Laboratório de programação

Estruturas de Controle

Universidade Estadual Vale do Acaraú – UVA

Paulo Regis Menezes Sousa

paulo_regis@uvanet.br

Estruturas de Controle

- Estruturas de Seleção
- Estruturas de Repetição
- Outras instruções de controle
- Exercícios

- As instruções em um programa são executadas na ordem em que foram escritas. Isto é chamado **execução sequencial**.
- Quando existe a necessidade de alterar o fluxo de execução de um programa pode-se utilizar estruturas específicas da linguagem para realizar desvios condicionais e repetições
- Para a realização de desvios condicionais a linguagem C fornece estruturas conhecidas como **estruturas de seleção**, são elas:

  if realiza (seleciona) uma ação se uma condição for verdadeira ou ignora a ação se a condição for falsa.

  if/else realiza uma ação se uma condição for verdadeira e realiza uma ação diferente se a condição for falsa.

  switch realiza uma entre muitas ações diferentes dependendo do valor de uma expressão.

- A palavra-chave **if** é usada para tomar decisões no seu código, baseadas em comparações simples.

```c
#include <stdio.h>

int main() {
    int a, b;

    a = 6;
    b = a - 2;

    if (a > b) {
        printf("%d é maior que %d\n", a, b);
    }

    return 0;
}
```

se 'a' é maior que 'b' então...

Operador	Exemplo	Verdadeiro quando
`!=`	`a != b`	a não é igual a b
`<`	`a < b`	a é menor que b
`<=`	`a <= b`	a é menor ou igual a b
`==`	`a == b`	a é igual a b
`>`	`a > b`	a é maior que b
`>=`	`a >= b`	a é maior ou igual a b

```c
#include <stdio.h>

int main() {
    int first, second;

    printf("Digite o primeiro valor: ");
    scanf("%d", &first);
    printf("Digite o segundo valor: ");
    scanf("%d", &second);

    puts("Avaliando...");
    if (first < second) {
        printf("%d é menor que %d\n", first, second);
    }
    if (first > second) {
        printf("%d é maior que %d\n", first, second);
    }

    return 0;
}
```

- O C fornece algumas maneiras para lidar com exceções, permitindo criar códigos que executem, baseados em possibilidades múltiplas.
- Para os tipos de comparação "ou-ou", a palavra chave `if` pode ser complementada com `else`:

```
1  if (condition) {
2      statement(x);
3  }
4  else {
5      statement(y);
6  }
```

```c
#include <stdio.h>

int main() {
    int a, b;

    a = 6;
    b = a - 2;

    if (a > b) {
        printf("%d é maior que %d\n", a, b);
    }
    else {
        printf("%d não é maior que %d\n", a, b);
    }

    return 0;
}
```

se 'a' é maior que 'b' então...

se não...

- Estruturas `if/else` aninhadas verificam vários casos inserindo umas estruturas `if/else` em outras.

```
1  if (condição lógica) {
2      //...
3  }
4  else if (condição lógica){
5      //...
6  }
7  else {
8      //...
9  }
```

- Blocos de estruturas com apenas uma linha de corpo não precisam dos delimitadores { }.
- Um conjunto de instruções dentro de um par de chaves é chamado uma instrução composta.

```
1  if (nota >= 7.0)
2      printf("Aprovado\n");
3  else
4      printf("Reprovado\n");
```

```
1  if (nivelBateria < 500) {
2      printf("Carga baixa.\n");
3      printf("Recarregue.\n");
4  }
```

```
1  if (nota >= 7.0)
2      printf("Aprovado\n");
3  else if (notaAF < 5.0)
4      printf("Reprovado\n");
5  else
6      printf("Aprovado\n");
```

```
1  if (5 < x) {
2      if (x <= 10)
3          printf("5<x<=10\n");
4      else
5          printf("x inválido\n);
6  }
```

Operador	Exemplo	Verdadeiro quando
`&&`	`A && B`	A e B são verdadeiros
`||`	`A || B`	A ou B são verdadeiros
`!`	`!A`	O item A é falso

- Para realizar comparações mais complexas em C podem-se utilizar os operadores lógicos.

```c
if (5 < x) {
    if (x <= 10)
        printf("5 < x <= 10\n");
    else
        printf("x inválido\n);
}
```

```c
if (5 < x && x <= 10)
    printf("5 < x <= 10\n");
else
    printf("x inválido\n);
```

- A linguagem C fornece o operador condicional (?:) que está intimamente relacionado com a estrutura if/else.

```
1  printf("%s\n", nota >= 7.0 ? "Aprovado" : "Reprovado");
```

```
1  nota >= 7.0 ? printf("Aprovado\n") : printf("Reprovado\n");
```

- O operador condicional é conhecido como operador ternário (porque utiliza três operandos).

- A estrutura **switch** consiste em uma série de rótulos **case** e de um caso opcional **default**.

```
1  switch (N) {
2      case 1:
3          printf("N = 1\n");
4          break;
5      case 2:
6          printf("N = 2\n");
7          break;
8      default:
9          printf("N não reconhecido.\n");
10         break;
11 }
```

- A linguagem C fornece três tipos de **estruturas de repetição**:

  while realiza um teste lógico e repete a execução de um trecho de código conforme o seu resultado.

  do/while executa um trecho de código e realiza um teste lógico repetindo a execução do trecho segundo o resultado do teste.

  for realiza um teste lógico e repete a execução de um trecho de código conforme o seu resultado.

- Uma estrutura de repetição permite ao programador especificar que uma ação deve ser repetida enquanto uma determinada condição for verdadeira.

Exemplo

Encontrar a primeira potência de 2 maior que 1000. Suponha que a variável inteira produto foi inicializada com o valor 2. Quando a estrutura de repetição a seguir terminar sua execução, produto conterá a resposta procurada.

```
1  produto = 2;
2  while (produto <= 1000)
3      produto = 2 * produto;
```

- Considere o seguinte enunciado de um problema:

Problema

Uma turma de dez alunos fez um teste. As notas (inteiros variando de 0 a 100) do teste estão disponíveis para você. Determine a média da turma no teste.

- A média da turma é igual à divisão da soma das notas pelo número de alunos.
- O algoritmo para resolver esse problema em um computador deve receber cada um das notas, realizar o cálculo da média e imprimir o resultado.

```c
#include <stdio.h>

int main () {
    int count = 1;
    float mean, grade, total = 0.0;

    while (count <= 10) {
        printf("Nota #%d: ", count);
        scanf("%f", &grade);

        total = total + grade;
        count++;
    }

    grade = total / 10.0;
    printf("\nA média da turma é: %.1f\n", mean);
    return 0;
}
```

- Vamos generalizar o problema do cálculo da média da turma. Considere o seguinte problema:

Problema 2

Desenvolva um programa para calcular a média de uma turma que processe um número arbitrário de notas cada vez que o programa for executado.

- Nesse exemplo, não há indicação de quantas notas serão fornecidos
- O programa deve processar um número arbitrário de notas.

- Uma maneira de resolver esse problema é usar um valor especial chamado valor sentinela para indicar o "final da entrada de dados".
- O usuário digita notas até que todos as notas válidos tenham sido fornecidas.
- Então o usuário digita o valor sentinela para indicar que o última nota foi fornecida.
- Obviamente, o valor sentinela deve ser escolhido de forma que não possa ser confundido com um valor aceitável de entrada.

```c
#include <stdio.h>

int main() {
    int i = 0;
    float media = 0.0, nota = 0.0, total = 0.0;

    printf("Digite as notas (-1 para encerrar)\n");
    while (nota != -1) {
        printf("Nota %d: ", i+1);
        scanf("%f", &nota);

        if (nota != -1) {
            total = total + nota;
            i++;
        }
    }

    if (i > 0) {
        media = total / i;
        printf("A média da turma é %.1f\n", media);
    }
    return 0;
}
```

```c
#include <stdio.h>

int main() {
    int op;
    float saldo = 35.7;

    do {
        printf("Selecione uma opção:\n\t1 Saldo\n\t2 Sair\n");
        scanf("%d", &op);

        switch (op) {
            case 1:
                printf("Seu saldo é R$ %.2f\n", saldo);
                break;
            case 2:
                printf("Até a próxima.\n");
                break;
            default:
                printf("Opção inválida.\n");
        }
    } while (op != 2);

    return 0;
}
```

- A estrutura de repetição for manipula automaticamente todos os detalhes da repetição controlada por contador.

```c
#include <stdio.h>

int main(){
    int contador;

    for (contador = 1; contador <= 10; contador++)
        printf("%d\n", contador);

    return 0;
}
```

- O próximo exemplo calcula juros compostos usando a estrutura for. Imagine o seguinte enunciado de problema:

Problema

Uma pessoa investe $1000,00 em uma conta de poupança que rende juros de 5 por cento. Admitindo que todos os juros são deixados em depósito na conta, calcule e imprima a quantia na conta ao final de cada ano, ao longo de 10 anos. Use a seguinte fórmula para determinar estas quantias:

$$m = c \times (1 + i)^n$$

onde: **c** é o capital investido originalmente **i** é a taxa anual de juros **t** é o tempo de investimento (número de anos), **m** é o montante existente em depósito no final do n-ésimo ano.

```c
#include <stdio.h>
#include <math.h>
int main(){
    int ano;
    double quantia, principal = 1000.0, taxa = .05;

    printf ("%4s%21s\n", "Ano", "Saldo na conta");

    for (ano = 1; ano <= 10; ano++){
        quantia = principal * pow(1.0 + taxa, ano);
        printf("%4d%21.2f\n", ano, quantia);
    }

    return 0;
}
```

- A estrutura de repetição **do/while** é similar à estrutura **while**. Na estrutura **while**, a condição de continuação do loop é testada em seu início antes de o corpo da estrutura ser executado.
- A estrutura **do/while** testa a condição de continuação depois de o corpo do loop ser executado, portanto ele será executado pelo menos uma vez.

```c
#include <stdio.h>

int main() {
    int contador = 1;

    do{
        printf("%d ", contador);
    } while (++contador <= 10);

    return 0;
}
```

- As instruções break e **continue** são usadas para alterar o fluxo de controle.
- A instrução **break**, quando executada em uma estrutura **while**, **for**, **do/while** ou **switch**, faz com que aconteça a saída imediata daquela estrutura.
- A instrução **continue**, quando executada em uma estrutura **while**, **for** ou **do/while**, ignora (salta sobre) as instruções restantes no corpo daquela estrutura e realiza a próxima iteração do loop.

```c
#include <stdio.h>

int main() {
    int x;
    for (x = 1; x <= 10; x++) {
        if(x == 5)
            break;
        if(x == 2)
            continue;
        printf("%d ", x);
    }

    printf ("\n Saiu do loop em x == %d\n", x);

    return 0;
}
```

Exercício 6

Escreva um programa que leia um número inteiro positivo N e em seguida imprima N linhas do chamado triângulo de Floyd. Exemplo, se N = 6 o seu programa deve imprimir 6 linhas.

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21

Exercício 7

Faça um algoritmo que leia um número positivo e imprima seus divisores. Exemplo: os divisores do número 66 são: 1, 2, 3, 6, 11, 22, 33 e 66.

Exercício 8

Faça um programa que leia um valor N inteiro e positivo. Calcule e mostre o valor de E, conforme a fórmula a seguir:

$$E = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{N!}$$

OBS: o valor de N não pode ser muito grande devido ao cálculo do fatorial :(