CURSO SUPERIOR DE ADS

Estruturas Iterativas

Prof. Fernando Marlon Soares Figueiredo

Disciplina: Programação Estruturada

Aula de Hoje

- Introdução às estruturas de repetição.
- 3 pilares da programação estruturada.
- Os comandos while, do while e for.
- Aplicar muitos exercícios

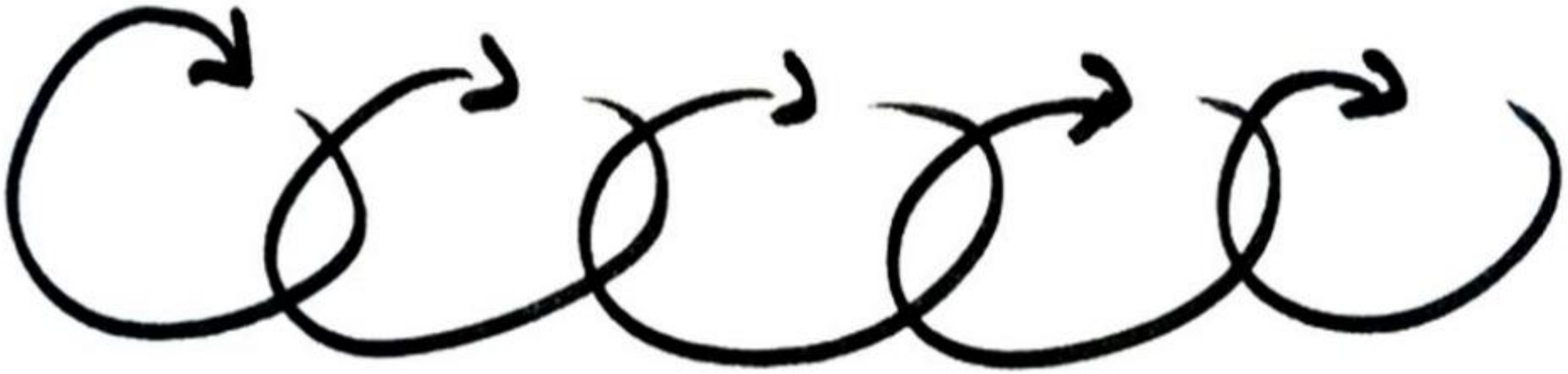
Os 3 Pilares do Paradigma da Programação Estruturada

- **Sequência**: Refere-se à **execução das instruções em ordem sequencial**, uma após a outra. Isso significa que as instruções são executadas na ordem em que estão escritas no código, a menos que haja estruturas de controle para alterar o fluxo de execução.
- **Seleção** (ou Decisão): Envolve a **capacidade de tomar decisões com base em condições**. A estrutura de seleção mais comum é a declaração "if-else" (ou "if-else if-else"), que permite executar um bloco de código somente se uma determinada condição for verdadeira. Isso permite que o programa escolha entre diferentes caminhos de execução com base em condições específicas.
- **Repetição** (ou Iteração): Trata da **execução repetida de um bloco de código enquanto uma condição for verdadeira**. Isso é feito usando estruturas de repetição, como "while" e "for". A repetição é fundamental para automatizar tarefas que precisam ser realizadas várias vezes, sem a necessidade de escrever o mesmo código repetidamente.

Hoje vamos discutir o 3º pilar

Repetição ou Iteração

- Os comandos de repetição permitem que determinados trechos de código sejam repetidos até que **determinada condição seja atingida** ou **um número específico de passos seja alcançado**.
- Estas duas formas distintas de controlar iterações (ou laços) são apresentadas por dois tipos de comandos: **while** e **do while**.



Comando while

- Permite que determinado trecho de código siga executando **enquanto** a sua condição for verdadeira. Quando a condição se tornar falsa, o programa deixa de executar o bloco de código e segue normalmente o fluxo de execução.
- **Sintaxe do while:**

```
while (condicao)

{

// codigo repetido enquanto a condicao eh verdadeira

}
```

O que esse programa faz?

```
#include <stdio.h>

int main ()
{
    int i;

    i = 0;
    while(i < 5)
    {
        printf(“Valor de i: %d\n”, i);
        i = i + 1;
    }

    printf(“Saiu do laco!\n”);

    return 0;
}
```

O que esse programa faz?

```c
#include <stdio.h>

int main ()
{
    int i;

    i = 0;
    while(i < 5)
    {
        printf("Valor de i: %d\n", i);
        i = i + 1;
    }

    printf("Saiu do laco!\n");

    return 0;
}
```

```
Valor de i: 0
Valor de i: 1
Valor de i: 2
Valor de i: 3
Valor de i: 4
Saiu do laco!
```

3 fatores importantes sobre as condições

1. Inicialização da variável

- A variável envolvida deve ser inicializada antes do comando, garantindo que ele entre no laço de execução da primeira vez (linha 7).
- Caso isso não aconteça, o laço pode nunca ser executado.

```c
1 #include <stdio.h>
2
3 int main ()
4 {
5     int i;
6
7     i = 0;
8     while(i < 5)
9     {
10        printf("Valor de i: %d\n", i);
11        i = i + 1;
12    }
13
14    printf("Saiu do laco!\n");
15
16    return 0;
17 }
```

2. Condição de Continuação

- O controle do laço deve conter uma condição que eventualmente termine de executar(linha 8).
- É importante que esta condição esteja de acordo tanto com a inicialização, quanto com o incremento.
- A inicialização deve permitir que a condição seja inicialmente verdadeira e o incremento deve garantir que ela pare de executar em algum momento.

```c
1 #include <stdio.h>
2
3 int main ()
4 {
5     int i;
6
7     i = 0;
8     while(i < 5)
9     {
10        printf(“Valor de i: %d\n”, i);
11        i = i + 1;
12     }
13
14     printf(“Saiu do laco!\n”);
15
16     return 0;
17 }
```

3. Incremento da Variável

- A variável envolvida deve ser modificada a cada iteração (linha 11), a fim de que a condição consiga chegar ao fim em determinado momento.

```
1 #include <stdio.h>
2
3 int main ()
4 {
5     int i;
6
7     i = 0;
8     while(i < 5)
9     {
10        printf(“Valor de i: %d\n”, i);
11        i = i + 1;
12    }
13
14    printf(“Saiu do laco!\n”);
15
16    return 0;
17 }
```

Exercício 1

- Faça um programa que pede para o usuário digitar um número inteiro. Na sequência o programa lê esse número e apresenta o número digitado no console.
- Caso o número digitado seja zero, o programa chega ao fim.

```
Digite um número inteiro (0 para sair): 4
Número digitado: 4
Digite um número inteiro (0 para sair): 2
Número digitado: 2
Digite um número inteiro (0 para sair): 0
Número digitado: 0
```

Exercício 1 - Resposta

```c
#include <stdio.h>

int main(void) {
  int valor = -1;
  while (valor!=0) {
    printf("Digite um número (0 pra sair)\n");
    scanf("%d", &valor);
    printf("Número digitado: %d\n", valor);
  }
  printf("Fim\n");
  return 0;
}
```

Exercício 1 - Resposta

- Temos uma quantidade definida ou indefinida de repetições?
- Indefinida. Não sabemos previamente quantos números diferentes de zero serão informados.

```c
#include <stdio.h>

int main ()
{
    int num;

    num = -1;
    while(num != 0)
    {
        printf(”Digite um numero inteiro (0 para sair):”);
        scanf("%d", &num);
        printf(”Numero digitado: %d\n”, num);
    }

    return 0;
}
```

Comando do while

- Note que tivemos que inicializar a variável **num** (no exercício anterior) com um valor diferente de zero para que o laço pudesse ser executado. Uma forma de evitar esse processo é através de uma variação do comando **while**, chamada **do while**.
- A sintaxe é:

```
do

{

// codigo repetido enquanto a condicao eh verdadeira

}

while (condicao);
```

Comando do while

- Esse comando garante que o código do laço será executado pelo menos uma vez no programa, já que a checagem da condição ocorre após o bloco.
- Uma das utilizações desse comando é para o **controle de menus**, onde é preciso que o menu de opções apareça para o usuário tomar um curso de ação.

Exemplo

```c
#include <stdio.h>

int main ()
{
    int opcao;
    printf(“Menu principal: \n”);
    printf(“1 – Imprimir mensagem 1\ n”);
    printf(“2 – Imprimir mensagem 2\ n”);
    printf(“3 – Sa i r \n”);
  do
  {
   printf (“Digite a opcao:”);
   scanf(“%d”, &opcao);
   switch(opcao)
   {
      case 1:
       printf(“Va em frente!\n”);
      break;
      case 2:
       printf(“Insista e nunca desista!\n”);
      break;
      case 3:
       printf(“Encerrando o programa...”);
      break;
      default:
       printf(“Opcao invalida.\n”);
   }
  }
  while(opcao != 3);

  return 0;
}
```

Exercício 2 – Jogo da Adivinhação

- Crie um programa que simule um jogo de adivinhação.
- O programa escolhe um número aleatório entre 1 e 100, e o usuário deve tentar adivinhar qual é esse número.
- O programa deve informar se o palpite do usuário está acima ou abaixo do número escolhido.
- O jogo continua até o usuário acertar o número.

SORTEIO DE NÚMERO ALEATÓRIO ENTRE 1 E 100

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
  srand(time(NULL));
  int valor = rand()% (100 + 1);
  printf("%d",valor);

  return 0;
}
```

Exercício 2 - Resposta

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
  srand(time(NULL));
  int valor = rand()% (100 + 1);
  printf("O valor a ser adivinhado é: %d\n",valor);
  int escolha = -1;
  do {
    printf("Digite um número: \n");
    scanf("%d", &escolha);
    if (valor > escolha) {
      printf("O valor procurado é maior\n");
    } else if (valor < escolha) {
      printf("O valor procurado é menor\n");
    } else {
      printf("Acertou!");
    }

  } while(escolha!=valor);
  return 0;
}
```

Comando for

- O comando pode ser visto como uma forma mais compacta do comando while, devendo ser usado para casos onde o número de iterações é conhecido previamente.
- A sintaxe do comando é:

```
for (inicial; condicao; incremento)

{

// codigo repetido enquanto a condicao eh verdadeira

}
```

Comando for

- Devemos inicializar a variável com uma **operação de atribuição.**
- A condição contém a **condição de parada** do laço.
- **Incremento atualiza o valor da variável de controle** a cada iteração com uma atribuição.

```
for (inicial; condicao; incremento)

{

// codigo repetido enquanto a condicao eh verdadeira

}
```

O que o programa faz?

```c
#include <stdio.h>
#define NUM_ALUNOS 5

int main ()
{
    int i;
    float nota;
  for (i = 0; i < NUM_ALUNOS; i++)
   {
        printf("Digite a nota do aluno %d:", i);
        scanf("%f", &nota);
        if(nota >= 6)
         printf("O aluno %d foi aprovado:)\n", i);
        else
         printf("O aluno %d foi reprovado:  (\ n", i);
   }

   return 0;
}
```

Contadores, Acumuladores e Sinalizadores

Estruturas iterativas geralmente vêm acompanhadas de categorias específicas de variáveis

Contadores

- São variáveis responsáveis por contar uma quantidade específica de vezes que algum trecho de código específico ocorre em iterações.
- A cada iteração, a variável tem seu valor acrescido em uma unidade. Por exemplo, detectar quantas vezes o usuário digitou seu nome antes de encerrar uma execução.

- cont = cont + 1; //ou cont++

Acumuladores

- Aglomeram valores lidos em iterações. A cada laço a variável soma um novo valor com o seu antigo.
- Podem ser utilizados para contar o valor total de vendas em uma loja a partir dos valores individuais de cada produto.

- acum = acum + 1
- acum = acum + novavenda // acum+=novavenda

Sinalizadores

- São utilizadas para indicar se determinado status no programa é verdadeiro ou falso. Geralmente estão vinculadas a estruturas condicionais.
- Por exemplo, descobrir se pelo menos uma pessoa maior de idade preencheu o cadastro. Quando alguém maior de 18 anos preencher o cadastro, a variável sinalizadora previamente definida como desligada (valor zero) é acionada (valor 1, ou seja, verdadeiro).

- flag = 0 //ou flag = 1;

Como escolher o comando de iteração adequado ao meu problema?

- Apesar do comando **while** poder ser utilizado para casos em que o número de iterações é conhecido, geralmente esta tarefa é deixada para o comando **for**.
- O uso principal dos comandos **while** e **do while** ocorre quando o número de iterações **não é conhecido a priori**.

Mais exercícios

Exercício 3

Faça um programa que peça para o usuário digitar diversos números inteiros até que um dos números seja negativo.

Exercício 3 - Resposta

```c
#include <stdio.h>

int main(void) {
  int num = 1;
  while (num >= 0) {
     printf("Digite um número: \n");
     scanf("%d", &num);
  }
  printf("Fim");

  return 0;
}
```

Exercício 4

Faça um programa que escreva os números de 10 a 20 na tela, separados por um espaço.

Exercício 4 - Resposta

```c
#include <stdio.h>

int main(void) {

  for (int i=10; i<=20; i++ ) {
    printf("%d ", i);
  }

  return 0;
}
```

Exercício 5

Faça um programa que peça para o usuário digitar diversos valores de compras não negativos e apresente a soma desses valores quando o usuário digitar o valor zero.

Exercício 5 - Resposta

```c
#include <stdio.h>

int main(void) {
  int valor = -1;
  int total = 0;

  while (valor!=0) {
    printf("Qual o valor da venda? \n");
    scanf("%d", &valor);
    total += valor;
  }
  printf("O total de vendas foi: R$ %d", total);
  return 0;
}
```

Exercício 6

Faça um programa que mostre a tabuada do número 5. Considere a tabuada dos números 1 ao 10.

```
>_ Console
> make -s
> ./main
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50
```

Exercício 6 - Resposta

```c
#include <stdio.h>

int main(void) {
  for (int i=1; i<=10; i++) {
    int valor = 5 * i;
    printf("%d * 5 = %d \n", i, valor);
  }
  return 0;
}
```

Exercício 7

Faça um programa que peça para o usuário digitar um número inteiro de 1 a 10 e gere sua respectiva tabuada de multiplicação (de 1 a 10).

Exercício 7 - Resposta

```c
#include <stdio.h>

int main(void) {
  printf("Digite um número: \n");
  int num;
  scanf("%d", &num);
  for (int i=1; i<=10; i++) {
    int valor = num * i;
    printf("%d * %d = %d \n", i, num, valor);
  }
  return 0;
}
```

Exercício 8

- Faça um programa que auxilie uma pesquisa a coletar as seguintes informações sobre cada um dos moradores de um bairro: idade (inteiro), sexo (inteiro 1, para masculino, 2 para feminino) e salário (float). O usuário deverá preencher as informações de cada habitante até que seja digitado o valor 0 para idade. Ao final o programa deve calcular e informar os seguintes dados:
- A maior idade.
- A média salarial
- A quantidade de mulheres
- Existe salário abaixo de 500?

Exercício 8 – Resposta – parte 1

```c
#include <stdio.h>

int main(void) {
  int idade = -1;
  int maior = -1;
  int sexo;
  int contfem = 0;
  float salario;
  int cont = 0, total = 0;
  float media = 0;
  float salmin = 0;
```

```c
  do {
    printf("Qual a idade?\n");
    scanf("%d", &idade);
    if (idade > maior) {
      maior = idade;
    }
    if (idade == 0 ) break;
    printf("Qual o sexo (1 - masc, 2 - fem)?\n");
    scanf("%d", &sexo);
    if (sexo == 2) {
      contfem++;
    }
```

Exercício 8 – Resposta – parte 2

```c
    printf("Qual o salário?\n");
    scanf("%f", &salario);
    if (salario < 500) {
      salmin = 1;
    }
    total = total + salario;
    cont++;

  } while (idade!=0);
```

```c
  printf("\nA maior idade é: %d", maior);

  media = total / cont;
  printf("\nA média salarial é: %.2f", media);
  printf("\nO total de mulheres é: %d", contfem);
  if (salmin) {
    printf("\nExiste salário abaixo de 500");
  } else {
    printf("\nNão existe salário abaixo de 500");
  }
  return 0;
}
```

Exercício 9

Faça um programa em C para gerar os n primeiros termos da sequência:

1 1 2 3 5 8 13 21 34 55 89 ...

Exercício 10

Faça um programa (utilizando a estrutura for) que leia 10 valores inteiros e:

- Encontre e mostre o maior valor.
- Encontre e mostre o menor valor.
- Calcule e mostre a média dos números lidos.

Exercício 11

- Faça um programa que receba um número inteiro e positivo, e em seguida, diga se esse número é primo ou não.
- Obs: Um número é primo somente quando for divisível por 1 e por ele mesmo.

Exercício 12

- Chico tem 1,70m e cresce 2 centímetros por ano, enquanto Juca tem 1,10m e cresce 3 centímetros por ano.
- Construir um programa em C que calcule e imprima quantos anos serão necessários para que Juca seja maior que Chico (utilize a estrutura while).