



LÓGICA DE PROGRAMAÇÃO

PROF^a. M.Sc. JULIANA H Q BENACCHIO

Laços **while** aninhados

- Por exemplo, para criar uma estrutura de tópico

```
1. Capítulo
  1.1 Seção
    1.1.1 Subseção
    1.1.2 Subseção
    1.1.3 Subseção
    1.1.4 Subseção
    1.1.5 Subseção
  1.2 Seção
    1.2.1 Subseção
    1.2.2 Subseção
    1.2.3 Subseção
    1.2.4 Subseção
    1.2.5 Subseção
```

Laços **while** aninhados

1.3 Secao

1.3.1 Subsecao

1.3.2 Subsecao

1.3.3 Subsecao

1.3.4 Subsecao

1.3.5 Subsecao

2. Capitulo

2.1 Secao

2.1.1 Subsecao

2.1.2 Subsecao

2.1.3 Subsecao

2.1.4 Subsecao

2.1.5 Subsecao

2.2 Secao

2.2.1 Subsecao

...

Laços while aninhados

```
int main(){
    int capitulo, secao, subsecao;
    capitulo = 1;
    while (capitulo <= 2)
    {
        printf("\n%d. Capitulo", capitulo);
        secao = 1;
        while (secao <= 3)
        {
            printf("\n\t%d.%d Secao", capitulo, secao);
            subsecao = 1;
            while (subsecao <= 5)
            {
                printf("\n\t\t%d.%d.%d Subsecao", capitulo, secao, subsecao);
                subsecao++;
            }
            secao++;
        }
        capitulo++;
    }
    return 0;
}
```

Laço infinito

- Ocasionalmente, você pode criar um loop infinito.
- Isso talvez inclua situações em que você quer que seu programa execute até o usuário interrompê-lo ou em você quer fornecer uma rota de escape com a instrução **break**.
- Como o valor **true** é condicional, o loop sempre encontrará sua condição como sendo verdadeira.

```
while (1)
```

Comandos **break** e **continue**

- Como vimos no **switch-case**, o comando **break** causa a saída imediata do comando
- Portanto, dentro do **while** o **break** irá causar a saída imediato do laço desconsiderando qualquer instrução abaixo do **break**
- Se o comando **break** estiver em estruturas de **while** aninhados afetará somente o laço que o contém e os laços internos a este

Comandos break e continue

```
int i, fat, n;
while(1)
{
    i = 1;
    fat = 1;
    printf("Calcular o fatorial de N:");
    scanf("%d", &n);
    if (n > 19){
        printf("Nao foi possivel calcular\n");
        break;
    }
    while(i <= n)
    {
        fat = fat * i;
        i++;
    }
    printf("Fatorial = %d\n", fat);
}
```

Comandos **break** e **continue**

- O comando **continue** é parecido com **break**, porém ao executá-lo saltamos para a próxima iteração do loop ao invés de terminá-lo
- Usar **continue** equivale a chegar ao final do bloco e a condição é reavaliada (qualquer que seja o loop atual)


```
int opcao = 0;
while (opcao != 5){
    printf("Escolha uma opção entre 1 e 5: ");
    scanf("%d", &opcao);
    /* se a opção for inválida, volta ao início do loop */
    if (opcao > 5 || opcao < 1){
        printf("\nOpcao invalida\n\n");
        continue;
    }
    switch (opcao){
        case 1:
            printf("\n --> Primeira opcao..\n\n");
            break;
        case 2:
            printf("\n --> Segunda opcao..\n\n");
            break;
        case 3:
            printf("\n --> Terceira opcao..\n\n");
            break;
        case 4:
            printf("\n --> Quarta opcao..\n\n");
            break;
        case 5:
            printf("\n --> Abandonando..\n");
            break;
    }
}
```

Estruturas de Repetição

- Relembrando, na linguagem C existem três tipos de laços:
 - `while`
 - `do-while`
 - `for`

O Laço `while`

- Execução:
 1. avalia a condição;
 2. se a condição for falsa, o comando é encerrado;
 3. se a condição for verdadeira, executa as instruções e retorna para o passo 1.

O Laço `while`



- Sua forma geral é:

```
while (condição)
{
    instrução1;
    instrução2;
}
```

O Laço do-while

- Execução:
 1. executa a sequência de instruções;
 2. avalia condição;
 3. se condição for falsa, encerra o comando;
 4. caso contrário, retorna ao passo 1.

O Laço do-while

- Sua forma geral é:

```
do
{
    instrução1;
    instrução2;
    instrução3;
} while (condição);
```

O Laço `do-while`

- O comando `do-while` tem um comportamento bastante semelhante ao `while`, diferenciando-se apenas em um único ponto: **Não é feito nenhum teste antes de entrar no loop pela primeira vez**
- Por verificar a condição ao final do laço, essa estrutura garante que a sequência de instruções será executada **pelo menos uma vez**

O Laço do-while

- O seguinte laço **do-while** lerá números digitados pelo usuário até que encontre um número maior ou igual a 100.

```
int num;  
do  
{  
    printf("Digite um numero: ");  
    scanf("%d", &num);  
}while(num < 100);
```


O Laço `do-while`

- Em geral, o laço `do-while` é usado para:
 - Garantir consistência de entrada de dados;
 - Repetir um processo com confirmação do usuário;
 - Implementar processos orientados por menus.

Consistência de entrada de dados

```
#include <stdio.h>

#include <math.h>

int main() {

    float n, r;

    do {

        printf("Digite um numero nao negativo: ");

        scanf("%f", &n);

    } while(n < 0);

    r = sqrt(n);

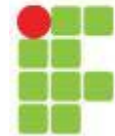
    printf("\nA raiz quadrada de %.0f = %.0f\n", n, r);

    return 0;

}
```

- Para executar corretamente, os programas precisam que o usuário forneça **dados consistentes**. No exemplo anterior, o programa não pode calcular a raiz quadrada de um número negativo, pois a execução da função `sqrt` terminará com erro fatal.
- Neste caso, em vez de aceitar qualquer valor digitado pelo usuário, o programa deve repetir a entrada de dados até que ela seja consistente.

Consistência de entrada de dados



INSTITUTO FEDERAL
PARANÁ

```
int n, r, c=1;

do {

    printf("Digite um numero entre 1 e 10: ");

    scanf("%d", &n);

} while(n < 1 || n > 10);

printf("\nTabuada do %d\n\n", n);

while (c <= 10){

    r = n * c;

    printf("%d x %2d = %3d\n", n, c, r);

    c++;

}
```

Repetição com confirmação do usuário

- A repetição com confirmação do usuário consiste num padrão em que um processo é executado e, ao final, o usuário é indagado se deseja continuar ou não.

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

```
#include <ctype.h>
```

```
int main() {
```

```
    int n, r, c;
```

```
    char op;
```

```
    ...
```

```
do {  
    do {  
        printf("Digite um numero entre 1 e 10: ");  
        scanf("%d%c", &n);  
    } while(n < 1 || n > 10);  
    printf("\nTabuada do %d\n\n", n);  
    c=1;  
    while (c <= 10) {  
        r = n * c;  
        printf("%d x %2d = %3d\n", n, c, r);  
        c++;  
    }  
    printf("\nContinua (s/n)? ");  
    scanf("%c", &op);  
} while (toupper(op) != 'N');
```

Processos orientados por menus

- Num processo orientado por menu, uma série de opções é apresentada e, conforme a opção escolhida pelo usuário, uma ação correspondente é executada.
- Em seguida, o menu de opções é reapresentado e o processo se repete até que o usuário decida finalizar a sua execução.

O Laço do-while

```
int op;

do {

    printf("Escolha a opcao:\n");

    printf("\t(1) Opção 1\n");

    printf("\t(2) Opcao 2\n");

    printf("\t(3) Opcao 3\n");

    printf("\t(4) Sair\n");

    scanf("%d", &op);

    switch (op) {

        case 1:

            printf("Voce escolheu a opcao 1.\n");

            break;
```


O Laço do-while

```
case 2:
```

```
    printf("Voce escolheu a opcao 2.\n");
```

```
    break;
```

```
case 3:
```

```
    printf("Voce escolheu a opcao 3.\n");
```

```
    break;
```

```
default:
```

```
    if (op != 4)
```

```
        printf("\nOpcao invalida!");
```

```
    }
```

```
} while (op != 4);
```

```
printf("\nFim do Programa!");
```