Lógica de programação

Estrutura de repetição

Walisson Pereira

walisson_pereira@uvanet.br
Universidade Estadual Vale do Acaraú

Roteiro

Comando de repetição

Comando de repetição

Os computadores podem repetir uma ou mais sequências de comandos quantas vezes forem necessárias, e é o programador quem decide o critério de parada das repetições.

As aplicações para os comandos de repetição são praticamente infinitas porque quase todas as tarefas contêm partes que devem ser executadas mais de uma vez.

Comando de repetição

Exemplo: Imprima três números começando de 1 até 3.

Método: Escrita direta

```c
#include <stdio.h>

int main () {
    printf("1\n");
    printf("2\n");
    printf("3\n");
}
```

Comando de repetição

Exemplo: Imprima três números começando de 1 até 3.

Método: Usando uma variável

```c
#include <stdio.h>

int main () {
    int x = 1;
    printf("%d\n", x);
    x = 2;
    printf("%d\n", x);
    x = 3;
    printf("%d\n", x);
}
```

Comando de repetição

Exemplo: Imprima três números começando de 1 até 3.

Método: Incrementando o valor da variável

```c
#include <stdio.h>

int main () {
    int x = 1;
    printf("%d\n", x);
    x = x + 1;
    printf("%d\n", x);
    x = x + 1;
    printf("%d\n", x);
}
```

Comando de repetição

Se o objetivo fosse escrever 100 números?

A escrita do programa ficaria muito **repetitiva**.

Basicamente o uso de exaustivo de **CONTROL+C** e **CONTROL+V**

Comando de repetição com precondição

Comando de repetição com precondição

Podemos usar o comando **while**.

```
1 while (<expressao logica>) {
2     <sequencia de comandos>
3 }
```

A estrutura **while** é o equivalente em português para:

Enquanto a condição for verdadeira, execute o que estiver no bloco.

Comando de repetição com precondição

Exemplo: Imprima três números começando de 1 até 3.

Método: Usando o comando **while**

```c
#include <stdio.h>

int main () {
    int x = 1;
    while (x <= 3) {
        printf("%d\n", x);
        x = x + 1;
    }
}
```

Comando de repetição com precondição

ATENÇÃO: antes de testar o código abaixo, analise-o e diga qual é o seu resultado.

```c
#include <stdio.h>

int main () {
    int x = 1;
    while (x <= 3) {
        printf("%d\n", x);
    }
    x = x + 1;
}
```

Comando de repetição com precondição

Exercícios:

1. Faça um programa que imprima os números de 1 a 100
2. Faça um programa que imprima os números de 50 a 100
3. Faça um programa para escrever a contagem regressiva do lançamento de um foguete. O programa deve imprimir 10, 9, 8, ..., 1, 0 e Fogo! na tela.

Contadores

Um **contador** é uma variável utilizada para contar o número de ocorrências de um determinado evento; neste caso, o número de repetições do **while**, que satisfaz às necessidade de nosso problema.

Contadores

A estrutura básica de um comando **while** usando um **contador** como variável de controle:

```c
#include <stdio.h>
int main () {
    int contador = 1;
    while (contador < 10) {
        printf("%d\n", contador);
        contador = contador + 1;
    }
}
```

Observe três trechos importantes do código:

1. **inicialização:** contador $= 1$
2. **condição de repetição:** contador < 10
3. **incremento:** contador $=$ contador $+ 1$

Contadores

Exemplo: Faça um programa que receba um valor inteiro e imprima os números inteiros entre 1 e o valor digitado.

```c
#include <stdio.h>

int main () {
    int fim;
    printf("Digite o ultimo numero a imprimir: ");
    scanf("%d", &fim);
    int contador = 1;
    while (contador <= fim) {
        printf("%d\n", contador);
        contador = contador + 1;
    }
}
```

Contadores

Exemplo: Faça um programa que receba um valor e escreva todos os números pares positivos até o valor digitado.

```c
#include <stdio.h>

int main () {
    int fim;
    printf("Digite o ultimo numero a imprimir: ");
    scanf("%d", &fim);
    int contador = 2;
    while (contador <= fim) {
        if (contador % 2 == 0) {
            printf("%d\n", contador);
        }
        contador = contador + 1;
    }
}
```

Contadores

Exemplo: Faça um programa que receba um valor e escreva todos os números pares positivos até o valor digitado.

```
1  #include <stdio.h>
2
3  int main () {
4      int fim;
5      printf("Digite o ultimo numero a imprimir: ");
6      scanf("%d", &fim);
7      int contador = 2;
8      while (contador <= fim) {
9          printf("%d\n", contador);
10         contador = contador + 2;
11     }
12 }
```

Contadores

Exercícios:

1. Faça um programa que receba um valor e escreva todos os números ímpares positivos até o valor digitado.
2. Faça um programa escreva os 10 primeiros números positivos múltiplos de 3.

Acumuladores

O próprio nome já indica o propósito de um **acumulador**.
Para ajudar a diferenciá-lo de um contador, podemos usar a seguinte regra:

- **Contador:** o valor adicionado é constante.
- **Acumulador:** o valor adicionado é variável.

Acumuladores

Exemplo: Um programa que calcula a soma de 10 números.

```c
#include <stdio.h>
int main () {
    int n = 1;
    int soma = 0;
    while (n <= 10) {
        int x;
        printf("Digite o %do. numero: ", n);
        scanf("%d", &x);
        soma = soma + x;
        n = n + 1;
    }
    printf("Soma = %d\n", soma);
}
```

Responda: Quem é o contador e o acumulador neste exemplo?

Comando de repetição com precondição

Exercícios:

1. Escreva um programa que pergunte o depósito inicial e a taxa de juros de uma poupança. Exiba os valores mês a mês para os 24 primeiros meses. Escreva o total ganho com juros no período.
2. Escreva um programa que pergunte o valor inicial de uma dívida e o juro mensal. Pergunte também o valor mensal que será pago. Imprima o número de meses para que a dívida seja paga, o total pago e o total de juro pago.

Operadores de atribuição especiais

Operadores de atribuição especiais

Operador	Exemplo	Equivalência
`+=`	`x += 1`	`x = x + 1`
`-=`	`x -= 1`	`x = x - 1`
`*=`	`c *= 2`	`c = c * 2`
`/=`	`d /= 2`	`d = d / 2`
`++`	`e++` ou `++e`	`e = e + 1`
`--`	`f--` ou `--f`	`f = f - 1`

Operadores de atribuição especiais

Existe diferença entre o ++ antes ou depois do nome da variável.

Exemplo: ++ depois do nome da variável

```c
#include <stdio.h>

int main () {
    int a = 0;
    int b = 0;
    a = b++;
    printf("a = %d e b = %d\n", a, b);
}
```

Resultado:

```
a = 0 e b = 1
```

Operadores de atribuição especiais

Existe diferença entre o ++ (ou --) antes ou depois do nome da variável.

Exemplo: ++ antes do nome da variável

```c
#include <stdio.h>

int main () {
    int a = 0;
    int b = 0;
    a = ++b;
    printf("a = %d e b = %d\n", a, b);
}
```

Resultado:

```
a = 1 e b = 1
```

Operadores de atribuição especiais

- No caso **a = b++;** o incremento da variável **b** é feito depois que seu valor for copiado para a variável **a**
- No caso **a = ++b;** o incremento da variável **b** é feito antes do seu valor ser copiado para a variável **a**

Operadores de atribuição especiais

No caso que o operador ++ ou – for usado apenas com a variável, o resultado será o mesmo independente da posição.

```c
#include <stdio.h>

int main () {
    int a = 0;
    a++;
    printf("a = %d\n", a);
    int b = 0;
    ++b;
    printf("b = %d\n", b);
}
```

Resultado:

```
a = 1
b = 1
```

Comando de repetição com pós-condição

Comando de repetição com pós-condição

Em contraste com o comando de precondição, o comando de repetição com pós-condição só efetua o teste da expressão lógica (condição de parada) após a primeira execução da sequência de comandos.

```
1 do {
2     <sequencia de comandos>
3 } while (<expressao logica>);
```

A estrutura **do-while** é o equivalente em português para:

Faça o que estiver no bloco enquanto a condição for verdadeira.

Comando de repetição com pós-condição

Exercício: Faça um programa que receba um número inteiro entre 0 a 10 (inclusive). Depois imprima o valor recebido. Não permita que o usuário digite um número fora do intervalo.

Comando de repetição com pós-condição

Uma possível solução seria:

```c
#include <stdio.h>
int main () {
    int numero;
    do {
        printf("Digite um numero entre 0 e 10 (inclusive): ");
        scanf("%d", &numero);
    } while ((0 > numero) || (numero > 10));
    printf("O numero digitado foi: %d\n", numero);
}
```

Comando de repetição condensado

Comando de repetição condensado

O comando de repetição condensado permite agrupar, em um único comando, a inicialização de uma variável, o incremento dessa variável e o teste de parada. Seu uso mais comum é um situação nas quais o número de respetições de sequência de comandos é conhecido antes mesmo do início do **for**.

```
1 for (<inicializacao>; <expressao logica>; <incremento>) {
2   <sequencia de comandos>
3 }
```

Comando de repetição condensado

Exercício: Faça um programa que escreva os n primeiros termos de uma progressão geométrica.

Comando de repetição condensado

Uma possível solução seria:

```
1  #include <stdio.h>
2  int main () {
3      float termo, razao, quant;
4      int i;
5      printf("Razao, primeiro termo e numero de termos da PG: ");
6      scanf("%f %f %f", &razao, &termo, &quant);
7      for (i = 0; i < quant; i = i + 1) {
8          printf ("%f\n", termo);
9          termo = termo * razao;
10     }
11 }
```

Interrompendo a repetição

Interrompendo a repetição

A estrutura **while** só verifica a sua condição de parada no início de cada repetição.

A instrução **break** é usada para caso seja desejado interromper a execução do **while** independente de sua condição.

Interrompendo a repetição

Exemplo: Um programa que lê um número e o soma. Caso o valor lido seja zero, termina o programa informando o total da soma.

```
1  #include <stdio.h>
2  int main () {
3      int soma = 0, valor = 1;
4      while (1) {
5          soma += valor;
6          printf("%d\n", soma);
7          if (soma == 5)
8            break;
9      }
10 }
```

Interrompendo a repetição

Exercício:

1. Escreva um programa que leia números inteiros do teclado. O programa deve ler os números até que o usuário digite: 0 (zero). No final da execução, exiba a quantidade de números digitados, assim como a soma e a média aritmética.

Interrompendo a repetição

Além do comando **break**, podemos usar o comando **continue**.

O comando **continue** é usado quando queremos pular para a próxima iteração do laço.

Interrompendo a repetição

Exemplo: Um programa que imprime de 1 a 10, menos os múltiplos de 3.

```c
#include <stdio.h>
int main () {
  int numero = 0;
  while (numero < 10) {
    numero += 1;
    if (numero % 3 == 0)
      continue;
      printf("%d\n", numero);
  }
}
```

Obs: Há um pequeno detalhe que deixa este código confuso. Você consegue dizer qual é?

Interrompendo a repetição

Exemplo: Um programa que imprime de 1 a 10, menos os múltiplos de 3.

```c
#include <stdio.h>
int main () {
  int numero = 0;
  while (numero < 10) {
    numero += 1;
    if (numero % 3 == 0)
      continue;
    printf("%d\n", numero);
  }
}
```

Obs: Veja a diferença que a indentação faz. Agora, fica claro para quem ler que o **printf** não faz parte do **if**.

Repetições aninhadas

Repetições aninhadas

Similar ao aninhamento de **if**, também podemos combinar vários **while** de forma a obter resultados mais interessantes.

Repetições aninhadas

Exemplo: Um programa que imprime a tabuada de multiplicação de 1 a 10.

```c
#include <stdio.h>
int main () {
    int tabuada = 1;
    while (tabuada <= 10) {
        int numero = 1;
        while (numero <= 10) {
            int valor = tabuada * numero;
            printf("%d x %d = %d\n", tabuada, numero, valor);
            numero += 1;
        }
        tabuada += 1;
    }
}
```

Referências

Referências

1 ASCENCIO, A.F.G.; CAMPOS, E.A.V. de. Fundamentos da Programção de Computadores. Algoritmos, Pascal e C/C++. São Paulo: Pearson Prentice Hall, 2012.

2 VAREJÃO, F. M. V. Introdução à programação: uma nova abordagem usando C. Rio de Janeiro: Elsevier, 2015.

3 BACKES, A. Linguagem C: completa e descomplicada. Rio de Janeiro: Elsevier, 2013.