

CP63B-DPGR3A

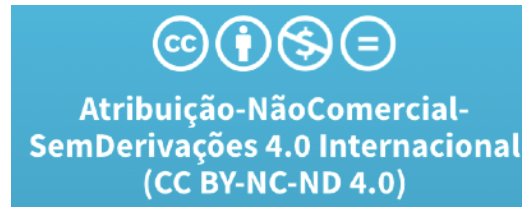
COMPUTAÇÃO 2

Aula 01A - Revisão

Prof. Rafael G. Mantovani

Licença

Este trabalho está licenciado com uma Licença CC BY-NC-ND 4.0:



maiores informações:

https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR

Roteiro

- 1 Objetivo**
- 2 Algoritmos**
- 3 Estrutura Programa em C**
- 4 Estruturas de Controle - Condicionais**
- 5 Exercícios**
- 6 Estruturas de Controle - Repetição**
- 7 Exercícios**
- 8 Referências**

Roteiro

- 1 Objetivo**
- 2 Algoritmos**
- 3 Estrutura Programa em C**
- 4 Estruturas de Controle - Condicionais**
- 5 Exercícios**
- 6 Estruturas de Controle - Repetição**
- 7 Exercícios**
- 8 Referências**

Objetivo



Relembrar

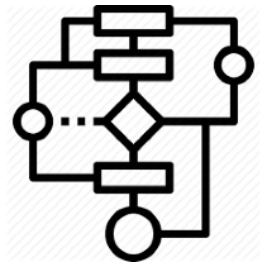
Objetivo

**Nosso objetivo nessa
aula é RELEMBRAR :)**



Relembrar

Objetivo

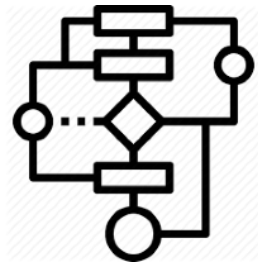


Algoritmo



Relembrar

Objetivo



Algoritmo

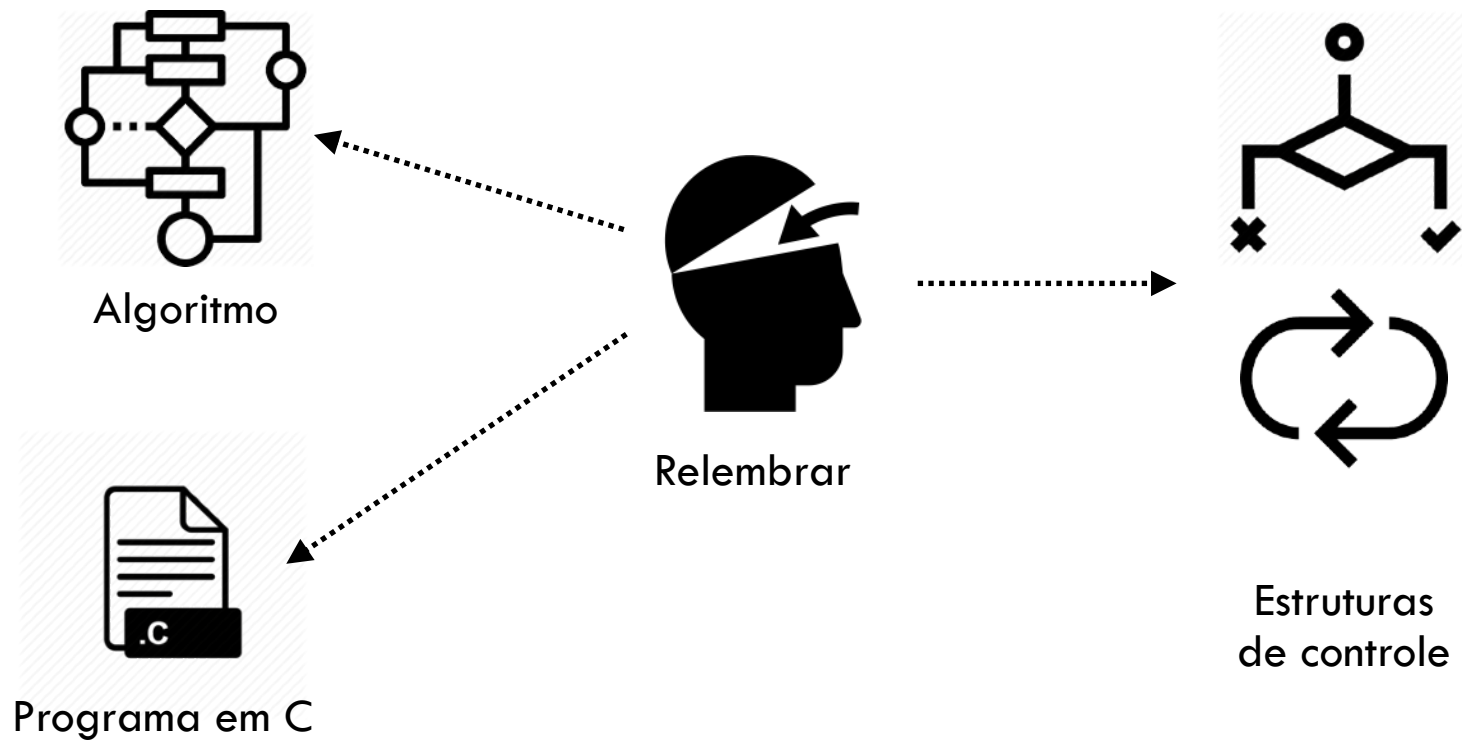


Relembrar

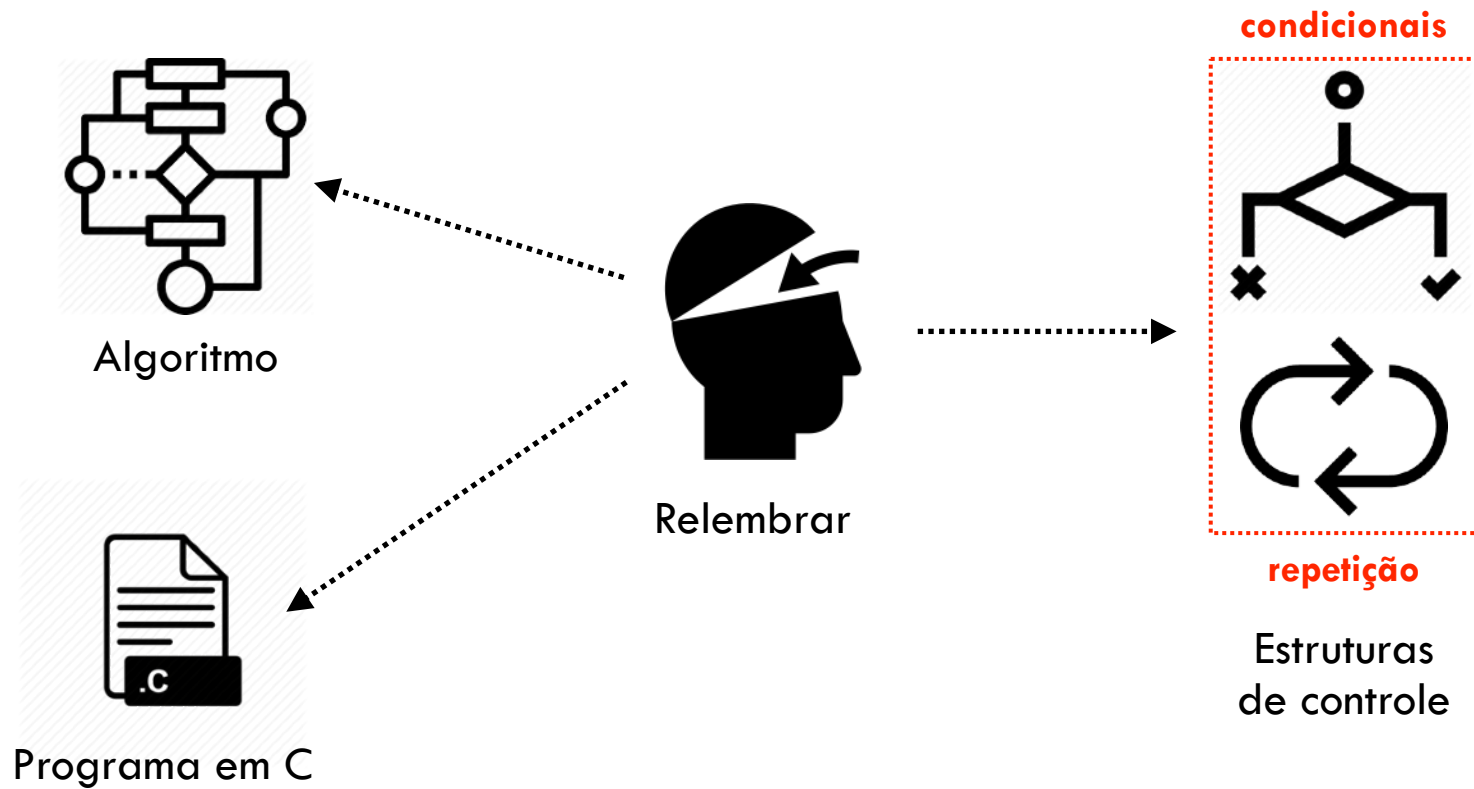


Programa em C

Objetivo



Objetivo



Roteiro

- 1 Objetivo
- 2 Algoritmos
- 3 Estrutura Programa em C
- 4 Estruturas de Controle - Condicionais
- 5 Exercícios
- 6 Estruturas de Controle - Repetição
- 7 Exercícios
- 8 Referências

Algoritmo?



Algoritmo?



**o que é um
ALGORITMO?**

Algoritmo?

**o que é um
ALGORITMO?**

"Conjunto de passos finitos e organizados, que quando executados, resolvem um determinado problema."

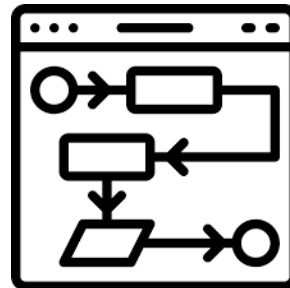
"Conjunto de regras para a solução de um problema."

"Sequência finita de ações executáveis que visam obter uma solução para um determinado tipo de problema".

Algoritmo?



Problema



algoritmo



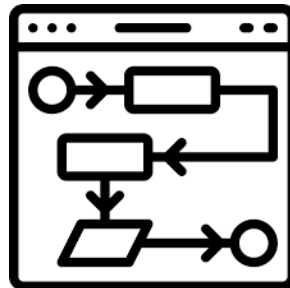
solução

Algoritmo?

Esse é nosso fluxo básico de programação



Problema



algoritmo



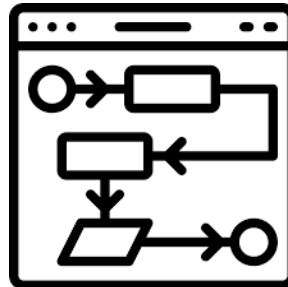
solução

Algoritmo?



Problema

**Problema do
mundo real
(dados)**



algoritmo

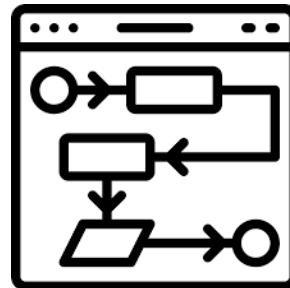


solução

Algoritmo?



Problema



algoritmo



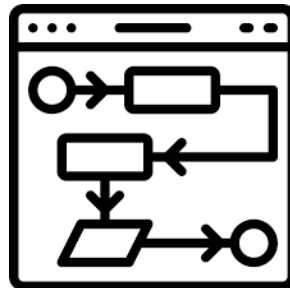
solução

**Algoritmo
(lógica) de
resolução**

Algoritmo?



Problema



algoritmo



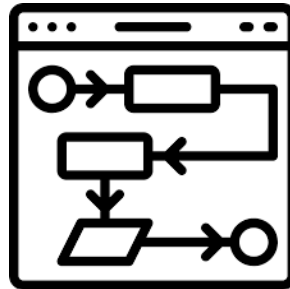
solução

**Programa
(C)**

Algoritmo?



Problema



algoritmo



solução

Algoritmo?



Problema

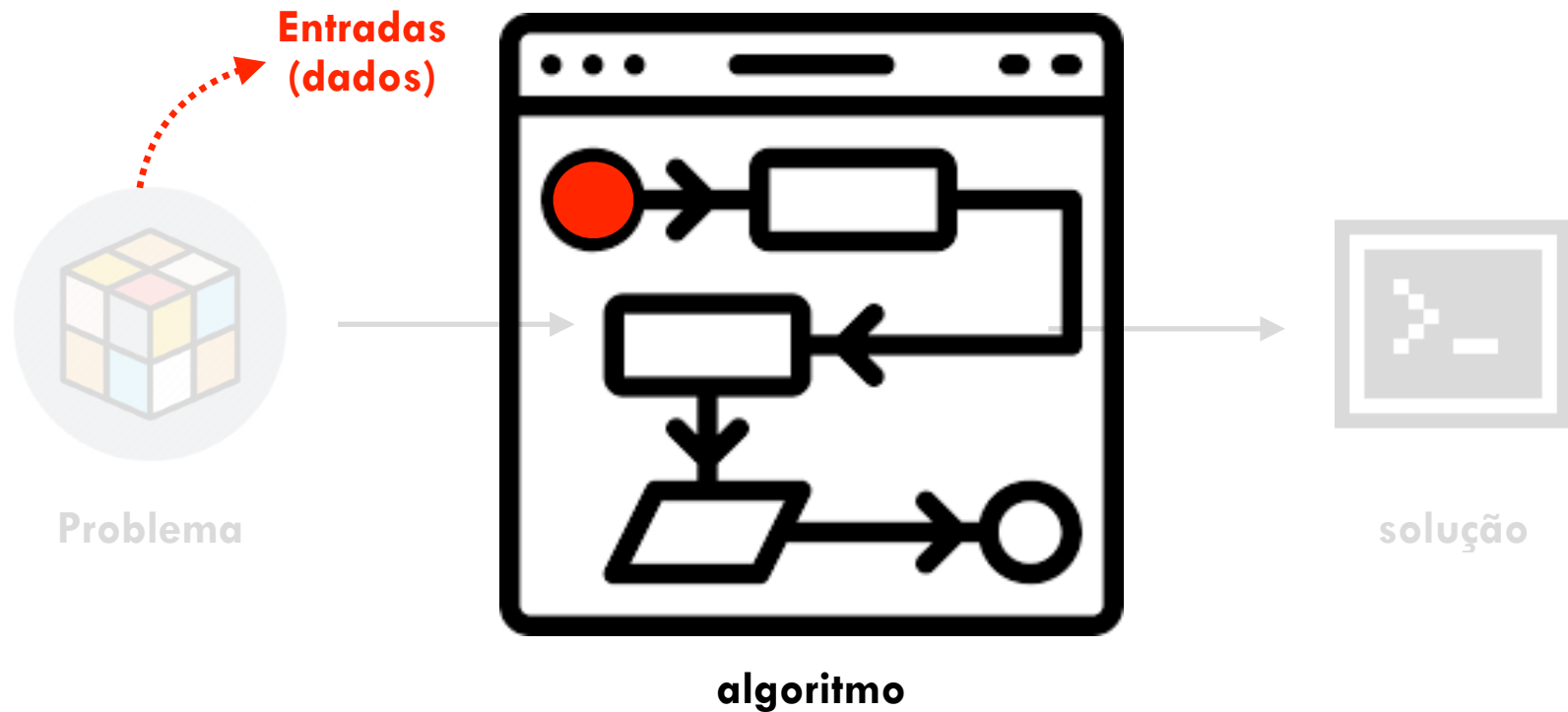


algoritmo



solução

Algoritmo?



Algoritmo?



Problema



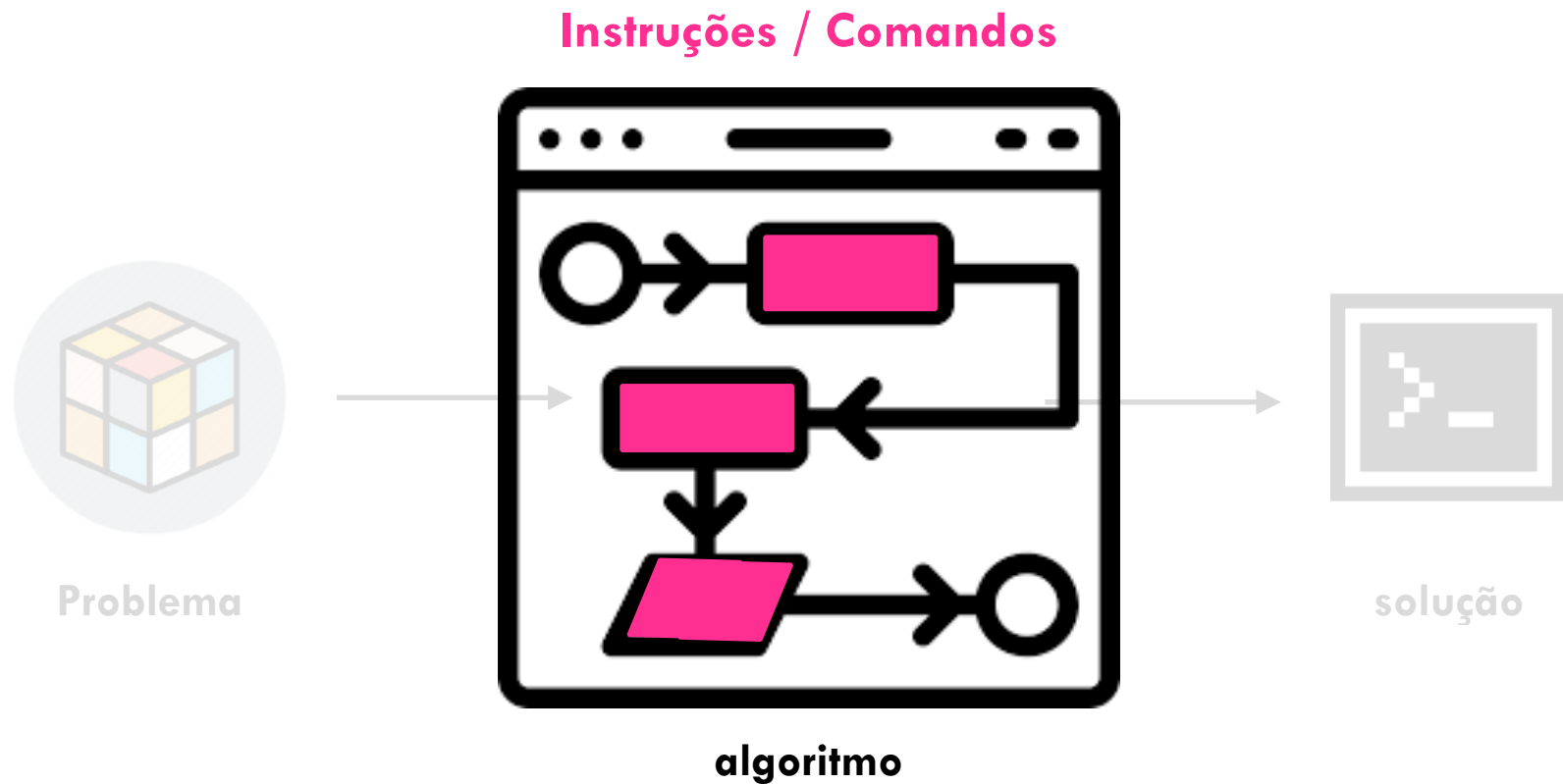
algoritmo

Saídas

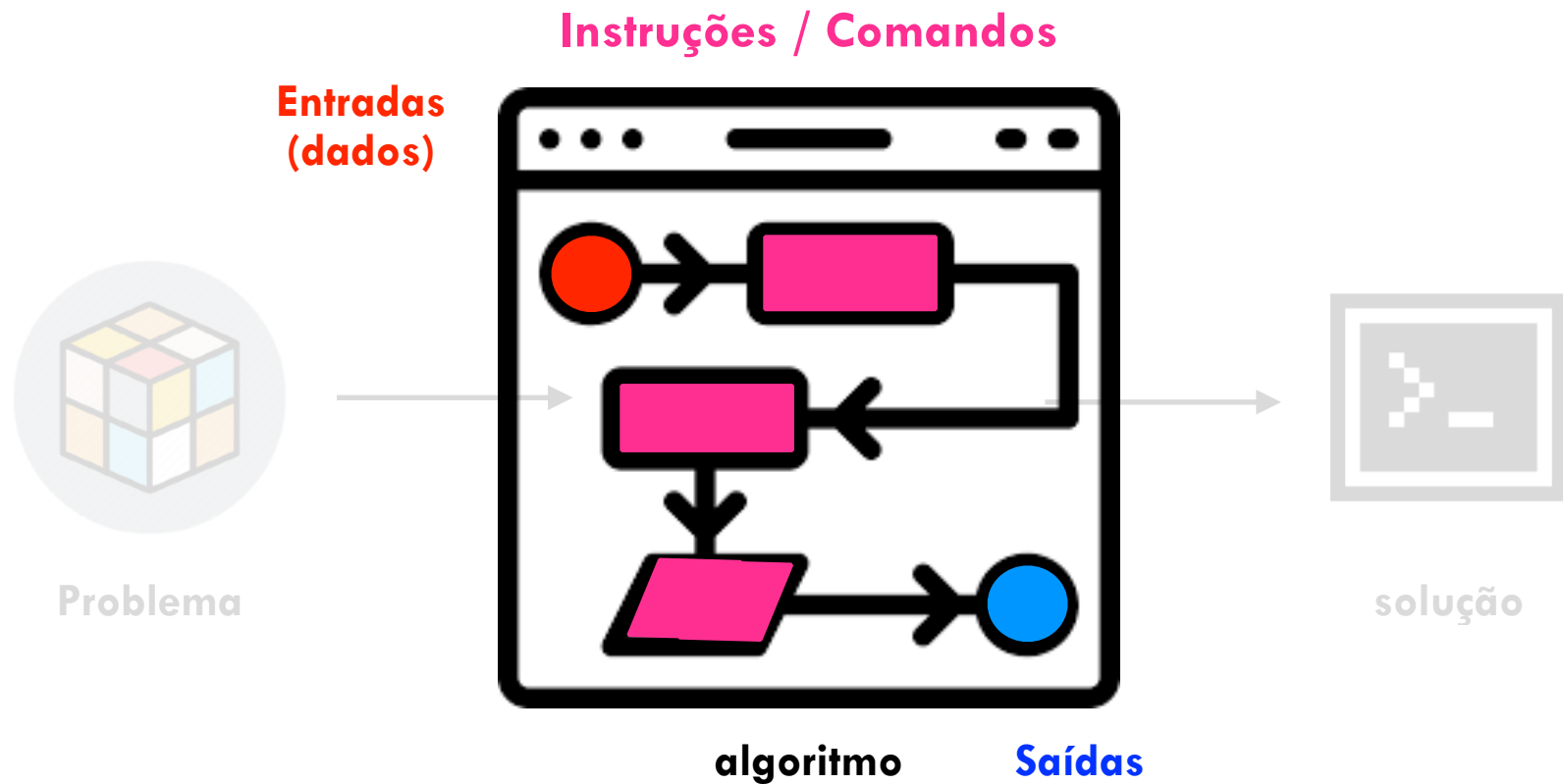


solução

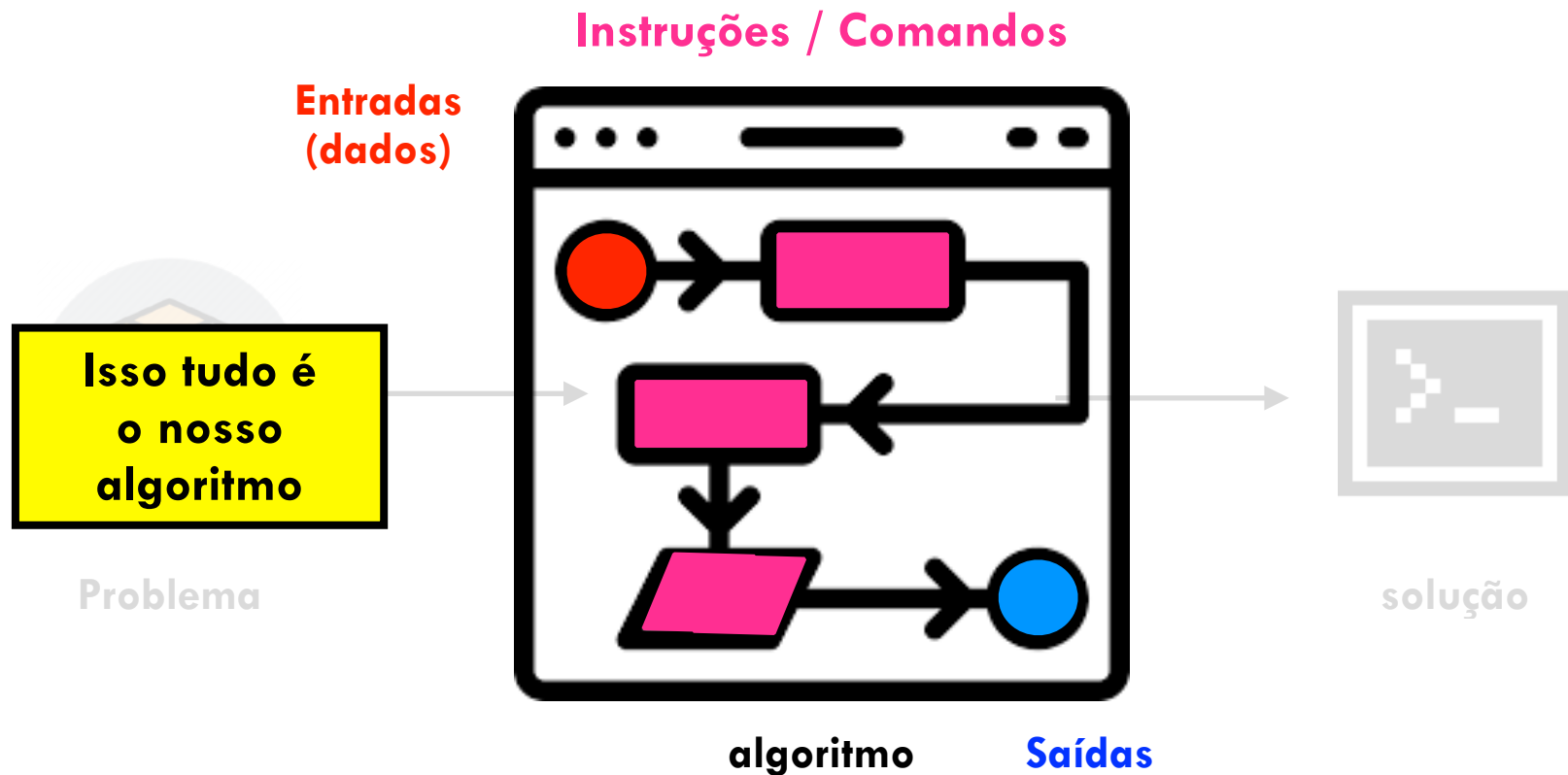
Algoritmo?



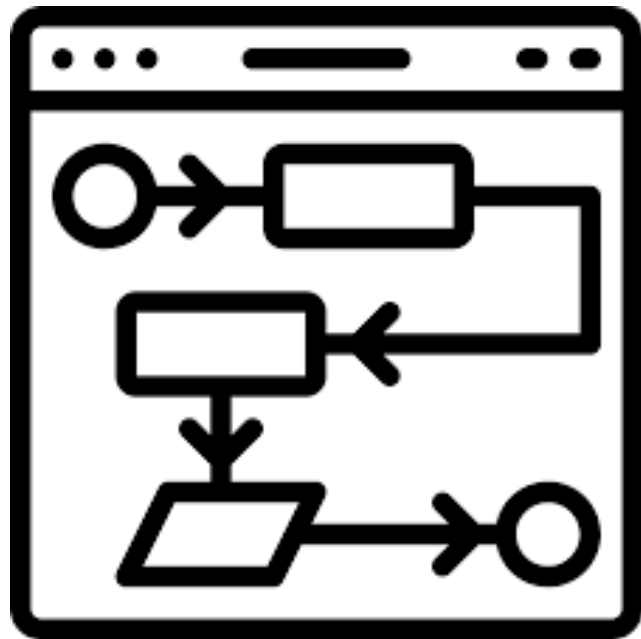
Algoritmo?



Algoritmo?



Algoritmo?

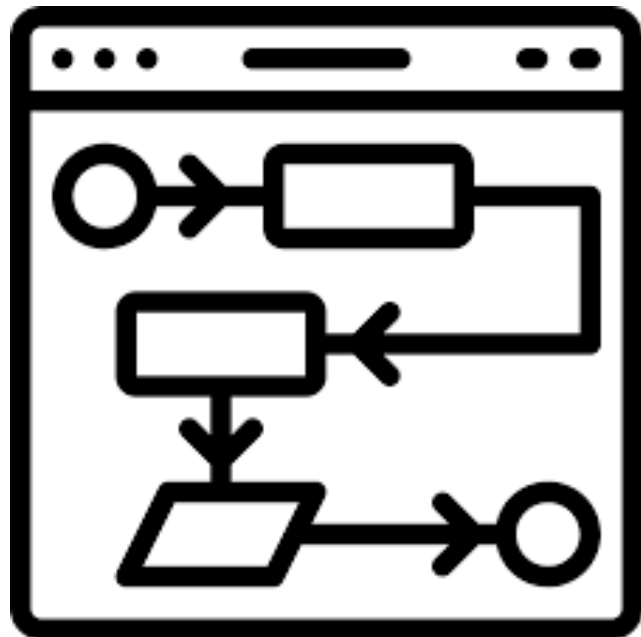


Pseudocódigo

```
Algoritmo Media
Var N1, N2, Media : real
Início
    Leia N1, N2
    Media  $\leftarrow$  (N1+N2)/2
    Se Media >= 7 Entao
        Escreva "Aprovado"
    Senao
        Escreva "Reprovado"
Fim.
```

Algoritmo?

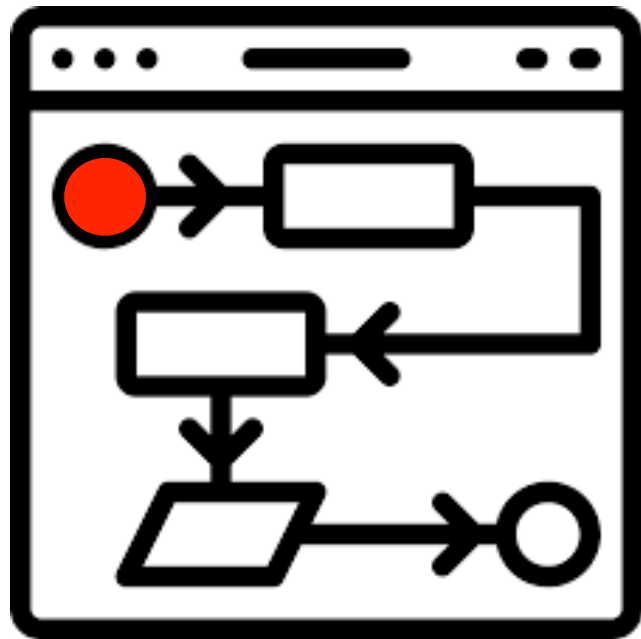
Convertemos nosso algoritmo para uma sequencia de passos em Pseudocódigo (lógica)



Pseudocódigo

```
Algoritmo Media
Var N1, N2, Media : real
Início
  Leia N1, N2
  Media ← (N1+N2)/2
  Se Media >= 7 Então
    Escreva "Aprovado"
  Senao
    Escreva "Reprovado"
Fim.
```

Algoritmo?



Pseudocódigo

Algoritmo Media

Var N1, N2, Media : real

Início

Leia N1, N2

Media \leftarrow (N1+N2)/2

Se Media \geq 7 **Entao**

Escreva "Aprovado"

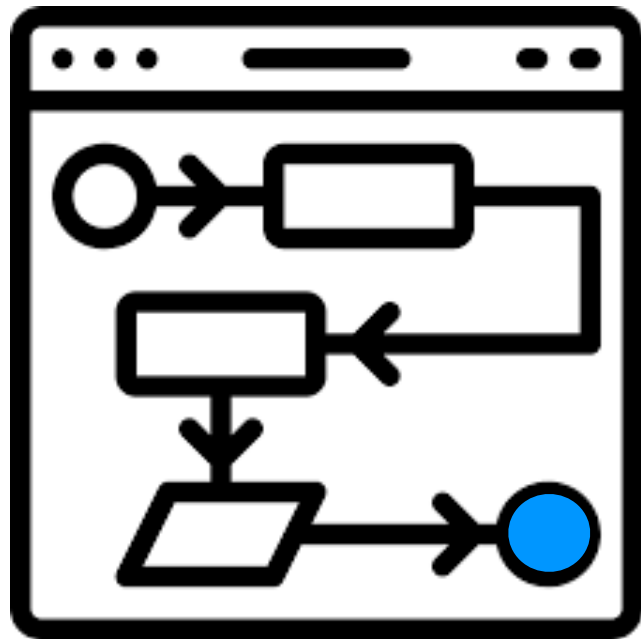
Senao

Escreva "Reprovado"

Fim.

**Entradas
(dados)**

Algoritmo?



Pseudocódigo

Algoritmo Media

Var N1, N2, Media : real

Início

Leia N1, N2

 Media \leftarrow (N1+N2)/2

Se Media \geq 7 **Entao**

Escreva "Aprovado"

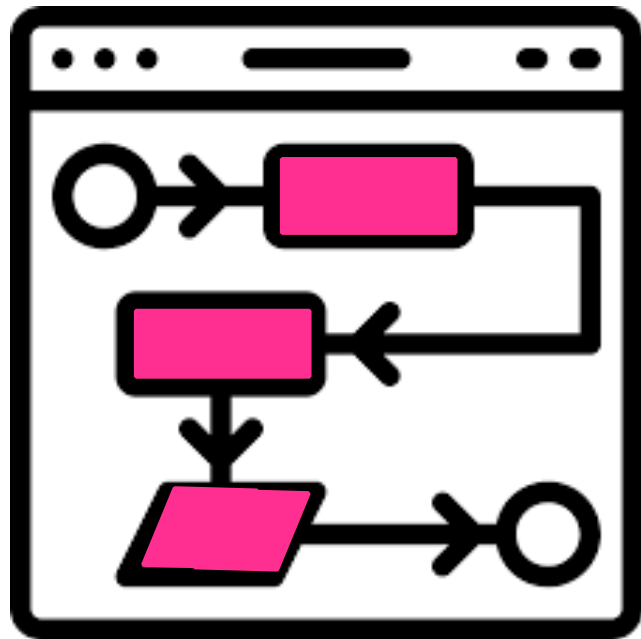
Senao

Escreva "Reprovado"

Fim.

Saídas

Algoritmo?



Pseudocódigo

Algoritmo Media

Var N1, N2, Media : real

Início

Leia N1, N2

Media \leftarrow (N1+N2)/2

Se Media \geq 7 **Entao**

Escreva "Aprovado"

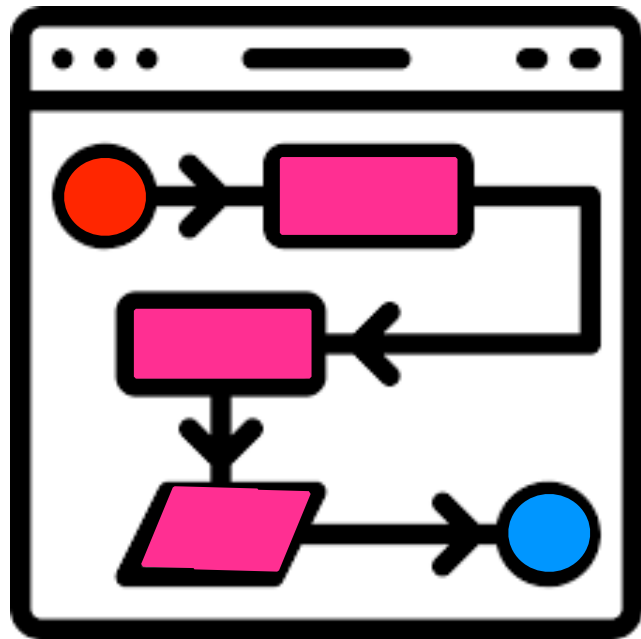
Senao

Escreva "Reprovado"

Fim.

Instruções / Comandos

Algoritmo?



Pseudocódigo

Algoritmo Media

Var N1, N2, Media : real

Início

Leia N1, N2

Media \leftarrow (N1+N2)/2

Se Media \geq 7 **Entao**

Escreva "Aprovado"

Senao

Escreva "Reprovado"

Fim.

**Entradas
(dados)**

Saídas

Instruções / Comandos

Exemplos



Exemplos



Exemplos

Algoritmo 1

```
Algoritmo AtravessarRua
  Olhar para a direita
  Olhar para a esquerda
  Se estiver vindo carro
    Não Atravesse
  Senão
    Atravesse
  Fim-Se
Fim-Algoritmo
```

Algoritmo 2

```
Algoritmo AtravessarRua
  Olhar para a esquerda
  Olhar para a direita
  Se não estiver vindo carro
    Atravesse
  Senão
    Não Atravesse
  Fim-Se
Fim-Algoritmo
```


Exemplos

Algoritmo 1



```
Algoritmo AtravessarRua
  Olhar para a direita
  Olhar para a esquerda
  Se estiver vindo carro
    Não Atravesse
  Senão
    Atravesse
  Fim-Se
Fim-Algoritmo
```

Algoritmo 2



```
Algoritmo AtravessarRua
  Olhar para a esquerda
  Olhar para a direita
  Se não estiver vindo carro
    Atravesse
  Senão
    Não Atravesse
  Fim-Se
Fim-Algoritmo
```

Exemplos

Algoritmo 1



```
Algoritmo AtravessarRua
  Olhar para a direita
  Olhar para a esquerda
  Se estiver vindo carro
    Não Atravesse
  Senão
    Atravesse
  Fim-Se
Fim-Algoritmo
```

Algoritmo 3

```
Algoritmo AtravessarRua
  Atravesse
  Se estiver vindo carro
    Olhar para a direita
  Senão
    Olhar para a esquerda
  Fim-Se
  Não Atravesse
Fim-Algoritmo
```

Exemplos

Algoritmo 1



```
Algoritmo AtravessarRua
  Olhar para a direita
  Olhar para a esquerda
  Se estiver vindo carro
    Não Atravesse
  Senão
    Atravesse
  Fim-Se
Fim-Algoritmo
```

Algoritmo 3



```
Algoritmo AtravessarRua
  Atravesse
  Se estiver vindo carro
    Olhar para a direita
  Senão
    Olhar para a esquerda
  Fim-Se
  Não Atravesse
Fim-Algoritmo
```

Exemplos

**Guia de sobrevivência:
como fazer um miojo?**



Dicas na construção de algoritmos

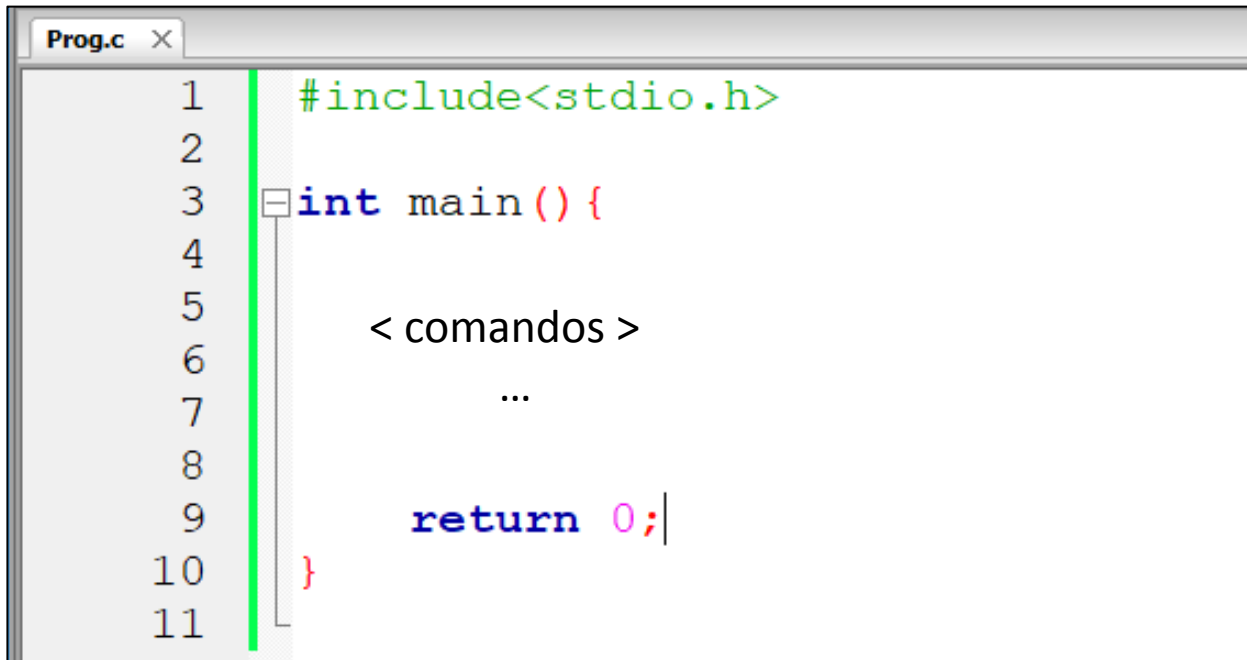
Seguir os seguintes passos:

- 1) Entender completamente o problema, destacando os pontos mais importantes.
- 2) **Definir os dados de entrada.**
- 3) Definir o processamento, ou seja, quais cálculos serão executados e as suas restrições. O processamento é responsável pela transformação dos dados de entrada em informações de saída.
- 4) **Definir os dados de saída**, ou seja, o que será gerado após o processamento.
- 5) Construir o algoritmo.
- 6) Testar o algoritmo realizando simulações

Roteiro

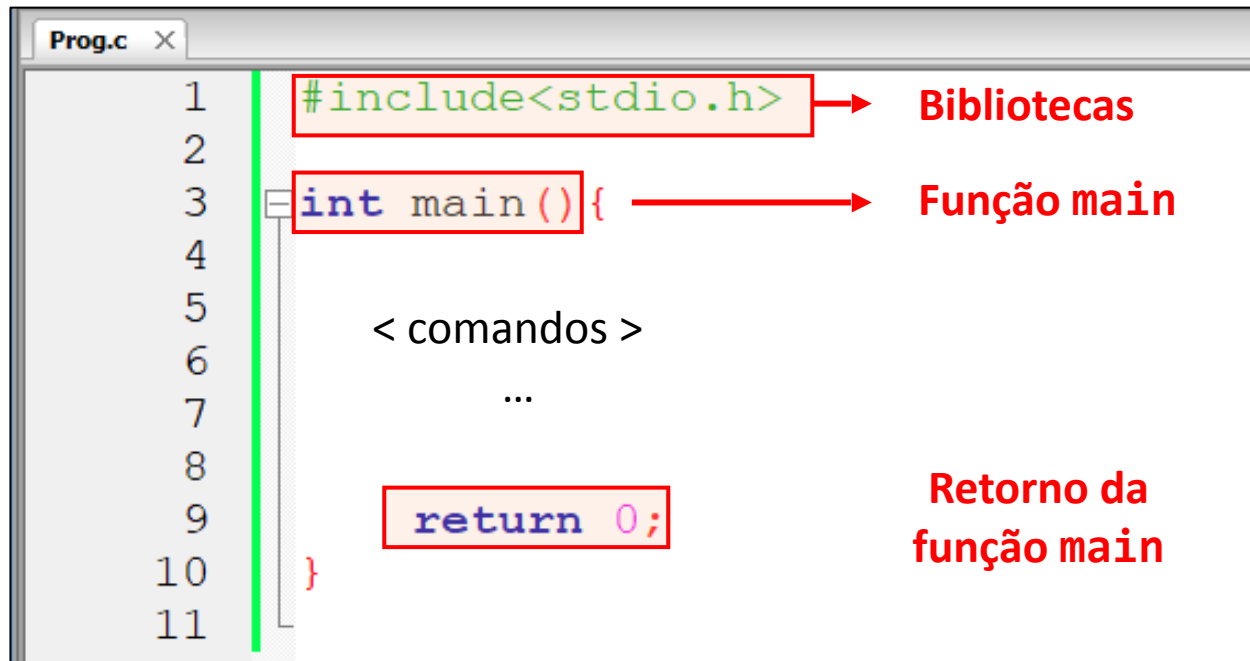
- 1 Objetivo
- 2 Algoritmos
- 3 Estrutura Programa em C
- 4 Estruturas de Controle - Condicionais
- 5 Exercícios
- 6 Estruturas de Controle - Repetição
- 7 Exercícios
- 8 Referências

Estrutura básica em C

A screenshot of a code editor window titled 'Prog.c'. The editor shows a basic C program structure. Line 1: #include<stdio.h>. Line 2: (empty). Line 3: int main() {. Line 4: (empty). Line 5: < comandos >. Line 6: ... (three dots). Line 7: (empty). Line 8: (empty). Line 9: return 0;. Line 10: }. Line 11: (empty). A vertical green line is positioned at the start of line 3. A small square icon is next to line 3. A vertical line with a bracket connects line 3 to line 10.

```
1  #include<stdio.h>
2
3  int main() {
4
5      < comandos >
6      ...
7
8
9      return 0;
10 }
11
```

Estrutura básica em C



The image shows a code editor window titled "Prog.c" with a line number column on the left (1 to 11). The code is as follows:

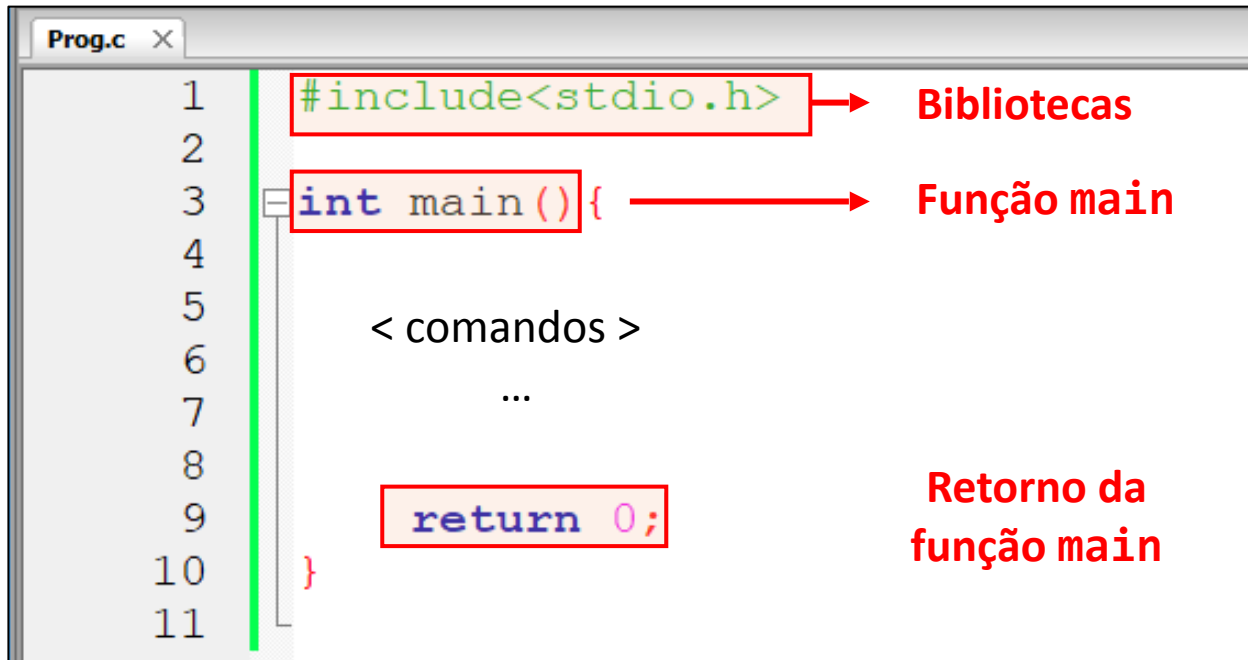
```
1 #include<stdio.h>
2
3 int main(){
4     < comandos >
5     ...
6
7     return 0;
8 }
9
10
11
```

Annotations with red boxes and arrows:

- A red box around line 1 (`#include<stdio.h>`) has an arrow pointing to the text **Bibliotecas**.
- A red box around line 3 (`int main(){`) has an arrow pointing to the text **Função main**.
- A red box around line 7 (`return 0;`) has an arrow pointing to the text **Retorno da função main**.

Indica o fim da execução da função main

Estrutura básica em C



The image shows a code editor window titled 'Prog.c' with line numbers 1 through 11. The code is as follows:

```
1 #include<stdio.h>
2
3 int main(){
4     < comandos >
5     ...
6
7     return 0;
8 }
9
10
11
```

Annotations with red boxes and arrows:

- A red box around line 1, `#include<stdio.h>`, with an arrow pointing to the text **Bibliotecas**.
- A red box around line 3, `int main(){`, with an arrow pointing to the text **Função main**.
- A red box around line 7, `return 0;`, with an arrow pointing to the text **Retorno da função main**.

Indica o fim da execução da função main

A execução do programa inicia-se pela função **main**.

Bibliotecas → contém conjunto de funções e instruções previamente estabelecidas e que podem ser usadas pela programa

Declaração de variáveis

- Uma variável sempre deve ser declarada antes da sua execução.
 - Declarar indica ao compilador para reservar um espaço na memória para guardar os valores dessa variável;

Exemplos de declaração

```
int main(){  
  
int num1;  
int num2;  
float resultado;  
float multiplicação;  
.  
.  
.
```

=

```
int main(){  
  
int num1, num2;  
float resultado, multiplicação;  
.  
.  
.
```

Entrada e saída de dados



Entrada e saída de dados



Entrada (IN):

```
scanf(formato, &variavel);
```

Exemplos:

```
scanf("%f", &n);  
scanf("%d", &num);  
scanf("%d", &cont);
```

Entrada e saída de dados



Entrada (IN):

```
scanf(formato, &variável);
```

Exemplos:

```
scanf("%f", &n);  
scanf("%d", &num);  
scanf("%d", &cont);
```

Saída (OUT):

```
printf(conteúdo, [parâmetros]);
```

Exemplos:

```
printf("%f", n);  
printf("Exame");  
printf("O numero e %d", num);  
printf("são necessárias %d moedas", cont);
```


Entrada e saída de dados

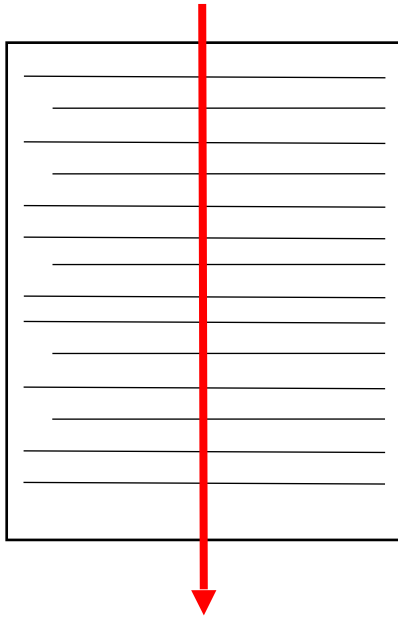
Códigos usados na função printf()

Código	Formato
%c	Caractere
%d	Inteiro decimal com sinal
%e	Notação científica
%f	Ponto flutuante decimal
%g	Menor representação entre %f e %e
%o	Octal sem sinal
%s	String de caracteres
%u	Inteiros decimais sem sinal (<i>unsigned int</i>)
%x	Hexadecimal sem sinal
%%	Símbolo %

Roteiro

- 1 Objetivo
- 2 Algoritmos
- 3 Estrutura Programa em C
- 4 Estruturas de Controle - Condicionais
- 5 Exercícios
- 6 Estruturas de Controle - Repetição
- 7 Exercícios
- 8 Referências

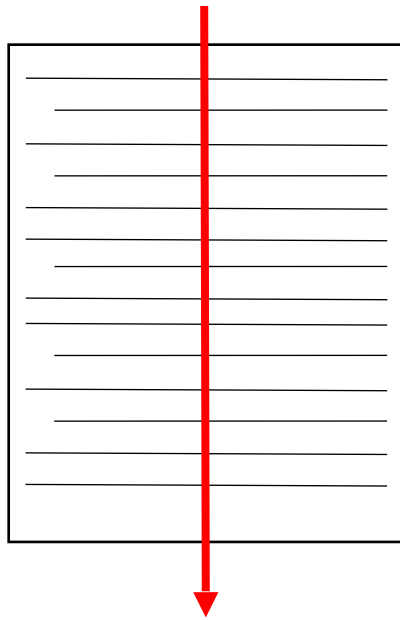
Estruturas de Controle



Fluxo de execução
Sequencial

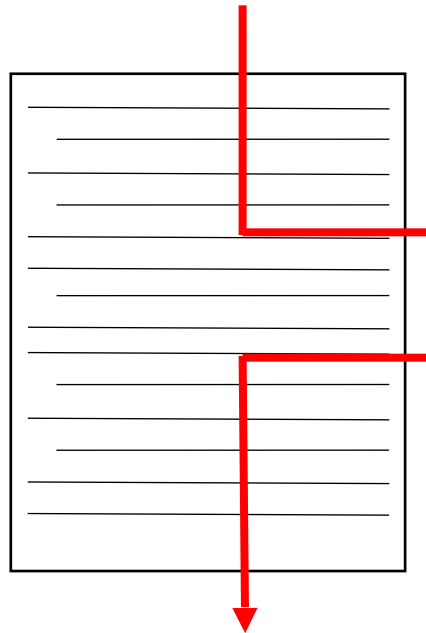
*Comandos são executados
um após o outro*

Estruturas de Controle



Fluxo de execução
Sequencial

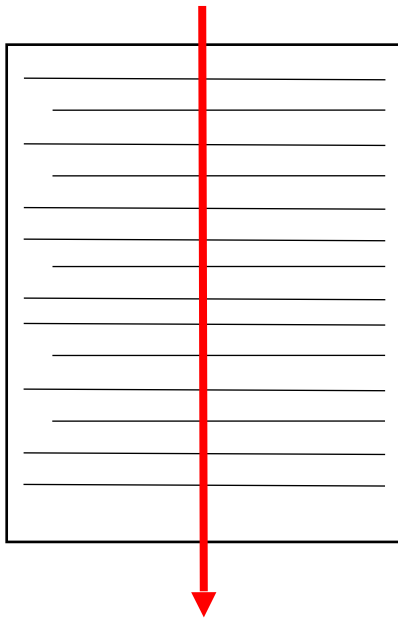
*Comandos são executados
um após o outro*



Fluxo de execução
com desvio

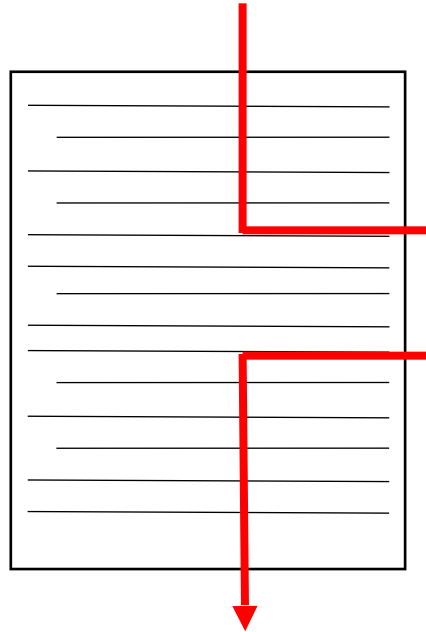
*Comandos são executados
dependendo do valor de uma
condição*

Estruturas de Controle



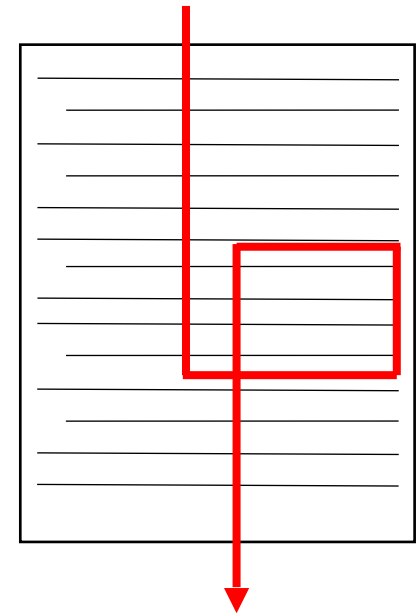
Fluxo de execução
Sequencial

*Comandos são executados
um após o outro*



Fluxo de execução
com desvio

*Comandos são executados
dependendo do valor de uma
condição*



Fluxo de execução
repetitivo

*Comandos são executados
de forma repetida*

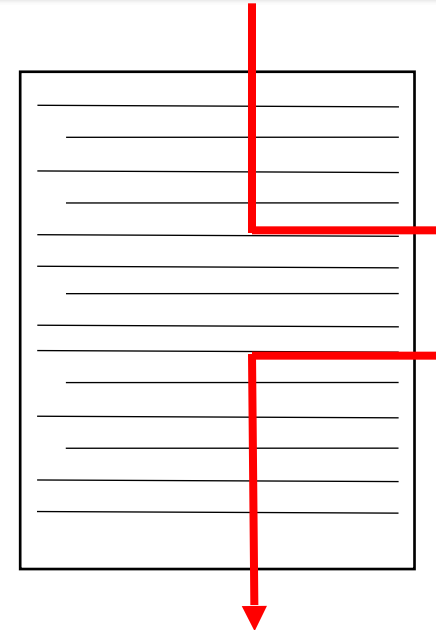
A) Estruturas Condicionais

- Estruturas de decisão em C/C++:

if

if-else

switch



Fluxo de execução
com desvio

*Comandos são executados
dependendo do valor de uma
condição*

A) Estruturas Condicionais

- Os operadores relacionais comparam dois valores e retornam um valor booleano
 - Verdadeiro (*true*);
 - Falso (*false*);

Operador	Descrição	X	Y	Lógico	Resultado
==	Igual a	2	3	X == Y	Falso
!=	Diferente de	2	3	X != Y	Verdadeiro
>	Maior que	2	3	X > Y	Falso
>=	Maior ou Igual	2	3	X >= Y	Falso
<	Menor que	2	3	X < Y	Verdadeiro
<=	Menor ou igual	2	3	X <= Y	Verdadeiro

Declaração IF

Estrutura condicional Simples

```
//comandos
```

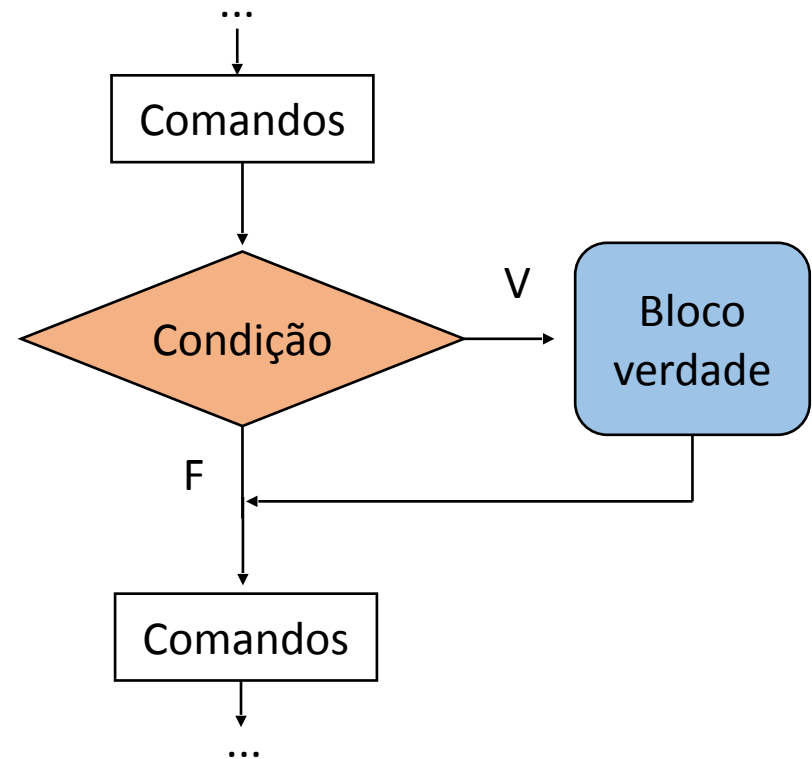
```
if(condição)
```

```
{
```

```
    //bloco verdade
```

```
}
```

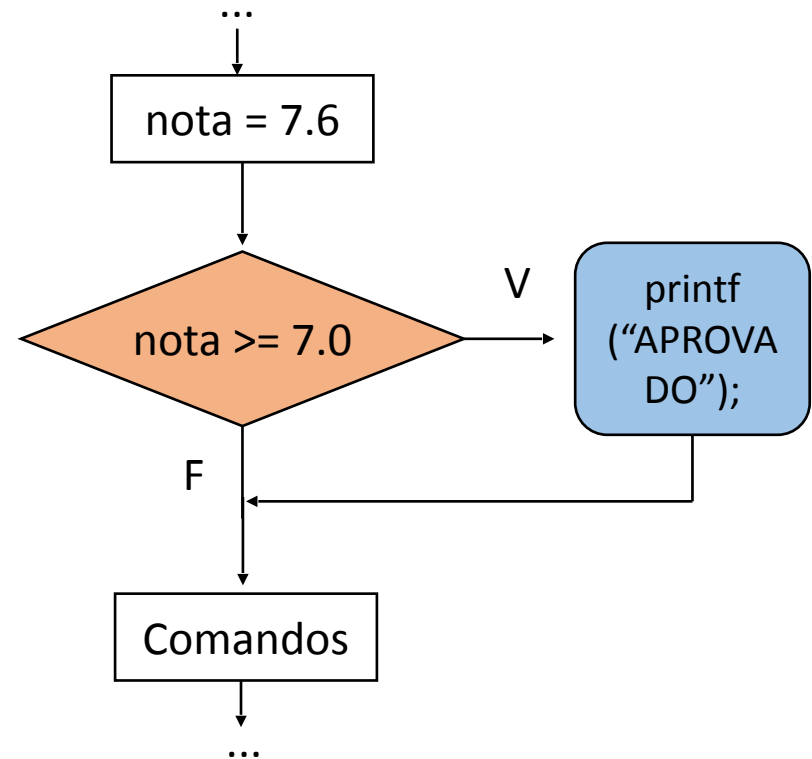
```
//comandos
```



Declaração IF

Estrutura condicional Simples

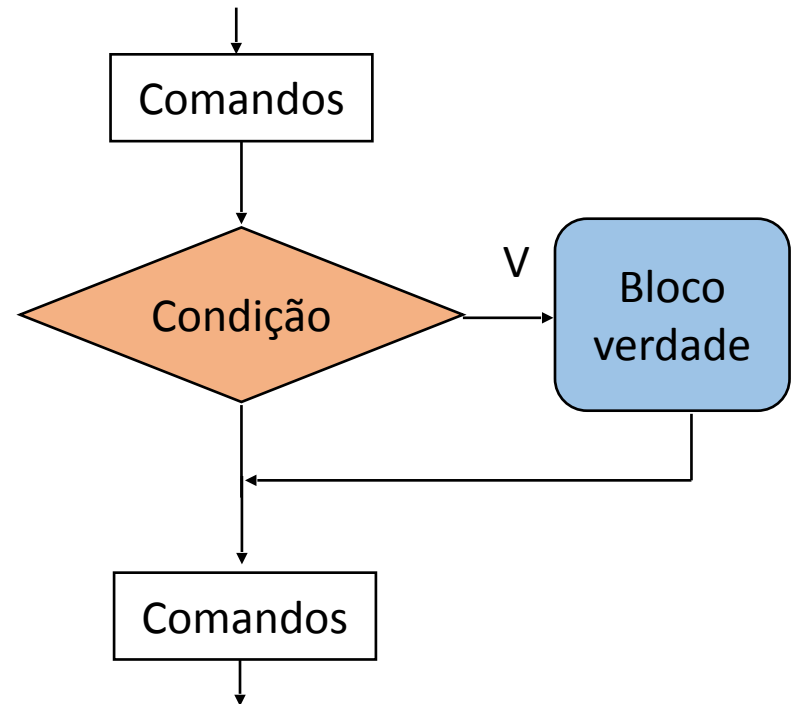
```
1. int main() {  
2.  
3.     float nota = 7.6;  
4.  
5.     if(nota >= 7.0)  
6.         printf("APROVADO!");  
7.  
8.     ...  
9.     ...  
10.  
11.     return 0;  
12. }
```



Declaração IF

Estrutura condicional Simples

```
1. int main( ) {  
2.  
3.     float nota1 = 7.6;  
4.     float nota2 = 5.0;  
5.     float media;  
6.  
7.     if(nota1 == 10.0)  
8.         printf("PARABÉNS!");  
9.  
10.    media = (nota1 + nota2)/2;  
11.    printf("Sua media e': %f", media);  
12.    ...  
13.    ...  
14.  
15.    return 0;  
16. }
```



Declaração IF-ELSE

```
//comandos
```

```
if(condição)
```

```
{
```

```
    //bloco verdade
```

```
}
```

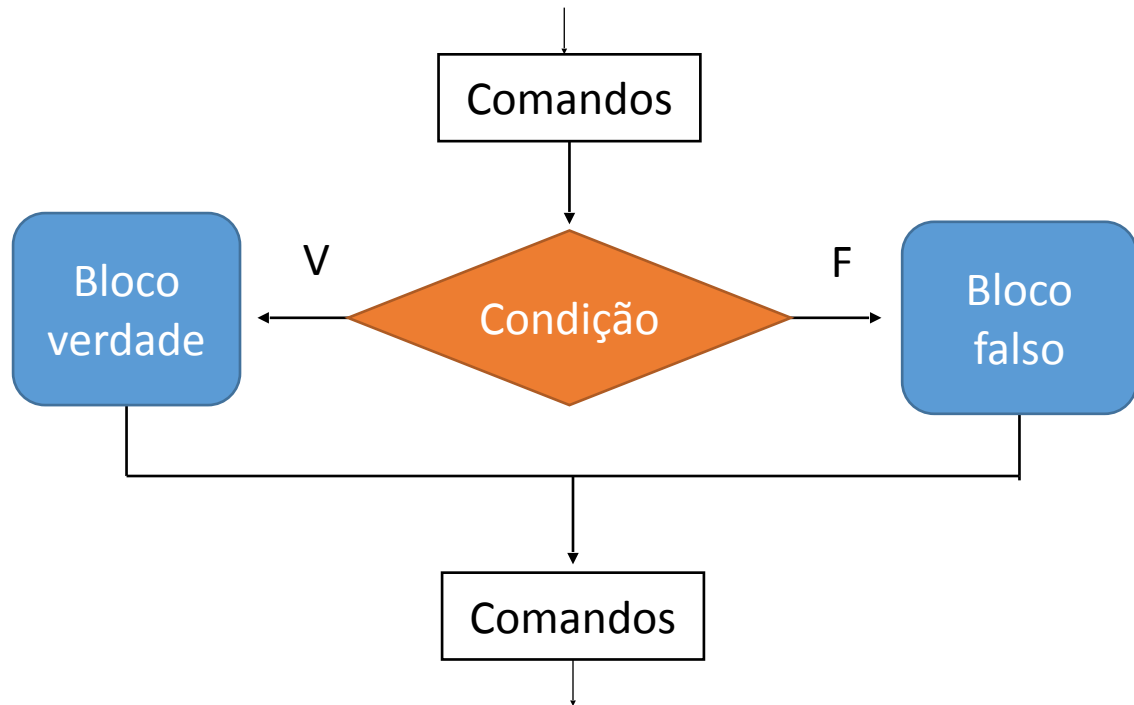
```
else
```

```
{
```

```
    //bloco falso
```

```
}
```

```
//comandos
```



Declaração IF-ELSE

```
//comandos
```

```
if(condição)
```

```
{
```

```
    //bloco verdade
```

```
}
```

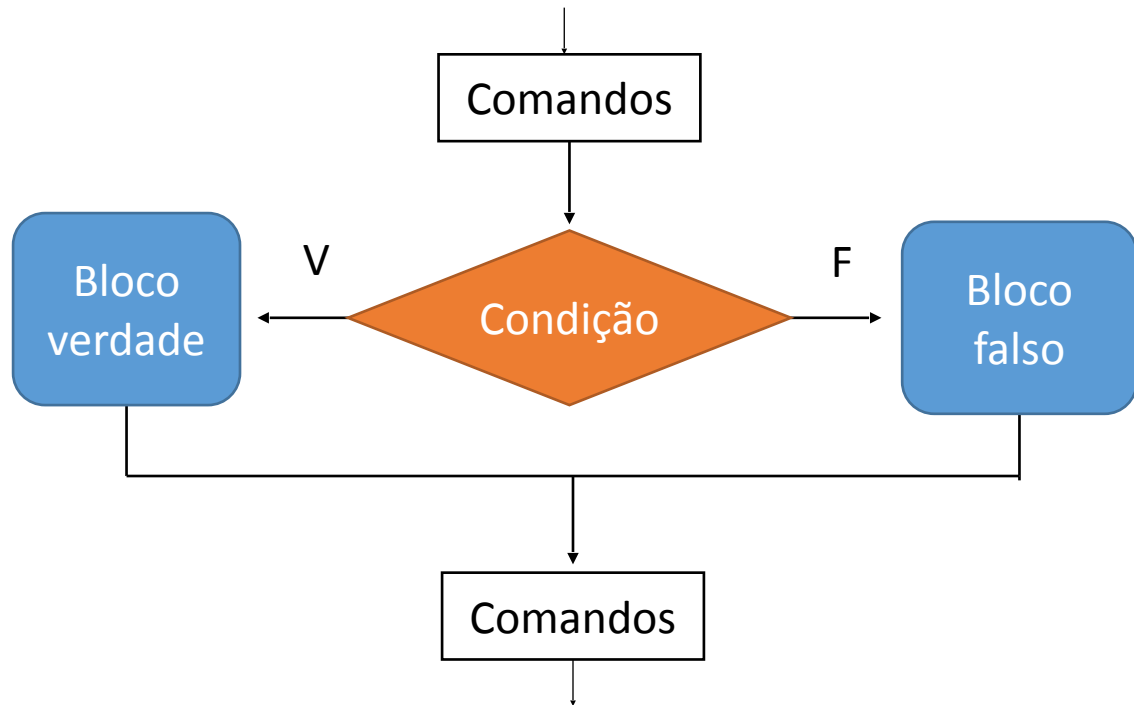
```
else
```

```
{
```

```
    //bloco falso
```

```
}
```

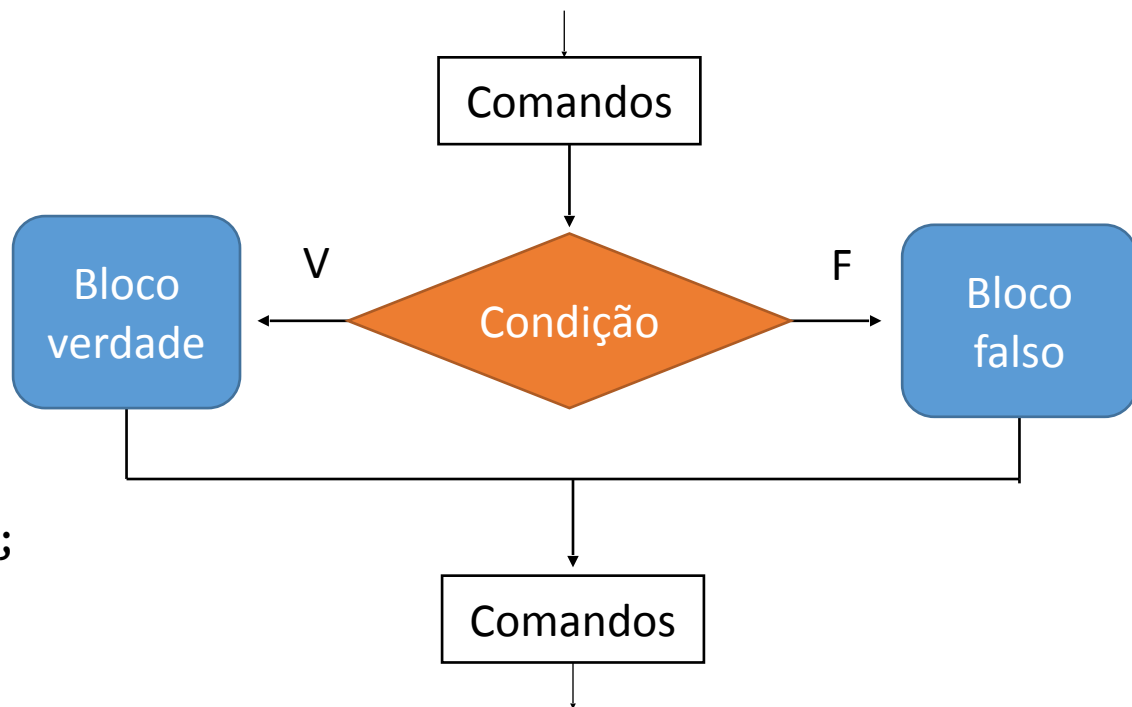
```
//comandos
```



A declaração *else* é opcional: pode-se utilizá-la para determinar um conjunto de comandos que serão executados caso a condição testada seja falsa

Declaração IF-ELSE

```
1. int main () {  
2.  
3.   int a = 8;  
4.  
5.   if(a == 5) {  
6.     printf("a vale 5");  
7.   }  
8.   else {  
9.     printf("a não vale 5");  
10.  }  
11.  a = 3;  
12.  
13.  return (0);  
14.}
```



Exemplo

```
1. int main( ) {
2.
3.     float saldo = 150.0;
4.     float saque = 50.0;
5.
6.     if(saldo - saque >= 0){
7.         saldo = saldo - saque;
8.         printf("Saque realizado com sucesso. Saldo atual = %f", saldo);
9.     }
10.    else
11.        printf("Impossivel realizar o saque.");
12.        printf("Informe um valor menor ou igual a %f", saldo);
13.
14.    return 0;
15. }
```

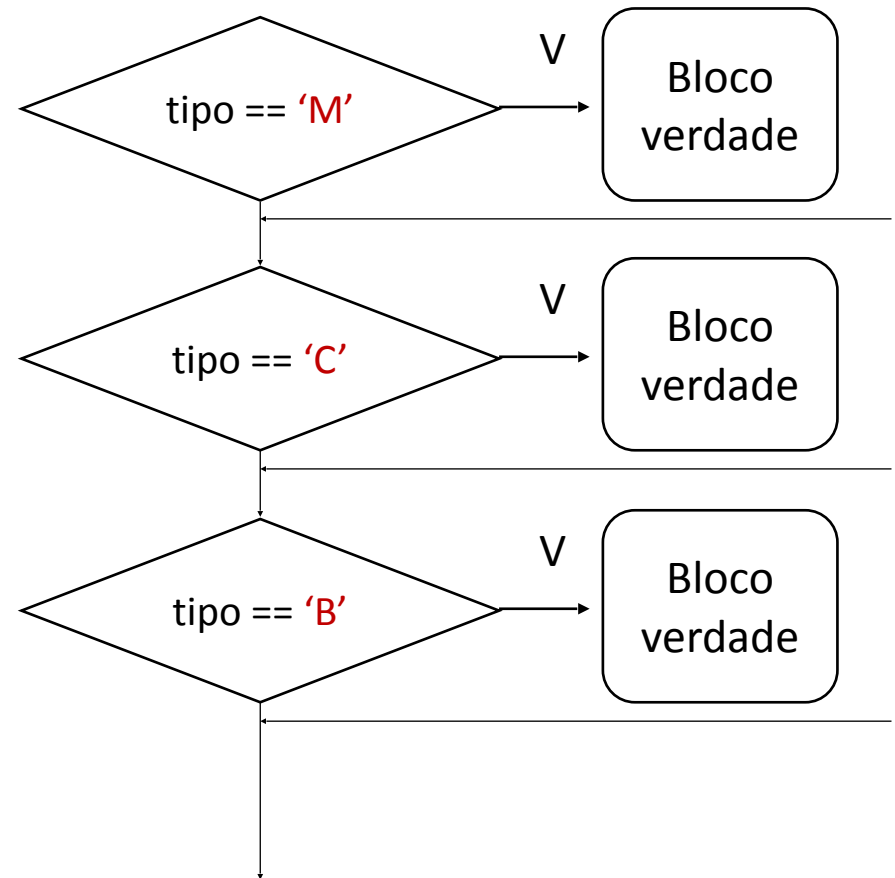
Exemplo

Onde está o erro?

```
1. int main( ) {  
2.  
3.     float saldo = 150.0;  
4.     float saque = 50.0;  
5.  
6.     if(saldo - saque >= 0){  
7.         saldo = saldo - saque;  
8.         printf("Saque realizado com sucesso. Saldo atual = %f", saldo);  
9.     }  
10.    else  
11.        printf("Impossivel realizar o saque.");  
12.        printf("Informe um valor menor ou igual a %f", saldo);  
13.  
14.    return 0;  
15. }
```

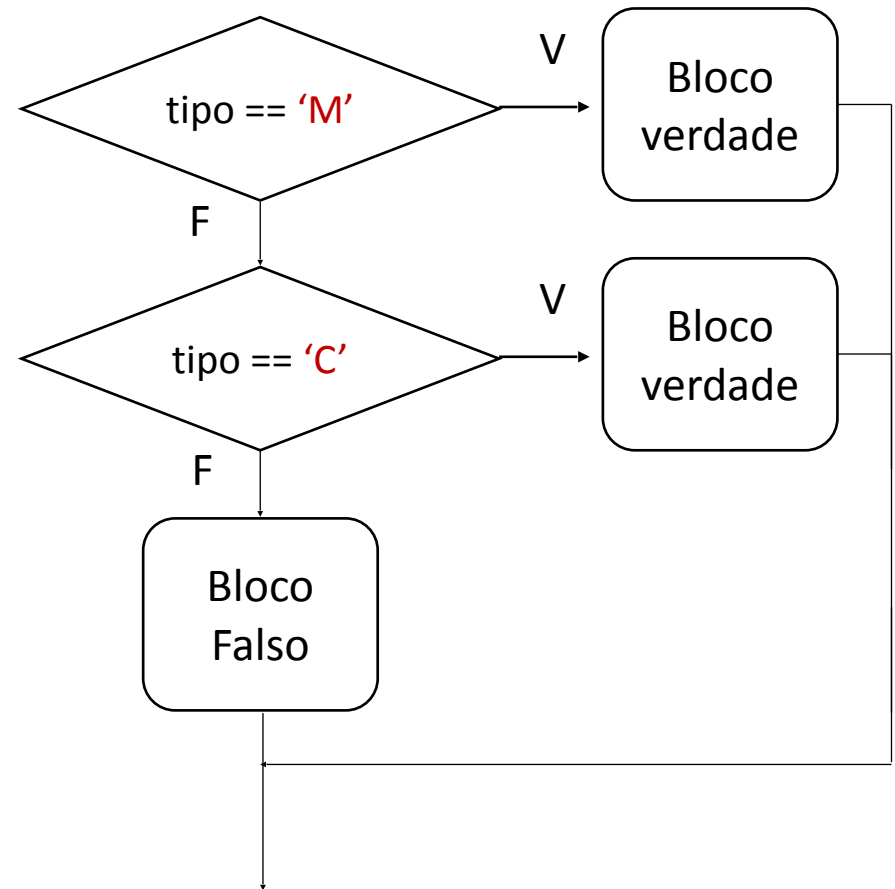
Estrutura Condicional Composta

```
1. int main() {  
2.     char tipo;  
3.     scanf("%c", &tipo);  
4.  
5.     if(tipo == 'M'){  
6.         printf("Mucarela");  
7.     }  
8.     if(tipo == 'C'){  
9.         printf("Calabresa");  
10.    }  
11.    if(tipo == 'B'){  
12.        printf("Bacon");  
13.    }  
14.    return (0);  
15. }
```



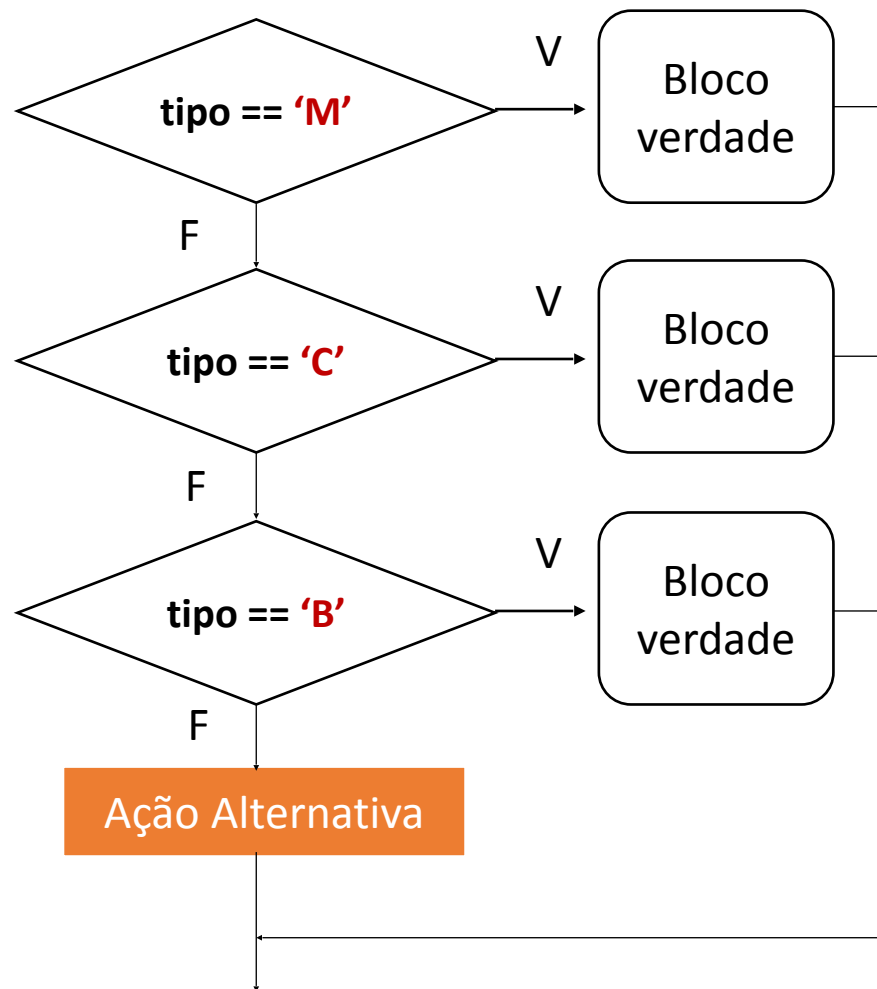
Estrutura Condicional Composta

```
1. int main() {  
2.     char tipo;  
3.     scanf("%c", &tipo);  
4.  
5.     if(tipo == 'M') {  
6.         printf("Mucarela");  
7.     }  
8.     else if(tipo == 'C') {  
9.         printf("Calabresa");  
10.    }  
11.    else {  
12.        printf("Bacon");  
13.    }  
14.    return 0;  
15. }
```



Declaração SWITCH

```
int main() {  
    char tipo;  
    scanf("%c", &tipo);  
  
    switch(tipo)  
    {  
        case 'M':  
            printf("Mucarela");  
            break;  
        case 'C':  
            printf("Calabresa");  
            break;  
        case 'B':  
            printf("Bacon");  
            break;  
        default:  
            printf("opção invalida");  
    }  
    return 0;  
}
```



Roteiro

- 1 Objetivo
- 2 Algoritmos
- 3 Estrutura Programa em C
- 4 Estruturas de Controle - Condicionais
- 5 Exercícios
- 6 Estruturas de Controle - Repetição
- 7 Exercícios
- 8 Referências

Exercícios - comandos condicionais

1.) Faça um programa que receba duas notas de um aluno. Calcule e mostre a média aritmética das notas e uma mensagem conforme a tabela a seguir.

Obs.: Não existe nota negativa.

<i>Média</i>	<i>Mensagem</i>
<i>[0, 3.9)</i>	<i>Reprovado</i>
<i>[4, 5.9)</i>	<i>Exame</i>
<i>[6, 10]</i>	<i>Aprovado</i>

Exercícios - comandos condicionais

2. Crie um programa onde:

- O usuário deve fornecer um valor inteiro;
- O programa deve responder com o nome do dia da semana correspondente;
- O programa não deve aceitar valores fora da faixa convencional, e deve apresentar uma mensagem de erro.

1 → Domingo
2 → Segunda-feira
3 → Terça-feira
4 → Quarta-feira
5 → Quinta-feira
6 → Sexta-feira
7 → Sábado

Roteiro

- 1 Objetivo
- 2 Algoritmos
- 3 Estrutura Programa em C
- 4 Estruturas de Controle - Condicionais
- 5 Exercícios
- 6 Estruturas de Controle - Repetição
- 7 Exercícios
- 8 Referências

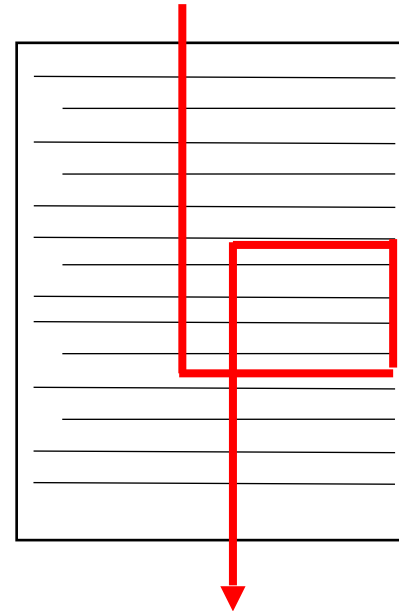
B) Estruturas de Repetição

- Estruturas de decisão em C/C++:

while enquanto ...

do-while faça ... enquanto

for para ...



Fluxo de execução
repetitivo

*Comandos são executados
de forma repetida*

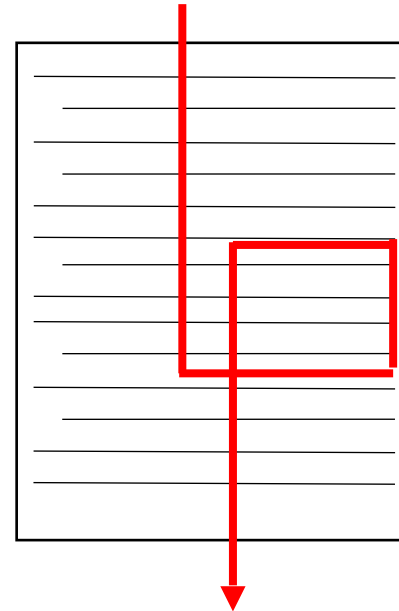
B) Estruturas de Repetição

- Estruturas de decisão em C/C++:

while enquanto ...

do-while faça ... enquanto

for para ...

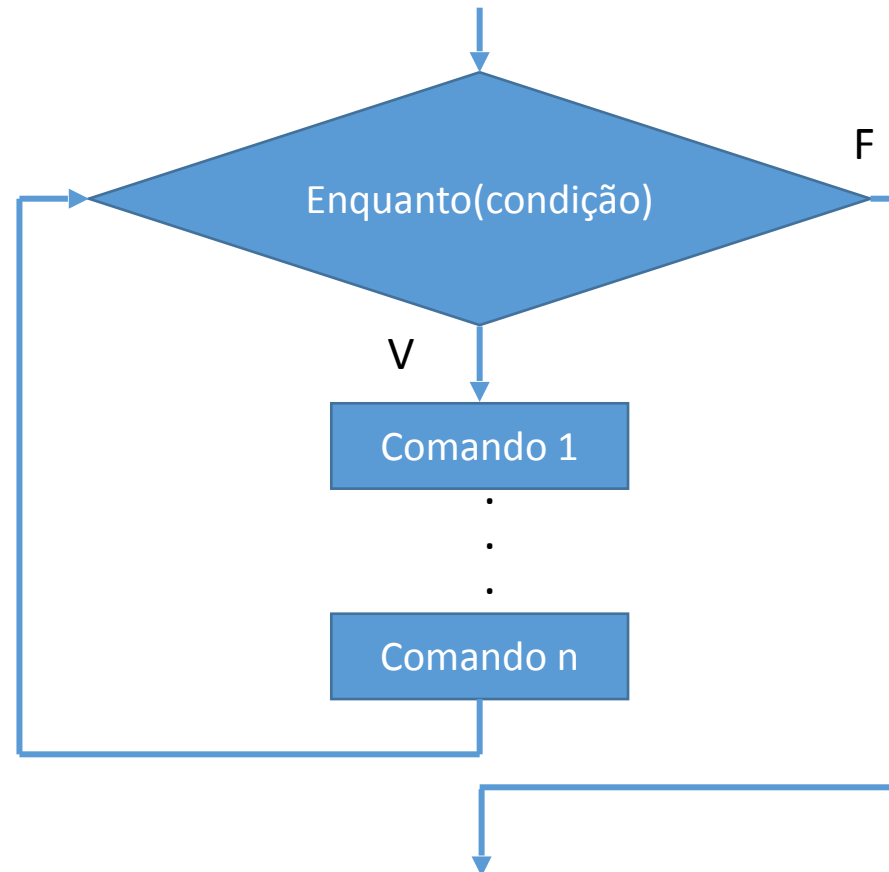


Fluxo de execução
repetitivo

*Comandos são executados
de forma repetida*

WHILE: teste no início

- **Exemplo:** calculando a tabuada de algum número
- Enquanto ... faça <comandos>



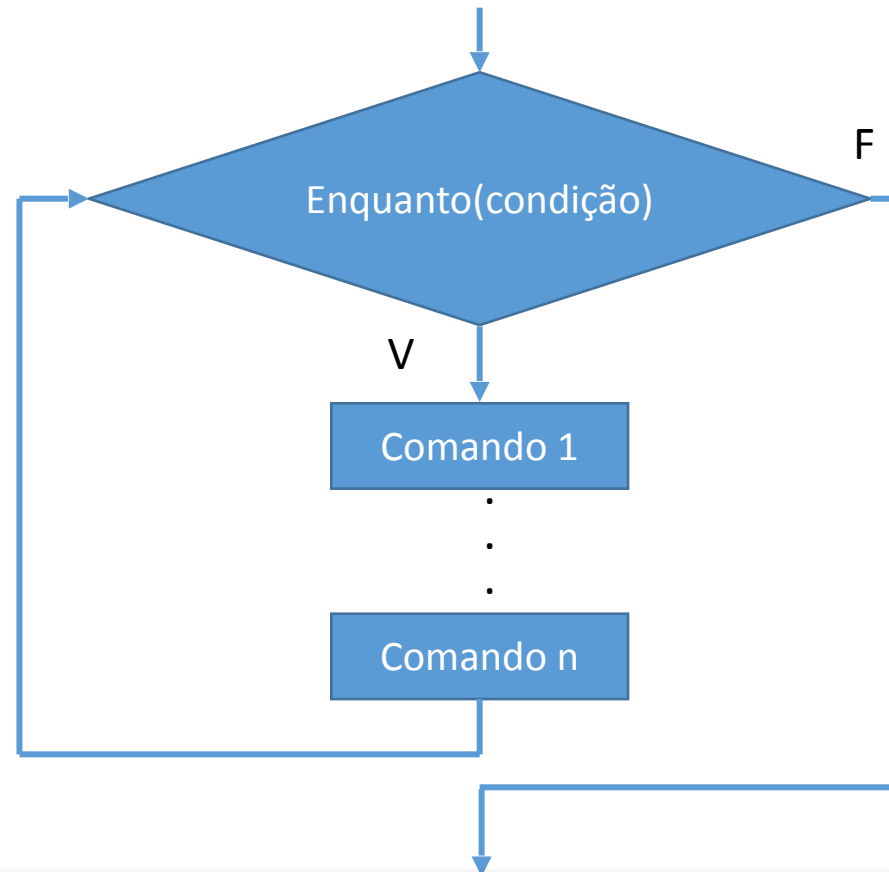
WHILE: teste no início

- **Exemplo:** calculando a tabuada de algum número
- Enquanto ... faça <comandos>

```
int multiplicador = 0, resultado, num;

printf("Tabuada de qual numero: ");
scanf("%d", &num);

while(multiplicador <= 10)
{
    resultado = num * multiplicador;
    printf("%d", resultado);
    multiplicador = multiplicador + 1;
}
```



Em alguns programas os comandos dentro da instrução *while* **podem não ser executados**, uma vez que a condição é verificada antes que eles possam ser executados.

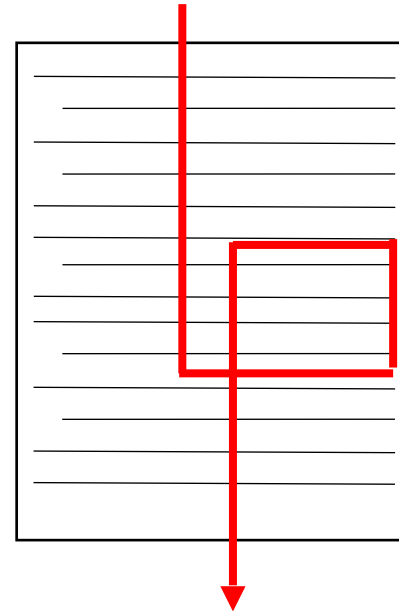
B) Estruturas de Repetição

- Estruturas de decisão em C/C++:

while enquanto ...

do-while faça ... enquanto

for para ...

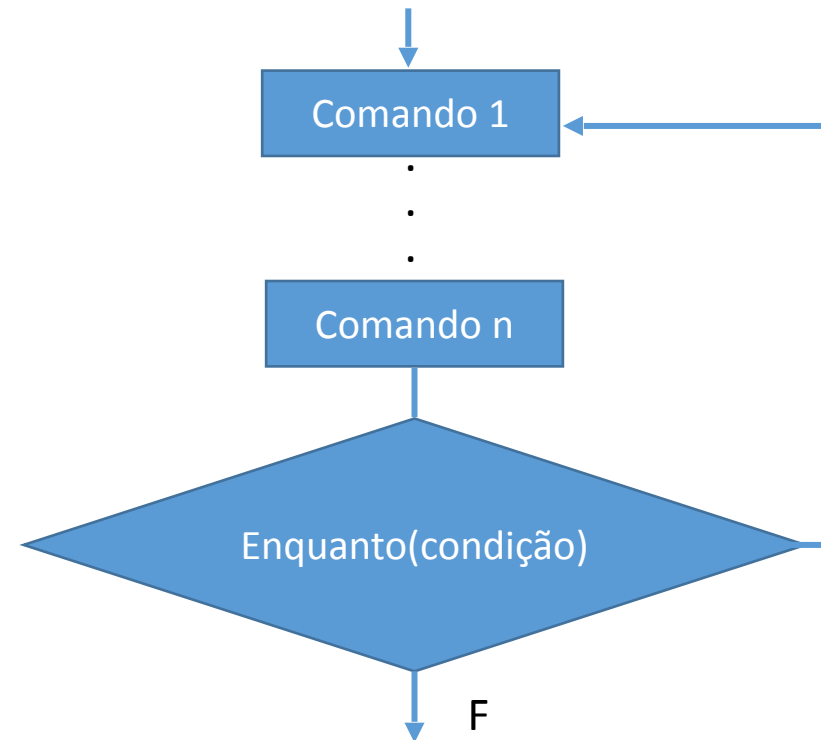


Fluxo de execução
repetitivo

*Comandos são executados
de forma repetida*

DO: teste no final

- **Exemplo:** verificar se o usuário digitou um valor positivo
- Faça... Enquanto

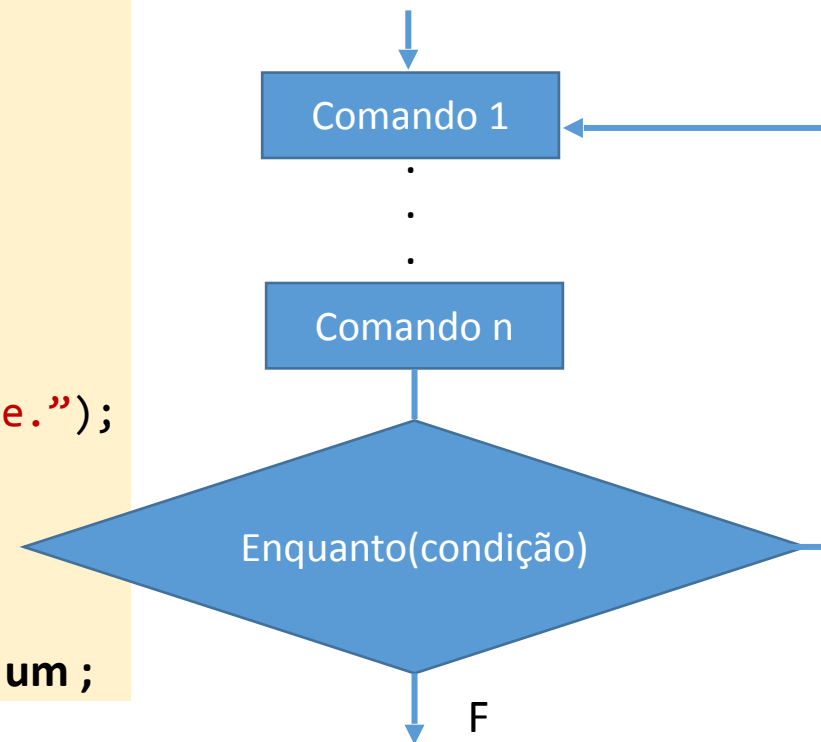


DO: teste no final

- **Exemplo:** verificar se o usuário digitou um valor positivo
- Faça... Enquanto

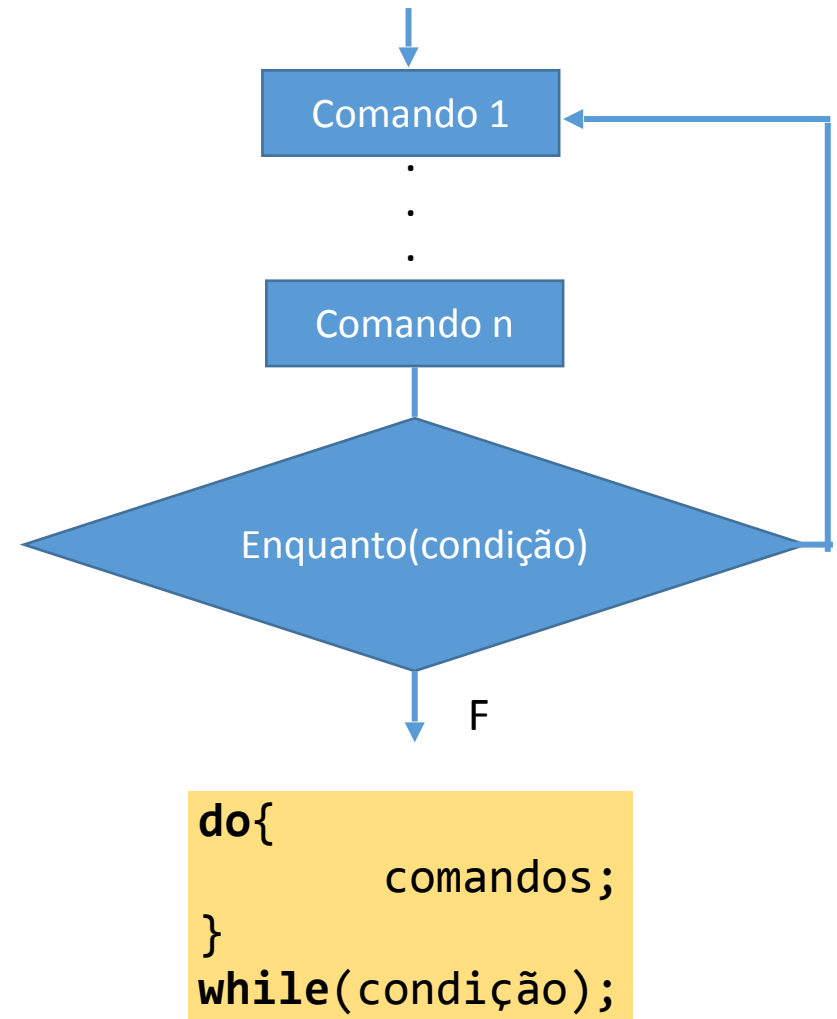
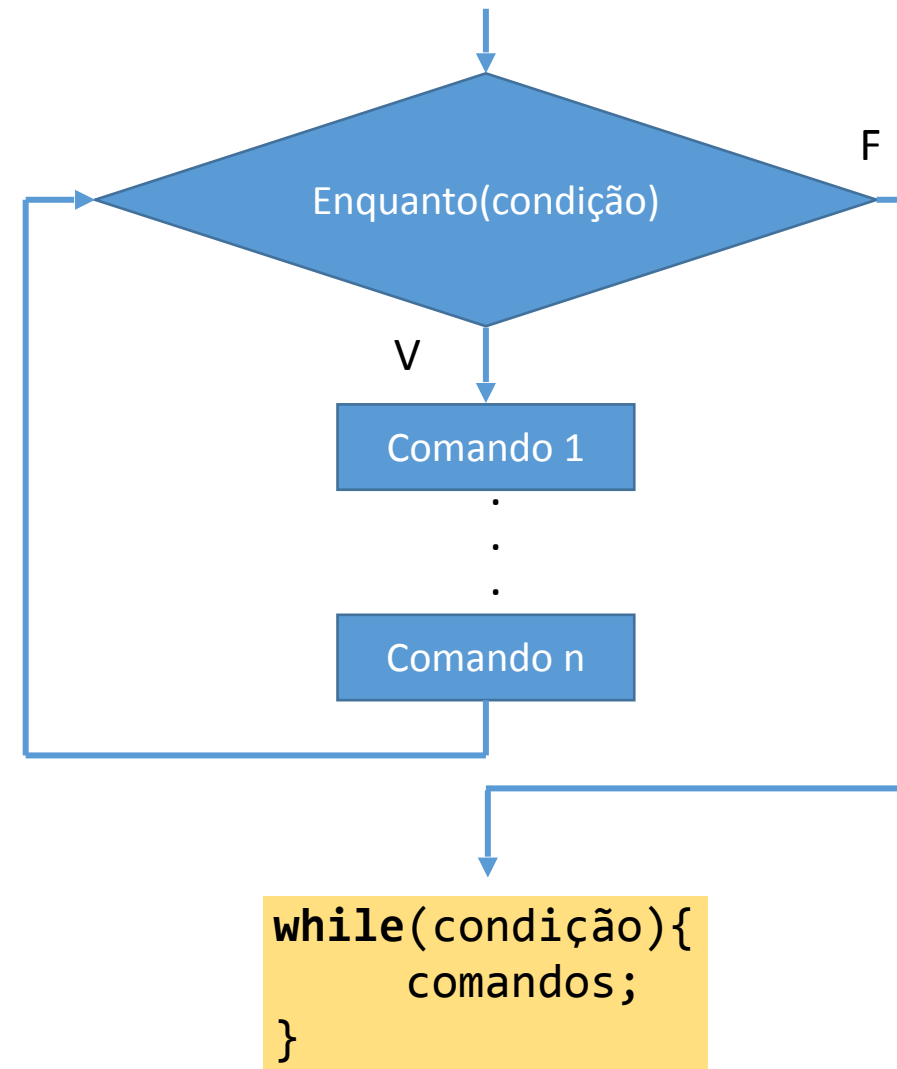
```
int num;  
  
do  
{  
    printf("Forneca um inteiro positivo: ");  
    scanf("%d", &num);  
  
    if(num < 0)  
        printf("Valor invalido. Tente novamente.");  
} while(num < 0);
```

Esse while tem um ;



Os comandos dentro da instrução **do** **são executadas** pelo menos uma vez, uma vez que o teste só é realizado no final.

Comparativo



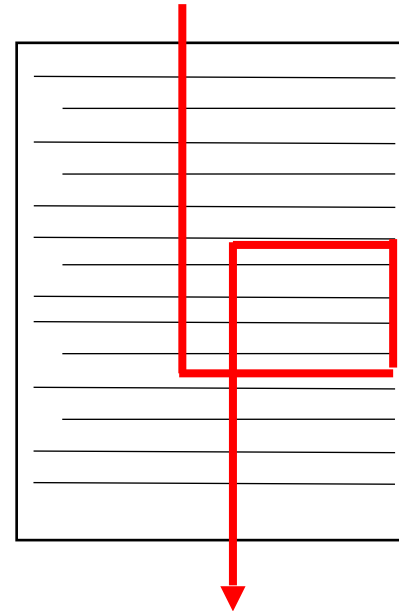
B) Estruturas de Repetição

- Estruturas de decisão em C/C++:

while enquanto ...

do-while faça ... enquanto

for para ...



Fluxo de execução
repetitivo

*Comandos são executados
de forma repetida*

FOR: número fixo de repetições

Calculando a Tabuada com o **for**:

```
int multiplicador, resultado, num;

printf("Tabuada de qual numero: ");
scanf("%d", &num);

for(multiplicador=0; multiplicador <= 10; multiplicador++){
    resultado = multiplicador * num;
    printf("%d", resultado);
}
```


FOR: número fixo de repetições

Calculando a Tabuada com o **for**:

```
int multiplicador, resultado, num;
```

```
printf("Tabuada de qual numero: ");  
scanf("%d", &num);
```

Inicialização	Condição	Incremento/ Decremento
for(multiplicador=0;	multiplicador <= 10;	multiplicador++)
{		
resultado = multiplicador * num;		
printf("%d", resultado);		
}		

FOR: número fixo de repetições

Calculando a Tabuada com o **for**:

```
int multiplicador, resultado, num;
```

```
printf("Tabuada de qual numero: ");  
scanf("%d", &num);
```

```
                Inicialização                Condição                Incremento/  
                                Decremento  
for(multiplicador=0; multiplicador <= 10; multiplicador++){  
    resultado = multiplicador * num;  
    printf("%d", resultado);  
}
```

```
for(inicialização; condição; incremento {  
    comandos;  
}
```

incremento e decremento

`i++`
`++i` \longrightarrow `i = i + 1`

```
i = 5;  
a = i++;
```

`a = 5, i = 6`

```
i = 5;  
a = ++i;
```

`a = 6, i = 6`

`i--`
`--i` \longrightarrow `i = i - 1`

```
i = 5;  
a = i--;
```

`a = 5, i = 4`

```
i = 5;  
a = --i;
```

`a = 4, i = 4`

Tomar cuidado com
atalhos de incremento
em operações de
atribuição

Comandos BREAK e CONTINUE

- Break
 - Utilizado para sair abruptamente da estrutura de controle;
- Continue
 - Ignora o resto do bloco de dados de uma iteração, mas continua executando a estrutura de controle;

```
for(int i=1; i <= 100; i++)  
{  
    if(i % 10 == 0)  
        continue;  
    else  
        printf("%d", i);  
}
```

```
for(int i=1; i <= 100; i++)  
{  
    if(i % 10 == 0)  
        break;  
    else  
        printf("%d", i);  
}
```

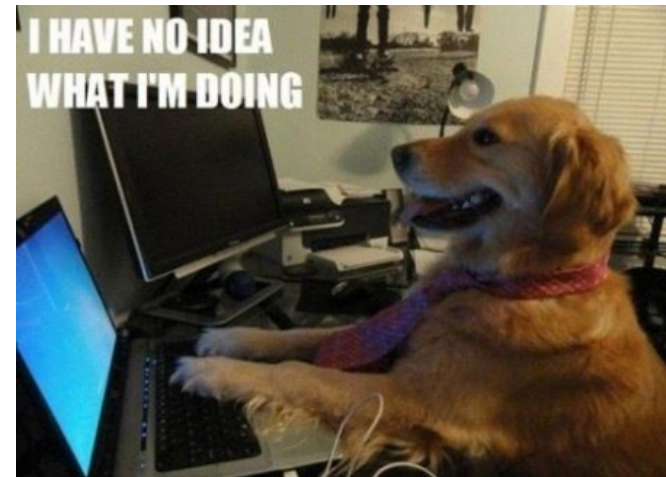
1 2 3 4 5 6 7 8 9 11 12 ... 19 21 ... 29 31 99

1 2 3 4 5 6 7 8 9

Loop Infinito

- Podem ocorrer por erros durante a programação:

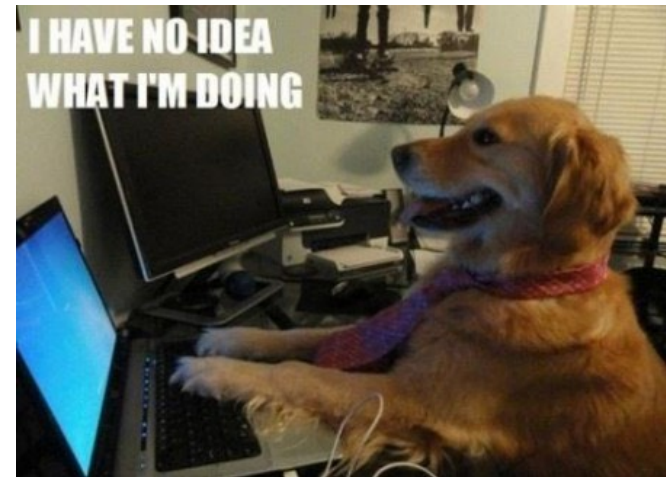
```
int multiplicador = 0, resultado;  
printf("Tabuada de qual numero: ");  
scanf("%d", &num);  
  
while(multiplicador <= 10)  
{  
    resultado = num * multiplicador;  
    printf("%d", resultado);  
}
```



Loop Infinito

- Podem ocorrer por erros durante a programação:

```
int multiplicador = 0, resultado;  
  
printf("Tabuada de qual numero: ");  
scanf("%d", &num);  
  
while(multiplicador <= 10)  
{  
    resultado = num * multiplicador;  
    printf("%d", resultado);  
}
```



O que está errado?

Qual o valor de multiplicador ao longo das iterações?

Quando o loop irá acabar?

Loop Infinito

- Podem ocorrer de propósito:

```
for(;;)
{
    printf("Digite um numero inteiro: ");
    scanf("%d", &n);
    if (n == 7)
    {
        printf("Saindo do loop...\n");
        break; //força a saída do loop
    }
    printf("Numero: %d\n", n);
}
printf("Fim de programa");
```

```
while(true)
{
    printf("Digite um numero inteiro: ");
    scanf("%d", &n);
    if (n == 7)
    {
        printf("Saindo do loop...\n");
        break; //força a saída do loop
    }
    printf("Numero: %d\n", n);
}
printf("Fim de programa");
```

Roteiro

- 1 Objetivo
- 2 Algoritmos
- 3 Estrutura Programa em C
- 4 Estruturas de Controle - Condicionais
- 5 Exercícios
- 6 Estruturas de Controle - Repetição
- 7 Exercícios
- 8 Referências

Exercícios - comandos repetição

3. Faça um Programa que leia 10 valores inteiros e escreva no final a soma dos valores lidos;
4. Faça um programa que receba a idade de dez pessoas, calcule e mostre a quantidade de pessoas com idade maior ou igual a 18 anos.
5. Faça um programa que receba dez números e mostre a quantidade de números entre 30 e 90.
6. Faça um programa para calcular $n!$
Lembrando que: $n! = n * (n - 1) * (n - 2) * \dots * 1$
 $0! = 1$, por definição
7. Calcule o resultado da série:
• $S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \dots + \frac{99}{50}$

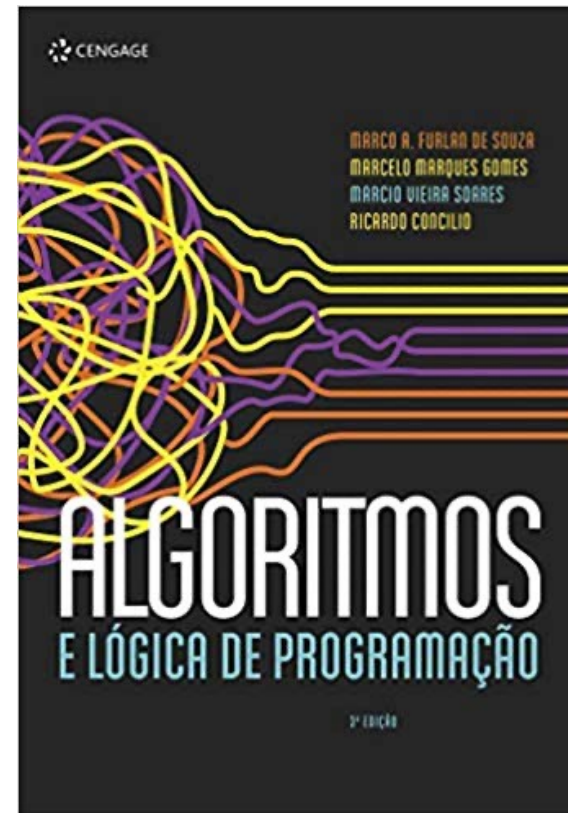
Roteiro

- 1 Objetivo
- 2 Algoritmos
- 3 Estrutura Programa em C
- 4 Estruturas de Controle - Condicionais
- 5 Exercícios
- 6 Estruturas de Controle - Repetição
- 7 Exercícios
- 8 Referências

Referências sugeridas



[MANZANO & OLIVEIRA, 2019]



[SOUZA et al, 2019]

Referências sugeridas



[SCHILDT, 2006]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br