Algoritmos e Estruturas de Dados I
Aula 4 -Estrutura de Repetição

Prof. Felipe Lara

Estrutura Condicional

Existem três estruturas básicas em programação:

- Estrutura Sequencial
- Estrutura Condicional
- **Estrutura Repetitiva**

Exercício 1

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

Exercício 1

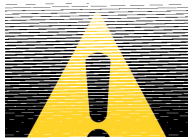
Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){
    printf(“1”);
    printf(“2”);
    printf(“3”);
    …
    printf(“1000”);
}
```

Exercício 1

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){
      printf(“1”);
      printf(“2”);
      printf(“3”);
      …
      printf(“1000”);
}
```



Solução não prática!

Estrutura de Repetição

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

Solução: estrutura de repetição!

- Certos tipos de problemas podem ser resolvidos com sequências de instruções executadas apenas uma vez;
  - Porém, outros algoritmos requerem a execução de determinados trechos de código várias vezes, por isso o surgimento das estruturas de repetição;

Estrutura de Repetição

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

Solução: estrutura de repetição!

- Certos tipos de problemas podem ser resolvidos com sequências de instruções executadas apenas uma vez;
  - Porém, outros algoritmos requerem a execução de determinados trechos de código várias vezes, por isso o surgimento das estruturas de repetição;
- Uma estrutura de repetição permite que uma sequência de instruções seja executada várias vezes até que uma condição seja satisfeita;
  - Quando utilizar? Analisar se uma mesma sequência de instruções necessita ser executada várias vezes.
- Comandos: ***while*** e ***for***.

Exercício 1

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){
    printf("1");
    printf("2");
    printf("3");
    ...
    printf("1000");
}
```

```
printf(numero);
```

Essa é a parte do código que repete por 1000 vezes

Exercício 1

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){

    printf("%d", numero);

}
```

Preciso variar o valor de numero de 1 até 1000

printf(numero);
Considerando que número é uma variável que vai variar de 1 até 1000, podemos usar um único printf

Exercício 1 - usando for

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){
    int numero;
    for (numero = 1; numero <= 1000; numero = numero + 1){
        printf("%d", numero);
    }
}
```

Exercício 1 - usando for

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){
      int numero;
      for (numero = 1; numero <= 1000; numero = numero + 1){
            printf("%d", numero);
      }
}
```

Obs: **início do loop**, **fim do loop**, **incremento/decremento do loop**, **parte do código que vai repetir (entre as { })**

Exercício 1 - usando while

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){
      int numero = 1;
      while (numero <=1000){
            printf("%d", numero);
            numero = numero + 1;
      }
}
```

Obs: **condição do loop**
parte do código que vai repetir (entre as { })

Exercício 1 - usando do while

Imagine um algoritmo que tenha que imprimir os primeiros 1000 números começando pelo 1

```
int main(){
      int numero = 1;
      do{
            printf("%d", numero);
            numero = numero + 1;
      }while (numero <=1000)
}
```

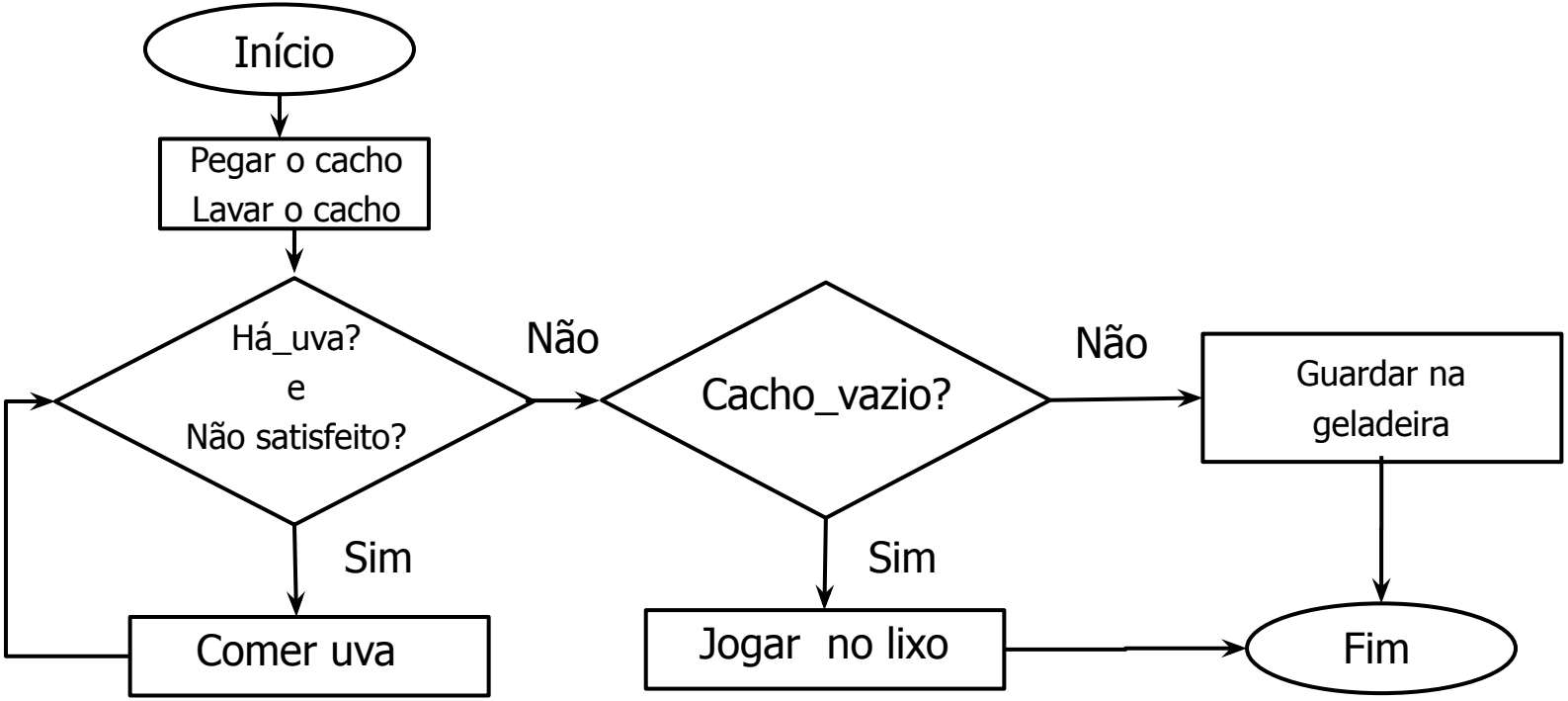
Obs: **condição do loop**
parte do código que vai repetir (entre as { })

Teoria sobre Estrutura de Repetição

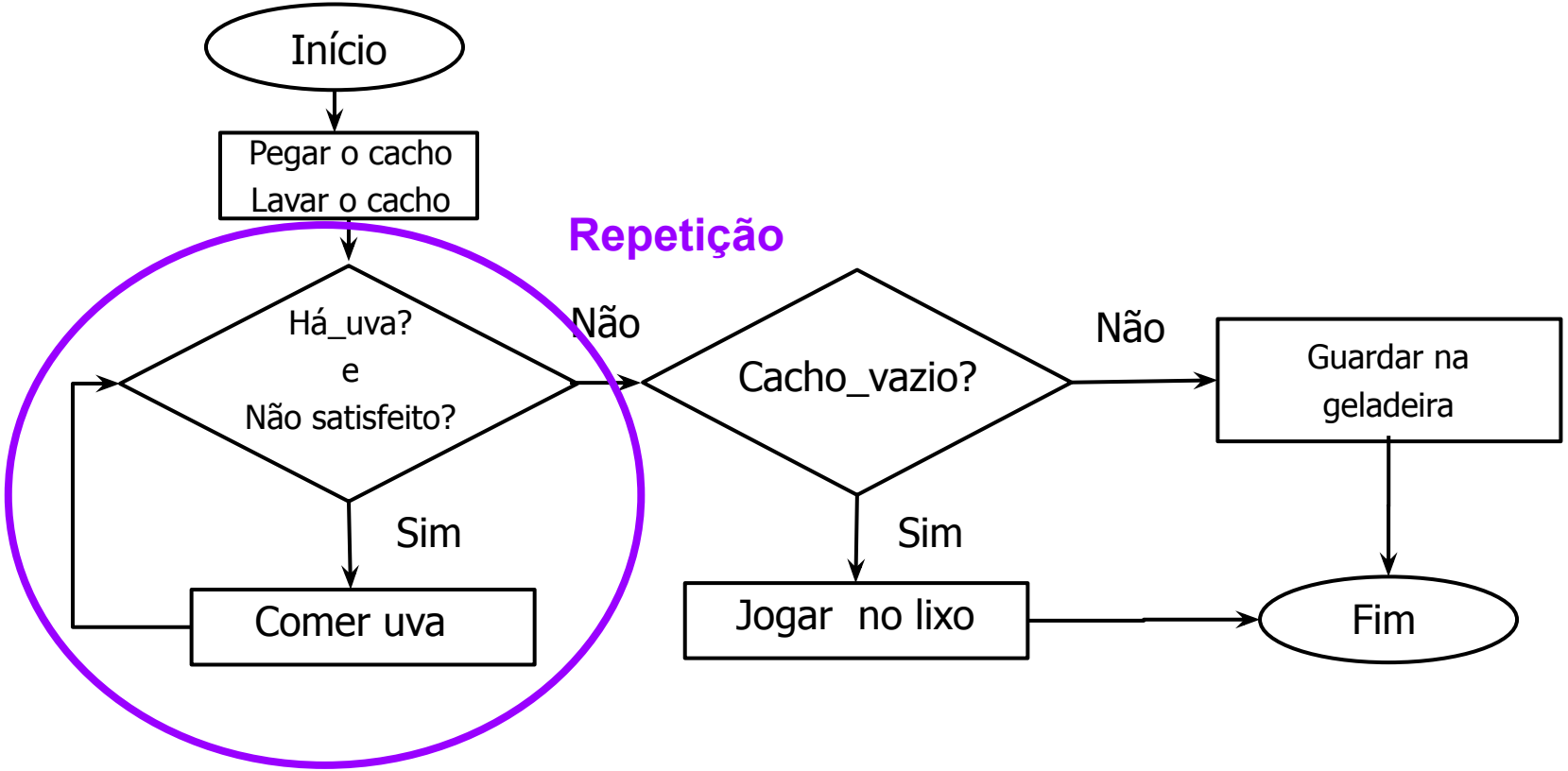
Algoritmo comer um cacho de uva

```
ALGORITMO "Comer cacho de uva"
    Pegar o cacho de uva
    Lavar o cacho de uva
    ENQUANTO (houver_uva && não_satisfeito) FAÇA
        Início
      Comer uva
        Fim
    SE (cacho_vazio) ENTÃO
        Início
            Jogar cacho no lixo
        Fim
    SENÃO
        Início
            Guardar cacho na geladeira
        Fim
FIM_ALGORITMO
```

Algoritmo comer um cacho de uva



Algoritmo comer um cacho de uva



Tipos de Estruturas de Repetição

- Existem três tipos básicos de estruturas de repetição:
  - Com número definido de repetições
    - Estrutura **PARA - FOR**
  - Com número indefinido de repetições e teste no início
    - Estrutura **ENQUANTO - WHILE**
  - Com número indefinido de repetições e teste no final
    - Estrutura **REPITA - DO WHILE**

Tipos de Estruturas de Repetição

- Existem três tipos básicos de estruturas de repetição:
  - Com número definido de repetições
    - Estrutura **PARA - FOR**
  - Com número indefinido de repetições e teste no início
    - Estrutura **ENQUANTO - WHILE**
  - Com número indefinido de repetições e teste no final
    - Estrutura **REPITA - DO WHILE**

FOR - Tipos de Estruturas de Repetição

- Número definido de repetições: **PARA**
  - Utilizada quando se sabe o número de repetições que o trecho do algoritmo deve ser repetido (teste controlado por contador)
  - Sintaxe:

FOR - Tipos de Estruturas de Repetição

- Número definido de repetições: **PARA**
  - Utilizada quando se sabe o número de repetições que o trecho do algoritmo deve ser repetido (teste controlado por contador)
  - Sintaxe:

```
int main(){
	int numero;
	for (numero = 1; numero <= 1000; numero = numero + 1){
		printf("%d", numero);
	}
}
```

Obs: **início do loop**, **fim do loop**, **incremento/decremento do loop**, **parte do código que vai repetir (entre as { })**

FOR - Tipos de Estruturas de Repetição

- Número definido de repetições: **PARA**
  - Utilizada quando se sabe o número de repetições que o trecho do algoritmo deve ser repetido (teste controlado por contador)
  - Sintaxe:

```
int main(){
	for (int i = 1; i <= 1000; i++){
		printf("%d", i);
	}
}
```

Resultado:

123456...9989991000

Obs: geralmente usa-se o i como variável do for

FOR - Tipos de Estruturas de Repetição

- Número definido de repetições: **PARA**
  - Utilizada quando se sabe o número de repetições que o trecho do algoritmo deve ser repetido (teste controlado por contador)
  - Sintaxe:

```
int main(){
	for (int i = 1; i <= 1000; i++){
		printf("%d - ", i);
	}
}
```

Resultado:

1 - 2 - 3 - 4 ... 998 - 999 - 1000 -

Obs: ajustando o código

FOR - Tipos de Estruturas de Repetição

- Número definido de repetições: **PARA**
  - Utilizada quando se sabe o número de repetições que o trecho do algoritmo deve ser repetido (teste controlado por contador)
  - Sintaxe:

```
int main(){
      for (int i = 1; i <= 1000; i++){
            printf("\nvalor de i:");
            printf("%d", i);
      }
}
```

Resultado:

valor de i: 1
valor de i: 2
...
valor de i: 1000

Obs: pode-se usar vários comandos dentro do for (entre { })

FOR - Tipos de Estruturas de Repetição

- Número definido de repetições: **PARA**
  - Utilizada quando se sabe o número de repetições que o trecho do algoritmo deve ser repetido (teste controlado por contador)
  - Sintaxe:

```
int main(){
    for (int i = 1; i <= 1000; i++){
        if (i %  10 == 0){
            printf("\n%d", i);
        }
    }
}
```

Resultado:

10

20

...

990

1000

Obs: pode-se usar vários comandos dentro do for (entre { }) - inclusive if

FOR - estrutura

Utilizada quando se sabe o número de repetições que o trecho do algoritmo deve ser repetido (teste controlado por contador)

```
for (inicio; parada; alteração a cada repetição){
    comando1
}
```

comando1 será executado utilizando repetidas vezes respeitando início, parada e alteração

```
for (inicio; parada; alteração a cada repetição){
    comando1
    comando2
}
```

comando1 e comando2 serão executados utilizando repetidas vezes respeitando início, parada e alteração

.

Exemplo 2 - For

```
int main(){
	for (int j = 1; j <= 1000; j = j + 2){
		printf("\nValor j: %d", j);
	}
}
```

O que acontecerá???

Exemplo 2 - For

```
int main(){
      for (int j = 1; j <= 1000; j = j + 2){
            printf("\nValor j: %d", j);
      }
}
```

Resultado:
Valor de j: 1
Valor de j: 3
Valor de j: 5
...
Valor de j: 997
Valor de j: 999

Exemplo 3 - For

```
int main(){
      for (int numero = 0; numero < 900; numero = numero + 5){
            printf("\nValor n: %d", numero);
      }
}
```

O que acontecerá???

Exemplo 3 - For

```
int main(){
	for (int numero = 0; numero < 900; numero = numero + 5){
		printf("\nValor n: %d", numero);
	}
}
```

Resultado:
Valor de n: 0
Valor de n: 5
Valor de n: 10
...
Valor de n: 895

Exemplo 4 - For

```
int main(){
      for (int numero = 50; numero >= 0; numero--){
            printf("\nValor n: %d", numero);
      }
}
```

O que acontecerá???

Exemplo 4 - For

```
int main(){
      for (int numero = 50; numero >= 0; numero--){
            printf("\nValor n: %d", numero);
      }
}
```

Resultado:
Valor de n: 50
Valor de n: 49
Valor de n: 48
...
valor de n: 1
Valor de n: 0

Exercícios

1. Contagem Simples - Escreva um programa que exiba os números de 1 a 100 usando um laço for. Exemplo de saída: 1 - 2 - 3 - 4 - … 98 - 99 - 100.

2. Contagem Regressiva - Faça um programa que exiba os números de 50 a 1 usando um laço while. Exemplo de saída: 50 - 49 - 48 - … 3 - 2 - 1.

3. Soma dos Números - Leia um número inteiro n e exiba a soma de todos os números de 1 até n usando um for.

Tipos de Estruturas de Repetição

- Existem três tipos básicos de estruturas de repetição:
  - Com número definido de repetições
    - Estrutura **PARA - FOR**
  - Com número indefinido de repetições e teste no início
    - Estrutura **ENQUANTO - WHILE**
  - Com número indefinido de repetições e teste no final
    - Estrutura **REPITA - DO WHILE**

While - Tipos de Estruturas de Repetição

- Nº indefinido de repetições e teste no início: ENQUANTO
  - Vantagem: não necessita conhecer o nº de repetições
  - Sintaxe:

While - Tipos de Estruturas de Repetição

- Nº indefinido de repetições e teste no início: ENQUANTO
  - Vantagem: não necessita conhecer o nº de repetições
  - Sintaxe:

```
int main(){
    while (condição){
        comando1
    }
}
```

O **comando1** será executado **enquanto** a **condição** for verdadeira

While - Tipos de Estruturas de Repetição

- Nº indefinido de repetições e teste no início: ENQUANTO
  - Vantagem: não necessita conhecer o nº de repetições
  - Sintaxe:

```
int main(){
    int numero = 1;
    while (numero <=1000){
        printf("%d", numero);
        numero = numero + 1;
    }
}
```

O **comando1** será executado **enquanto** a **condição** for verdadeira

Exemplo 2 - While

```
int main(){
        int j = 1;
        while (j <= 1000){
                printf("\nValor j: %d", j);
                j = j + 2
        }
}
```

O que acontecerá???

Exemplo 2 - While

```
int main(){
        int j = 1;
        while (j <= 1000){
                printf("\nValor j: %d", j);
                j = j + 2
        }
}
```

Resultado:
Valor de j: 1
Valor de j: 3
Valor de j: 5
...
Valor de j: 997
Valor de j: 999

For x While

```
int main(){
	for (int j = 1; j <= 1000; j = j + 2){
		printf("\nValor j: %d", j);
	}
}
```

```
int main(){
	int j = 1;
	while (j <= 1000){
		printf("\nValor j: %d", j);
		j = j + 2
	}
}
```

For x While

```
int main(){
	for (int j = 1; j <= 1000; j = j + 2){
		printf("\nValor j: %d", j);
	}
}
```

```
int main(){
	int j = 1;
	while (j <= 1000){
		printf("\nValor j: %d", j);
		j = j + 2
	}
}
```

Exemplo 3 - while

```
int main(){
	int numero = 0;
	while (numero < 900){
		printf("\nValor n: %d", numero);
		numero = numero + 5;
	}
}
```

O que acontecerá???

Exemplo 3 - while

```
int main(){
	int numero = 0;
	while (numero < 900){
		printf("\nValor n: %d", numero);
		numero = numero + 5;
	}
}
```

Resultado:
Valor de n: 0
Valor de n: 5
Valor de n: 10
...
Valor de n: 895

Exemplo 3 - while

```
int main(){
	int numero = 0;
	while (numero <= 900){
		printf("\nValor n: %d", numero);
		numero = numero + 5;
	}
}
```

Resultado:
Valor de n: 0
Valor de n: 5
Valor de n: 10
...
Valor de n: 895
Valor de n: 900

Exemplo 4 - while

```
int main(){
	int numero = 10;
	while (numero != 0){
		printf(“\nDigite um numero: ”);
		scanf(“%d”, &numero);
		printf(“\n%d”, numero);
	}
}
```

O que acontecerá???

Exemplo 4 - while

```
int main(){
	int numero = 10;
	while (numero != 0){
		printf(“\nDigite um numero: ”);
		scanf(“%d”, &numero);
		printf(“\n%d”, numero);
	}
}
```

Resultado:
O código vai ler números enquanto o valor lido for diferente de 0

Exercícios

6. Média de Números - Peça ao usuário para inserir 5 números e calcule a média desses números usando do while.

10. Média de Notas com Validação - Solicite notas de um aluno até que ele insira uma nota válida (entre 0 e 10). Use do while para garantir a validação.

Tipos de Estruturas de Repetição

- Existem três tipos básicos de estruturas de repetição:
  - Com número definido de repetições
    - Estrutura **PARA - FOR**
  - Com número indefinido de repetições e teste no início
    - Estrutura **ENQUANTO - WHILE**
  - Com número indefinido de repetições e teste no final
    - Estrutura **REPITA - DO WHILE**

Do - While - Tipos de Estruturas de Repetição

- Nº indefinido de repetições e teste no final: **REPITA**
  - Vantagem: não necessita conhecer o nº de repetições
  - Sintaxe:

Do - While - Tipos de Estruturas de Repetição

- Nº indefinido de repetições e teste no final: **REPITA**
  - Vantagem: não necessita conhecer o nº de repetições
  - Sintaxe:

```
int main(){
    do{
        comando1
        comando2
    }while (condição)
}
```

Os **comandos** se **repetirão** até que a **condição** se torne **verdadeira**

Exemplo 3 - do - while

```
int main(){
	int numero = 0;
	do{
		printf("\nValor n: %d", numero);
		numero = numero + 5;
	}while (numero < 900)
}
```

O que acontecerá???

Exemplo 3 - do - while

```
int main(){
	int numero = 0;
	do{
		printf("\nValor n: %d", numero);
		numero = numero + 5;
	}while (numero < 900)
}
```

Resultado:
Valor de n: 0
Valor de n: 5
Valor de n: 10
...
Valor de n: 895

Exemplo 4 - do - while

```
int main(){
	int numero = 10;
	do{
		printf(“\nDigite um numero: ”);
		scanf(“%d”, &numero);
		printf(“\n%d”, numero);
	}while (numero != 0)
}
```

O que acontecerá???

Exemplo 4 - do - while

```
int main(){
	int numero = 10;
	do{
		printf(“\nDigite um numero: ”);
		scanf(“%d”, &numero);
		printf(“\n%d”, numero);
	}while (numero != 0)
}
```

Resultado:
O código vai ler números enquanto o valor lido for diferente de 0

Exemplo 4 - do - while

```
int main(){
      int numero;
      do{
            printf("\nDigite um numero: ");
            scanf("%d", &numero);
            printf("\n%d", numero);
      }while (numero != 0)
}
```

Obs: como a verificação está no fim, eu poderia deixar numero sem um valor inicial

While x Do - While

```
int main(){
      int numero;
      do{
            printf("\nDigite numero: ");
            scanf("%d", &numero);
            printf("\n%d", numero);
      }while (numero != 0)
}
```

```
int main(){
      int numero = 10;
      while (numero != 0){
            printf("\nDigite numero: ");
            scanf("%d", &numero);
            printf("\n%d", numero);
      }
}
```

Teste de mesa – Exemplo

```
int main(){
        int x;
        for (x = 0; x < 5; x++){
                printf("%d", x);
        }
}
```

Tela	x	
	0	Valor inicial
Valor de x = 0	1	Valores obtidos dentro da estrutura da repetição
Valor de x = 1	2	
Valor de x = 2	3	
Valor de x = 3	4	
Valor de x = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de x depois que sair da estrutura = 5		

Flags como Variáveis de Controle

- Uma flag é uma variável (geralmente do tipo int ou bool) usada para representar um estado (por exemplo, ”continuar” ou ”interromper”).
- O uso de uma flag é vantajoso quando a condição para interromper o loop não é algo que se possa definir de forma direta e única na expressão do while.
- Outros casos que o uso de flags é vantajoso:
    - Interrupção em pontos específicos do loop.
    - Melhor legibilidade.

Exemplo de uso da flag

Desenvolva um programa que solicite a entrada de 10 números inteiros, um de cada vez.

Se o número 5 for digitado durante o processo, o programa deve interromper imediatamente as leituras e exibir a mensagem ”Número encontrado!”.

Se, após as 10 entradas, o número 5 não tiver sido informado, o programa deverá exibir a mensagem ”O número não foi encontrado.”.

Exemplo de uso da flag

```
#include <stdio.h>
#include <stdbool.h>

int main() {
    int numero, contador;
    bool flag = false;
    contador = 0;

    while (contador < 10 && !flag) {
        printf("Digite um Número:\n");
        scanf("%d", &numero);

        if (numero == 5) {
            flag = true;
        }
        contador++;
    }

    if (flag) {
        printf("Número encontrado!\n");
    }
    else {
        printf("Número não encontrado.\n");
    }

    return 0;
}
```

Exemplo de uso da flag como int

```
#include <stdio.h>
#include <stdbool.h>

int main() {
    int numero, contador;
    int flag = 0;
    contador = 0;

    while (contador < 10 && flag == 0) {
        printf("Digite um Número:\n");
        scanf("%d", &numero);

        if (numero == 5) {
            flag = 1;
        }
        contador++;
    }

    if (flag == 1) {
        printf("Número encontrado!\n");
    }
    else {
        printf("Número não encontrado.\n");
    }

    return 0;
}
```

Comando break

- O break é uma instrução usada para interromper imediatamente a execução de um laço (while, for) ou de um switch.

- Quando o break é executado, o controle do programa salta para a próxima instrução após o bloco onde ele está inserido.

Exemplo de uso do break

Desenvolva um programa que solicite a entrada de 10 números inteiros, um de cada vez.

Se o número 5 for digitado durante o processo, o programa deve interromper imediatamente as leituras e exibir a mensagem ”Número encontrado!”.

Se, após as 10 entradas, o número 5 não tiver sido informado, o programa deverá exibir a mensagem ”O número não foi encontrado.”.

Exemplo de uso do break

```
#include <stdio.h>
#include <stdbool.h>

int main() {
    int numero, contador;
    bool flag = false;
    contador = 0;

    while (contador < 10 && !flag) {
        printf("Digite um Número:\n");
        scanf("%d", &numero);

        if (numero == 5) {
            flag = true;
            break;
        }
        contador++;
    }

    if (flag) {
        printf("Número encontrado!\n");
    }
    else {
        printf("Número não encontrado.\n");
    }

    return 0;
}
```

Exercício - Desafio!

Faça um programa que imprime a soma e a média dos salários dos funcionários de uma empresa.

Os salários serão digitados pelo usuário.

A leitura dos salários é interrompida quando o usuário digitar um salário menor que 0. O programa deve imprimir, também, a quantidade de salários digitados.

Comando FOR - operador vírgula

- Qualquer uma das expressões do for pode conter várias instruções separadas por vírgulas.
- A vírgula é um operador que significa ”faça isso e depois isso”.
- Um par de expressões separadas por vírgula é avaliado da esquerda para direita

Comando FOR - operador vírgula - Exemplo

Imprimir os números de 0 a 10 e de 10 a 0 ao mesmo tempo

Comando FOR - operador vírgula - Exemplo

Imprimir os números de 0 a 10 e de 10 a 0 ao mesmo tempo

```
int main(){
        int i,j;
        for(i=0, j=10; i<=10; i++, j--){
                printf("\n%d | %d ", i, j);
        }
        return 0;
}
```

Obs: **início do loop**, **fim do loop**,
incremento/decremento do loop,
parte do código que vai repetir (entre as { })

Comando FOR - operador vírgula - Exemplo

Imprimir os números de 0 a 10 e de 10 a 0 ao mesmo tempo

```
int main(){
        int i,j;
        for(i=0, j=10; i<=10; i++, j--){
                printf("\n%d | %d ", i, j);
        }
        return 0;
}
```

Resultado:

0 | 10
1 | 9
2 | 8
...
9 | 1
10 | 0

PUC Minas

Dúvidas?