Introdução a Algoritmos

Raoni F. S. Teixeira

Aula 5 - Comandos de Repetição

1 Introdução

Até este ponto, aprendemos a escrever programas que executam instruções de forma sequencial e, quando necessário, tomam decisões utilizando o comando `if`. Entretanto, muitos problemas exigem a execução de uma mesma ação diversas vezes.

A repetição é um conceito central na programação e também no raciocínio humano: aprendemos e automatizamos tarefas repetindo-as. Da mesma forma, os computadores executam comandos de forma iterativa para resolver problemas mais complexos.

Três perguntas principais nos ajudam a utilizar um comando de repetição:

1. O que deve ser repetido?
2. Quantas vezes deve se repetir?
3. Qual condição pode ser utilizada para representar essa repetição?

2 Motivação: o problema da repetição manual

```
printf("1");
printf("2");
printf("3");
printf("4");
```

```
printf("1");
printf("2");
...
printf("100");
```

A expansão mostra um padrão: o mesmo comando `printf` é repetido, mudando apenas o número impresso. Ao reconhecer o padrão, podemos substituí-lo por uma estrutura que automatiza a repetição.

3 Descobrindo o padrão

Antes de introduzir a sintaxe, podemos raciocinar como o fascículo orienta:

1. **O que se repete?** A instrução `printf()`.
2. **Quantas vezes se repete?** Cem vezes, de 1 até 100.
3. **Qual condição de repetição?** Enquanto o número atual for menor ou igual a 100.

Essas respostas nos permitem criar um **comando de repetição** — o `while`.

4 Comando `while`

4.1 Sintaxe

```
while (condicao) {
    comandos;
}
```

4.2 Funcionamento

1. Testa a condição. Se for verdadeira, executa o bloco de comandos.
2. Ao final, volta para o teste da condição.
3. Quando a condição se torna falsa, a repetição termina.

4.3 Exemplo — Números de 1 a 100

```
int i = 1;
while (i <= 100) {
    printf("%d␣", i);
    i++;
}
```

Neste exemplo:

- O que se repete: a impressão do número.
- Quantas vezes: 100 vezes.
- Condição: `i <= 100`.

4.4 Exemplo — Números até um valor lido do teclado

```
int i = 1, n;
scanf("%d", &n);
while (i <= n) {
    printf("%d␣", i);
    i++;
}
```

O número de repetições agora depende da entrada do usuário. Assim, a estrutura é adaptável e escalável.

5 Exemplo de Descoberta de Padrão — Potências de 2

Queremos imprimir as n primeiras potências de 2.

Perguntas:

1. O que deve ser repetido? O cálculo e a impressão de uma potência.
2. Quantas vezes? n vezes.
3. Qual condição? Enquanto o contador i for menor ou igual a n.

```
int i = 1, n, pot = 2;
scanf("%d", &n);
while (i <= n) {
    printf("2^%d␣=␣%d\n", i, pot);
    pot = pot * 2;
    i++;
}
```

6 Explorando o raciocínio algorítmico

Ao utilizar o comando `while`, seguimos um processo semelhante ao método de resolução de problemas de Polya (aplicado no Fascículo):

1. Entendemos o problema (identificamos o padrão de repetição);
2. Construímos uma solução (um bloco que se repete);
3. Executamos a solução (simulamos o laço);
4. Avaliamos o resultado (condição de parada e correção do padrão).

7 Casos Especiais

```
while (a != a) a = a + 1;
```

O laço não é executado.

```
while (a == a) a = a + 1;
```

8 Exemplo Prático — Divisão inteira com soma e subtração

Problema: Calcular a divisão inteira de dois números usando apenas soma e subtração.

Raciocínio:

1. O que se repete? Subtrair o divisor do dividendo.
2. Quantas vezes? Enquanto o dividendo ainda for maior ou igual ao divisor.
3. Qual condição? `temporario >= divisor`.

```
#include <stdio.h>

int main() {
    int dividendo, divisor, temporario, contador;

    scanf("%d␣%d", &dividendo, &divisor);

    temporario = dividendo;
    contador = 0;

    while (temporario >= divisor) {
        temporario = temporario - divisor;
        contador++;
    }

    printf("%d\n", contador);
    return 0;
}
```

Reflexão: A repetição controla o processo de subtração sucessiva até o resultado final. A condição de parada evita um loop infinito e define o limite lógico da operação.

9 Comando `do-while`

9.1 Sintaxe

```
do {
    comandos;
} while (condicao);
```

9.2 Funcionamento

O bloco é executado ao menos uma vez, antes da verificação da condição.

```
int i = 1, n;
scanf("%d", &n);
do {
    printf("%d\n", i);
    i++;
} while (i <= n);
```

Discussão: Mesmo que `n = 0`, o laço executa uma vez — esse comportamento pode ser desejado em situações onde é necessário exibir algo antes da verificação.

10 Conclusão e Síntese Conceitual

O estudo das estruturas de repetição combina raciocínio lógico, identificação de padrões e abstração algorítmica. Assim como nas estratégias de Polya, a repetição deve ser compreendida como um **processo iterativo de construção de conhecimento**, tanto para o computador quanto para o programador.

Em resumo:

- O comando `while` testa a condição antes de repetir.
- O comando `do-while` testa a condição depois da execução.
- Todo laço deve ter:
    1. Uma variável de controle;
    2. Uma condição de parada;
    3. Um comando que altera o estado da variável.

11 Exercícios

1. Escreva um programa que leia um número n e imprima todos os números pares de 1 até n.

2. Escreva um programa que calcule a soma dos números de 1 até n utilizando um laço `while`.
3. Reescreva o programa anterior utilizando o comando `do-while`.
4. Identifique, em cada exercício, as três perguntas fundamentais:
    - O que se repete?
    - Quantas vezes se repete?
    - Qual é a condição de parada?