SCC0222 - Laboratório de Introdução à Ciência de Computação I

Estruturas de Repetição

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Estruturas de Repetição

Estruturas de Repetição

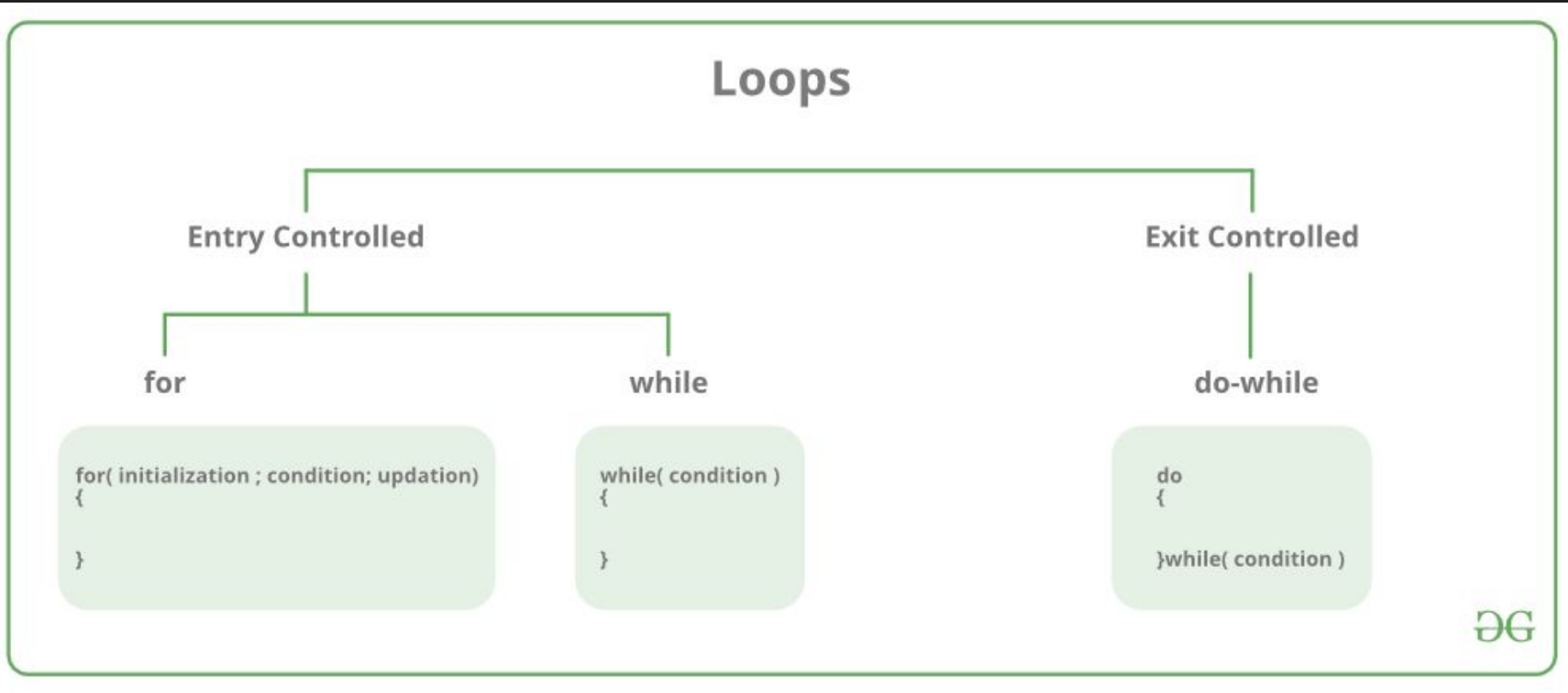
- São usadas para *repetir* um bloco de comandos
- Sequência de instruções que são repetidas até que certa(s) condição(ões) sejam atendidas
- Podem ser controlados na entrada ou saída do corpo do laço

Estruturas de Repetição

- Entrada
  - Condição é testada antes de entrar no corpo do laço
  - *For* e *While*
- *for* (inicialização; teste; atualização) { corpo }
- Inicialização; *while* (teste) {corpo; atualização}

Estruturas de Repetição

- Saída
  - Condição é testada ao final do corpo do laço
  - O corpo do laço é sempre executado **pelo menos** uma vez
  - *do* {corpo; atualização} *while* (teste)



Fonte: https://www.geeksforgeeks.org/loops-in-c-and-cpp/

For

- Uma variável de laço controla o laço
- Inicializamos esta variável com um valor
- Checamos se a variável é menor ou maior que um valor
  - Se for verdadeiro
    - Continua para o bloco
    - Atualiza a variável do laço
  - Se for falso
    - Sai do laço

For

```c
int main ()
{
  int i;
  for(i = 0; i < 3; i++)
  {
    //Fazer algo
    printf("%d\n", i);
  }
  return 0;
}
```

for com mais de um comando

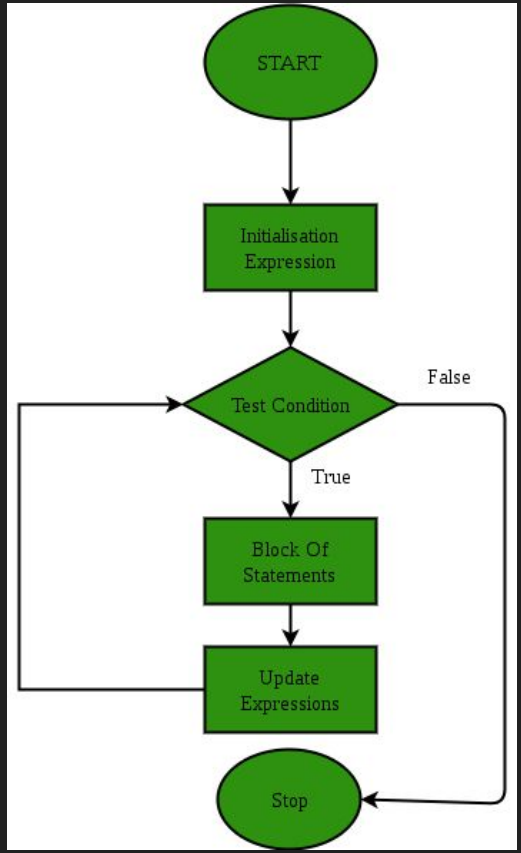
```c
int main ()
{
  int i, j;
  for (i = 0, j = 1; i < 3; i++, j++)
  {
    //Fazer algo
    printf ("%d %d\n", i, j);
  }
  return 0;
}
```

for um pouco mais complexo

```c
int main () {
  int i, j, k;
  k = 10;
  for (i = 0, j = k; i < 3 && j > 0; i++, j-=2){
    //Fazer algo
    printf ("%d %d\n", i, j);
  }
  return 0;
}
```

for aninhado

```c
int main ()
{
    int i, j;
    for (i = 0; i < 3; i++)
    {
        for(j = 0; j < 3; j++)
        {
            printf("%d %d\n", i, j);
        }
        printf("------\n");
    }
    return 0;
}
```



Fonte: https://www.geeksforgeeks.org/loops-in-c-and-cpp/

While

- Geralmente, os laços de *for* são feitos quando sabemos quantas iterações queremos. Quando não sabemos, é mais comum usar um *while*
- Normalmente é possível usar qualquer um dos dois para laços
  - Mas a legibilidade pode ficar difícil ou não intuitiva

While

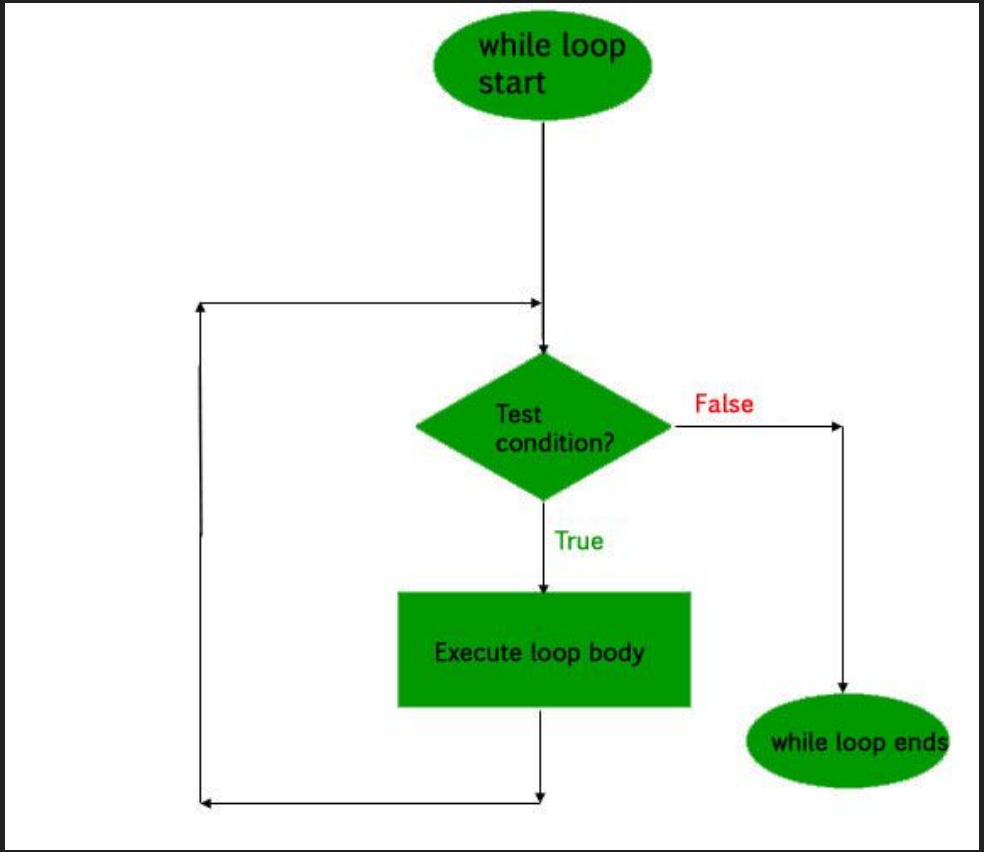
```c
int main ()
{
    int i;
    i = 0;
    while ( i < 3)
    {
        //Fazer algo
        printf ("%d\n", i);
        ++i;
    }
    return 0;
}
```

While com mais de uma condição

```c
int main () {
    int i, j;
    i = 0;
    j = 10;
    while ( i < 3 && j > 0) {
        //Fazer algo
        printf ("%d %d\n", i, j);
        ++i;
        j-=i;
    }
    return 0;
}
```

While aninhado

```c
int main () {
    int i, j;
    i = 0;
    while ( i < 3) {
        j = i+1;
        while(j > 0) {
            //Fazer algo
            printf ("%d %d\n", i, j);
            j--;
        }
        ++i;
    }
      return 0;
}
```

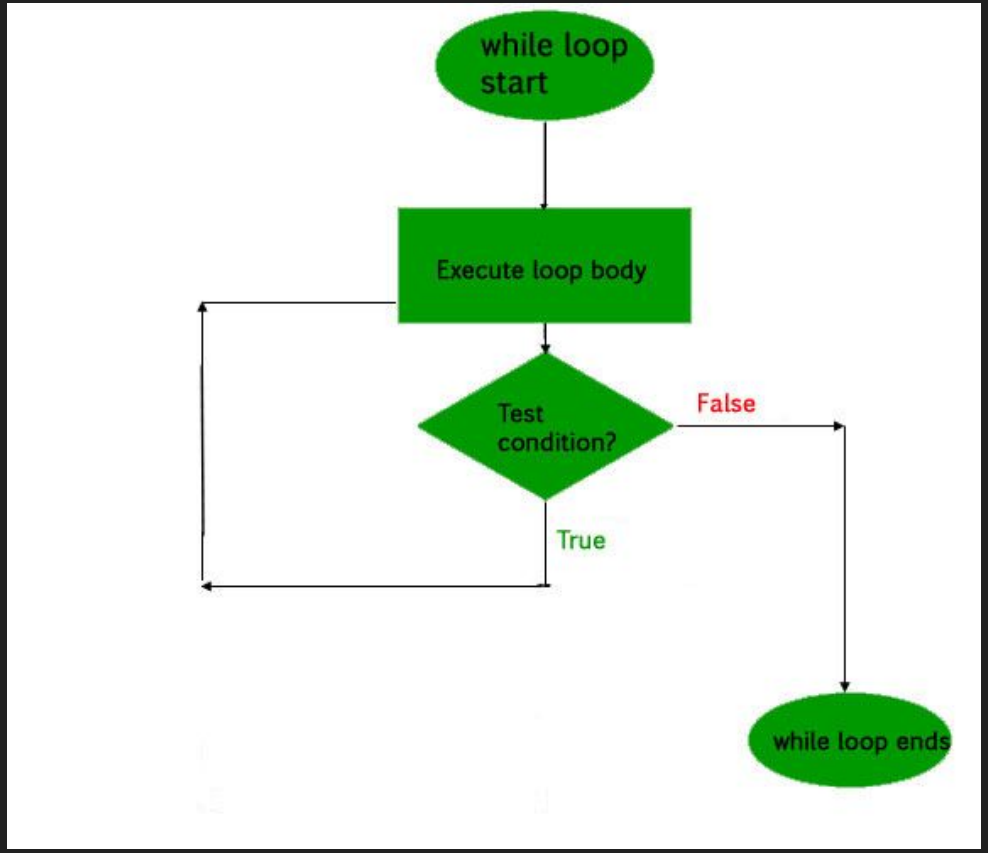

while loop start
Test condition?
False
True
Execute loop body
while loop ends

Do While

- Igual ao *while*, mas a condição é testada depois do corpo do laço
- Ou seja, o corpo é executado pelo menos uma vez

Do While

```c
int main () {
    int i, j;
    i = 0;
    j = 10;
    do {
        //Fazer algo
        printf ("%d %d\n", i, j);
        ++i;
        j-=i;
    } while ( i < 3 && j > 0);
    return 0;
}
```



Fonte: https://www.geeksforgeeks.org/loops-in-c-and-cpp/

Referências

- https://www.geeksforgeeks.org/loops-in-c-and-cpp/
- https://www.devmedia.com.br/estrutura-de-repeticao-c/24121
- https://www.programiz.com/c-programming/c-do-while-loops