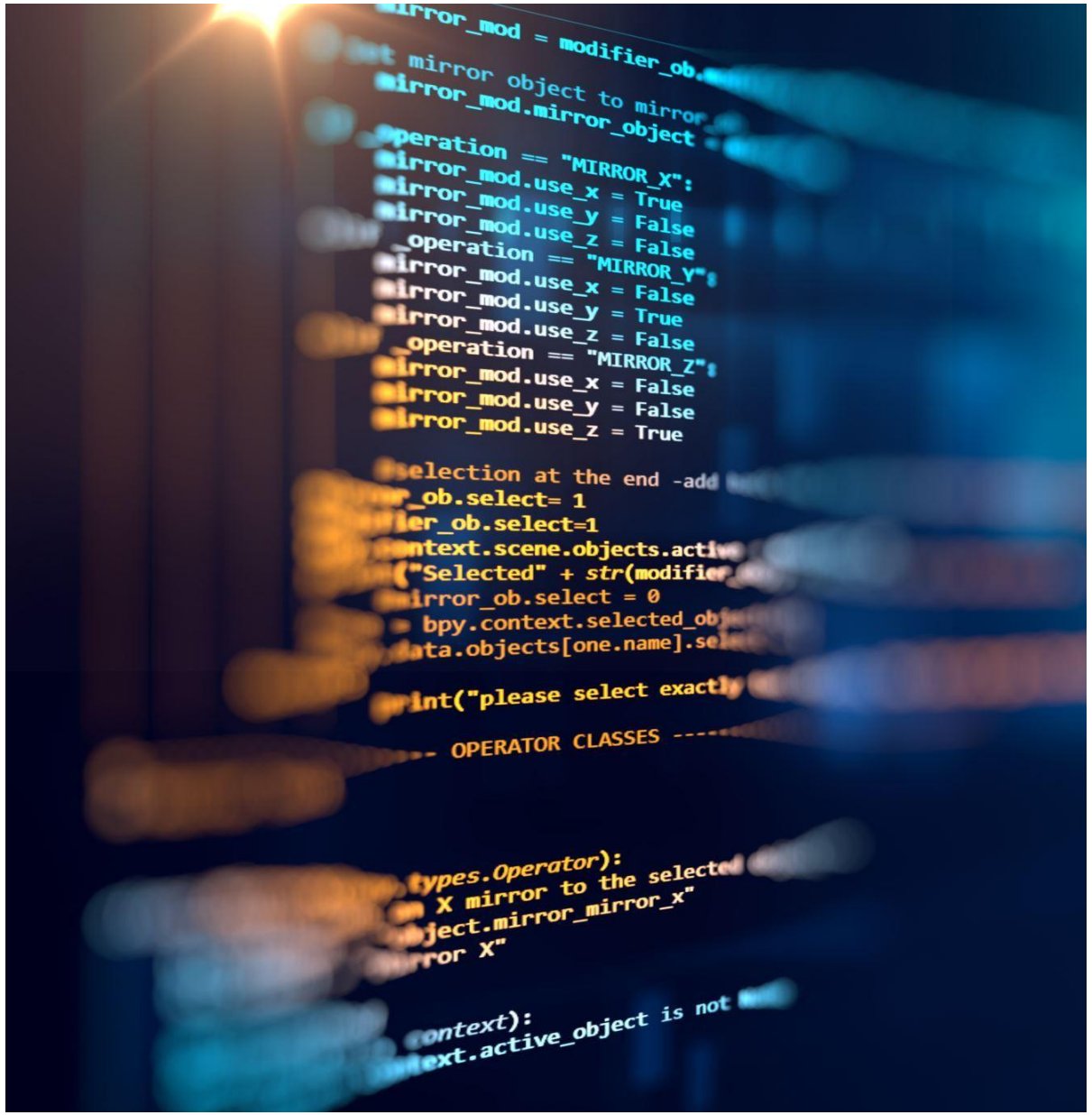
Aula 1 – Introdução à lógica e à linguagem C

Fundamentos de programação e arquitetura de computadores

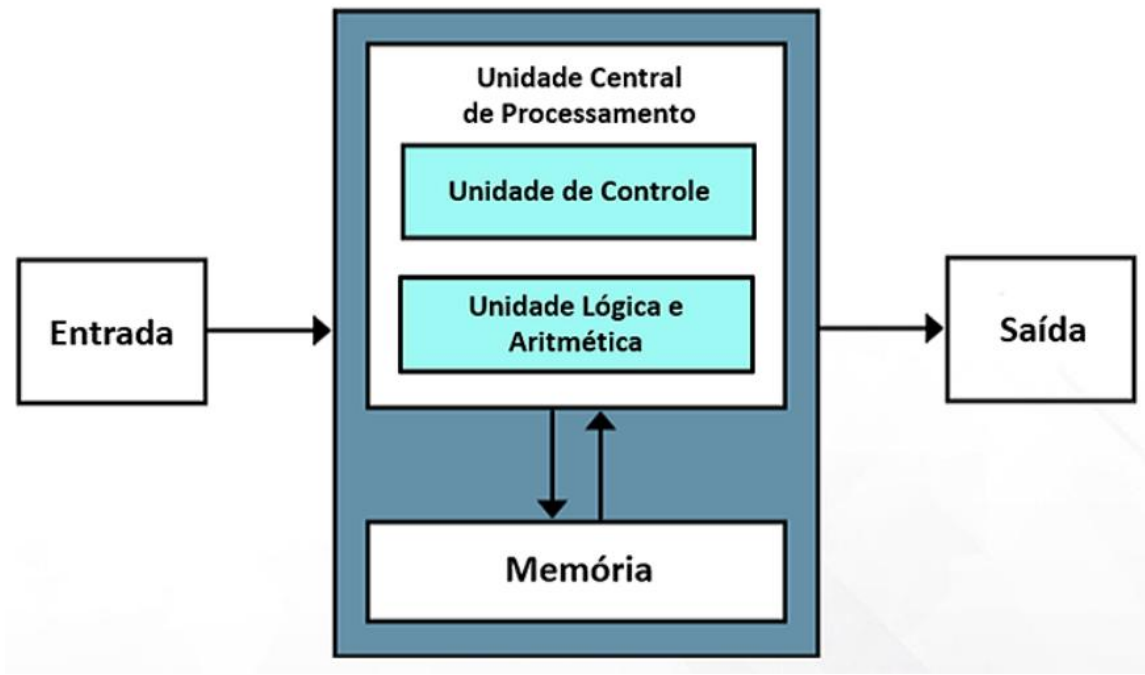


Tópicos da Aula

- Modelo de von Neumann
- O que é lógica de programação?
- Introdução à linguagem C
- Estrutura básica de um programa em C
- Ambiente de desenvolvimento (OnlineGDB)
- Primeiro programa: 'Hello, World!'
- Entrada e saída padrão com printf() e scanf()

Modelo de von Neumann

Arquitetura e componentes



Unidade de Controle

A unidade de controle gerencia o fluxo de dados e instruções dentro do sistema, coordenando as operações das outras unidades.

Unidade Lógica e Aritmética

A unidade lógica e aritmética é responsável por executar operações matemáticas e lógicas, fundamentais para o processamento de dados.

Memória

A memória armazena dados e programas temporariamente, permitindo acesso rápido durante o processamento.

Dispositivos de Entrada e Saída

Os dispositivos de entrada e saída permitem a interação do usuário com o sistema e a troca de informações com o mundo exterior.

O que é lógica de programação?

Definição de lógica de programação

Raciocínio Estruturado

A lógica de programação é baseada em um raciocínio estruturado que organiza soluções para problemas, facilitando a implementação.

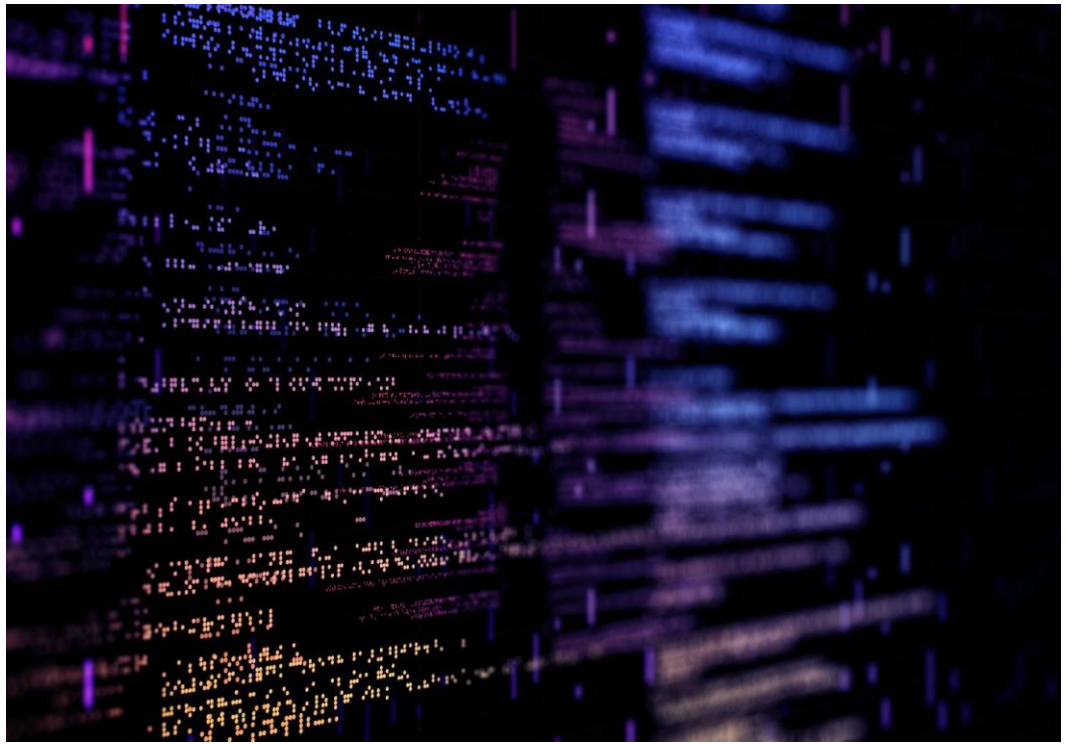
Decomposição de Problemas

Decompor problemas em partes menores é fundamental para entender e resolver questões complexas em programação.

Instruções Claras

Definir instruções claras é essencial para que o computador execute a lógica de programação de maneira eficaz.

Principais conceitos e estruturas



Variáveis em Programação

Variáveis são elementos fundamentais que armazenam dados que podem ser manipulados em um programa. Elas desempenham um papel crucial na construção de qualquer algoritmo.

Condições e Decisões

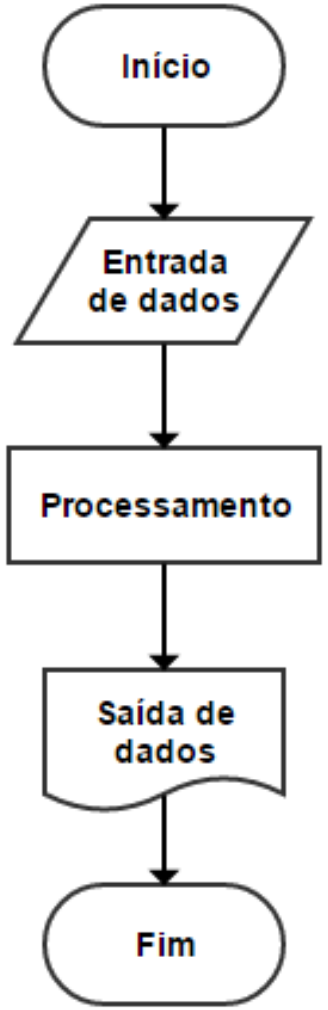
Condições permitem que o programa tome decisões com base em informações específicas. Elas são essenciais para o fluxo de controle em algoritmos.

Loops e Repetição

Loops são estruturas que permitem a repetição de um conjunto de instruções até que uma condição específica seja atendida, economizando tempo e esforço em programação.

Funções e Modularidade

Funções permitem que o código seja dividido em partes menores e reutilizáveis, promovendo a modularidade e facilitando a manutenção do programa.



Exemplos práticos e aplicação

Lógica em Algoritmos

A lógica é fundamental na construção de algoritmos. Vamos analisar como a lógica orienta a resolução de problemas práticos.

Resolução de Problemas Reais

Exemplos práticos demonstram como a lógica pode ser aplicada para resolver problemas do dia a dia de forma eficiente.

Eficiência na Programação

Aplicar a lógica corretamente aumenta a eficiência dos programas, tornando-os mais rápidos e eficazes.

A mensagem do evangelho em codigo

```
#include <stdio.h>


int main() {
                printf("=== ALGORITMO DO EVANGELHO ===\n\n");
                printf("1. Deus criou o ser humano para se relacionar com Ele. (Gênesis 1:26-27)\n");
                printf("2. O ser humano desobedeceu e se separou de Deus pelo pecado. (Romanos 3:23)\n");
                printf("3. Deus, por amor, enviou Seu Filho Jesus Cristo. (João 3:16)\n");
                printf("   -> Jesus veio em carne (João 1:14), viveu sem pecado (Hebreus 4:15),\n");
                 printf("      morreu na cruz pelos nossos pecados (1 Pedro 2:24) e ressuscitou ao terceiro dia. (1 Coríntios 15:3-4)\n");
                printf("4. Quem crê em Jesus é reconciliado com Deus e recebe a vida eterna. (Romanos 5:1, João 5:24)\n");
                printf("5. Um novo céu e uma nova terra aguardam os que permanecem em Cristo. (Apocalipse 21:1-5)\n");
                printf("\nResultado: Creia no evangelho e viva para sempre com Deus! (Romanos 10:9-10)\n");
                return 0;
}
```

Introdução à linguagem C

História e evolução da linguagem C

Criação da Linguagem C

Desenvolvida por Dennis Ritchie na década de 1970, a linguagem C foi projetada para ser poderosa e versátil, adaptando-se a diversas necessidades de programação.

Poder e Flexibilidade

A linguagem C permite o desenvolvimento eficiente de sistemas operacionais e software de alto desempenho, sendo uma escolha popular entre programadores.

Evolução na Programação

A evolução da linguagem C ao longo das décadas solidificou seu lugar na programação moderna, influenciando muitas linguagens que vieram depois.

Principais características e vantagens

Eficiência da Linguagem C

A linguagem C é reconhecida por sua eficiência em termos de uso de recursos, tornando-a adequada para aplicações críticas que exigem alto desempenho.

Portabilidade

A linguagem C é altamente portátil, permitindo que programas sejam executados em diferentes plataformas e sistemas operacionais sem modificações significativas.

Controle de Baixo Nível

O controle de baixo nível sobre o hardware é uma das características mais importantes da linguagem C, permitindo acesso direto à memória e aos recursos do sistema.

Comparação com outras linguagens de programação

Sintaxe da Linguagem C

A linguagem C tem uma sintaxe que é considerada mais complexa em comparação com Python e Java, mas oferece grande controle sobre o hardware.

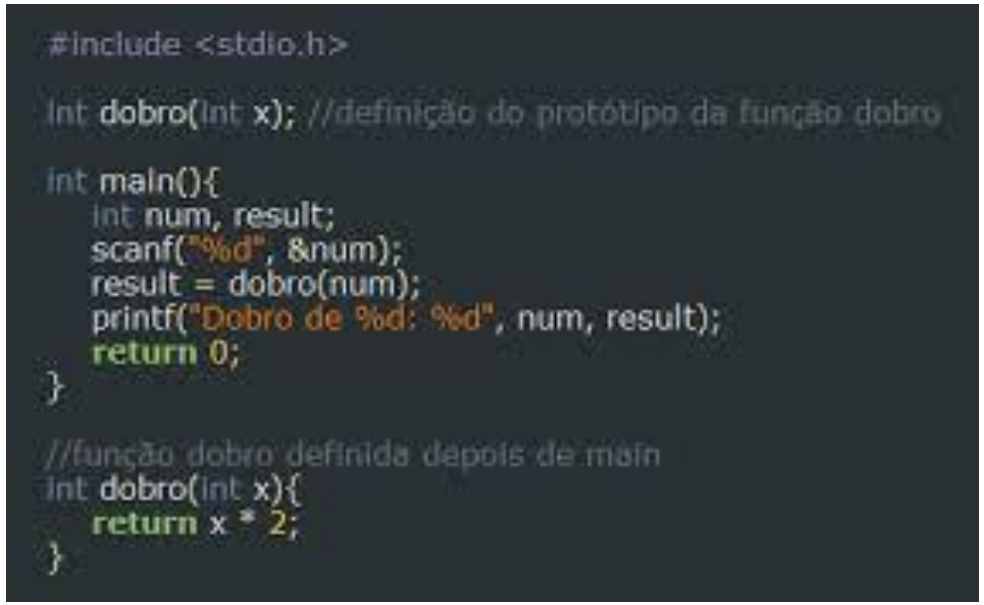
Desempenho e Eficiência

C é conhecida por seu desempenho superior e eficiência em comparação com Python e Java, especialmente em aplicações de sistemas e embutidas.

Casos de Uso

C é amplamente utilizada em desenvolvimento de sistemas, software embarcado e aplicações que exigem alto desempenho.

Estrutura básica de um programa em C



```c
#include <stdio.h>

int dobro(int x); //definição do protótipo da função dobro

int main(){
    int num, result;
    scanf("%d", &num);
    result = dobro(num);
    printf("Dobro de %d: %d", num, result);
    return 0;
}

//função dobro definida depois de main
int dobro(int x){
    return x * 2;
}
```

Elementos fundamentais de um programa em C

Funções em C

As funções são blocos de código que executam tarefas específicas e podem ser reutilizadas ao longo do programa.

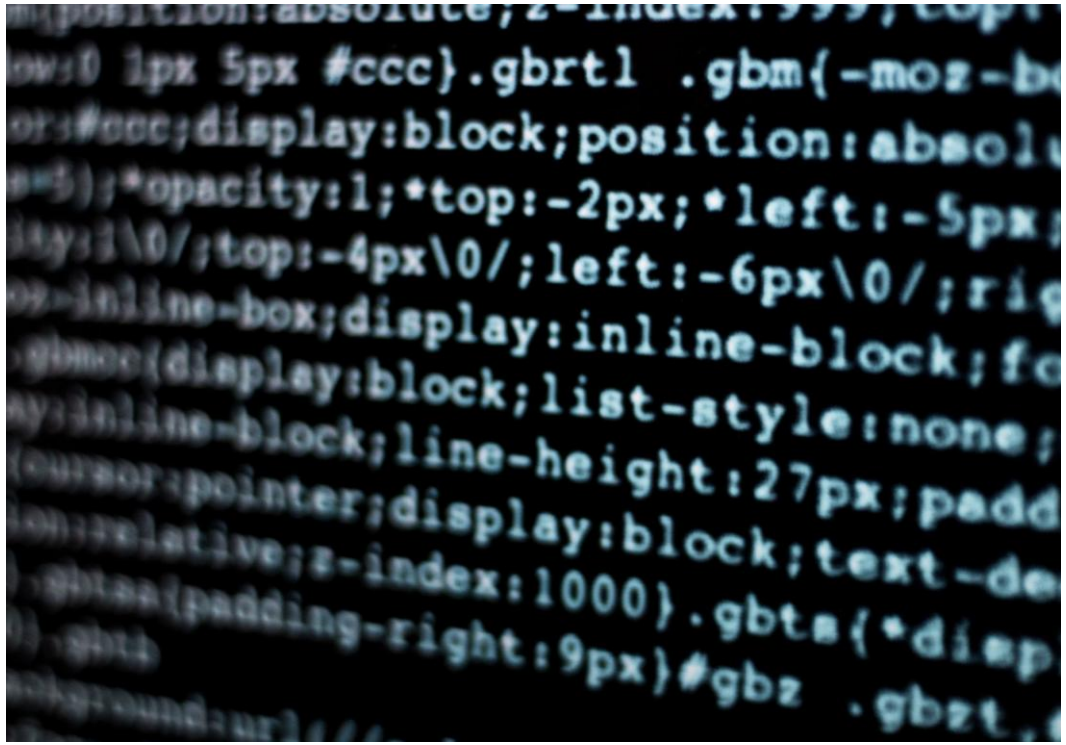
Declarações de Variáveis

As declarações de variáveis definem espaços na memória para armazenar dados que serão utilizados no programa.

Ponto de Entrada 'main'

A função 'main' é o ponto de entrada do programa, onde a execução começa e termina.

Sintaxe básica e organização



Rigor da Sintaxe em C

A linguagem C possui uma sintaxe rigorosa que deve ser seguida para evitar erros de compilação e garantir a funcionalidade do programa.

Definição de Funções

As funções em C são blocos de código que realizam tarefas específicas e são essenciais para a organização do código.

Declarações de Variáveis

As declarações de variáveis são fundamentais em C, permitindo o armazenamento e manipulação de dados dentro do programa.

Importância dos Pontos e Vírgulas

Os pontos e vírgulas são utilizados para terminar as instruções em C, sendo essenciais para a correta execução do código.

Exemplo de código e explicação detalhada

Exemplo de Programa em C

Apresentaremos um exemplo simples de um programa em C, servindo como base para a compreensão da linguagem.

Explicação de Cada Linha

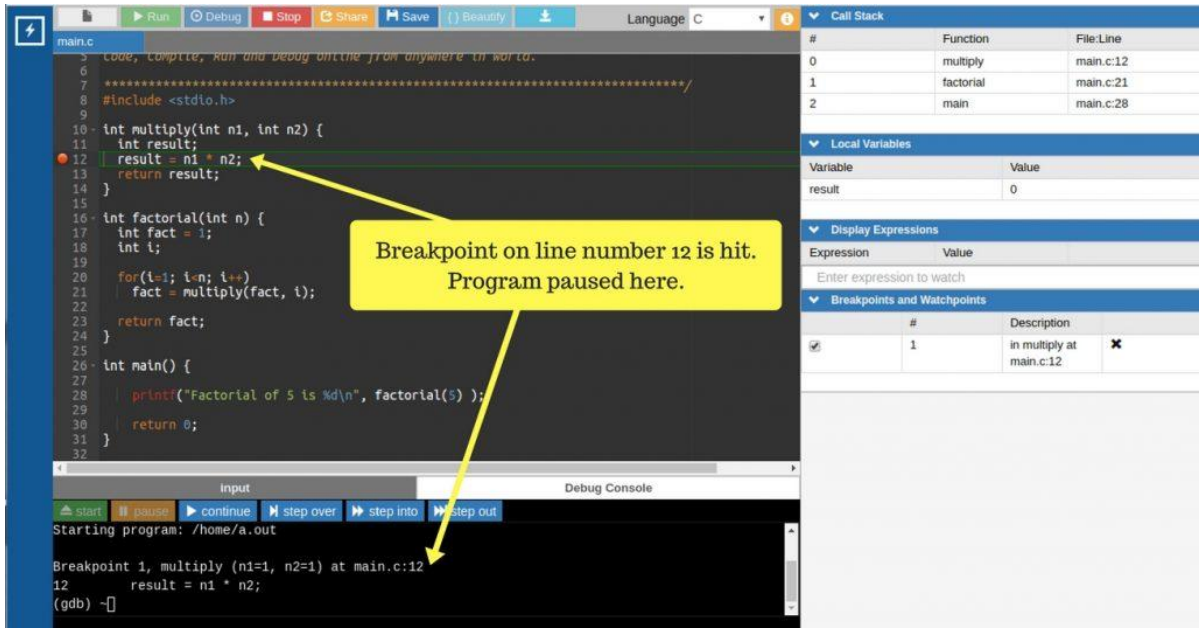
Analisaremos detalhadamente cada linha do código, explicando sua função e importância na lógica do programa.

Estrutura e Lógica

Este exemplo ajudará a solidificar a compreensão da estrutura e da lógica por trás da programação em C.

Ambiente de desenvolvimento (OnlineGDB)

Introdução ao OnlineGDB



Ferramenta Acessível

O OnlineGDB é uma ferramenta acessível que permite que programadores de todos os níveis escrevam e testem código facilmente.

Interface Intuitiva

A interface do OnlineGDB é intuitiva e fácil de navegar, tornando a programação mais simples para iniciantes.

Funcionalidades Poderosas

Oferece funcionalidades poderosas que ajudam programadores experientes a otimizar seu trabalho e a colaborar em projetos.

Configuração e uso do ambiente

Criar um Novo Arquivo

Aprenderemos a criar um novo arquivo no OnlineGDB, que é o primeiro passo para escrever código. Seguir essas etapas é essencial para configurar corretamente o ambiente.

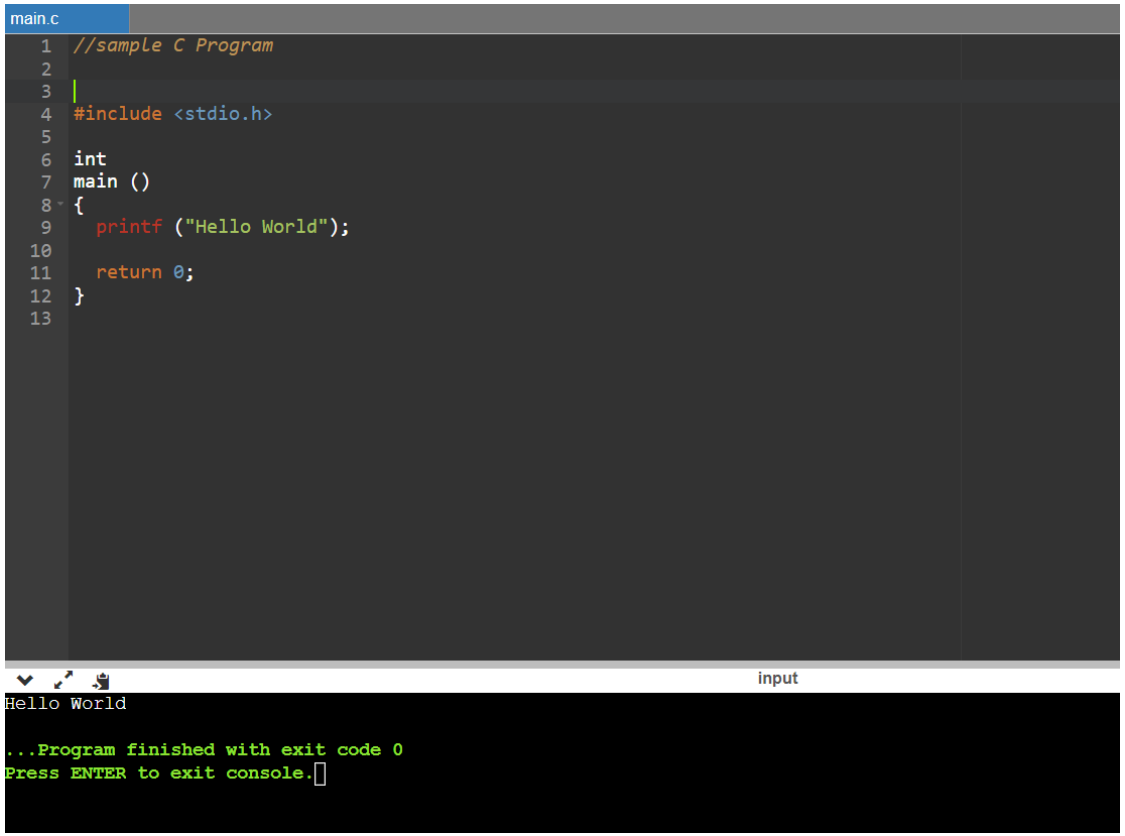
Compilar o Código

Após criar o arquivo, é crucial saber como compilar o código. Isso transforma o código que você escreveu em um programa executável.

Executar Programas

Finalmente, aprenderemos como executar programas no OnlineGDB para ver os resultados do seu código. Isso é fundamental para testar suas ideias e soluções.

Vantagens e recursos disponíveis



Depuração Eficiente

A funcionalidade de depuração do OnlineGDB permite identificar e corrigir erros facilmente, facilitando o aprendizado.

Suporte a Múltiplas Linguagens

O suporte a várias linguagens de programação no OnlineGDB permite a prática e o aprendizado em diferentes contextos de codificação.

Interface Intuitiva

A interface intuitiva do OnlineGDB melhora a experiência do usuário, tornando o desenvolvimento em C mais acessível e agradável.

Primeiro programa: 'Hello, World!'

Escrevendo o código do 'Hello, World!'

Introdução ao 'Hello, World!'

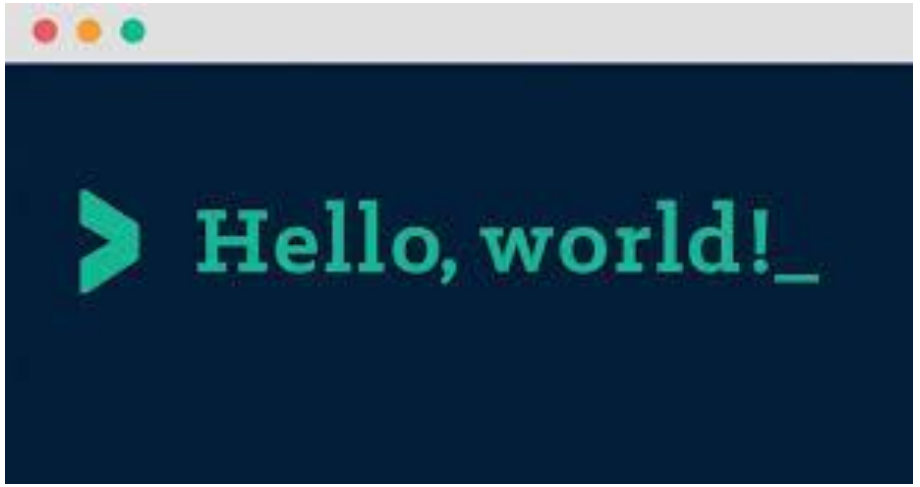
O programa 'Hello, World!' é um exemplo clássico que ajuda novos programadores a entender a estrutura básica de um programa em C.

Sintaxe da Linguagem C

Escrever o código 'Hello, World!' nos ensina a sintaxe básica da linguagem C, incluindo a declaração de funções e instruções de saída.

Importância do Programa

Este exemplo simples é muitas vezes o primeiro programa que novos programadores escrevem, simbolizando o início de sua jornada de programação.



Compilação e execução do programa

Processo de Compilação

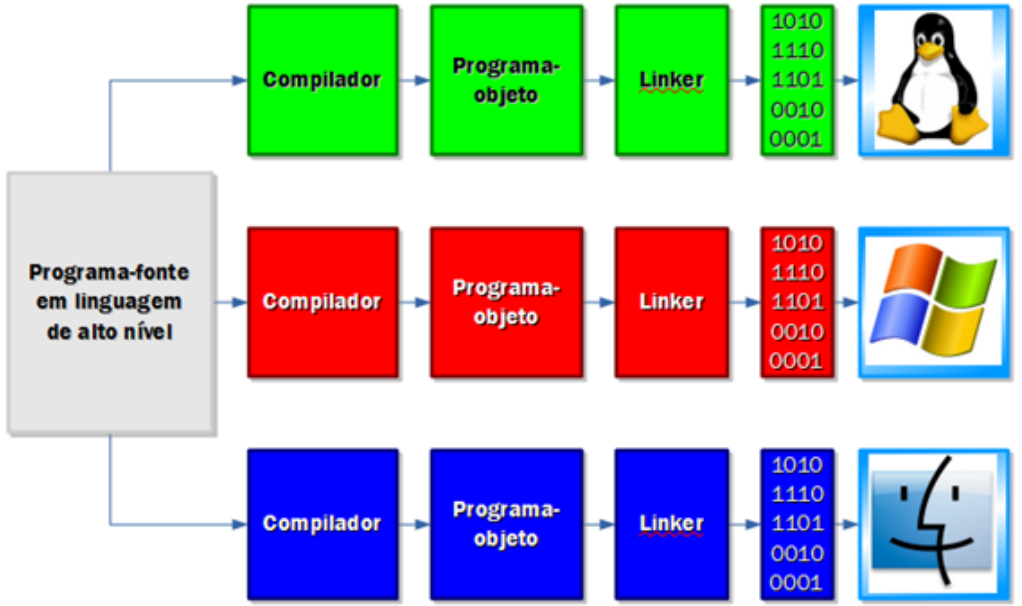
A compilação é o processo de traduzir o código-fonte em um programa executável, fundamental para a programação.

Importância da Compilação

A compilação garante que o código seja analisado e transformado corretamente, ajudando a identificar e corrigir erros antes da execução.

Execução do Programa

Após a compilação, o programa é executado, permitindo que o código realize as funções para as quais foi criado.



Entrada e saída padrão com printf() e scanf()

Uso do printf() para saída de dados

Sintaxe do printf()

A sintaxe da função printf() inclui especificadores de formato que ajudam a controlar como os dados são apresentados na tela.

Formatando a Saída

A formatação de saída permite que você organize os dados de maneira legível, utilizando diferentes especificadores de formato.

Exibindo Diferentes Tipos

O printf() pode ser usado para exibir diferentes tipos de dados, incluindo inteiros, floats e strings, com precisão controlada.

Uso do scanf() para entrada de dados

Função scanf()

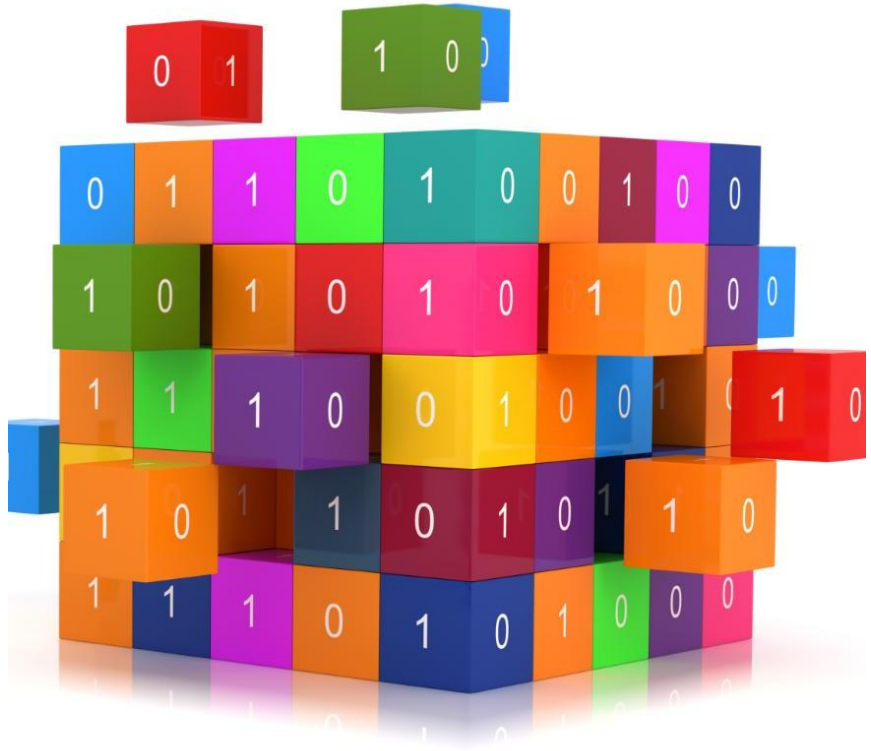
A função scanf() é utilizada em C para permitir a entrada de dados pelo usuário, essencial em muitos programas.

Tipos de Dados

Aprender a lidar com diferentes tipos de dados de entrada, como inteiros, float e strings, é crucial para o uso da scanf().

Uso Correto

Vamos explorar as melhores práticas e erros comuns ao usar a função scanf() para garantir uma entrada de dados eficiente.



Exercícios práticos para consolidação

Importância da Prática

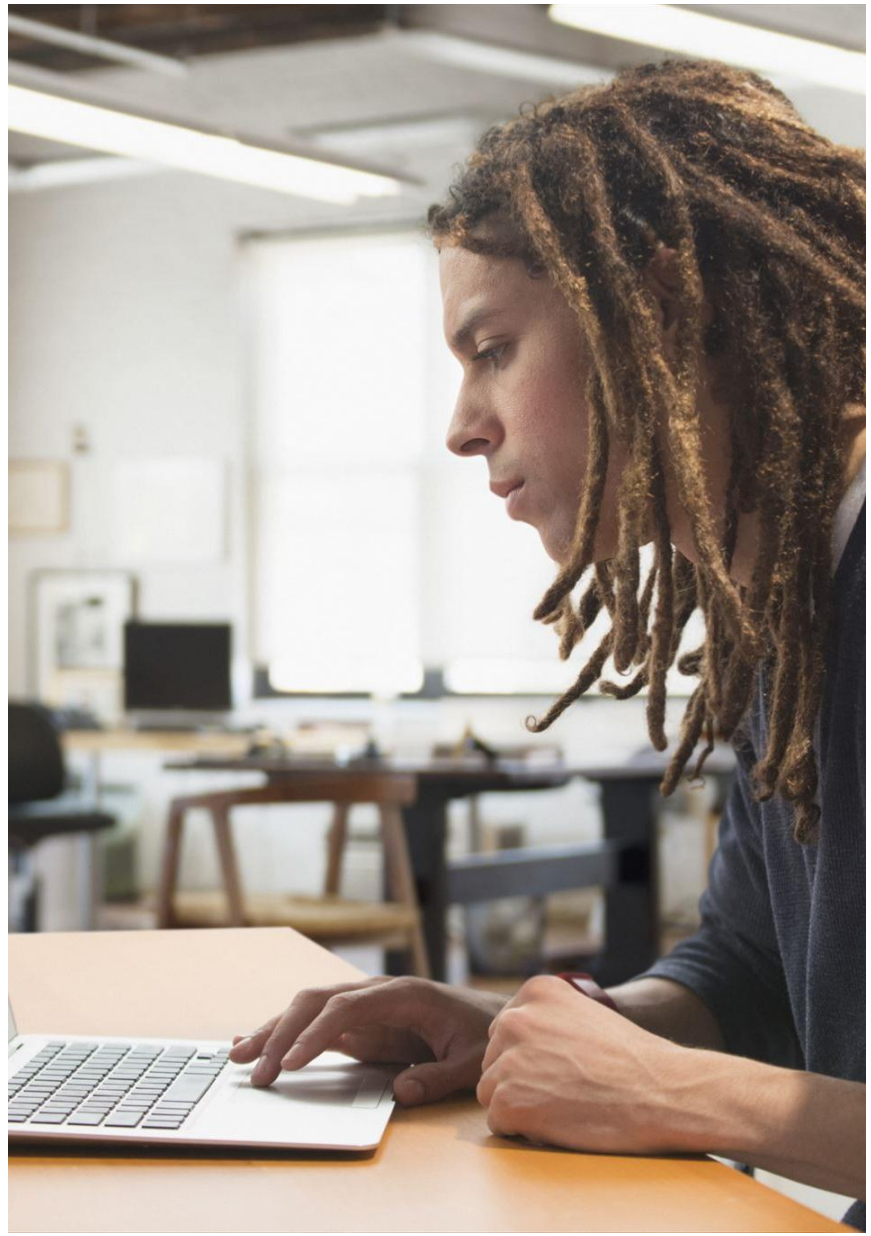
Realizar exercícios práticos é essencial para consolidar o aprendizado e aplicar o conhecimento teórico em situações reais.

Uso de printf()

O printf() é uma função fundamental em C, usada para a saída de dados formatados, e será amplamente utilizada nos exercícios.

Uso de scanf()

O scanf() permite a entrada de dados do usuário, e é crucial para a interação em programas. Esse conceito será aplicado nos exercícios.



Joao 5:24

```
#include <stdio.h>

int main() {
    printf("Etapa 1: Ouvir a Palavra de Jesus (João 5:24)\n");
    printf("Você está ouvindo agora...\n");
    printf("\nEtapa 2: Crer naquele que o enviou (Deus)\n");
    printf("Se você crê em Deus, prossiga...\n");
    printf("\nEtapa 3: Recebe a Vida Eterna\n");
    printf("Você tem a vida eterna!\n");
    printf("\nEtapa 4: Não entra em juízo\n");
    printf("Você foi justificado pela fé.\n");
    printf("\nEtapa 5: Passou da morte para a vida\n");
    printf("Parabéns! Você agora vive em Cristo.\n");
    printf("\nMensagem final: João 5:24 - Quem ouve e crê tem a vida eterna.\n");
    return 0;
}
```

Conclusão

Conceitos de Lógica de Programação

Aprendemos os conceitos fundamentais da lógica de programação, que são essenciais para resolver problemas de maneira estruturada.

Modelo de von Neumann

O Modelo de von Neumann é a base da arquitetura de computadores e fundamental para entender a execução de programas.

Estrutura de um Programa em C

Entender a estrutura de um programa em C é crucial, pois nos ajuda a escrever código eficaz e organizado.

Ambiente de Desenvolvimento

Aprendemos a usar um ambiente de desenvolvimento, que facilita a escrita, teste e depuração de código.