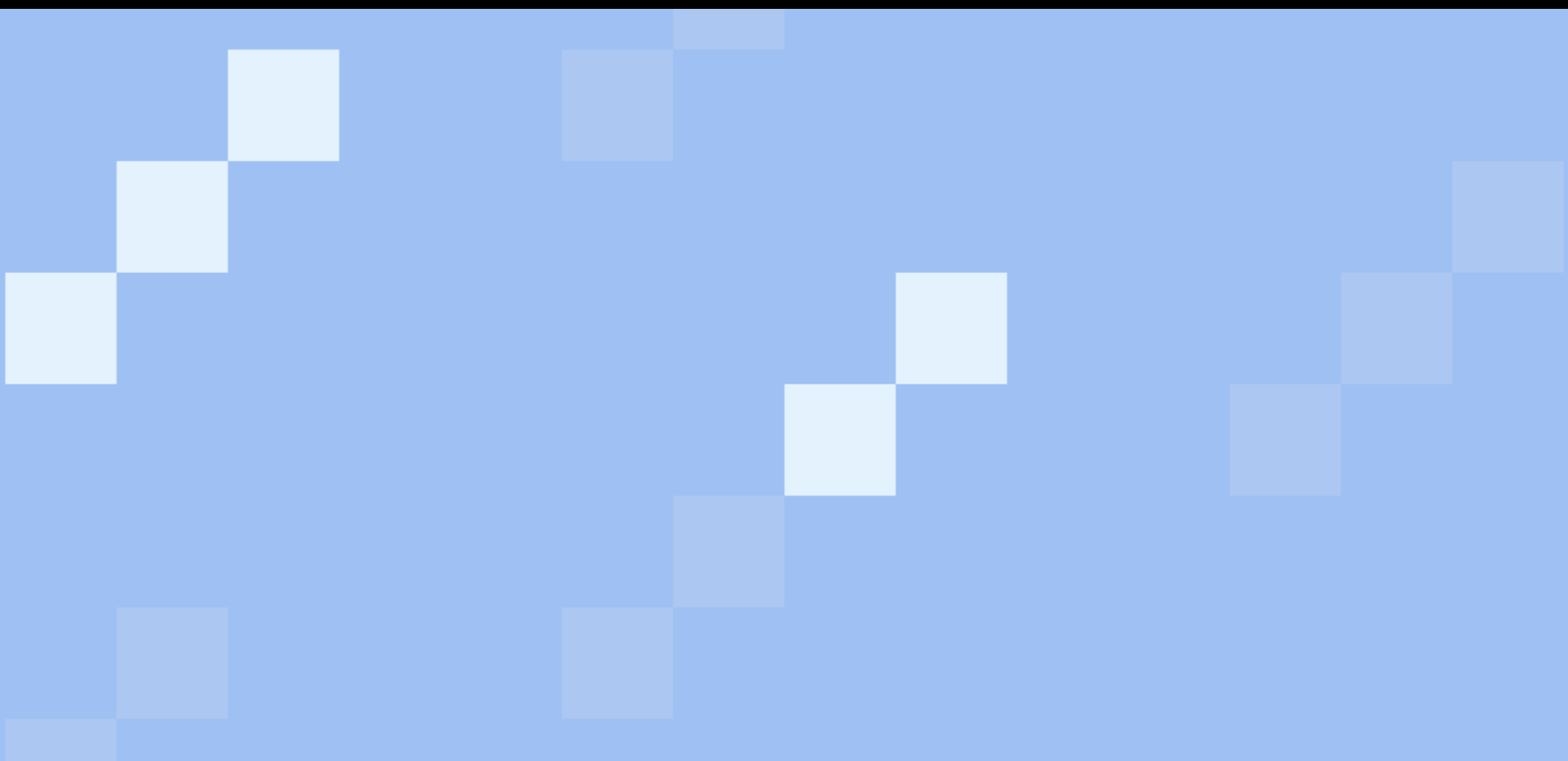
int main() {
    char nome[50];
    printf("Digite seu nome: ");
    scanf("%s", nome);
    printf("Olá, %s!\n", nome);
    return 0;
}
```

A função ***printf*** é utilizada para escrever uma mensagem na tela;

A função ***scanf*** é utilizada para ler os dados de entrada do usuário, e armazená-los num local na memória, indicado por um ***ponteiro***.

05

PRÁTICAS DE PROGRAMAÇÃO

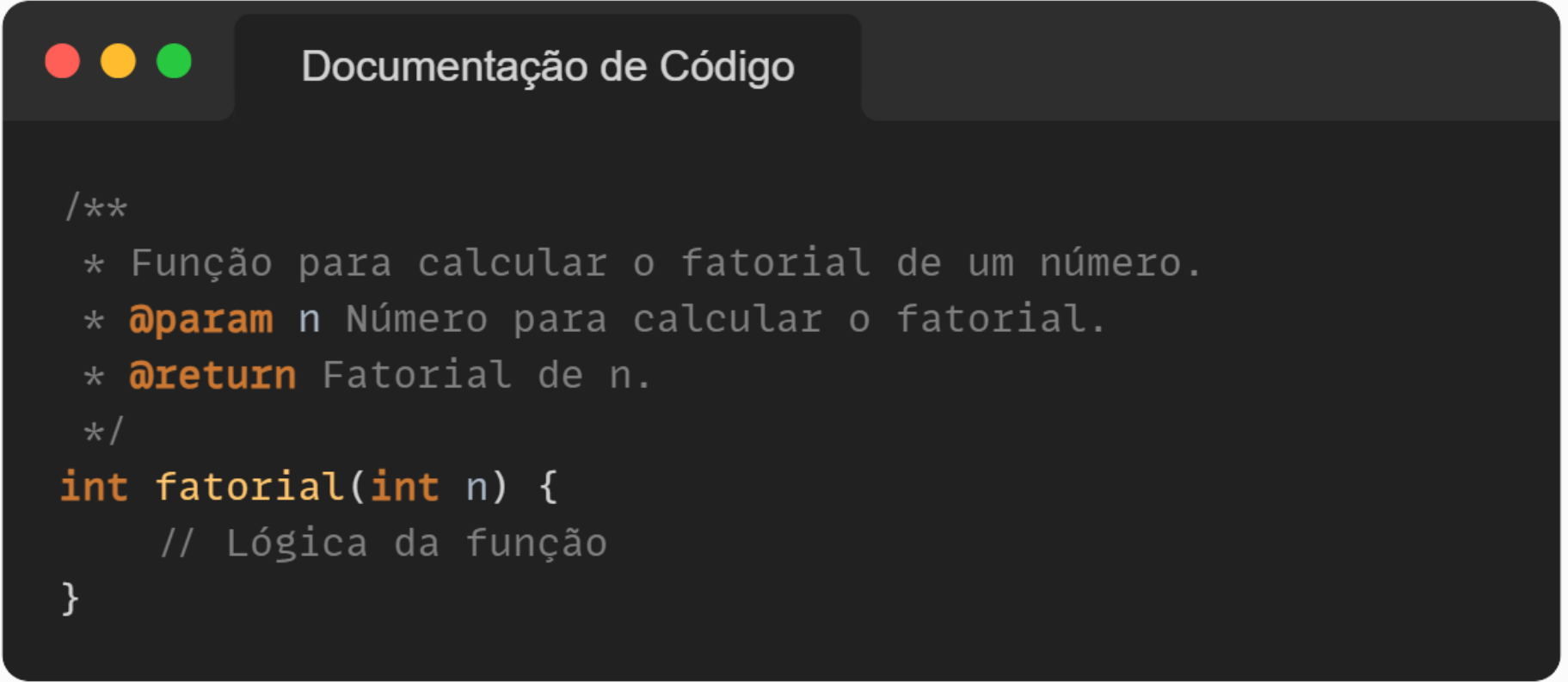


BOAS PRÁTICAS DE CODIFICAÇÃO

- Use nomes de variáveis descritivos.
- Comente o código para explicar lógicas mais complexas.
- Divida o código em funções para melhorar a legibilidade.

DOCUMENTAÇÃO DE CÓDIGO

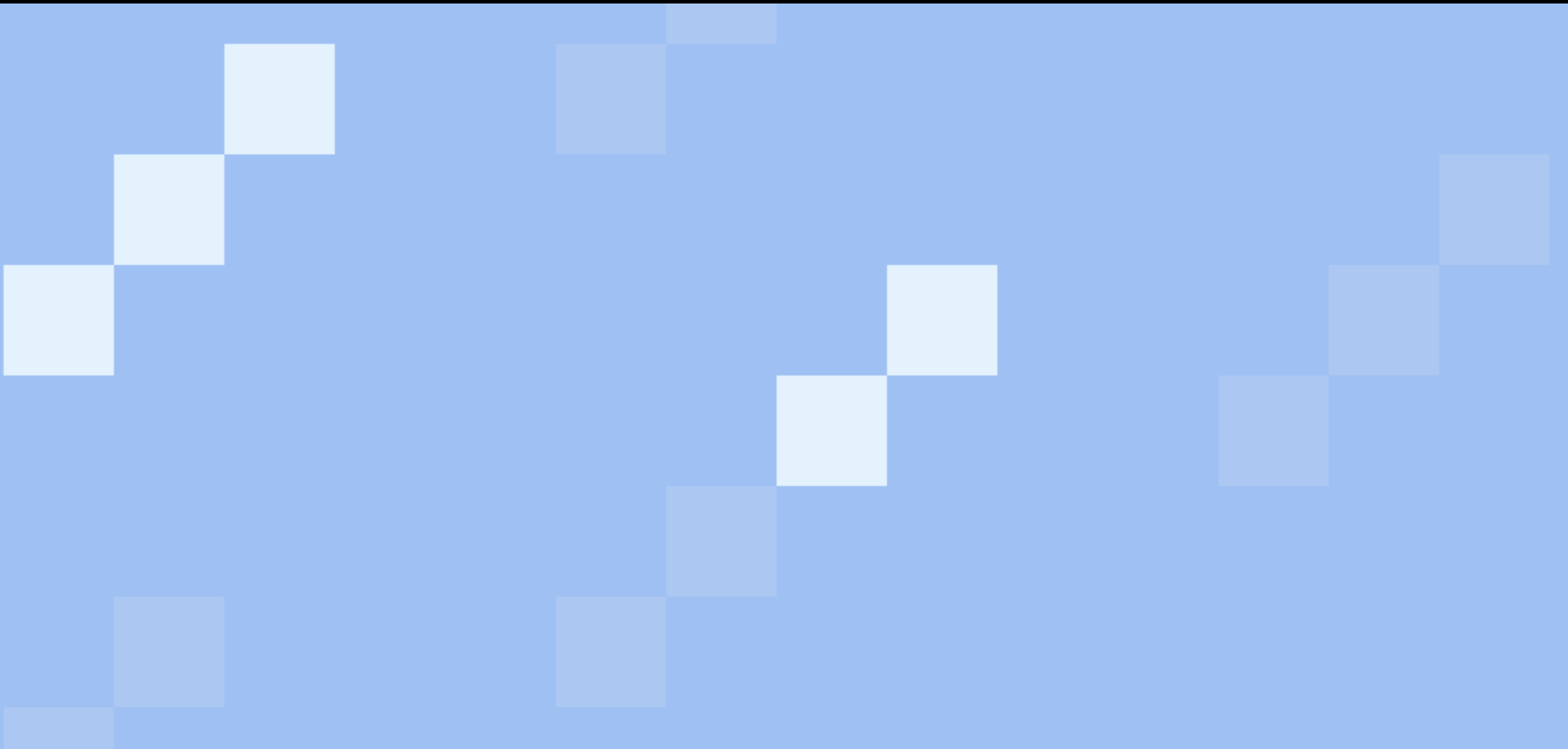
Documente o código com comentários para facilitar a manutenção e colaboração:



```
/**
 * Função para calcular o fatorial de um número.
 * @param n Número para calcular o fatorial.
 * @return Fatorial de n.
 */
int fatorial(int n) {
    // Lógica da função
}
```

10

BÔNUS

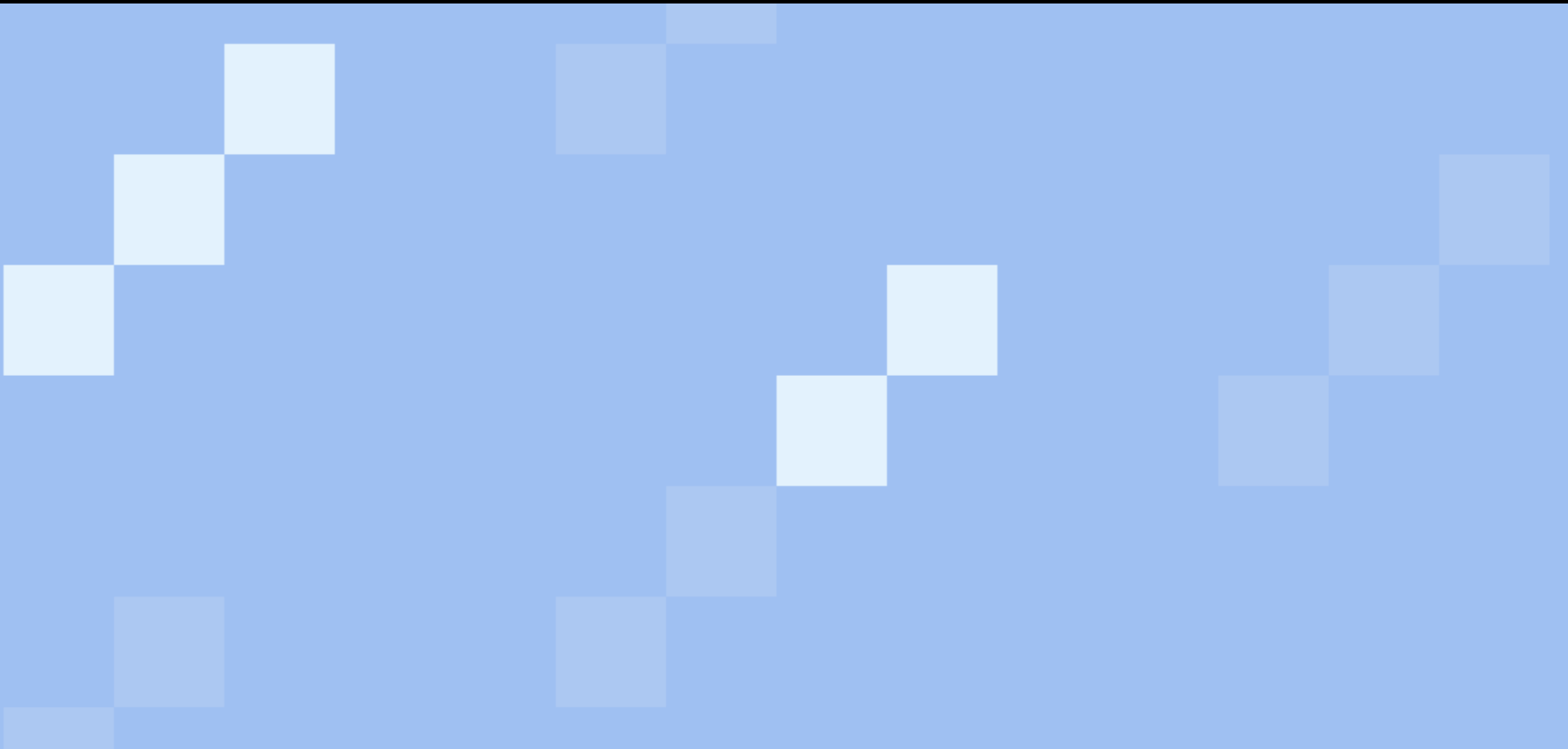


SITES PARA PRATICAR PROGRAMAÇÃO

- [Codecademy](#): Oferece cursos interativos de programação para iniciantes com exercícios práticos e projetos divertidos.
- [HackerRank](#): Plataforma com desafios de programação em várias linguagens, incluindo C. Ideal para praticar e melhorar suas habilidades.
- [LeetCode](#): Site com problemas de programação que variam de fácil a difícil. Excelente para treinar lógica de programação e algoritmos.
- [CodeWars](#): Oferece desafios de programação em formato de katas, que são pequenos exercícios para praticar e aprimorar suas habilidades.
- [Codingame](#): Combina jogos com programação, permitindo que você resolva desafios de programação de maneira divertida e interativa.



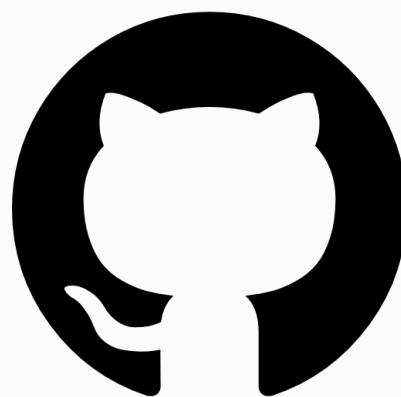
AGRADECIMENTOS



OBRIGADA POR LER ATÉ AQUI!

Este ebook foi gerado por IA, e diagramado por humano. O passo a passo se encontra no meu Github.

Este conteúdo foi gerado para fins didáticos, porém foi feita uma validação humana breve quanto aos conteúdos abordados e à facilidade de assimilação.



AUTORA



MARIA CLARA NASCIMENTO SILVA