

SI100

Algoritmos e Programação de Computadores I

1º Semestre - 2018

Tópico 4: Estruturas de Controle – Repetição

Unidade 8

Prof. Guilherme Palermo Coelho
guilherme@ft.unicamp.br

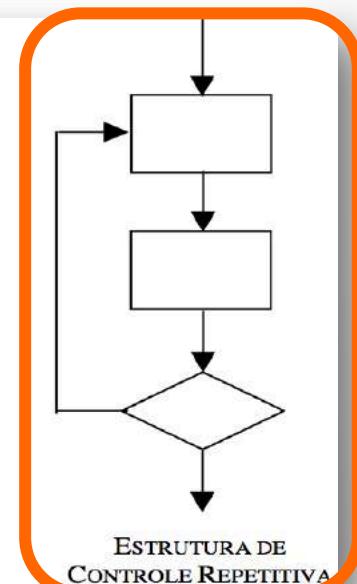
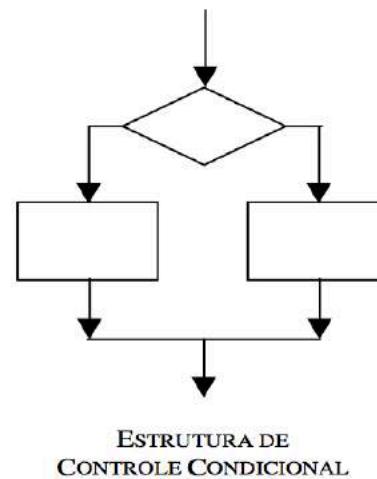
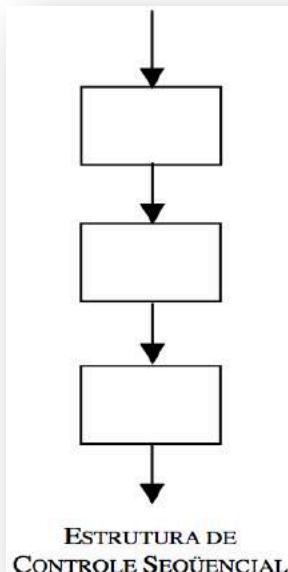
Roteiro

- Introdução;
- O laço *for*;
- O laço *while*;
- O laço *do-while*;
- Os comandos *break* e *continue*;
- Exercícios;
- Referências.

INTRODUÇÃO

Programação Estruturada

- Como vimos nas aulas anteriores, C é uma linguagem de **programação estruturada**:
 - Um programa em C é composto por **blocos elementares** que são interligados através de **três** mecanismos básicos:
 - Sequência;
 - Seleção;
 - Iteração.



Comandos de Repetição

- **Exemplo:**
 - 1) Acionar o interruptor;
 - 2) **Enquanto a lâmpada não acender, faça:**
 - 3) Posicionar a escada embaixo da lâmpada;
 - 4) Pegar uma lâmpada nova;
 - 5) Subir na escada;
 - 6) Retirar a lâmpada velha;
 - 7) Colocar a lâmpada nova;
 - 8) Descer da escada;



Esta sequência de passos de um programa, que é repetida um certo número de vezes, é conhecida como **laço** (ou *loop*).

O LAÇO *FOR*

O laço *FOR*

- O laço *for* engloba 3 expressões em uma única;
- É útil principalmente quando é necessário repetir algo um número fixo de vezes.
- Forma geral de um laço *for*:

```
for (inicialização; teste; incremento)
    instrução ou bloco de instruções
```

- A expressão de um *for* como um todo é chamada de *expressão de laço*.
 - É dividida em *expressão de inicialização*, *expressão de teste* e *expressão de incremento*.

O laço *FOR*

```
for (inicialização; teste; incremento)  
    instrução ou bloco de instruções
```

- As três expressões podem ser compostas por quaisquer instruções válidas em C;
- Inicialização: em sua forma mais simples, é uma atribuição (a uma variável de controle);
 - Executada uma única vez antes do início do laço.
- Teste: é uma instrução de condição que *controla* o laço.
 - É avaliada **toda vez** que o laço for iniciado ou reiniciado;
 - Se **verdadeira**, corpo do laço é executado.
 - Se **falsa**, o laço é encerrado e o controle passa para a instrução seguinte.

O laço *FOR*

```
for (inicialização; teste; incremento)  
    instrução ou bloco de instruções
```

- Incremento: define a forma com que a *variável de controle* será alterada a cada repetição do laço;
 - É executada sempre ao final de cada repetição do laço.
- As expressões de um **for** SEMPRE devem estar separadas por ponto-e-vírgula (“;”);
- Se o corpo do **for** for um bloco de instruções, elas devem estar entre “{“ “}” (de maneira análoga ao que foi visto para **if**)
- NUNCA coloque ponto-e-vírgula entre o “)” após a expressão de incremento e o “{“ do bloco de instruções.

O laço *FOR* - exemplo

- Programa que conte de 0 a 3:

```
/*Teste do comando FOR - contar até 3*/
#include <stdio.h>

int main() {
    int conta; // variável de controle do loop
    for (conta=0; conta <= 3; conta++)
        printf("conta=%d\n", conta);

    return 0;
}
```

```
conta=0
conta=1
conta=2
conta=3
```

O laço *FOR* - exemplo

- Programa que conte de 3 a 0 (em ordem **decrescente**):

```
/*Teste do comando FOR - contar até 3*/
#include <stdio.h>

int main() {
    int conta;
    for (conta=3; conta >= 0; conta--)
        printf("Conta=%d\n", conta);

    return 0;
}
```

Conta=3

Conta=2

Conta=1

Conta=0

O laço *FOR* - flexibilidade

- O comando *for* apresenta uma grande flexibilidade quanto às expressões de **inicialização, teste e incremento**;
 - Cada uma delas pode conter **várias instruções**, separadas por **vírgula** (as múltiplas instruções são avaliadas da esquerda para a direita);

```
/*Imprimir num. de 0 a 98 com incremento de 2*/
#include <stdio.h>

int main() {
    int x,y;
    for (x=0, y=0; x+y < 100; x=x+1, y=y+1)
        printf("%d ", x+y);
    return 0;
}
```

O laço *FOR* - flexibilidade

- O comando *for* apresenta uma grande flexibilidade quanto às expressões de **inicialização, teste e incremento**;
 - É possível usar **caracteres** ao invés de valores numéricos:

```
/*Imprimir letras minúsculas do alfabeto e seu código ASCII*/
#include <stdio.h>

int main() {
    char ch;

    for (ch='a' ; ch < 'z' ; ch++)
        printf("O valor ASCII de %c é %d.\n", ch, ch);

    return 0;
}
```

O laço *FOR* - flexibilidade

- O comando *for* apresenta uma grande flexibilidade quanto às expressões de **inicialização, teste e incremento**;
 - Podem ser chamadas **funções** em qualquer expressão do laço:

```
/* Lê um caractere e imprime o seguinte até que seja X*/
#include <stdio.h>

int main() {
    char ch;

    for (ch=getchar(); ch != 'X'; ch=getchar())
        printf("%c", ch+1);

    return 0;
}
```

O laço *FOR* - flexibilidade

- O comando *for* apresenta uma grande flexibilidade quanto às expressões de **inicialização, teste e incremento**;
 - Qualquer uma das três expressões de um *for* pode ser **omitida**;
 - Os separadores (“;”) **devem** ser mantidos.
 - Se a expressão de **teste** for omitida, ela é considerada **permanentemente verdadeira**.

```
#include <stdio.h>

int main() {
    for (;;)
        printf("Loop infinito\n");

    return 0;
}
```

O laço *FOR* - flexibilidade

- O **corpo** do laço também pode ser **vazio**.
- Neste caso, é necessário colocar um “;” para indicar o corpo vazio.

```
#include <stdio.h>

int main() {
    char c;
    for (; (c = getchar()) != 'X' ; printf("%c", c+1))
        ;
    return 0;
}
```

O Programa acima lê um caractere digitado pelo usuário e exibe o caractere seguinte, até que seja digitado X. É equivalente ao do slide 14.

Laços *FOR* aninhados

- Da mesma maneira que vimos para *if*, laços *for* podem estar *aninhados* (um no corpo de outro).
- Exemplo: programa que imprime a tabuada

TABUADA	DO	2	TABUADA	DO	3	TABUADA	DO	4	TABUADA	DO	5
$2 \times 1 = 2$			$3 \times 1 = 3$			$4 \times 1 = 4$			$5 \times 1 = 5$		
$2 \times 2 = 4$			$3 \times 2 = 6$			$4 \times 2 = 8$			$5 \times 2 = 10$		
$2 \times 3 = 6$			$3 \times 3 = 9$			$4 \times 3 = 12$			$5 \times 3 = 15$		
$2 \times 4 = 8$			$3 \times 4 = 12$			$4 \times 4 = 16$			$5 \times 4 = 20$		
$2 \times 5 = 10$			$3 \times 5 = 15$			$4 \times 5 = 20$			$5 \times 5 = 25$		
$2 \times 6 = 12$			$3 \times 6 = 18$			$4 \times 6 = 24$			$5 \times 6 = 30$		
$2 \times 7 = 14$			$3 \times 7 = 21$			$4 \times 7 = 28$			$5 \times 7 = 35$		
$2 \times 8 = 16$			$3 \times 8 = 24$			$4 \times 8 = 32$			$5 \times 8 = 40$		
$2 \times 9 = 18$			$3 \times 9 = 27$			$4 \times 9 = 36$			$5 \times 9 = 45$		
TABUADA	DO	6	TABUADA	DO	7	TABUADA	DO	8	TABUADA	DO	9
$6 \times 1 = 6$			$7 \times 1 = 7$			$8 \times 1 = 8$			$9 \times 1 = 9$		
$6 \times 2 = 12$			$7 \times 2 = 14$			$8 \times 2 = 16$			$9 \times 2 = 18$		
$6 \times 3 = 18$			$7 \times 3 = 21$			$8 \times 3 = 24$			$9 \times 3 = 27$		
$6 \times 4 = 24$			$7 \times 4 = 28$			$8 \times 4 = 32$			$9 \times 4 = 36$		
$6 \times 5 = 30$			$7 \times 5 = 35$			$8 \times 5 = 40$			$9 \times 5 = 45$		
$6 \times 6 = 36$			$7 \times 6 = 42$			$8 \times 6 = 48$			$9 \times 6 = 54$		
$6 \times 7 = 42$			$7 \times 7 = 49$			$8 \times 7 = 56$			$9 \times 7 = 63$		
$6 \times 8 = 48$			$7 \times 8 = 56$			$8 \times 8 = 64$			$9 \times 8 = 72$		
$6 \times 9 = 54$			$7 \times 9 = 63$			$8 \times 9 = 72$			$9 \times 9 = 81$		

Laços *FOR* aninhados

- Exemplo: programa que imprime a tabuada

```
#include <stdio.h>

int main() {
    int i, j, k;

    printf("\n");

    for (k=0; k<=1; k++) {
        printf("\n");
        for (i=1; i<5; i++)
            printf("TABUADA DO %2d\t", i+4*k+1);
        printf("\n");
        for (i=1; i<=9; i++) {
            for (j=2+4*k; j<=5+4*k; j++)
                printf("%2d x %-2d = %2d\t", j, i, j*i);
            printf("\n");
        }
    }
    return 0;
}
```

Laços *FOR* aninhados

- Exemplo: programa que imprime a tabuada

TABUADA	DO	2	TABUADA	DO	3	TABUADA	DO	4	TABUADA	DO	5
$2 \times 1 = 2$			$3 \times 1 = 3$			$4 \times 1 = 4$			$5 \times 1 = 5$		
$2 \times 2 = 4$			$3 \times 2 = 6$			$4 \times 2 = 8$			$5 \times 2 = 10$		
$2 \times 3 = 6$			$3 \times 3 = 9$			$4 \times 3 = 12$			$5 \times 3 = 15$		
$2 \times 4 = 8$			$3 \times 4 = 12$			$4 \times 4 = 16$			$5 \times 4 = 20$		
$2 \times 5 = 10$			$3 \times 5 = 15$			$4 \times 5 = 20$			$5 \times 5 = 25$		
$2 \times 6 = 12$			$3 \times 6 = 18$			$4 \times 6 = 24$			$5 \times 6 = 30$		
$2 \times 7 = 14$			$3 \times 7 = 21$			$4 \times 7 = 28$			$5 \times 7 = 35$		
$2 \times 8 = 16$			$3 \times 8 = 24$			$4 \times 8 = 32$			$5 \times 8 = 40$		
$2 \times 9 = 18$			$3 \times 9 = 27$			$4 \times 9 = 36$			$5 \times 9 = 45$		

k=0

TABUADA	DO	6	TABUADA	DO	7	TABUADA	DO	8	TABUADA	DO	9
$6 \times 1 = 6$			$7 \times 1 = 7$			$8 \times 1 = 8$			$9 \times 1 = 9$		
$6 \times 2 = 12$			$7 \times 2 = 14$			$8 \times 2 = 16$			$9 \times 2 = 18$		
$6 \times 3 = 18$			$7 \times 3 = 21$			$8 \times 3 = 24$			$9 \times 3 = 27$		
$6 \times 4 = 24$			$7 \times 4 = 28$			$8 \times 4 = 32$			$9 \times 4 = 36$		
$6 \times 5 = 30$			$7 \times 5 = 35$			$8 \times 5 = 40$			$9 \times 5 = 45$		
$6 \times 6 = 36$			$7 \times 6 = 42$			$8 \times 6 = 48$			$9 \times 6 = 54$		
$6 \times 7 = 42$			$7 \times 7 = 49$			$8 \times 7 = 56$			$9 \times 7 = 63$		
$6 \times 8 = 48$			$7 \times 8 = 56$			$8 \times 8 = 64$			$9 \times 8 = 72$		
$6 \times 9 = 54$			$7 \times 9 = 63$			$8 \times 9 = 72$			$9 \times 9 = 81$		

k=1

Laços *FOR* aninhados

- Exemplo: programa que imprime a tabuada

```
#include <stdio.h>

int main() {
    int i, j, k;

    printf("\n");
    for (k=0; k<=1; k++) {
        printf("\n");
        for (i=1; i<5; i++)
            printf("TABUADA DO %2d\t", i+4*k+1);
        printf("\n");
        for (i=1; i<=9; i++) {
            for (j=2+4*k; j<=5+4*k; j++)
                printf("%2d x %-2d = %2d\t", j, i, j*i);
            printf("\n");
        }
    }
    return 0;
}
```



Laços *FOR* aninhados

- Exemplo: programa que imprime a tabuada

i=1	i=2	i=3	i=4	k=0
TABUADA DO 2	TABUADA DO 3	TABUADA DO 4	TABUADA DO 5	
$2 \times 1 = 2$	$3 \times 1 = 3$	$4 \times 1 = 4$	$5 \times 1 = 5$	
$2 \times 2 = 4$	$3 \times 2 = 6$	$4 \times 2 = 8$	$5 \times 2 = 10$	
$2 \times 3 = 6$	$3 \times 3 = 9$	$4 \times 3 = 12$	$5 \times 3 = 15$	
$2 \times 4 = 8$	$3 \times 4 = 12$	$4 \times 4 = 16$	$5 \times 4 = 20$	
$2 \times 5 = 10$	$3 \times 5 = 15$	$4 \times 5 = 20$	$5 \times 5 = 25$	
$2 \times 6 = 12$	$3 \times 6 = 18$	$4 \times 6 = 24$	$5 \times 6 = 30$	
$2 \times 7 = 14$	$3 \times 7 = 21$	$4 \times 7 = 28$	$5 \times 7 = 35$	
$2 \times 8 = 16$	$3 \times 8 = 24$	$4 \times 8 = 32$	$5 \times 8 = 40$	
$2 \times 9 = 18$	$3 \times 9 = 27$	$4 \times 9 = 36$	$5 \times 9 = 45$	
TABUADA DO 6	TABUADA DO 7	TABUADA DO 8	TABUADA DO 9	k=1
$6 \times 1 = 6$	$7 \times 1 = 7$	$8 \times 1 = 8$	$9 \times 1 = 9$	
$6 \times 2 = 12$	$7 \times 2 = 14$	$8 \times 2 = 16$	$9 \times 2 = 18$	
$6 \times 3 = 18$	$7 \times 3 = 21$	$8 \times 3 = 24$	$9 \times 3 = 27$	
$6 \times 4 = 24$	$7 \times 4 = 28$	$8 \times 4 = 32$	$9 \times 4 = 36$	
$6 \times 5 = 30$	$7 \times 5 = 35$	$8 \times 5 = 40$	$9 \times 5 = 45$	
$6 \times 6 = 36$	$7 \times 6 = 42$	$8 \times 6 = 48$	$9 \times 6 = 54$	
$6 \times 7 = 42$	$7 \times 7 = 49$	$8 \times 7 = 56$	$9 \times 7 = 63$	
$6 \times 8 = 48$	$7 \times 8 = 56$	$8 \times 8 = 64$	$9 \times 8 = 72$	
$6 \times 9 = 54$	$7 \times 9 = 63$	$8 \times 9 = 72$	$9 \times 9 = 81$	

Laços *FOR* aninhados

- Exemplo: programa que imprime a tabuada

```
#include <stdio.h>

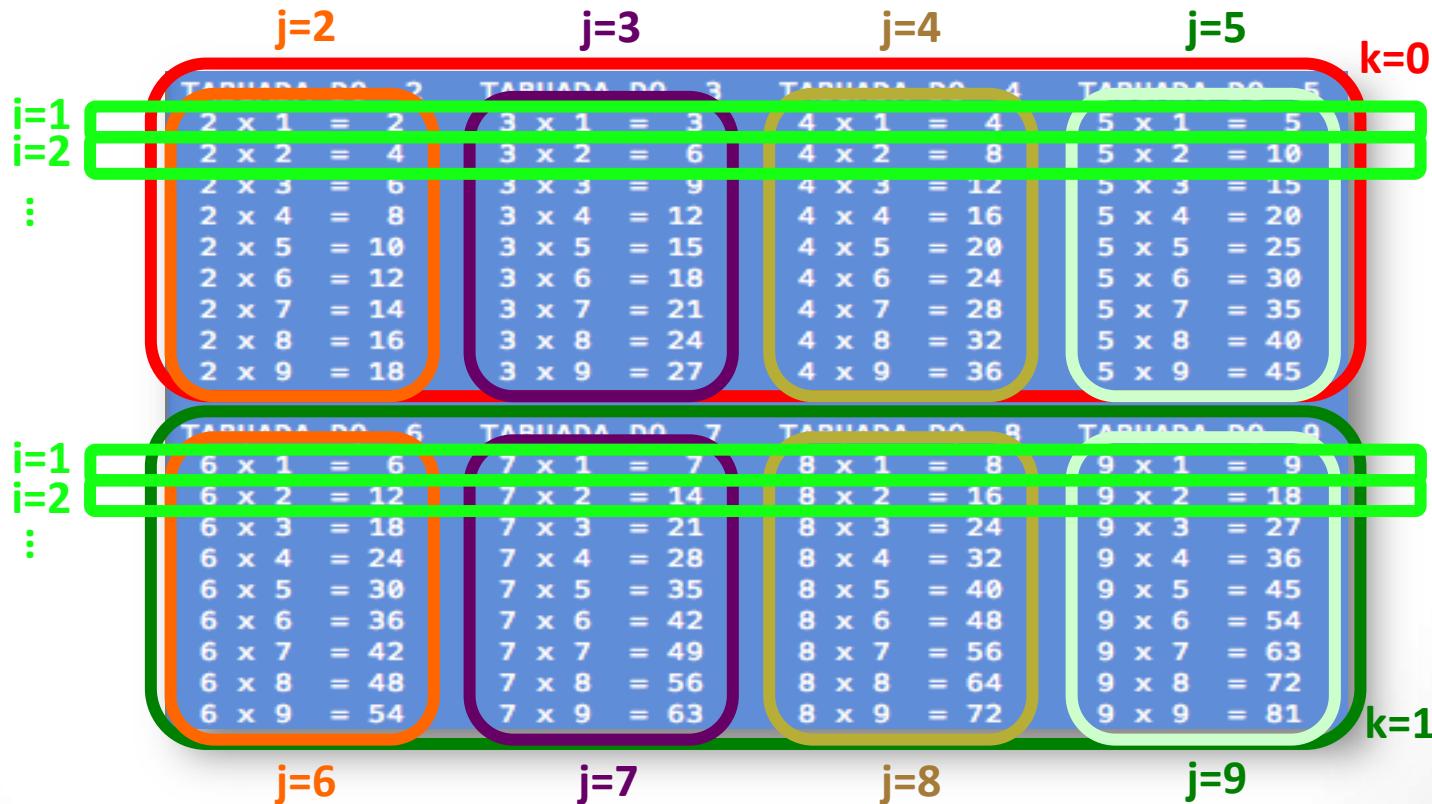
int main() {
    int i, j, k;

    printf("\n");
    for (k=0; k<=1; k++) {
        printf("\n");
        for (i=1; i<5; i++)
            printf("TABUADA DO %2d\t", i+4*k+1);
        printf("\n");
        for (i=1; i<=9; i++) {
            for (j=2+4*k; j<=5+4*k; j++)
                printf("%2d x %-2d = %2d\t", j, i, j*i);
            printf("\n");
        }
    }
    return 0;
}
```



Laços *FOR* aninhados

- Exemplo: programa que imprime a tabuada



Laços *FOR* aninhados

- Exemplo: programa que imprime a tabuada

TABUADA	DO	2	TABUADA	DO	3	TABUADA	DO	4	TABUADA	DO	5
$2 \times 1 = 2$			$3 \times 1 = 3$			$4 \times 1 = 4$			$5 \times 1 = 5$		
$2 \times 2 = 4$			$3 \times 2 = 6$			$4 \times 2 = 8$			$5 \times 2 = 10$		
$2 \times 3 = 6$			$3 \times 3 = 9$			$4 \times 3 = 12$			$5 \times 3 = 15$		
$2 \times 4 = 8$			$3 \times 4 = 12$			$4 \times 4 = 16$			$5 \times 4 = 20$		
$2 \times 5 = 10$			$3 \times 5 = 15$			$4 \times 5 = 20$			$5 \times 5 = 25$		
$2 \times 6 = 12$			$3 \times 6 = 18$			$4 \times 6 = 24$			$5 \times 6 = 30$		
$2 \times 7 = 14$			$3 \times 7 = 21$			$4 \times 7 = 28$			$5 \times 7 = 35$		
$2 \times 8 = 16$			$3 \times 8 = 24$			$4 \times 8 = 32$			$5 \times 8 = 40$		
$2 \times 9 = 18$			$3 \times 9 = 27$			$4 \times 9 = 36$			$5 \times 9 = 45$		
TABUADA	DO	6	TABUADA	DO	7	TABUADA	DO	8	TABUADA	DO	9
$6 \times 1 = 6$			$7 \times 1 = 7$			$8 \times 1 = 8$			$9 \times 1 = 9$		
$6 \times 2 = 12$			$7 \times 2 = 14$			$8 \times 2 = 16$			$9 \times 2 = 18$		
$6 \times 3 = 18$			$7 \times 3 = 21$			$8 \times 3 = 24$			$9 \times 3 = 27$		
$6 \times 4 = 24$			$7 \times 4 = 28$			$8 \times 4 = 32$			$9 \times 4 = 36$		
$6 \times 5 = 30$			$7 \times 5 = 35$			$8 \times 5 = 40$			$9 \times 5 = 45$		
$6 \times 6 = 36$			$7 \times 6 = 42$			$8 \times 6 = 48$			$9 \times 6 = 54$		
$6 \times 7 = 42$			$7 \times 7 = 49$			$8 \times 7 = 56$			$9 \times 7 = 63$		
$6 \times 8 = 48$			$7 \times 8 = 56$			$8 \times 8 = 64$			$9 \times 8 = 72$		
$6 \times 9 = 54$			$7 \times 9 = 63$			$8 \times 9 = 72$			$9 \times 9 = 81$		

O LAÇO *WHILE*

O laço *WHILE*

- A estrutura de laço **while** da linguagem C possui os mesmos elementos da estrutura **for**, mas eles ficam distribuídos de forma diferente ao longo do código.
- Forma geral de um laço **while**:

```
while (expressão de teste)
      instrução ou bloco de instruções
```

- Se a **expressão de teste** for verdadeira, o corpo do laço **while** é executado uma vez e a expressão é reavaliada.
- Este **ciclo de teste-execução** é repetido até que a expressão de teste se torne falsa.
 - Quando isto ocorre, o controle passa para a linha seguinte ao laço.

O laço *WHILE*

- Ex.: programa que conta de 0 a 3.

```
#include <stdio.h>

int main() {
    int conta=0;
    while (conta <= 3) {
        printf("conta=%d\n", conta);
        conta++;
    }

    return 0;
}
```

```
conta=0
conta=1
conta=2
conta=3
```

O laço *WHILE*

- Dado que temos *estruturas de laços for e while*, quando usar cada um deles?
 - **For:** mais indicado para situações em que se *conhece de antemão* o número de repetições que deverão ser feitas.
 - **While:** é mais indicado para situações em que *não se sabe* com certeza quando a repetição será encerrada.

```
#include <stdio.h>

int main() {
    char ch;
    while((ch = getchar()) != 'X')
        printf("%c\n", ch);
    return 0;
}
```

O laço *WHILE* - exemplo

- Ex.: cálculo do fatorial de um número

```
#include <stdio.h>

int main() {
    int num;
    long resposta;
    while (1) {
        printf("Digite o numero: ");
        scanf("%d", &num);
        resposta = 1;
        while(num > 1)
            resposta *= num--;
        printf("O fatorial eh: %ld\n", resposta);
    }
    return 0; }
```

O LAÇO *DO-WHILE*

O laço *DO-WHILE*

- O laço ***do-while*** é uma estrutura de repetição muito semelhante ao ***while*** ;
- Diferenças:
 - ***While*** faz o teste da condição antes da execução do laço;
 - ***Do-while*** faz o teste da condição depois do laço ser executado;
- Forma geral de um laço ***do-while***:

```
do {  
    instrução ou bloco de instruções  
} while (expressão de teste);
```

Não esquecer do “;”!

O laço *DO-WHILE*

- Exemplo:

```
#include <stdio.h>

int main() {
    char ch;
    do {
        printf("Digite um caractere: ");
        scanf("%c", &ch);

        if ((ch >= 'a') && (ch <= 'z'))
            ch = ch - 32;
        printf("%c\n", ch);
    } while (ch != '=');

    return 0;
}
```

Neste programa, atenção aos caracteres ocultos que são inseridos sempre que se pressiona “Enter”!

Os COMANDOS *BREAK* E *CONTINUE*

O comando *BREAK*

- O comando ***break*** pode ser usado no corpo de qualquer estrutura de laço em C;
- Ele provoca a **saída imediata** do laço;
 - Se estiver em estruturas de laços aninhados, ele afetará ***apenas o laço que o contém*** (e os laços internos a ele);

```
#include <stdio.h>

int main() {
    int i;
    for(i=0; i <= 100; i++) {
        printf("%d\n", i);
        if (i == 5)
            break;
    }
    return 0; }
```

0
1
2
3
4
5

O comando *CONTINUE*

- O comando ***continue*** força o início da próxima iteração do laço, descartando todas as instruções restantes do código atual;
 - Em laços ***while*** e ***do-while*** → execução pula para verificação de condição;
 - Em laços ***for*** → é feito o incremento e depois a verificação de condição

```
#include <stdio.h>

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

0
1
2
3
4
6 ←
7
8
9

EXERCÍCIOS

Exercícios

1. Escreva um programa que imprima o quadrado de todos os inteiros de 1 a 20, um por linha.
2. Faça um programa, utilizando o laço ***while***, que solicite caracteres ao usuário e imprima na tela seus respectivos códigos ASCII. O programa deve terminar quando o usuário digitar o caractere '#' (**dica:** consulte a tabela ASCII - <http://www.asciitable.com/>);
3. Escreva um programa que, usando um laço ***for***, imprima na tela os caracteres de código ASCII 32 a 255 decimal. O programa deve imprimir cada caractere, seu código ASCII em formato decimal e seu código ASCII em formato hexadecimal.

Exercícios

4. O número de combinações de **n** objetos diferentes onde **s** objetos são escolhidos de cada vez é dado pela seguinte fórmula:

$$C_s^n = \binom{n}{s} = \frac{n!}{s! \cdot (n - s)!}$$

Escreva um programa que calcule o número de combinações de **n** objetos tomados **s** de cada vez. Os valores de **n** e **s** devem ser solicitados ao usuário.

5. Usando as estruturas de loop vistas em aula, escreva um programa que leia dez números inteiros e imprima na tela quantos destes números são pares e quantos são ímpares.

Exercícios

6. Uma forma alternativa para encontrar o **quadrado** de um **número inteiro positivo** é dado pelo seguinte algoritmo:

“O quadrado de um número positivo n é igual à soma dos n primeiros números ímpares”

Por exemplo: $3^2 = 1 + 3 + 5 = 9$

Sendo assim, escreva um programa que implemente o algoritmo acima para obtenção do quadrado de um número inteiro **n** positivo. O valor de **n** deve ser solicitado ao usuário.

7. Utilizando as estruturas de **loop** vistas em aula, implemente um programa que escreva na tela o padrão de caracteres dado na figura a seguir:

Exercícios



Exercícios

8. Escreva um programa que encontre o menor número inteiro positivo n que aceite as seguintes condições:

$n/3$ possui resto 2

$n/5$ possui resto 3

$n/7$ possui resto 4.

Exercícios

- **Observação:**
 - Os seguintes exercícios devem ser entregues via SuSy.
 - Exercício 1;
 - Exercício 4;
 - Exercício 5.
 - Veja os enunciados atualizados no site do sistema:
 - <https://susy.ic.unicamp.br:9999/si100a> (Turma A);
 - <https://susy.ic.unicamp.br:9999/si100b> (Turma B);

REFERÊNCIAS

Referências

- MIZRAHI, V. V., *Treinamento em Linguagem C – Curso Completo*, 2a Edição, Pearson Makron Books, 2005.
- MIYAZAWA, F. K., DE SOUZA, C. C. & KOWALTOWSKI, T.. *Notas de Aula da disciplina Algoritmos e Programação de Computadores*. Instituto de Computação, Unicamp.