Material de Leitura - Semana 2

Variáveis, Tipos e Operadores

Introdução a Técnicas de Programação (2025.2)

Objetivos de Aprendizagem

Ao final desta semana, você será capaz de:

- Compreender os conceitos fundamentais de variáveis e tipos de dados em C
- Declarar e inicializar variáveis de diferentes tipos
- Utilizar operadores aritméticos, relacionais e lógicos
- Implementar entrada e saída de dados básica
- Aplicar boas práticas de nomenclatura e estruturação de código

---

1. Introdução à Linguagem C

1.1 História e Características

A linguagem C foi desenvolvida por Dennis Ritchie no início dos anos 1970 nos laboratórios Bell. É uma linguagem de programação de propósito geral, caracterizada por:

- **Portabilidade**: Programas em C podem ser compilados em diferentes sistemas operacionais
- **Eficiência**: Oferece controle direto sobre recursos de hardware
- **Simplicidade**: Conjunto relativamente pequeno de palavras-chave
- **Flexibilidade**: Permite tanto programação estruturada quanto de baixo nível

1.2 Estrutura Básica de um Programa em C

Todo programa em C segue uma estrutura fundamental:

```c
#include <stdio.h>  // Diretiva de preprocessamento
#include <math.h>   // Biblioteca para funções matemáticas

int main() {        // Função principal
    // Corpo do programa
    printf("Olá, mundo!\n");

    return 0;       // Retorno da função main
}
```

Componentes principais:

- **Diretivas de preprocessamento**: Instruções que começam com `#include`
- **Função main()**: Ponto de entrada do programa

- **Instruções**: Comandos que terminam com ponto e vírgula (`;`)
- **Comentários**: Texto explicativo usando `//` ou `/* */`

1.3 Processo de Compilação

O processo de transformação do código-fonte em programa executável ocorre em etapas:

1. **Preprocessamento**: Processamento das diretivas `#include` e `#define`
2. **Compilação**: Conversão do código em linguagem assembly
3. **Montagem**: Conversão para código objeto
4. **Ligação**: Combinação com bibliotecas para gerar o executável

Comando básico de compilação:

```
gcc programa.c -o programa
```

---

2. Variáveis e Tipos de Dados

2.1 Conceito de Variável

Uma **variável** é um espaço nomeado na memória do computador que armazena um valor que pode ser modificado durante a execução do programa. Cada variável possui:

- **Identificador**: Nome único para referenciá-la
- **Tipo**: Determina que tipo de dados pode armazenar
- **Endereço**: Localização na memória
- **Valor**: Dados armazenados

2.2 Declaração e Inicialização

Em C, todas as variáveis devem ser declaradas antes de serem utilizadas:

```
int idade;              // Declaração simples
float altura = 1.75;    // Declaração com inicialização
int a, b = 10, c;       // Múltiplas declarações
```

Boa prática: Sempre inicialize suas variáveis para evitar comportamentos indefinidos:

```
int contador = 0;       // ✓ Correto
float taxa = 0.0;       // ✓ Correto
char letra = 'A';       // ✓ Correto
```

2.3 Tipos Primitivos em C

2.3.1 Tipos Inteiros

Tipo	Tamanho	Faixa de Valores	Especificador
`char`	1 byte	-128 a 127	`%c`
`int`	4 bytes	-2.147.483.648 a 2.147.483.647	`%d`
`long long`	8 bytes	±9.223.372.036.854.775.807	`%lld`

Variantes unsigned (apenas valores não negativos):

```
unsigned int positivo = 42;     // 0 a 4.294.967.295
unsigned char codigo = 255;     // 0 a 255
```

2.3.2 Tipos de Ponto Flutuante

Tipo	Tamanho	Precisão	Especificador
`float`	4 bytes	~7 dígitos	`%f`
`double`	8 bytes	~15 dígitos	`%lf`

```
float pi = 3.14159f;           // Precisão simples
double precisao = 3.141592653589793; // Precisão dupla
```

2.3.3 Tipo Caractere

O tipo `char` armazena um único caractere usando a tabela ASCII:

```
char letra = 'A';              // Valor ASCII: 65
char digito = '5';             // Valor ASCII: 53 (não é o número 5!)
char especial = '\n';          // Caractere de nova linha
```

Sequências de escape comuns:

- `\n` - Nova linha
- `\t` - Tabulação
- `\"` - Aspas duplas
- `\'` - Aspas simples
- `\\` - Barra invertida

2.4 Regras para Identificadores

Regras obrigatórias:

- Deve começar com letra (a-z, A-Z) ou underscore (_)
- Pode conter letras, dígitos (0-9) e underscores

- Não pode ser uma palavra-chave da linguagem
- É sensível a maiúsculas e minúsculas (case-sensitive)

Exemplos válidos:

```
int idade;
float _temperatura;
char primeiraLetra;
int contador2024;
```

Exemplos inválidos:

```
int 2contador;      // ✗ Começa com dígito
float for;          // ✗ Palavra-chave reservada
char minha-var;     // ✗ Contém hífen
int número;         // ✗ Contém caractere especial
```

Convenções recomendadas:

- Use nomes descritivos: `salario` em vez de `s`
- Para nomes compostos, use camelCase: `salarioAnual`
- Constantes em maiúsculas: `PI`, `MAX_VALOR`
- Evite identificadores muito longos

---

3. Entrada e Saída de Dados

3.1 Saída de Dados com printf()

A função `printf()` é usada para exibir dados na tela:

```
#include <stdio.h>

int main() {
    printf("Mensagem simples\n");
    printf("Número inteiro: %d\n", 42);
    printf("Número real: %.2f\n", 3.14159);
    return 0;
}
```

Especificadores de Formato

Especificador	Tipo	Exemplo
`%d` ou `%i`	int	`printf("%d", 123)`

Especificador	Tipo	Exemplo
`%f`	float/double	`printf("%.2f", 3.14)`
`%c`	char	`printf("%c", 'A')`
`%s`	string	`printf("%s", "texto")`
`%x`	hexadecimal	`printf("%x", 255)`
`%p`	ponteiro	`printf("%p", &variavel)`

Formatação de Precisão

```c
float numero = 3.141592653;

printf("%.2f\n", numero);    // 3.14
printf("%.4f\n", numero);    // 3.1416
printf("%8.2f\n", numero);   // "    3.14" (8 caracteres totais)
printf("%-8.2f\n", numero);  // "3.14    " (alinhado à esquerda)
```

3.2 Entrada de Dados com scanf()

A função `scanf()` lê dados do teclado:

```c
#include <stdio.h>

int main() {
    int idade;
    float altura;
    char inicial;

    printf("Digite sua idade: ");
    scanf("%d", &idade);

    printf("Digite sua altura: ");
    scanf("%f", &altura);

    printf("Digite sua inicial: ");
    scanf(" %c", &inicial);  // Note o espaço antes de %c

    printf("Idade: %d, Altura: %.2f, Inicial: %c\n",
           idade, altura, inicial);

    return 0;
}
```

Pontos importantes sobre scanf():

- Use o operador `&` antes do nome da variável (endereço)

- Para `char`, use um espaço antes de `%c` para ignorar whitespace
- Para múltiplas entradas: `scanf("%d %f", &idade, &altura)`

---

4. Operadores

4.1 Operadores Aritméticos

Operador	Descrição	Exemplo	Resultado
`+`	Adição	`5 + 3`	`8`
`-`	Subtração	`5 - 3`	`2`
`*`	Multiplicação	`5 * 3`	`15`
`/`	Divisão	`5 / 2`	`2` (divisão inteira)
`%`	Módulo (resto)	`5 % 2`	`1`

Atenção à divisão:

```c
int a = 5, b = 2;
int resultado_int = a / b;      // 2 (divisão inteira)

float resultado_float = (float)a / b;  // 2.5 (conversão de tipo)
```

4.2 Operadores de Atribuição

Operador	Equivale a	Exemplo
`=`	Atribuição simples	`x = 5`
`+=`	`x = x + y`	`x += 3`
`-=`	`x = x - y`	`x -= 2`
`*=`	`x = x * y`	`x *= 4`
`/=`	`x = x / y`	`x /= 2`
`%=`	`x = x % y`	`x %= 3`

4.3 Operadores Relacionais

Os operadores relacionais comparam valores e retornam 1 (verdadeiro) ou 0 (falso):

Operador	Descrição	Exemplo	Resultado
`==`	Igual a	`5 == 3`	`0` (falso)
`!=`	Diferente de	`5 != 3`	`1` (verdadeiro)

Operador	Descrição	Exemplo	Resultado
`>`	Maior que	`5 > 3`	`1` (verdadeiro)
`<`	Menor que	`5 < 3`	`0` (falso)
`>=`	Maior ou igual	`5 >= 5`	`1` (verdadeiro)
`<=`	Menor ou igual	`3 <= 5`	`1` (verdadeiro)

Cuidado comum: Confundir `=` (atribuição) com `==` (comparação)

```
int x = 5;
if (x = 3) {        // ✗ Erro! Atribuição em vez de comparação
    // código
}

if (x == 3) {       // ✓ Correto! Comparação
    // código
}
```

4.4 Operadores Lógicos

Os operadores lógicos trabalham com valores booleanos:

Operador	Descrição	Exemplo	Resultado
`!`	NÃO (negação)	`!(5 > 3)`	`0` (falso)
`&&`	E (conjunção)	`(5 > 3) && (2 < 4)`	`1` (verdadeiro)
`||`	OU (disjunção)	`(5 < 3) || (2 < 4)`	`1` (verdadeiro)

Tabela verdade:

A	B	!A	A && B	A || B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

4.5 Precedência de Operadores

A precedência determina a ordem de avaliação das expressões:

Precedência	Operadores	Associatividade
1 (maior)	`()`	Esquerda → Direita

Precedência	Operadores	Associatividade
2	`!`	Direita → Esquerda
3	`*`, `/`, `%`	Esquerda → Direita
4	`+`, `-`	Esquerda → Direita
5	`<`, `<=`, `>`, `>=`	Esquerda → Direita
6	`==`, `!=`	Esquerda → Direita
7	`&&`	Esquerda → Direita
8	`||`	Esquerda → Direita
9 (menor)	`=`, `+=`, `-=`, etc.	Direita → Esquerda

Exemplo:

```
int resultado = 2 + 3 * 4;      // 14, não 20
int resultado2 = (2 + 3) * 4;   // 20
```

---

5. Funções Matemáticas Básicas

A biblioteca `math.h` fornece funções matemáticas essenciais:

```
#include <stdio.h>
#include <math.h>

int main() {
    double x = 16.0, y = 2.0;

    printf("sqrt(%.1f) = %.2f\n", x, sqrt(x));        // 4.00
    printf("pow(%.1f, %.1f) = %.2f\n", x, y, pow(x, y)); // 256.00
    printf("fabs(-5.5) = %.1f\n", fabs(-5.5));        // 5.5
    printf("ceil(3.2) = %.0f\n", ceil(3.2));          // 4
    printf("floor(3.8) = %.0f\n", floor(3.8));        // 3

    return 0;
}
```

Compile com a flag -lm:

```
gcc programa.c -o programa -lm
```

---

6. Expressões e Conversões de Tipo

6.1 Avaliação de Expressões

Uma expressão em C combina variáveis, constantes e operadores para produzir um valor:

```c
int a = 5, b = 3, c = 2;
int resultado;

resultado = a + b * c;          // 11 (não 16, devido à precedência)
resultado = (a + b) * c;        // 16 (parênteses alteram a precedência)
resultado = a / b;              // 1 (divisão inteira)
resultado = a % b;              // 2 (resto da divisão)
```

6.2 Conversões de Tipo (Casting)

Conversão Implícita

O compilador automaticamente converte tipos quando necessário:

```c
int i = 5;
float f = i;        // int → float (5.0)
char c = 65;        // int → char ('A')
```

Conversão Explícita (Cast)

O programador força a conversão:

```c
int a = 5, b = 2;
float resultado = (float)a / b;     // 2.5 em vez de 2
int truncado = (int)3.14;          // 3
```

---

7. Boas Práticas de Programação

7.1 Estilo de Código

Indentação e Espaçamento:

```c
// ✓ Bem formatado
#include <stdio.h>

int main() {
    int idade = 25;
    float altura = 1.75;
```

```
    if (idade >= 18) {
        printf("Maior de idade\n");
    }

    return 0;
}
```

Nomenclatura Consistente:

```
// ✓ Nomes descritivos
int idadeUsuario;
float salarioMensal;
char primeiraLetra;

// ✗ Nomes pouco descritivos
int x;
float s;
char c;
```

7.2 Comentários Eficazes

```
// Calcula o IMC (Índice de Massa Corporal)
float calcularIMC(float peso, float altura) {
    return peso / (altura * altura);
}

/*
 * Programa para cálculo de juros compostos
 * Fórmula: M = C * (1 + i)^t
 * Onde: M = montante, C = capital, i = taxa, t = tempo
 */
```

7.3 Tratamento de Entrada

```
#include <stdio.h>

int main() {
    float peso, altura;

    // Solicita entrada com mensagens claras
    printf("Digite seu peso (kg): ");
    scanf("%f", &peso);

    printf("Digite sua altura (m): ");
    scanf("%f", &altura);
```

```
    // Verifica se os valores são válidos
    if (peso <= 0 || altura <= 0) {
        printf("Erro: Valores devem ser positivos!\n");
        return 1;  // Encerra o programa com código de erro
    }

    float imc = peso / (altura * altura);
    printf("Seu IMC é: %.2f\n", imc);

    return 0;
}
```

---

8. Exemplos Práticos

Exemplo 1: Calculadora de IMC

```
#include <stdio.h>

int main() {
    float peso, altura, imc;

    printf("=== Calculadora de IMC ===\n");

    printf("Digite seu peso (kg): ");
    scanf("%f", &peso);

    printf("Digite sua altura (m): ");
    scanf("%f", &altura);

    imc = peso / (altura * altura);

    printf("\nSeu IMC é: %.2f kg/m²\n", imc);

    return 0;
}
```

Exemplo 2: Conversor de Temperatura

```
#include <stdio.h>

int main() {
    float celsius, fahrenheit, kelvin;

    printf("Digite a temperatura em Celsius: ");
    scanf("%f", &celsius);

    fahrenheit = (celsius * 9.0 / 5.0) + 32;
    kelvin = celsius + 273.15;
```

```
    printf("%.1f°C = %.1f°F = %.1fK\n",
           celsius, fahrenheit, kelvin);

    return 0;
}
```

---

9. Exercícios Propostos

1. **Calculadora Básica**: Implemente uma calculadora que leia dois números e execute as quatro operações básicas.
2. **Área e Perímetro**: Calcule a área e o perímetro de um retângulo dados comprimento e largura.
3. **Conversão de Tempo**: Converta um tempo em segundos para horas, minutos e segundos.
4. **Média Ponderada**: Calcule a média ponderada de três notas com pesos 2, 3 e 5.

Próximos Passos

Na **Semana 3**, estudaremos:

- Estruturas condicionais (if, else, switch)
- Operadores lógicos avançados
- Estruturas de decisão aninhadas
- Boas práticas de controle de fluxo