Lógica de Programação para Engenharia

Outras Estruturas de Repetição

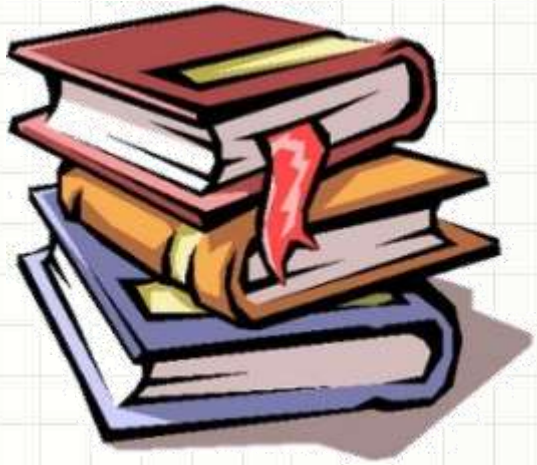
Prof. Dr. Daniel Caetano

2011 - 2

Visão Geral

1. Contagem com While
2. Repetição com For
3. Repetição com Do~While

Material de Estudo



Material	Acesso ao Material
Notas de Aula	http://www.caetano.eng.br/aulas/lpe/ (Aula 13)
Apresentação	http://www.caetano.eng.br/aulas/lpe/ (Aula 13) – PARCIAL / COMPLETO
Material Didático	Fundamentos da Programação de Computadores – Parte 2 – Páginas 93 a 144.

Objetivos

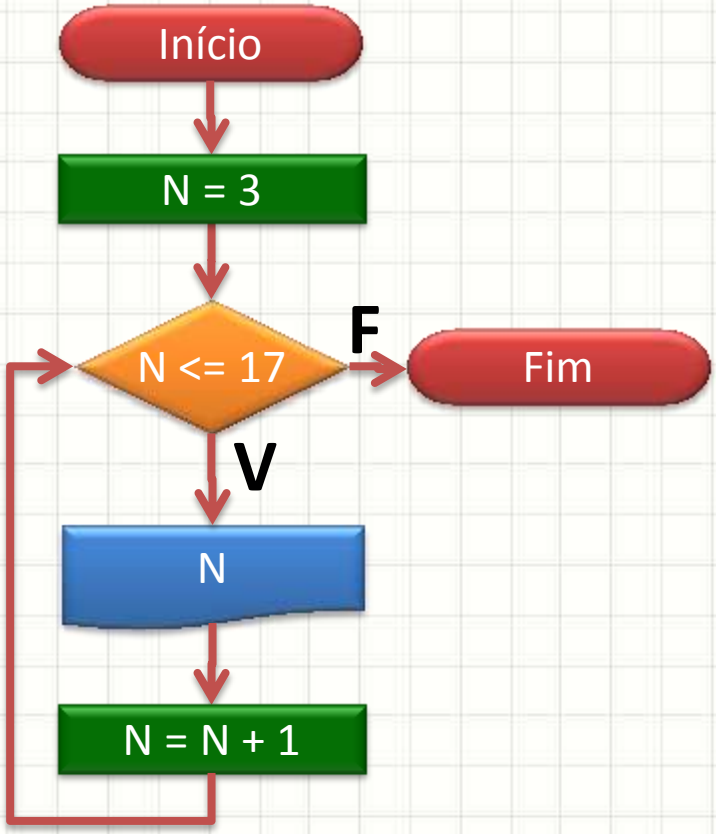
- Conhecer as várias estruturas de repetição da linguagem C/C++
- Compreender o uso de cada uma destas estruturas
- Capacitar para a criação de algoritmos que envolvam repetição

- **PARA CASA**
  - **Lista de Exercícios 2 está ONLINE!**

Recordando o While

Recordando o While

- Aula passada: estrutura de repetição **while**
  - **O que faz?**

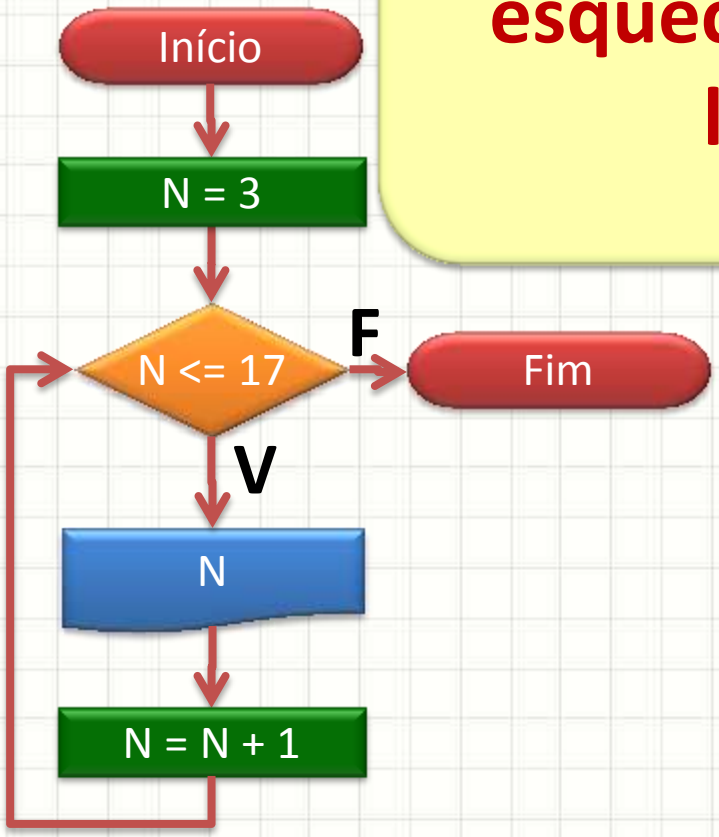



```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int N;

      N = 3;
      while ( N <= 17 ) {
          cout << N << endl;
          N = N + 1;
          }

getchar();
}
```

Recordando o While

- Aula passada: estrutura de repetição **while**
  - **O que fa**

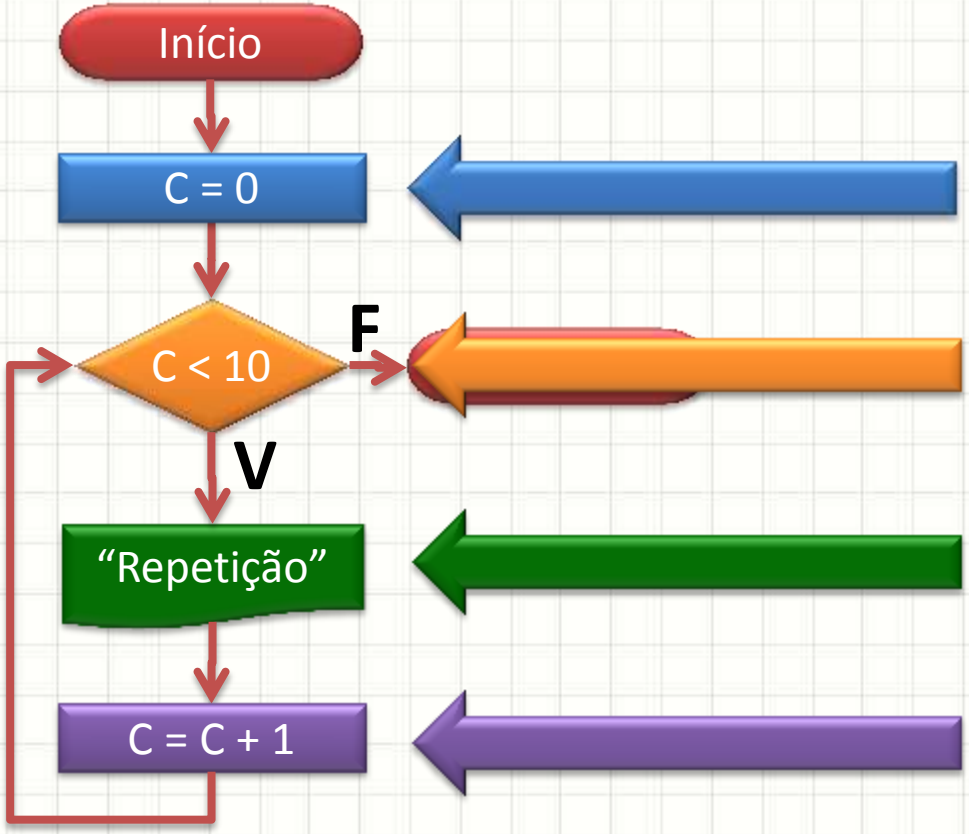
O que acontece se esquecermos essa linha?




```
N =
while( N <= 17 ) {
    cout << N << endl;
    N = N + 1;
    }

getchar();
}
```

Recordando o While

- Observe:
  - O que faz?



INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

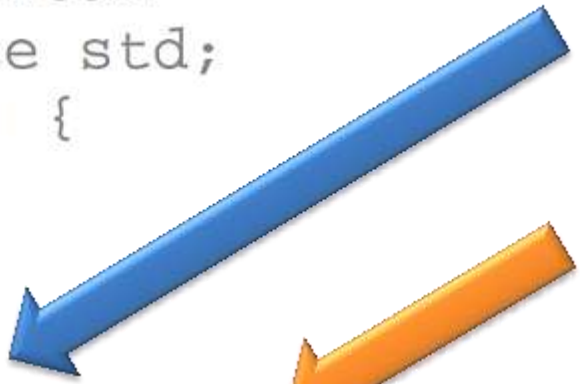
BLOCO

ATUALIZAÇÃO

Recordando o While

- No código...

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int CONT;

      CONT = 0;
      while ( CONT < 10 ) {
         cout << "Isso é uma repetição" << endl;
         CONT = CONT + 1;
         }

getchar();
}
```



INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

BLOCO

ATUALIZAÇÃO

le

Fácil esquecer um deles!

```
#i
#i
usi
int main(void) {

      int CONT;

      CONT = 0;
      while ( CONT < 10 ) {
         cout << "Isso é uma repetição" << endl;
         CONT = CONT + 1;
         }

getchar();
}
```

INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

BLOCO

ATUALIZAÇÃO

A Estrutura de Repetição FOR

O que é a estrutura **for**

- Todos os elementos em uma única linha
  - Só o bloco fica “isolado”

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int CONT;

      CONT = 0;
      while ( CONT < 10 ) {
         cout << "Isso é uma repetição" << endl;
         CONT = CONT + 1;
         }

getchar();
}
```

O que é a estrutura **for**

- Todos os elementos em uma única linha
  - Só o bloco fica “isolado”

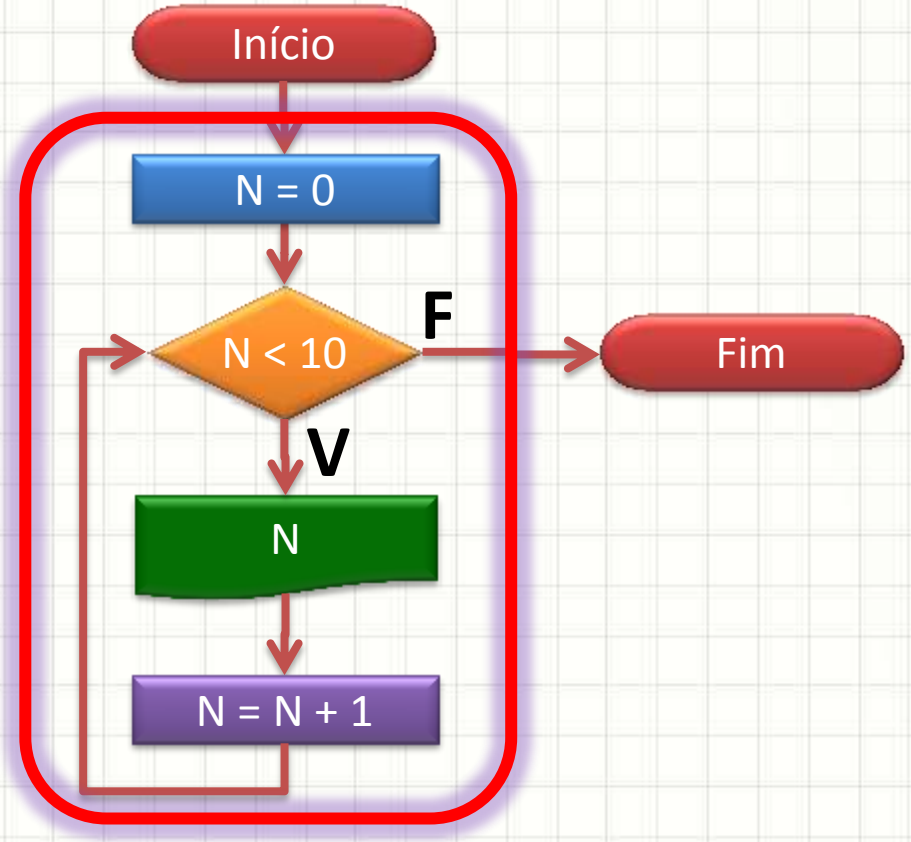
```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int CONT;

      for (CONT = 0; CONT < 10; CONT = CONT + 1) {
         cout << "Isso é uma repetição" << endl;
         }

getchar();
}
```

O que é a estrutura **for**

- Todos os elementos em uma única linha
  - Só o bloco fica “isolado”

Forma Geral do **for**

```
for ( inicialização; condição de repetição; atualização)  {
    Executa enquanto a proposição for verdadeira
}
```



Leitura do **for**

```
for ( X = 0 ; X < 7 ; X = X + 2 ) {
    cout << X << endl;
}
```

Faça, a partir de X = 0, enquanto X < 7 e contando de 2 em 2, a impressão de X.

EXERCÍCIO

A) Faça um programa que apresente os números de 50 a 75.

EXERCÍCIO

A) Faça um programa que apresente os números de 50 a 75.

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int C;

      for ( C = 50; C <= 75; C = C + 1) {
         cout << C << endl;
         }

getchar();
}
```

EXERCÍCIO

B) Modifique o programa anterior para que ele conte de 2 em 2.

EXERCÍCIO

B) Modifique o programa anterior para que ele conte de 2 em 2.

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int C;

      for ( C = 50; C <= 75; C = C + 2) {
         cout << C << endl;
         }

getchar();
}
```

EXERCÍCIO

C) Modifique o programa para que imprima só números divisíveis por 5.

EXERCÍCIO

C) Modifique para que imprima só números divisíveis por 5.

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int C;

      for ( C = 50; C <= 75; C = C + 2) {
         if ( C%5 == 0 ) {
            cout << C << endl;
            }
         }

getchar();
}
```

Repetição com Do~While

Repetição com Do~While

- Algumas vezes queremos que um procedimento seja executado “pelo menos uma vez”.

- Quando?
  - Precisamos esperar que um dado específico seja digitado
  - Precisamos esperar que um valor específico seja lido de um sensor
  - Etc.

Repetição com Do~While

- Observe:

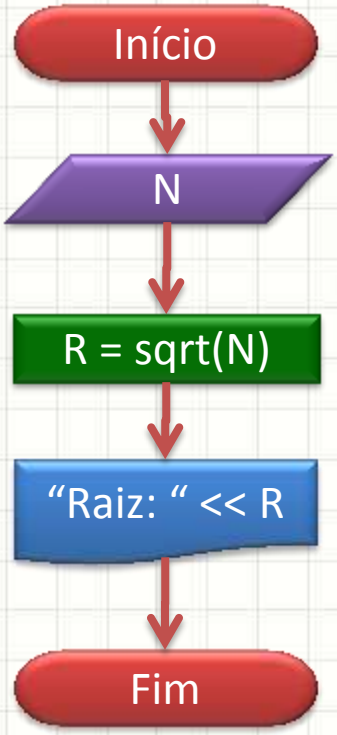
```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    cout << “Digite um número positivo: ”;
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```

E se o usuário digitar um número negativo?

Não seria legal poder repetir a pergunta?

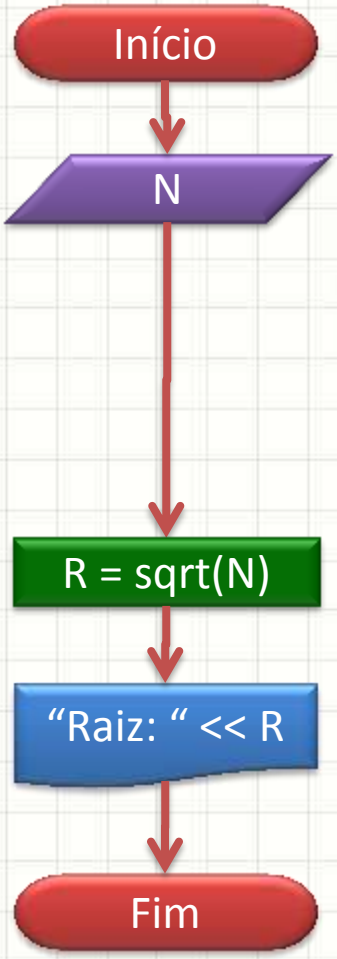
Repetição com Do~While

```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    cout << “Digite no. > 0: ”;
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```



Repetição com Do~While

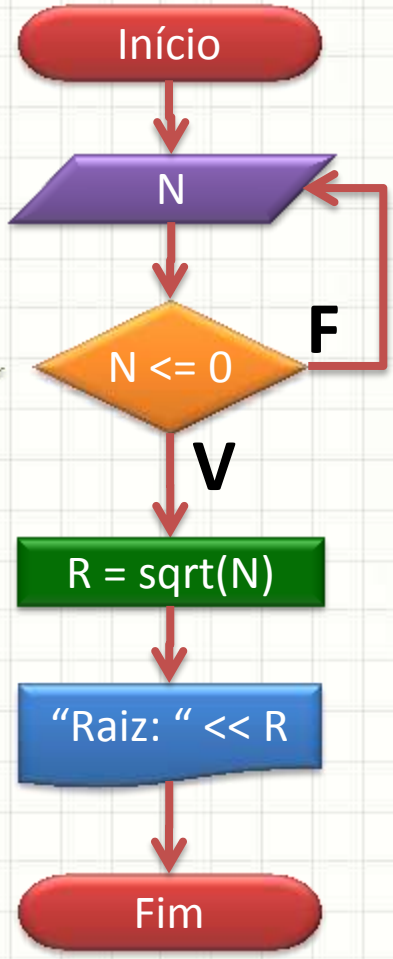
```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    cout << “Digite no. > 0: ”;
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```



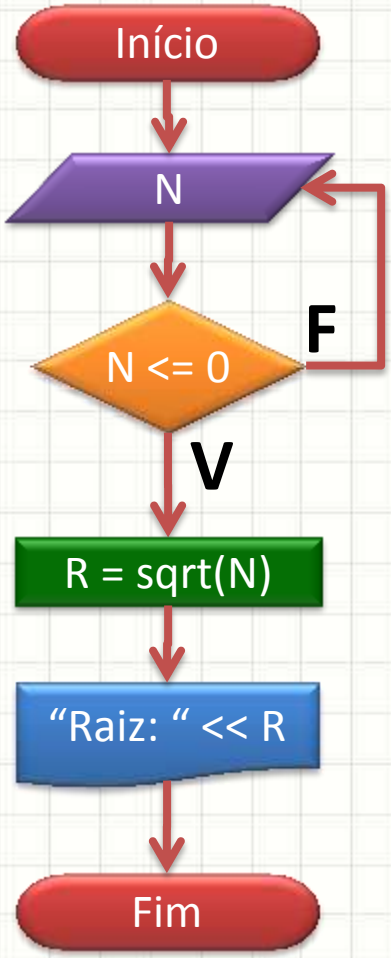
Repetição com Do~While

```
#include <stdio>
#include <math>
#include <iostream>
using nam
int main(
    float
    cout <
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```

while serve?



Repetição com Do~While




```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    do {
        cout << “Digite no. > 0: ”;
        cin >> N;
        } while ( N <= 0 );
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```

Forma Geral do **do~while**

```
do {
    Executa enquanto a proposição for verdadeira
} while ( condição de repetição );
```

- Qual a diferença com relação ao **while** ?

```
while ( condição de repetição ) {
    Executa enquanto a proposição for verdadeira
}
```

EXERCÍCIO

A) Crie um menu para que ele contenha as seguintes opções:

1- Saldo

2- Extrato

3- Saque

4- Depósito

Para cada opção ele deve imprimir um texto que indique a opção selecionada... e ele não deve aceitar opções inválidas

EXERCÍCIO

A) Crie um men
as seguintes op

1- Saldo

2- Extrato

3- Saque

4- Depósit

Para cada opçã
texto que indiq
e ele não deve

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
      int N;

      cout << "1- Saldo" << endl;
      cout << "2- Extrato" << endl;
      cout << "3- Saque" << endl;
      cout << "4- Depósito" << endl;
      do {
         cin >> N;
         } while (N < 1 || N > 4);

      if (N == 1) {
         cout << "Saldo!" << endl;
         }
      if (N == 2) {
         cout << "Extrato!" << endl;
         }
      if (N == 3) {
         cout << "Saque!" << endl;
         }
      if (N == 4) {
         cout << "Depósito!" << endl;
         }

getchar();
}
```

EX
A)
as

Pa
te
e e

```
int N;

cout << "1- Saldo" << endl;
cout << "2- Extrato" << endl;
cout << "3- Saque" << endl;
cout << "4- Depósito" << endl;
do {
   cin >> N;
   } while (N < 1 || N > 4);

if (N == 1) {
   cout << "Saldo!" << endl;
   }
if (N == 2) {
   cout << "Extrato!" << endl;
   }
if (N == 3) {
   cout << "Saque!" << endl;
   }
if (N == 4) {
   cout << "Depósito!" << endl;
   }
```

EXERCÍCIO

B.1) Analise os códigos e descubra qual é mais adequado para **while** e qual para **do~while**:

Código 1

a) Leia um número N;

b) N = N * 2;

c) Se N for menor que 32, volta para o passo (b);

d) Imprima N.

Código 2

a) Leia um número N;

b) Enquanto N for menor que 32, repita (c)

c) N = N * 2;

d) Imprima N.

EXERCÍCIO

B.1) Analise os códigos e descubra qual é mais adequado para **while** e qual para **do~while**:

Código 1

a) Leia um número N;
b) N = N * 2;
c) Se N for menor que 32, volta para o passo (b);
d) Imprima N.

do~while

Código 2

a) Leia um número N;
b) Enquanto N for menor que 32, repita (c)
c) N = N * 2;
d) Imprima N.

while

EXERCÍCIO

B.2) Implemente ambos os códigos (pode ser no mesmo programa):

Código 1

a) Leia um número N;

b) N = N * 2;

c) Se N for menor que 32, volta para o passo (b);

d) Imprima N.

Código 2

a) Leia um número N;

b) Enquanto N for menor que 32, repita (c)

c) N = N * 2;

d) Imprima N.

EXERCÍCIO

B.2) Implemente ambos os códigos (pode ser no mesmo programa):

Código 1

a) Leia um número N;

b) N = N * 2;

c) Se N for menor que 32, volta para o passo (b);

d) Imprima N.

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

    float N;

    cout << "Digite um número: ";
    cin >> N;

    do {
       N = N * 2;
       } while (N < 32);

    cout << "Resultado: " << N << endl;

getchar();
}
```

EXERCÍCIO

B.2) Implemente ambos os códigos (pode ser no mesmo programa):

Código 2

a) Leia um número N;

b) Enquanto N for menor que 32, repita (c)

c) N = N * 2;

d) Imprima N.

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

    float N;

    cout << "Digite um número: ";
    cin >> N;

    while (N < 32) {
       N = N * 2;
       }

    cout << "Resultado: " << N << endl;

getchar();
}
```

EXERCÍCIO

B.3) Execute ambos os códigos para as entradas:

0

20

40

E responda: os resultados são sempre iguais? Por quê?

EXERCÍCIO

B.3) Execute ambos os códigos para as entradas:

0

20

40

E responda: os resultados são sempre iguais? Por quê?

Valor de Entrada:	Resultado Código 1	Resultado Código 2
0	Travou	Travou
20	40	40
40	80	40

EXERCÍCIO

C.1) Faça um programa que receba dois inteiros N1 e N2 e calcule a soma de todos os números entre eles (inclusive N1 e N2).

Exemplo:

Se N1 = 10 e N2 = 16, o programa deve calcular

10 + 11 + 12 + 13 + 14 + 15 + 16

EXERCÍCIO

C.1)

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
    int N1, N2, SOMA;
    cout << "Digite o primeiro número: ";
    cin >> N1;
    cout << "Digite o segundo número: ";
    cin >> N2;
    SOMA = 0;
    do {
       SOMA = SOMA + N1;
       N1 = N1 + 1;
       } while (N1 <= N2);
    cout << "Resultado: " << SOMA << endl;
getchar();
}
```

EXERCÍCIO

C.2) Faça um programa que receba o salário atual do funcionário (SAL) a taxa de aumento anual (TAXA) e o número de anos (ANOS) e calcule o salário do funcionário depois que esses anos se passarem.

EXERCÍCIO

C.2)

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

    float SAL, TAXA;

    int ANOS;

    cout << "Digite o salário atual: ";
    cin >> SAL;
    cout << "Digite a taxa de reajuste anual: ";
    cin >> TAXA;
    cout << "Digite o número de anos: ";
    cin >> ANOS;

    while (ANOS > 0) {
       SAL = SAL + (SAL * TAXA);
       ANOS = ANOS - 1;
       }

    cout << "Salário final: " << SAL << endl;

getchar();
}
```

EXERCÍCIO

C 2)

```
#include <stdio>
float SAL, TAXA;

int ANOS;

cout << "Digite o salário atual: ";
cin >> SAL;
cout << "Digite a taxa de reajuste anual: ";
cin >> TAXA;
cout << "Digite o número de anos: ";
cin >> ANOS;

while (ANOS > 0) {
    SAL = SAL + (SAL * TAXA);
    ANOS = ANOS - 1;
    }

cout << "Salário final: " << SAL << endl;
}
```

EXERCÍCIO

C.3) Faça um programa que leia o número de cidades N e, para cada uma delas, pergunte o número de nascimentos (NASC) no último ano. O programa deve responder o número de nascimentos da cidade onde ocorreu mais nascimentos e também a média de nascimentos por cidade.

EXERCÍCIO

C.3)

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

    int N, CIDADES, NASC;
    int MAX, TOTAL;
    float MEDIA;

    cout << "Digite o número de cidades: ";
    cin >> CIDADES;

    TOTAL = 0;
    MAX = 0;
    for (N = CIDADES; N > 0; N = N -1) {
       cout << "Quantos nascimentos na cidade " << N << "? ";
       cin >> NASC;

       TOTAL = TOTAL + NASC;
       if (NASC > MAX) MAX = NASC;
       }

    MEDIA = (float)TOTAL/(float)CIDADES;

    cout << "Máximo: " << MAX << endl;
    cout << "Média: " << MEDIA << endl;

getchar();
}
```

EXERCÍCIO

```
int N, CIDADES, NASC;
int MAX, TOTAL;
float MEDIA;

cout << "Digite o número de cidades: ";
cin >> CIDADES;

TOTAL = 0;
MAX = 0;
for (N = CIDADES; N > 0; N = N -1) {
   cout << "Quantos nascimentos na cidade " << N << "? ";
   cin >> NASC;

   TOTAL = TOTAL + NASC;
   if (NASC > MAX) MAX = NASC;
   }

MEDIA = (float)TOTAL/(float)CIDADES;

cout << "Máximo: " << MAX << endl;
cout << "Média: " << MEDIA << endl;
```

```
getchar();
}
```

Conclusões

Resumo

- Existem diversos tipos de estruturas de decisão
- Elas são intercambiáveis, isto é, tudo que se faz com um uma, é possível fazer com outra
- Dependendo da situação, cada uma delas é mais apropriada!

- **TAREFA!**
  - **Lista de Exercícios 2!**

Próxima Aula

- Chegamos ao fim do curso de lógica...
  - Treinar... treinar... e treinar!
  - Aulas finais: exercícios diversos!

PERGUNTAS?

Bom Descanso a Todos!