

Comandos de Repetição

DCC 199 – Algoritmos



Repetição

Todo aluno de Algoritmos sabe que precisa fazer o teste de mesa para saber se seu programa está correto. No entanto, em todas as aulas, os professores precisam lembrar os alunos deste requisito importante na resolução dos exercícios.

Para alunos atentos, é suficiente lembrar uma vez; alunos desatentos precisam ser lembrados 2 ou 3 vezes; mas, às vezes, é necessário repetir até 5 vezes a mesma frase, para que os alunos mais desatentos resolvam fazer o teste de mesa.

Repetição

Para que essa tarefa fique mais fácil para o professor, esta aula começa com o seguinte problema:

Elaborar um programa que pergunte ao professor quantas vezes a frase

`"Faça o teste de mesa de cada exercício."` precisa ser impressa. O programa deve imprimir a frase conforme a indicação do professor.

```
#include <iostream>
using namespace std;
int main()
{
    int numVezes;
    cout << "Quantas vezes a impressao deve ser feita? ";
    cin >> numVezes;
    if( numVezes == 1 )
        cout << "Faca o teste de mesa de cada exercicio\n";
    else
        if( numVezes == 2 ){
            cout << "Faca o teste de mesa de cada exercicio\n";
            cout << "Faca o teste de mesa de cada exercicio\n";
        }
        else
            if( numVezes == 3 ){
                cout << "Faca o teste de mesa de cada exercicio\n";
                cout << "Faca o teste de mesa de cada exercicio\n";
                cout << "Faca o teste de mesa de cada exercicio\n";
            }
            else
                if( numVezes == 5 ){
                    cout << "Faca o teste de mesa de cada exercicio\n";
                    cout << "Faca o teste de mesa de cada exercicio\n";
                    cout << "Faca o teste de mesa de cada exercicio\n";
                    cout << "Faca o teste de mesa de cada exercicio\n";
                    cout << "Faca o teste de mesa de cada exercicio\n";
                }
    return 0;
}
```

Repetição

Até a semana passada, este programa funcionava muito bem para os professores. Mas, todo professor sabe que, à medida que os exercícios ficam mais complexos, maior é a importância do teste de mesa e maior é a dificuldade dos alunos.

O professor prevê que será necessário repetir a frase até 20 vezes nas próximas aulas...

... mas escrever um programa com uma sequência de tantos `if/else` é desanimador!

Repetição

Nesta aula, vamos aprender a repetir comandos.

Em nosso programa:

- que comando precisamos repetir?
- quantas vezes precisamos repetir?

Repetição

Nesta aula, vamos aprender a repetir comandos.
Em nosso programa:

- que comando precisamos repetir?

A impressão da frase.

- quantas vezes precisamos repetir?

O número de vezes indicado pelo usuário.

Enquanto o número de impressões já realizadas for menor que o número de impressões desejado, é necessário continuar imprimindo.

Repetição

- Chamadas de estruturas iterativas, iterações, laços ou loops;
- Permitem repetir a execução de uma ação várias vezes;
- Podem ser:
 1. Repetição com Teste no Início
 2. Repetição com Teste no Fim
 3. Repetição com Variável de Controle

Repetição com Teste no Início

```
while ( condicao )  
{  
    BlocoDeComandos1;  
}  
BlocoDeComandos2;
```

- Enquanto a condição for **VERDADEIRA**, a seqüência de comandos será repetida.
- Quando a condição fornecer resultado **FALSO**, o controle sai da estrutura passando para o comando seguinte ao final do bloco.

Repetição com Teste no Início

```
while ( condicao )
{
    BlocoDeComandos1;
}
BlocoDeComandos2;
```

Enquanto o número de impressões já realizadas for menor que o número de impressões desejado, é necessário continuar imprimindo.

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVezes, contador;
6
7     cout << "Quantas vezes a impressao deve ser feita? ";
8     cin >> numVezes;
9
10    contador = 0;
11
12    while( contador < numVezes)
13    {
14        cout << "Faca o teste de mesa dos exercicios\n";
15        contador++; //soma 1 à variável
16    }
17
18    return 0;
19 }
```

[illegible]

[illegible]

[illegible]

Quantas vezes a impressao deve ser feita?

[illegible]

[illegible]

Quantas vezes a impressao deve ser feita? 3

[illegible]

Quantas vezes a impressao deve ser feita? 3

[illegible]

Quantas vezes a impressao deve ser feita? 3

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios

[illegible]

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVeze, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVeze;
8
9     contador = 0;
10
11 while( contador < numVeze)
12 {
13     cout << "Faca o teste de mesa dos exercicios\n";
14     contador++; //soma 1 à variável
15 }
16
17 return 0;
18 }
```

Linha	numVeze	contador	teste da condição
3	?	?	--
7	3	?	--
9	3	0	--
11	3	0	V
14	3	1	--
11	3	1	V

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVezeas, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVezeas;
8
9     contador = 0;
10
11     while( contador < numVezeas)
12     {
13         cout << "Faca o teste de mesa dos exercicios\n";
14         contador++; //soma 1 à variável
15     }
16
17     return 0;
18 }
```

Linha	numVezeas	contador	teste da condição
3	?	?	--
7	3	?	--
9	3	0	--
11	3	0	V
14	3	1	--
11	3	1	V

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVeze, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVeze;
8
9     contador = 0;
10
11     while( contador < numVeze)
12     {
13         cout << "Faca o teste de mesa dos exercicios\n";
14         contador++; //soma 1 à variável
15     }
16
17     return 0;
18 }
```

Linha	numVeze	contador	teste da condição
3	?	?	---
7	3	?	---
9	3	0	---
11	3	0	V
14	3	1	---
11	3	1	V
14	3	2	---

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVezeas, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVezeas;
8
9     contador = 0;
10
11 while( contador < numVezeas)
12 {
13     cout << "Faca o teste de mesa dos exercicios\n";
14     contador++; //soma 1 à variável
15 }
16
17 return 0;
18 }
```

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios

Linha	numVezeas	contador	teste da condição
3	?	?	---
7	3	?	---
9	3	0	---
11	3	0	V
14	3	1	---
11	3	1	V
14	3	2	---
11	3	2	V

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVeze, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVeze;
8
9     contador = 0;
10
11     while( contador < numVeze)
12     {
13         cout << "Faca o teste de mesa dos exercicios\n";
14         contador++; //soma 1 à variável
15     }
16
17     return 0;
18 }
```

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios

Linha	numVeze	contador	teste da condição
3	?	?	---
7	3	?	---
9	3	0	---
11	3	0	V
14	3	1	---
11	3	1	V
14	3	2	---
11	3	2	V

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVeze, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVeze;
8
9     contador = 0;
10
11     while( contador < numVeze)
12     {
13         cout << "Faca o teste de mesa dos exercicios\n";
14         contador++; //soma 1 à variável
15     }
16
17     return 0;
18 }
```

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios

Linha	numVeze	contador	teste da condição
3	?	?	---
7	3	?	---
9	3	0	---
11	3	0	V
14	3	1	---
11	3	1	V
14	3	2	---
11	3	2	V
14	3	3	---

Repetição

```

1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVezes, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVezes;
8
9     contador = 0;
10
11 while( contador < numVezes)
12 {
13     cout << "Faca o teste de mesa dos exercicios\n";
14     contador++; //soma 1 à variável
15 }
16
17 return 0;
18 }
    
```

Quantas vezes a impressao deve ser feita? 3
 Faca o teste de mesa dos exercicios
 Faca o teste de mesa dos exercicios
 Faca o teste de mesa dos exercicios

Linha	numVezes	contador	teste da condição
3	?	?	---
7	3	?	---
9	3	0	---
11	3	0	V
14	3	1	---
11	3	1	V
14	3	2	---
11	3	2	V
14	3	3	---
11	3	3	F

Repetição

```

1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVezeas, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVezeas;
8
9     contador = 0;
10
11     while( contador < numVezeas)
12     {
13         cout << "Faca o teste de mesa dos exercicios\n";
14         contador++; //soma 1 à variável
15     }
16
17     return 0;
18 }

```

Quantas vezes a impressao deve ser feita? 3
 Faca o teste de mesa dos exercicios
 Faca o teste de mesa dos exercicios
 Faca o teste de mesa dos exercicios

Linha	numVezeas	contador	teste da condição
3	?	?	---
7	3	?	---
9	3	0	---
11	3	0	V
14	3	1	---
11	3	1	V
14	3	2	---
11	3	2	V
14	3	3	---
11	3	3	F

Repetição

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVezeas, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVezeas;
8
9     contador = 0;
10
11     while( contador < numVezeas)
12     {
13         cout << "Faca o teste de mesa dos exercicios\n";
14         contador++; //soma 1 à variável
15     }
16
17     return 0;
18 }
```

Quantas vezes a impressao deve ser feita? 3
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios
Faca o teste de mesa dos exercicios

Linha	numVezeas	contador	teste da condição
3	?	?	---
7	3	?	---
9	3	0	---
11	3	0	V
14	3	1	---
11	3	1	V
14	3	2	---
11	3	2	V
14	3	3	---
11	3	3	F

Exercício

Faça o teste de mesa com entrada 2 e com entrada -1.

```

1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numVezes, contador;
6     cout << "Quantas vezes a impressao deve ser feita? ";
7     cin >> numVezes;
8
9     contador = 0;
10
11     while( contador < numVezes)
12     {
13         cout << "Faca o teste de mesa dos exercicios\n";
14         contador++; //soma 1 à variável
15     }
16
17     return 0;
18 }

```

Repetição

Situações comuns no uso de repetições:

1. Número de repetições é conhecido:
Neste caso, é necessário contar quantas vezes a repetição é feita para que seja possível sair do laço.

Repetição - exemplo

Desenvolva uma função que calcule o número de pontos de cada um dos times que participa de um campeonato. A função deve receber como parâmetro o número de times e, para cada time:

- ler o número de vitórias, empates e derrotas;
- calcular e imprimir o número de pontos (cada vitória vale 3 pontos, cada empate vale 1 ponto).

Repetição - exemplo

Desenvolva uma função que calcule o número de pontos de cada um dos times que participa de um campeonato. A função deve receber como parâmetro o número de times e, para cada time:

- ler o número de vitórias, empates e derrotas;
- calcular e imprimir o número de pontos (cada vitória vale 3 pontos, cada empate vale 1 ponto).

NumTimes → indica o número de repetições.
 → é necessário contar os times.
 → para cada time, será necessário ler e imprimir valores.

Repetição - exemplo

Desenvolva uma função que calcule o número de pontos de cada um dos times que participa de um campeonato. A função deve receber como parâmetro o número de times e, para cada time:

- ler o número de vitórias, empates e derrotas;
- calcular e imprimir o número de pontos (cada vitória vale 3 pontos, cada empate vale 1 ponto).

Inicializa o contador

Enquanto **estiver faltando time**

{

 Lê vitórias, empates e derrotas de um time

 Imprime número de pontos do time

}

Repetição - exemplo

Desenvolva uma função que calcule o número de pontos de cada um dos times que participa de um campeonato. A função deve receber como parâmetro o número de times e, para cada time:

- ler o número de vitórias, empates e derrotas;
- calcular e imprimir o número de pontos (cada vitória vale 3 pontos, cada empate vale 1 ponto).

Inicializa o contador

Enquanto **contador < número de times**

{

 Lê vitórias, empates e derrotas de um time

 Imprime número de pontos do time

}

Repetição - exemplo



```
void imprimePontosNoCampeonato(int numTimes)
{
    int cont, vitorias, empates, derrotas;

    cout << "\nPara cada time, digite o numero ";
    cout << "de vitorias, empates e derrotas: ";

    cont = 0;

    while( cont < numTimes ){

        cout << "\nTime " << cont + 1 << ": ";
        cin >> vitorias >> empates >> derrotas;
        cout << "    Total: " <<
            (3 * vitorias) + (1 * empates) << " pontos";

        cont++;
    }
}
```

Exercício



Faça o teste
de mesa com
3 times.

```
1 #include <iostream>
2 using namespace std;
3 void imprimePontosNoCampeonato(int numTimes)
4 {
5     int cont, vitorias, empates, derrotas;
6     cout << "\nPara cada time, digite o numero ";
7     cout << "de vitorias, empates e derrotas: ";
8
9     cont = 0;
10    while( cont < numTimes ){
11        cout << "\nTime " << cont + 1 << ": ";
12        cin >> vitorias >> empates >> derrotas;
13        cout << "    Total: " <<
14            (3 * vitorias) + (1 * empates) << " pontos";
15        cont++;
16    }
17 }
18 int main()
19 {
20     int times;
21     cout << "\nDigite o numero de times: ";
22     cin >> times;
23     imprimePontosNoCampeonato(times);
24     return 0;
25 }
```

Repetição

Situações comuns no uso de repetições:

2. Número de repetições é indefinido:
Muitas vezes, a repetição é realizada até que o usuário solicite o término das repetições (o usuário insere um valor específico para indicar este término).

Repetição

- Um valor específico (chamado de **FLAG**) indica o fim dos dados de entrada.
- Quando este valor é lido, não deve ser tratado como parte da sequência de entrada. Isto é, não deve ser processado no interior do laço como os outros dados da sequência.
- A leitura do **FLAG** informa ao programa que os dados de entrada terminaram e que ele deve partir para a execução da finalização de seu processamento (cálculos finais, impressões finais etc.).

Repetição - exemplo

Desenvolva um algoritmo que leia uma sequência de números inteiros, calculando e imprimindo o quadrado de cada número lido. A sequência deve terminar quando o número 0 (zero) for lido.

Repetição - exemplo

Desenvolva um algoritmo que leia uma sequência de números inteiros, calculando e imprimindo o quadrado de cada número lido. A sequência deve terminar quando o número 0 (zero) for lido.

ZERO → indica o final da sequência de valores.
 → encerra as repetições.
 → não deve ser processado (o quadrado de zero não deve ser impresso).

Repetição - exemplo

Desenvolva um algoritmo que leia uma sequência de números inteiros, calculando e imprimindo o quadrado de cada número lido. A sequência deve terminar quando o número 0 (zero) for lido.

```
Lê o primeiro número
Enquanto o número lido não for zero
{
    Calcula e imprime o quadrado do número
    Lê o próximo número
}
```

Repetição - exemplo

```
#include <iostream>
using namespace std;
int main()
{
    int num, quad;
    // imprime uma msg e le o 1o. inteiro

    cout << "Digite um inteiro (0 para sair): ";
    cin >> num;

    while( num != 0 )
    {
        quad = num * num;
        cout << "\nQuadrado de " << num << ": " << quad;
        cout << "\nDigite um inteiro (0 para sair): ";
        cin >> num;
    }
    return 0;
}
```

TESTE DE MESA

--

--

[illegible]

TESTE DE MESA

Digite um inteiro (0 para sair):

Digite um inteiro (0 para sair): 5

[illegible]

TESTE DE MESA

Digite um inteiro (0 para sair): 5

Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14            << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---

Digite um inteiro (0 para sair): 5

Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14            << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---

Digite um inteiro (0 para sair): 5
Quadrado de 5: 25

TESTE DE MESA

[illegible]

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair):
```

Repetição - exemplo

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14             << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
```

Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14             << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---
10	2	25	V

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
```

Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14             << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---
10	2	25	V
12	2	4	---

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
```

Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14            << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---
10	2	25	V
12	2	4	---

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
Quadrado de 2: 4
```

Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14             << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---
10	2	25	V
12	2	4	---

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
Quadrado de 2: 4
Digite um inteiro (0 para sair):
```


Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14             << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---
10	2	25	V
12	2	4	---
16	0	4	---

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
Quadrado de 2: 4
Digite um inteiro (0 para sair): 0
```

Repetição - exemplo

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---
10	2	25	V
12	2	4	---
16	0	4	---
10	0	4	F

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14             << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
Quadrado de 2: 4
Digite um inteiro (0 para sair): 0
```

Repetição - exemplo

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num, quad;
6     // imprime uma msg e le o 1o. inteiro
7     cout << "Digite um inteiro (0 para sair): ";
8     cin >> num;
9
10    while( num != 0 )
11    {
12        quad = num * num;
13        cout << "\nQuadrado de " << num << ": "
14             << quad;
15        cout << "\nDigite um inteiro (0 para sair): ";
16        cin >> num;
17    }
18    return 0;
19 }
```

TESTE DE MESA

linha	num	quad	teste
3	?	?	---
8	5	?	---
10	5	?	V
12	5	25	---
16	2	25	---
10	2	25	V
12	2	4	---
16	0	4	---
10	0	4	F
18	0	4	---

```
Digite um inteiro (0 para sair): 5
Quadrado de 5: 25
Digite um inteiro (0 para sair): 2
Quadrado de 2: 4
Digite um inteiro (0 para sair): 0
```

Estrutura da repetição

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int num, quad;
6      // imprime uma msg e le o 1o. inteiro
7      cout << "Digite um inteiro (0 para sair): ";
8      cin >> num;
9
10     while( num != 0 )
11     {
12         quad = num * num;
13         cout << "\nQuadrado de " << num << ": "
14             << quad;
15         cout << "\nDigite um inteiro (0 para sair): ";
16         cin >> num;
17     }
18     return 0;
19 }

```

Estrutura da repetição

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int num, quad;
6      // imprime uma msg e le o 1o. inteiro
7      cout << "Digite um inteiro (0 para sair): ";
8      cin >> num;
9
10     while ( num != 0 )
11     {
12         quad = num * num;
13         cout << "\nQuadrado de " << num << ": "
14             << quad;
15         cout << "\nDigite um inteiro (0 para sair): ";
16         cin >> num;
17     }
18     return 0;
19 }

```

Condição

Teste normalmente envolve ao menos uma variável.

Estrutura da repetição

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int num, quad;
6      // imprime uma msg e le o 1o. inteiro
7      cout << "Digite um inteiro (0 para sair): ";
8      cin >> num;
9
10     while( num != 0 )
11     {
12         quad = num * num;
13         cout << "\nQuadrado de " << num << ": "
14             << quad;
15         cout << "\nDigite um inteiro (0 para sair): ";
16         cin >> num;
17     }
18     return 0;
19 }

```

Condição

Teste normalmente envolve ao menos uma variável.

Inicialização

Toda variável da condição precisa ser inicializada antes do laço, através de atribuição ou leitura.

Estrutura da repetição

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int num, quad;
6      // imprime uma msg e le o 1o. inteiro
7      cout << "Digite um inteiro (0 para sair): ";
8      cin >> num;
9
10     while( num != 0 )
11     {
12         quad = num * num;
13         cout << "\nQuadrado de " << num << ": "
14             << quad;
15         cout << "\nDigite um inteiro (0 para sair): ";
16         cin >> num;
17     }
18     return 0;
19 }

```

Condição

Teste normalmente envolve ao menos uma variável.

Inicialização

Toda variável da condição precisa ser inicializada antes do laço, através de atribuição ou leitura.

Atualização

Ao menos uma variável da condição precisa ser atualizada no interior do laço.

Estrutura da repetição

```
void imprimePontosNoCampeonato(int numTimes)
{
    int cont, vitorias, empates, derrotas;

    cout << "\nPara cada time, digite o numero ";
    cout << "de vitorias, empates e derrotas: ";

    cont = 0;

    while( cont < numTimes ){

        cout << "\nTime " << cont + 1 << ": ";
        cin >> vitorias >> empates >> derrotas;

        cout << "    Total: " <<
            (3 * vitorias) + (1 * empates) << " pontos";

        cont++;
    }
}
```

Condição

Teste normalmente envolve ao menos uma variável.

Inicialização

Toda variável da condição precisa ser inicializada antes do laço, através de atribuição ou leitura.

Atualização

Ao menos uma variável da condição precisa ser atualizada no interior do laço.

Estrutura da repetição

```
void imprimePontosNoCampeonato(int numTimes)
{
    int cont, vitorias, empates, derrotas;

    cout << "\nPara cada time, digite o numero ";
    cout << "de vitorias, empates e derrotas: ";

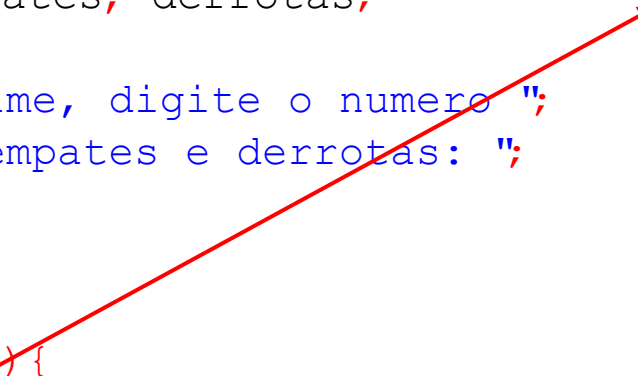
    cont = 0;

    while (cont < numTimes) {

        cout << "\nTime " << cont + 1 << ": ";
        cin >> vitorias >> empates >> derrotas;

        cout << "    Total: " <<
            (3 * vitorias) + (1 * empates) << " pontos";

        cont++;
    }
}
```



Condição

Teste normalmente envolve ao menos uma variável.

Inicialização

Toda variável da condição precisa ser inicializada antes do laço, através de atribuição ou leitura.

Atualização

Ao menos uma variável da condição precisa ser atualizada no interior do laço.

Estrutura da repetição

```
void imprimePontosNoCampeonato(int numTimes)
{
    int cont, vitorias, empates, derrotas;

    cout << "\nPara cada time, digite o numero ";
    cout << "de vitorias, empates e derrotas: ";

    cont = 0;

    while( cont < numTimes ){

        cout << "\nTime " << cont + 1 << ": ";
        cin >> vitorias >> empates >> derrotas;

        cout << "    Total: " <<
            (3 * vitorias) + (1 * empates) << " pontos";

        cont++;
    }
}
```

Condição

Teste normalmente envolve ao menos uma variável.

Inicialização

Toda variável da condição precisa ser inicializada antes do laço, através de atribuição ou leitura.

Atualização

Ao menos uma variável da condição precisa ser atualizada no interior do laço.

Estrutura da repetição

```
void imprimePontosNoCampeonato(int numTimes)
{
    int cont, vitorias, empates, derrotas;

    cout << "\nPara cada time, digite o numero ";
    cout << "de vitorias, empates e derrotas: ";

    cont = 0;

    while( cont < numTimes ){

        cout << "\nTime " << cont + 1 << ": ";
        cin >> vitorias >> empates >> derrotas;

        cout << "    Total: " <<
            (3 * vitorias) + (1 * empates) << " pontos";

        cont++;
    }
}
```

Condição

Teste normalmente envolve ao menos uma variável.

Inicialização

Toda variável da condição precisa ser inicializada antes do laço, através de atribuição ou leitura.

Atualização

Ao menos uma variável da condição precisa ser atualizada no interior do laço.

Repetição - exemplo

Imagine um programa de um caixa eletrônico com as seguintes opções de menu:

- 1 – imprime o saldo da conta
- 2 – imprime o extrato da conta
- 3 – efetua pagamento de boleto
- 4 – efetua um saque
- 5 – sai do programa

Imagine que o usuário pode executar mais de uma operação, ou ainda, repetir operações.

O usuário poderá repetir as operações enquanto a opção selecionada não for a opção **5 – sai do programa**.

```
#include <iostream>
using namespace std;
void imprimeOpcoesMenu();
void imprimeSaldo();
void imprimeExtrato();
void pagaBoleto();
void fazSaque();
int main()
{
    int opcaoSelecionada;

    imprimeOpcoesMenu();
    cin >> opcaoSelecionada;

    while( opcaoSelecionada != 5 ) {
        switch( opcaoSelecionada ) {
            case 1: imprimeSaldo(); break;
            case 2: imprimeExtrato(); break;
            case 3: pagaBoleto(); break;
            case 4: fazSaque(); break;
            //case 5 nunca seria executado, pois laço termina antes
            default: cout << "Opcao invalida!"; break;
        }
        imprimeOpcoesMenu();
        cin >> opcaoSelecionada;
    }
    cout << "Programa encerrado";
    return 0;
}
```

Exercício

1. Faça o teste de mesa do programa ao lado. Apresente as saídas que serão impressas pelo programa.

```

1. #include <iostream>
2. using namespace std;
3. int funcao(int a){
4.     int b=1;
5.     while ( b<a){
6.         if (b%3==0)
7.             return b;
8.         else
9.             cout << "A" << b+a;
10.        b = b + 1;
11.    }
12.    return b;
13. }
14. int main (){
15.     int a;
16.     a = 4;
17.     cout << "\n" << funcao(a);
18.     return 0;
19. }
```

Exercício

2. Faça o teste de mesa do programa ao lado. Apresente as saídas que serão impressas pelo programa.

```

1. #include <iostream>
2. using namespace std;
3. int funcao(int m, int n){
4.     int r=0;
5.     while (m-n > 1){
6.         if (n%2 == 0){
7.             r = r + m;
8.         }
9.         else{
10.            r = r - n;
11.        }
12.        m--;
13.        n++;
14.        cout << n << ", " << m << ", " <<
            r << "\n";
15.    }
16.    return r;
17. }
18. int main (){
19.     int x, y, z;
20.     x = 7;
21.     y = 1;
22.     z = funcao(x, y);
23.     cout << "R: " << z;
24.     return 0;
25. }

```

Exercício

3. **a)** Escreva um programa que leia a nota de um aluno cujo valor está entre 0 e 100. O programa deve escrever “REPROVADO” se a nota for inferior a 60 e escrever “APROVADO”, caso contrário.
- b)** Reescreva o programa para que ele repita a leitura e impressão para um grupo de alunos. A quantidade de notas lidas é desconhecida. O programa deve ser encerrado se for lido um valor inválido (fora do intervalo entre 0 e 100). Teste seu programa com a sequência:
87 59 94 108
- c)** Reescreva o programa para que o usuário comece informando o número de alunos. Neste caso, a quantidade de notas lidas deve ser igual ao número informado. Teste seu programa com: 3 64 92 31

Exercício

4. **a)** Escreva uma função que recebe um número inteiro n como parâmetro e imprime a tabuada de n . Por exemplo, se n for 3, a função deve imprimir as linhas do quadro à direita.
- b)** Faça um programa que leia um valor inteiro do teclado e use a função da letra a para imprimir a tabuada deste número.
- c)** Reescreva o programa para que ele imprima a tabuada de todo valor inteiro que o usuário digitar. O programa deverá ser encerrado quando o usuário digitar o valor zero.
- d)** Reescreva o programa para que ele imprima a tabuada de todo valor inteiro de 1 a 9.

Tabuada do 3				
3	x	0	=	0
3	x	1	=	3
3	x	2	=	6
3	x	3	=	9
3	x	4	=	12
3	x	5	=	15
3	x	6	=	18
3	x	7	=	21
3	x	8	=	24
3	x	9	=	27
3	x	10	=	30

Exercício

5. **a)** Escreva uma função que lê um caractere do teclado que indique o estado civil de uma pessoa. A função deve retornar apenas o valor 'C' (casado) ou o valor 'S' (solteiro), por isso, enquanto o usuário digitar um valor inválido, a função deve solicitar que ele digite um destes dois valores.
- b)** Escreva uma função que leia um valor inteiro do teclado representando a idade de uma pessoa. Espera-se que a função retorne um valor entre 0 e 130. Assim, enquanto o usuário digitar um valor inválido, a função deve solicitar novamente o valor.
- c)** Escreva um programa que leia o estado civil e a idade de 15 pessoas, chamando as funções desenvolvidas acima para garantir que serão informados dados válidos.

Exercício

6. **a)** Escreva uma função que recebe como parâmetro um caractere representando um operador inteiro ('+', '-', '*', '/' ou '%'). Se o caractere for realmente um operador, a função deve ler do teclado dois valores inteiros e imprimir o resultado da operação solicitada. Se o caractere não for um operador ou se a operação não puder ser realizada (divisor nulo), a função deve imprimir uma mensagem informando o problema.
- b)** Escreva um programa que leia um caractere do teclado e chame a função acima. Teste seu programa com: $- 8 \quad 43$
- c)** Reescreva o programa para que ele execute uma sequência de operações, até que o usuário digite o caractere 'F', indicando o fim da execução. Teste seu programa com:
 $+ \quad 3 \quad 4 \quad \$ \quad / \quad 4 \quad 0 \quad - \quad 2 \quad -3 \quad A \quad F$

Exercício

7. **a)** Escreva uma função que leia caracteres do teclado. Toda vez que o usuário digitar um espaço em branco, a função deverá mudar de linha (imprimir "\n"). Quando o usuário digitar um ponto final, a leitura deverá ser encerrada.
- b)** Escreva um programa para testar a função acima.
8. **a)** Escreva uma função que receba um inteiro X como parâmetro e retorne um valor aleatório entre 0 e $X-1$. Use a função `rand() % X` (biblioteca `cstdlib`) para obter um número aleatório entre 0 e $X-1$.
- b)** Escreva uma função que lê um caractere do teclado que indica se o usuário prefere Par ('P') ou Impar ('I'). Em seguida, a função deve sortear um número entre 0 e 99 (chamando a função da letra a) e escrever se o usuário ganhou ou não.
- c)** Escreva um programa que pergunte ao usuário se ele deseja jogar Par ou Impar (opção 'J') ou sair do programa ('S'). Enquanto o usuário quiser jogar, chame a função da letra b.

Comandos de Repetição

DC5199 - Algoritmos - Prática



while

- Sintaxe

```
// Inicializacao
while (condicao)
{
    blocoDeComandos;
    // Atualizacao
}
```

```
int main()
{
    int i;
    i=0;
    while (i<=10) {
        cout << i << "\n";
        i++;
    }
    return 0;
}
```

```
int main()
{
    int i;
    cout << "2 x ";
    cin >> i;
    while (i != 0) {
        cout << " = " << 2 * i << "\n";
        cout << "2 x ";
        cin >> i;
    }
    cout << " = 0\n FIM \n";
    return 0;
}
```

Exercício

1. Escreva uma função que recebe como parâmetro um número inteiro n . A função deve ler n valores do teclado e retornar quantos destes valores são negativos. Faça também um programa principal que leia do teclado o que for necessário para executar a chamada correta desta função e imprimir seu resultado na tela.
2. Escreva uma função que leia os valores $n1$ e $n2$ e imprima o intervalo fechado entre esses dois valores. Exemplo: se os valores lidos forem 5 e 2, a saída deverá ser 5 4 3 2. Faça também um programa principal que leia do teclado o que for necessário para executar a chamada correta desta função.

Exercício

3. **a)** Escreva uma função que recebe um inteiro N como parâmetro e imprime uma linha contendo os números de 1 até N. Por exemplo, se o valor do parâmetro for 6, a função deverá imprimir: 1 2 3 4 5 6
- b)** Faça um programa para testar a função, solicitando que o usuário digite o valor de N.
- c)** Refaça seu programa para que, após o usuário digitar o valor de N (por exemplo, 6), o programa imprima o triângulo:

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6

```


Exercício

4. Uma professora de matemática quer fazer um programa para seus alunos estudarem a tabuada.
 - a) Escreva uma função que sorteia dois números entre 0 e 9 (use a função `rand()` da biblioteca `cstdlib`). A função deve pedir que o usuário digite o resultado da multiplicação dos números sorteados. Enquanto o usuário não acertar, a função deve indicar que houve erro e pedir ao usuário que tente novamente. Ao final, a função deve imprimir uma mensagem indicando o acerto.
 - b) Faça um programa que peça a um aluno que resolva 15 operações, chamando a função acima.
- DESAFIO:** Refaça a função para que o usuário tenha até 5 chances de acertar a multiplicação. Assim, a repetição deve terminar quando o usuário acertar ou quando as chances acabarem. Ao final, se o usuário não conseguiu acertar, indique o valor correto. Se ele conseguiu, parabeneze-o.

Exercício

5. Uma empresa tem 30 funcionários e resolveu conceder a todos um auxílio mensal de R\$60,00 por mês para cada dependente menor de 18 anos.
Escreva um programa que, para cada um dos 30 funcionários, leia seu número de dependentes. Em seguida, o programa deve ler a idade de cada dependente e imprimir o valor total do auxílio que este funcionário receberá.
Você pode criar funções, se achar necessário.
 6. Para controlar seus gastos mensais, Ana anota todas as suas despesas. Assim, quando vai ao shopping, em cada loja que entra, ela registra quantos itens comprou e qual o valor de cada um dos itens. Faça um programa para ajudá-la. O programa deverá ler o número de itens comprados em cada loja e o valor de cada um. O programa será encerrado quando for informada uma quantidade negativa de itens.
- DESAFIO:** Altere o programa para imprimir o gasto total de Ana.

Exercício

7. Para organizar os alunos em duplas para uma atividade, um professor resolveu separá-los de acordo com os números de matrícula: o primeiro fará dupla com o último, o segundo com o antepenúltimo, e assim em diante. Se o número de alunos for ímpar, um dos alunos ficará sozinho no final.
- a) Escreva uma função que recebe o número de alunos como parâmetro e imprime como será feita a divisão. Se o número de alunos for 5, por exemplo, a função deverá imprimir:
- ```

1 e 5
2 e 4
3

```
- b) Faça um programa para separar os alunos de várias turmas. O programa deve pedir que o usuário digite o número de alunos de cada turma e deve ser encerrado quando este valor for menor ou igual a zero.