Introdução a Algoritmos

Raoni F. S. Teixeira

Aula 2 - Variáveis, Atribuição, Estrutura de Programa, Escrita e Leitura em C e Operações aritméticas

1 Variáveis

Uma variável é um **local de memória** onde armazenamos valores durante a execução de um programa. Cada variável é caracterizada por um **nome** (identificador) e um **tipo**, que determina o tipo de dado armazenado.

1.1 Declaração

Em linguagem C, declaramos uma variável da seguinte forma:

```
<tipo> <nome_variavel>;
```

Exemplo:

```
int soma;
float preco_abacaxi;
char resposta;
```

1.2 Tipos de variáveis

Inteiras. Usadas para armazenar números inteiros.

- `int` — inteiro padrão, 32 bits em máquinas modernas.
- `unsigned int` — inteiro sem sinal.
- `long int`, `short int` — versões maiores e menores.

Caractere. `char` armazena um único símbolo, por exemplo: `'A'`, `'8'`, `'$'`.

Ponto flutuante. Armazenam números reais com casas decimais.

- `float` — precisão simples (32 bits)
- `double` — precisão dupla (64 bits)

1.3 Regras para nomes de variáveis

- Devem começar por uma letra ou sublinhado (_);
- Não podem começar por número;
- Podem conter letras, números e sublinhados;
- São sensíveis a maiúsculas/minúsculas (`a` e `A` são diferentes);
- Palavras reservadas da linguagem não podem ser usadas (por exemplo: `int`, `return`, `if`, `while`, `double`, etc.).

2 Constantes

Constantes são valores fixos definidos no programa. Podem ser de qualquer tipo válido (inteiro, real, caractere, string):

- Inteira: `10`, `145`, `1000000`
- Real: `2.3456`, `5.0`
- Caractere: `'A'`
- String: `"Hello, world!"`

3 Comando de Atribuição

O operador de atribuição é o sinal =. À esquerda deve haver o nome de uma variável, e à direita uma expressão cujo valor será calculado e armazenado na variável.

```
int a;
float c;
a = 5 + 5 + 10;
c = 67.89 + 8 - 9;
```

4 Estrutura Básica de um Programa em C

Um programa em C possui a seguinte estrutura:

```
#include <stdio.h>

int main() {
    // Declara  o de vari veis
    int a, b, c;

    // Comandos
    a = 7 + 9;
    b = a + 10;
    c = b - a;

    return 0;
}
```

5 Escrita de Dados (`printf`)

A função `printf` imprime texto ou valores na tela.

```
printf("Ol  pessoal!\n");
```

O caractere especial `\n` representa quebra de linha.

5.1 Escrevendo variáveis

Podemos incluir variáveis na saída usando **especificadores de formato**:

```
printf("A vari vel %s cont m o valor %d", "a", a);
```

Principais especificadores:

- `%d` — inteiro
- `%u` — inteiro sem sinal
- `%f` — ponto flutuante
- `%lf` — double
- `%c` — caractere
- `%s` — string

```
int a = 12;
printf("O␣valor␣  ␣%d\n", a);
```

6 Leitura de Dados (`scanf`)

A função `scanf` lê valores digitados pelo usuário.

```
#include <stdio.h>

int main() {
    int n;
    printf("Digite␣um␣n mero:␣");
    scanf("%d", &n);
    printf("O␣valor␣digitado␣foi␣%d\n", n);
    return 0;
}
```

Note que é necessário usar o operador & antes do nome da variável — ele indica o endereço de memória onde o valor será armazenado.

6.1 Lendo várias variáveis

```
int m, n, o;
printf("Digite␣tr s␣n meros:␣");
scanf("%d␣%d␣%d", &m, &n, &o);
printf("Valores␣lidos:␣%d␣%d␣%d\n", m, n, o);
```

6.2 Formatos de leitura

Os formatos de `scanf` são semelhantes aos de `printf`:

`%c`	lê um único caractere
`%s`	lê uma string (sem espaços)
`%d`	lê um inteiro decimal
`%u`	lê um inteiro sem sinal
`%l`	lê um inteiro longo
`%f`	lê um número em ponto flutuante
`%lf`	lê um double

7 Comentários no Código

Comentários servem para documentar o código e são ignorados pelo compilador.

```
#include <stdio.h>

/* Este    o meu primeiro programa */
// Isto tamb m    um coment rio

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

8 Exercício

Escreva um programa que:

1. Solicite ao usuário um número inteiro n;
2. Calcule o quadrado e o dobro de n;
3. Mostre os resultados na tela usando `printf`.

```
#include <stdio.h>

int main() {
    int n;
    printf("Digite um n mero: ");
    scanf("%d", &n);
    printf("O quadrado    %d e o dobro    %d\n", n*n, 2*n);
    return 0;
}
```

9 Expressões Aritméticas

Constantes e variáveis são expressões. Uma **expressão** também pode ser um conjunto de operações aritméticas, lógicas ou relacionais utilizadas para realizar cálculos sobre os valores das variáveis.

```
a + b  // soma o valor de a e b
```

9.1 Operadores aritméticos em C

Os principais operadores aritméticos são:

+	Soma
-	Diferença
*	Produto
/	Divisão
%	Resto da divisão inteira
- ¡expr¿	Inversão de sinal

Exemplos:

```
a = a + b;
a = a - b;
a = a * b;
a = a / b;
a = a % b;
a = -b;
```

9.2 Criando expressões

Operadores aritméticos podem ser combinados para formar expressões mais complexas:

```
a = -b + 2 + c - (9 + d * 8);
```

```
5 + 10 % 3    // resultado: 6
5 * 10 % 3    // resultado: 2
```

10 Precedência de Operadores

A **precedência** define a ordem em que as operações são avaliadas. Em C, a ordem padrão é:

1. *, / e % — da esquerda para a direita;
2. + e - — da esquerda para a direita.

```
int x = 8 + 10 * 6; // x = 68
```

10.1 Alterando a precedência com parênteses

Para modificar a ordem de avaliação, utilizamos parênteses:

```
(5 + 10) % 3   // resultado: 0
```

Regra da paridade: O número de parênteses que abrem deve ser igual ao número de parênteses que fecham.

Dica: use parênteses sempre que a ordem de avaliação não for óbvia.

11 Operadores de Incremento e Decremento

Algumas operações comuns possuem atalhos em C:

- Incremento: c++ (soma 1 à variável)
- Decremento: c-- (subtrai 1 da variável)

```
#include <stdio.h>
int main(void) {
    int a = 10;
    printf("%d", ++a); // imprime 11
}
```

```
#include <stdio.h>
int main(void) {
    int a = 10;
    printf("%d", a++); // imprime 10
}
```

Em expressões, ++ e -- são avaliados antes dos demais operadores. Evite expressões confusas como:

```
printf("%d", a * ++a); // imprime 121, comportamento confuso
```

12 Atribuições Simplificadas

É possível simplificar atribuições repetitivas em C. Por exemplo:

```
a = a + b;
```

pode ser escrito como:

```
a += b;
```

Tabela de operadores de atribuição:

Comando	Exemplo	Equivalente a
+=	`a += b;`	`a = a + b;`
-=	`a -= b;`	`a = a - b;`
*=	`a *= b;`	`a = a * b;`
/=	`a /= b;`	`a = a / b;`
%=	`a %= b;`	`a = a % b;`

13 Conversão de Tipos

Em C, uma expressão pode ser avaliada em outro tipo de dado. Há dois tipos de conversão:

13.1 Conversão implícita

Acontece automaticamente quando o tipo de destino tem maior tamanho (sem perda de informação).

```
int a; short b;
a = b;

float x; int y = 10;
x = y;
```

13.2 Conversão explícita (casting)

O tipo de destino é informado explicitamente pelo programador. Pode haver perda de informação.

```
a = (int)((float)b / (float)c);
```

Exemplo incorreto:

```
int a;
(float)a = 1.0; // inv lido
```

14 Uso Prático: Divisão

O operador de divisão / possui dois comportamentos:

1. **Divisão inteira**: se ambos os operandos são inteiros. Exemplo: `10 / 3` resulta em `3`.
2. **Divisão real**: se pelo menos um operando for ponto flutuante. Exemplo: `1.5 / 3` resulta em `0.5`.

```
#include <stdio.h>
int main(void) {
    int a = 10, b = 3;
    printf("%f", a / (float)b); // imprime 3.333333
}
```

15 Exercício

Analise o programa abaixo:

```
#include <stdio.h>
int main(void){
    int n;
    scanf("%d", &n);
    printf("O␣valor␣da␣expressao␣eh:␣%d\n",
        1 + 2 * 30 / (++n % 4 * -5));
}
```

1. Indique a ordem em que as operações são executadas.
2. Reescreva o programa de forma que cada comando contenha apenas uma operação aritmética.
3. (Desafio) Existe um problema nesse programa. Qual é e quando ocorre?