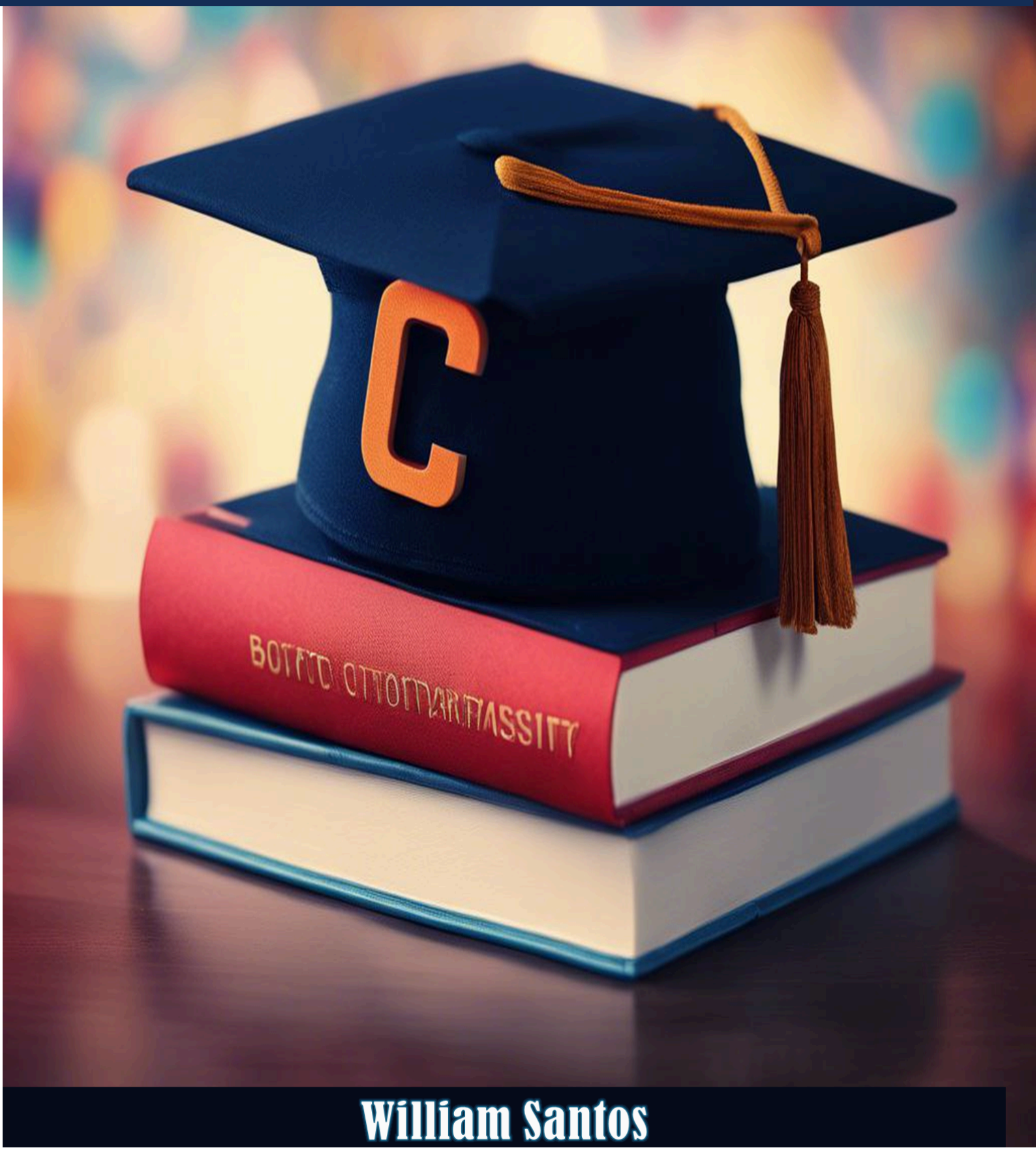
PROGRAMAÇÃO EM C BÁSICA PARA UNIVERSITÁRIOS



William Santos

Capítulo 1: Mergulhando no Mundo C: Do Zero à Sua Primeira Linha de Código!

E aí, futuro programador! Preparado para desvendar os mistérios da linguagem C e criar seus próprios programas? Você veio ao lugar certo!

Este guia foi feito especialmente para você que está começando do zero e quer aprender a programar em C de forma fácil e divertida. Sem enrolação, vamos direto ao ponto, com explicações simples e muitos exemplos práticos para você colocar a mão na massa!

Mas afinal, o que é essa tal de linguagem C?

Imagine que você quer ensinar um amigo a fazer um sanduíche. Para isso, você precisa dar instruções passo a passo, em uma linguagem que ele entenda, certo? Na

programação, a linguagem C é como um conjunto de instruções que o computador entende e usa para realizar tarefas.

Pense no C como um idioma que te permite conversar com o computador e dar ordens a ele! Legal, né?

A linguagem C foi criada na década de 1970 e é super famosa por ser **rápida, eficiente e versátil**. Programas escritos em C podem rodar em vários tipos de computadores, como PCs, Macs e até celulares!

E por que aprender C?

- **Alicerce Forte:** Aprender C é como construir uma base sólida para sua carreira de programador. Você vai entender a fundo como os programas funcionam e como interagem com o computador.
- **Abre Caminhos:** Dominar C te ajuda a aprender outras linguagens de programação mais facilmente, já que muitos conceitos são parecidos. É como aprender inglês para depois mandar bem em espanhol!
- **Mercado Aquecido:** C é usado em muitos sistemas importantes, como jogos, aplicativos e até sistemas

operacionais. Saber programar em C te deixa pronto para diversas oportunidades de trabalho.
- **Rapidez e Poder:** Com C, você pode criar programas super rápidos e eficientes, ideais para jogos e aplicações que precisam de muito desempenho.

Mãos à Obra: Configurando Seu Ambiente de Programação

Antes de começar a codar, precisamos preparar o "escritório" do programador! Para isso, vamos precisar de duas ferramentas:

1. **Compilador:** É como um tradutor que transforma o código que você escreve em linguagem C para a linguagem que o computador entende.
2. **Editor de Texto:** É o bloco de notas do programador, onde você vai escrever seus códigos.

Existem várias opções de compiladores e editores de texto gratuitos e fáceis de usar. Uma sugestão bacana para iniciantes é o **Code::Blocks**, que já vem com tudo pronto para você começar a programar em C.

"Olá, Mundo!": Sua Primeira Conquista na Linguagem C

Pronto para escrever seu primeiro programa em C? É super simples! Digite o código abaixo no Code::Blocks e vamos lá!

```c
#include <stdio.h>

int main() {
  printf("Olá, Mundo!\n");
  return 0;
}
```

Vamos entender o que cada linha faz:

- **#include <stdio.h>:** Essa linha é como um "manual de instruções" que diz ao compilador como lidar com comandos de entrada e saída de dados. É essencial para usarmos o comando printf.
- **int main() { }:** Todo programa em C precisa ter essa linha! É como a "porta de entrada" do seu programa, onde tudo começa.
- **printf("Olá, Mundo!\n");:** Essa linha é a estrela do show! O comando printf exibe uma mensagem na

tela. No nosso caso, a mensagem é "Olá, Mundo!". O \n no final serve para pular uma linha.

- **return 0;:** Essa linha indica que o programa terminou com sucesso. É como dizer "missão cumprida!" ao computador.

Para compilar e executar o código no Code::Blocks, basta clicar no botão verde com um ícone de "play". Se tudo der certo, você verá a mensagem "Olá, Mundo!" na tela!

E aí, curtiu criar seu primeiro programa em C? Agora que você já deu o primeiro passo, prepare-se para o próximo capítulo, onde vamos explorar o mundo das variáveis!

Capítulo 2: Dominando as Variáveis: A Arte de Guardar Informações no Mundo C

No capítulo anterior, você aprendeu a criar um programa em C que exibe uma mensagem na tela. Bacana, né? Mas e se quisermos guardar informações

dentro do programa para usarmos depois? É aí que entram as variáveis!

As Variáveis: Caixas Mágicas para Guardar Informações

Imagine as variáveis como caixas mágicas onde você pode guardar diferentes tipos de informações. Cada caixa tem um nome e um tipo específico, que define o que pode ser guardado dentro dela.

Por exemplo, podemos ter uma caixa chamada idade do tipo inteiro para guardar a idade de uma pessoa, ou uma caixa chamada nome do tipo texto para guardar o nome de alguém.

Tipos de Dados: Escolhendo a Caixa Certa para Cada Informação

A linguagem C oferece vários tipos de dados para você escolher a caixa perfeita para guardar suas informações. Vamos conhecer os principais:

- **int (inteiro):** Para guardar números inteiros, como idade, quantidade de produtos ou número de

alunos.
    - **Exemplo:** int idade = 25;
- **float (ponto flutuante):** Para guardar números com casas decimais, como altura, peso ou preço de um produto.
    - **Exemplo:** float altura = 1.75;
- **char (caractere):** Para guardar um único caractere, como uma letra, um número ou um símbolo.
    - **Exemplo:** char inicial = 'F';

Declarando Variáveis: Criando Caixas e Dando Nomes a Elas

Para criar uma variável em C, você precisa **declarar** ela, informando o tipo de dado e o nome que ela terá. É como escrever na caixa o que ela pode guardar e dar um nome para identificá-la.

A sintaxe para declarar uma variável é:

```
tipo_de_dado nome_da_variavel;
```

Exemplo:

```
int idade; // Declara uma variável chamada 'idade'
do tipo inteiro.
float peso; // Declara uma variável chamada 'peso'
do tipo ponto flutuante.
char sexo; // Declara uma variável chamada 'sexo'
do tipo caractere.
```

Atribuindo Valores: Guardando Informações nas Caixas

Depois de declarar uma variável, você pode guardar informações nela usando o operador de atribuição **=**. É como colocar algo dentro da caixa.

Exemplo:

```
idade = 30;  // Guarda o valor 30 na variável
'idade'.
peso = 72.5; // Guarda o valor 72.5 na variável
'peso'.
sexo = 'M';   // Guarda o caractere 'M' na variável
'sexo'.
```

Você também pode declarar e atribuir um valor à variável na mesma linha:

```
int dia = 10;   // Declara 'dia' como inteiro e
guarda o valor 10.
```

Exemplo Prático: Calculando a Média de Duas Notas

Vamos ver como as variáveis podem ser usadas em um programa real? Imagine que queremos calcular a média de duas notas de um aluno. Podemos usar variáveis para guardar as notas e o resultado da média.

```c
#include <stdio.h>

int main() {
  float base, altura, area;

  printf("Digite a base do retângulo: ");
  scanf("%f", &base);

  printf("Digite a altura do retângulo: ");
  scanf("%f", &altura);

  area = base * altura; // Calcula a área usando o operador de multiplicação.

  printf("A área do retângulo é: %.2f\n", area);

  return 0;
}
```

Pronto para o próximo nível? No próximo capítulo, vamos aprender sobre os operadores em C, que nos permitem realizar cálculos, comparar valores e muito mais!

Capítulo 3: Operadores em C: Dando Ordens ao Computador como um Maestro

No mundo da programação, os operadores são como as batutas de um maestro, comandando o computador a realizar diversas operações com os dados que você fornece. Através deles, você pode realizar cálculos, comparar valores, tomar decisões e muito mais!

Operadores Aritméticos: Fazendo Contas com o Computador

Os operadores aritméticos são os mais básicos e intuitivos, permitindo realizar as quatro operações matemáticas básicas, além do módulo (resto da divisão):

Operador	Descrição	Exemplo	Resultado
`+`	Adição	`5 + 2`	`7`
`-`	Subtração	`10 - 3`	`7`
`*`	Multiplicação	`4 * 6`	`24`
`/`	Divisão	`15 / 3`	`5`
`%`	Módulo (resto da divisão)	`10 % 3`	`1`

Exemplo prático:

Imagine que você quer calcular a área de um retângulo. Para isso, você precisa multiplicar a base pela altura.

```c
#include <stdio.h>

int main() {
  float base, altura, area;

  printf("Digite a base do retângulo: ");
  scanf("%f", &base);

  printf("Digite a altura do retângulo: ");
  scanf("%f", &altura);

  area = base * altura; // Calcula a área usando o operador de multiplicação.

  printf("A área do retângulo é: %.2f\n", area);

  return 0;
}
```

Operadores Relacionais: Comparando Valores para Tomar Decisões

Os operadores relacionais permitem comparar dois valores e determinar se a relação entre eles é verdadeira ou falsa.

Operador	Descrição	Exemplo	Resultado
`==`	Igual a	`5 == 5`	`verdadeiro`
`!=`	Diferente de	`7 != 3`	`verdadeiro`
`>`	Maior que	`8 > 4`	`verdadeiro`
`<`	Menor que	`2 < 9`	`verdadeiro`
`>=`	Maior ou igual a	`6 >= 6`	`verdadeiro`
`<=`	Menor ou igual a	`1 <= 5`	`verdadeiro`

Exemplo prático:

Imagine que você quer verificar se um aluno foi aprovado em uma disciplina. Para isso, você precisa comparar a média do aluno com a nota mínima para aprovação.

```c
#include <stdio.h>

int main() {
  float media;

  printf("Digite a média do aluno: ");
  scanf("%f", &media);

  if (media >= 7.0) { // Verifica se a média é maior ou igual a
7.0
    printf("Aluno aprovado!\n");
  } else {
    printf("Aluno reprovado.\n");
  }

  return 0;
}
```

Operadores Lógicos: Combinando Condições para Decisões Complexas

Os operadores lógicos permitem combinar múltiplas condições para criar decisões mais complexas.

Operador	Descrição	Exemplo	Resultado
`&&`	E (AND)	`(5 > 2) && (3 < 7)`	`verdadeiro`
`||`	OU (OR)	`(10 == 5) || (8 != 1)`	`verdadeiro`
`!`	NÃO (NOT)	`!(4 < 2)`	`verdadeiro`

Exemplo prático:

Imagine que você quer verificar se um cliente pode obter um empréstimo. Para isso, você precisa verificar

se a renda do cliente é maior que um valor mínimo **e** se ele não possui dívidas em atraso.

```c
#include <stdio.h>

int main() {
  float renda;
  int possuiDividas;

  printf("Digite a renda do cliente: ");
  scanf("%f", &renda);

  printf("Possui dívidas em atraso? (1 para sim, 0 para não): ");
  scanf("%d", &possuiDividas);

  if (renda > 1500 && possuiDividas == 0) { // Verifica as duas
condições usando o operador '&&' (E).
    printf("Empréstimo aprovado!\n");
  } else {
    printf("Empréstimo negado.\n");
  }

  return 0;
}
```

Ufa! Que jornada pelos operadores em C! Com este conhecimento, você pode comandar o computador com maestria, criando programas cada vez mais sofisticados e poderosos!

Dedicatória

Este ebook é dedicado a todos os futuros programadores que, assim como eu, estão no início da jornada no mundo da programação. Que este guia sirva de inspiração e suporte para suas primeiras conquistas na linguagem C. Lembre-se: cada linha de código é um passo rumo à realização de seus sonhos!

Com carinho, William Santos