Daniel Da Silva - Análise e Desenvolvimento de Sistemas - Uninove - 2025©

Estrutura Básica de um Programa em C

Um programa em C é composto por algumas partes fundamentais. Vou te explicar isso com um exemplo simples:

```
#include <stdio.h>  // Biblioteca padrão de entrada e saída

// Função principal do programa
int main() {
    // Instruções do programa
    printf("Olá, Mundo!\n");  // Imprime "Olá, Mundo!" na tela
    return 0;  // Retorna 0, indicando que o programa foi executado com sucesso
}
```

Explicando cada parte:

1. **`#include <stdio.h>`:**
    - Esta linha inclui a biblioteca padrão de entrada e saída (stdio), que contém funções como `printf()` (para imprimir algo na tela) e `scanf()` (para ler entradas do usuário).
    - O `#include` é um pré-processador, ou seja, ele traz esse conteúdo externo para dentro do seu código.
2. **`int main() { ... }`:**
    - Toda aplicação em C começa na função `main()`. Ela é a função principal e obrigatória, e onde a execução do programa começa.
    - O `int` antes de `main` significa que a função vai retornar um valor do tipo inteiro (neste caso, um código de status, geralmente 0 para indicar que o programa terminou com sucesso).
3. **`printf("Olá, Mundo!\n");`:**
    - O `printf` é uma função usada para imprimir texto na tela. O texto que você quer imprimir vai entre aspas.

- O `\n` no final é um caractere especial que cria uma nova linha, ou seja, vai pular para a linha seguinte após imprimir a mensagem.

4. **`return 0;`:**

   - O `return` termina a execução da função `main` e retorna um valor. O valor 0 é um código de sucesso, indicando que o programa terminou sem erros.

Componentes e Conceitos Importantes

Além da estrutura básica, é importante entender alguns outros conceitos em C:

1. **Variáveis:**

   - Em C, você pode armazenar dados em variáveis. Você precisa declarar o tipo da variável (como `int`, `float`, `char`, etc.).

Exemplo de declaração e uso de uma variável:

```
 int idade = 25;
printf("Sua idade é %d\n", idade);
```

   -

2. **Operadores:**

   - C possui diversos operadores para realizar operações matemáticas, lógicas, comparações, etc.

   - **Operadores matemáticos**: `+`, `-`, `*`, `/`, `%` (módulo)

   - **Operadores lógicos**: `&&` (E lógico), `||` (OU lógico)

   - **Operadores de comparação**: `==` (igual a), `!=` (diferente de), `>` (maior que), `<` (menor que)

3. **Controle de Fluxo:**

Condicionais (se e senão):

```
 if (idade >= 18) {
    printf("Você é maior de idade.\n");
} else {
    printf("Você é menor de idade.\n");
}
```

-
- **Laços de repetição**:
    - `for`: Usado quando sabemos o número de repetições.
    - `while`: Usado quando a condição é verificada antes de cada repetição.
    - `do while`: A condição é verificada após a execução do bloco, garantindo ao menos uma execução.

4. **Funções:**
    - Funções são blocos de código que realizam uma tarefa específica e podem ser chamadas dentro do programa. A principal função é a `main()`, mas você pode criar outras.

Exemplo de uma função:

```
 int soma(int a, int b) {
    return a + b;
}
```

📌 Especificadores comuns em C:

Código	Significado	Exemplo de uso
`%d`	Inteiro decimal (int)	`printf("Idade: %d", idade);`
`%f`	Número de ponto flutuante (float/double)	`printf("Altura: %.2f", altura);`
`%c`	Caractere único (char)	`printf("Letra: %c", letra);`
`%s`	String (vetor de caracteres)	`printf("Nome: %s", nome);`

`%u`	Inteiro sem sinal (unsigned int)	`printf("Valor: %u", valor);`
`%x`	Inteiro em hexadecimal (minúsculo)	`printf("Hex: %x", num);`
`%p`	Ponteiro (endereço de memória)	`printf("Endereço: %p", ptr);`

Exemplo Completo com Funções e Controle de Fluxo

Aqui vai um exemplo mais complexo para ilustrar como você pode juntar essas ideias:

```
#include <stdio.h>

// Função que soma dois números
int soma(int a, int b) {
    return a + b;
}

// Função principal
int main() {
    int num1, num2, resultado;

    // Solicitando a entrada do usuário
    printf("Digite o primeiro número: ");
    scanf("%d", &num1);
    printf("Digite o segundo número: ");
    scanf("%d", &num2);

    // Chamando a função soma
    resultado = soma(num1, num2);

    // Mostrando o resultado
    printf("A soma de %d e %d é %d\n", num1, num2, resultado);

    return 0;
}
```

O que este código faz?

1. Ele pede para o usuário digitar dois números.

2. A função `soma()` é chamada com esses dois números como argumentos.

3. O resultado da soma é então impresso na tela.

Conclusão

Em resumo, a linguagem C é poderosa e permite um controle muito preciso do que acontece no computador. A estrutura básica inclui funções, variáveis, operadores e controle de fluxo. À medida que você vai avançando, aprenderá conceitos mais complexos como ponteiros, alocação dinâmica de memória, estruturas e manipulação de arquivos.