

<http://www.nickgentry.com/>

Algoritmos e Programação de Computadores

Disciplina 113476

Prof. Alexandre Zaghetto
<http://alexandre.zaghetto.com>
zaghetto@unb.com

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação



O presente conjunto de *slides* não pode ser reutilizado ou republicado sem a permissão do instrutor.

Módulo 05

Algoritmos com Alternativas

1. Algoritmos com Alternativas

- Na vida real tomamos decisões a partir da análise de algumas condições.

✓ Exemplo: Se eu tiver pelo menos R\$ 50,00 (na conta corrente ou poupança), então irei ao cinema hoje à noite.

Trata-se de uma expressão lógica, uma vez que a pergunta: “Tenho R\$ 50,00 sobrando?” Pode ser respondida com um “Sim” ou com um “Não”.

➤ Analogamente, em linguagens de programação um determinado bloco básico de comandos **será executado ou não**, dependendo da avaliação de expressões lógico-aritmético-relacionais. A isso chamados de **algoritmo com alternativa**.



2. Relembrando o Tipo Lógico

- **Portugol:** conjunto de valores FALSO ou VERDADEIRO em proposições lógicas.

- **Linguagem C:**

- O tipo lógico não é um tipo básico em C.

- ✓ **0** está para **FALSO**.

- ✓ Qualquer outro valor está para **VERDADEIRO**.

3. Operadores Lógicos

NOT

A	X
0	1
1	0

$\neg A$

AND

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

$A \&\& B$

OR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

$A || B$

3. Operadores Lógicos

Operador	Ação
!	NAO (Negação)
&&	E (Conjunção)
	OU (Disjunção inclusiva)

! 0 && 0 || 0
! (0 && 0) || 0

ATENÇÃO: O operador “!” tem mais alta precedência que qualquer operador aritmético. Já os operadores “&&” e “||” têm mais baixa precedência.

3. Operadores Relacionais

Operador	Ação
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual
==	Igual
!=	Diferente

$A > B+8$

ATENÇÃO: Esses operadores são menores em precedência que qualquer operador aritmético.

4. Precedência entre operadores lógicos e relacionais

Operador	Precedência
!	Mais alta
> >= < <=	.
== !=	.
&&	.
	Mais baixa

10 > 5 && !(10 < 9) || 3 <=4

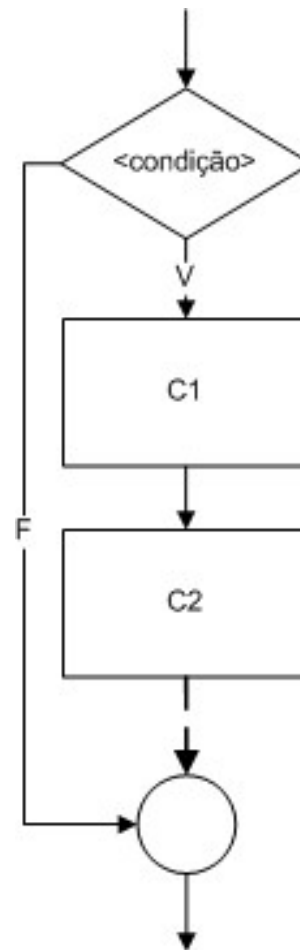
5. Alternativa Simples

- Forma geral em Portugal:

```
se <condição> entao  
    <ações (uma ou mais) a serem realizadas se a condição for verdadeira>  
fimse
```

5. Alternativa Simples

- Fluxograma:



5. Alternativa Simples

Exemplo 1: Escreva um algoritmo para calcular a área de um círculo, fornecido o valor do raio, **que deve ser positivo**. Em seguida o algoritmo deve imprimir o valor da área na tela do computador.



5. Alternativa Simples

```
algoritmo "Area1"  
var  
real: Area, Raio  
inicio  
    escreva ("Entre com o raio do círculo:")  
    leia (Raio)  
    se Raio > 0 entao  
        Area <- 3.14*pow(Raio,2)  
        escreva ("A área é ", Area)  
    fimse  
    se Raio <= 0 entao  
        escreva ("Raio não pode ser nulo ou negativo!")  
    fimse  
fimalgoritmo
```



5. Alternativa Simples

```
algoritmo "Area1"  
var  
real: Area, Raio  
inicio  
    escreva ("Entre com o raio do círculo")  
    leia (Raio)  
    se Raio > 0 entao  
        Area <- 3.14*pow(Raio,2)  
        escreva ("A área é ", Area)  
    fimse  
    se Raio <= 0 entao  
        escreva ("Raio não pode ser nulo ou negativo!")  
    fimse  
fimalgoritmo
```

5. Alternativa Simples

algoritmo "Areal"

var

real: Area, Raio:

inicio

escreva ("Entre com o raio do círculo")

leia (Raio)

se Raio > 0 **entao**

 Area <- 3.14*pow(Raio, 2)

escreva ("A área é ", Area)

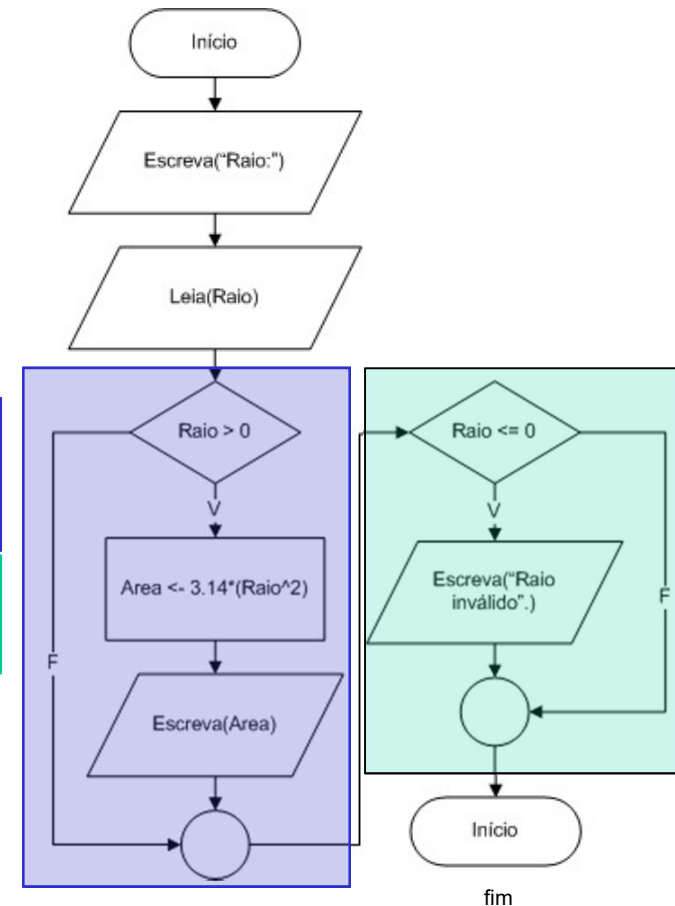
fimse

se Raio <= 0 **entao**

escreva ("Raio não pode ser nulo ou negativo!")

fimse

fimalgoritmo



5. Alternativa Simples

- Forma geral em C:

```
if (<expressão>) {  
    <comandos>  
}
```


5. Alternativa Simples

Exemplo 2: Escreva um algoritmo que lê o valor da mercadoria, calcula e mostra o valor final a ser pago incluindo, se for o caso, a taxa de embrulho para presente que é de R\$ 1,50.

5. Alternativa Simples

algoritmo "caixa"

var

real: valor;

caracter: presente;

inicio

escreva ("Informe o valor da mercadoria:");

leia (valor);

escreva ("Deverá ser embrulhada para presente?");

leia (presente);

se presente == 'S' entao

 valor = valor + 1.50;

fimse

escreva ("Total a pagar: ", valor);

fimalgoritmo



5. Alternativa Simples

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

int main()
{
    float valor;
    char presente;

    printf("Informe o valor da mercadoria:");
    scanf("%f", &valor);
    getchar();
    printf("Deverá ser embrulhada para presente?");
    scanf("%c", &presente);
```



5. Alternativa Simples

```
if (presente == 'S') {  
    valor = valor + 1.50;  
}  
  
printf("Total a pagar: %f", valor);  
  
return 0;  
}
```

6. Funções Úteis

- As funções ***getchar()*** e ***putchar()***:

✓ Um dos inconvenientes de se utilizar a função ***scanf()*** para se ler um dado do tipo ***char*** (%c) é que o *enter* (\n) que fica armazenado no *buffer* de entrada do teclado é levando em consideração pelo programa.

✓ Para se evitar esse inconveniente, utiliza-se a função ***getchar()*** logo após a leitura (***scanf***) anterior:

```
printf("Informe o valor da mercadoria:");  
scanf("%f", &valor);  
getchar();           → captura o enter  
printf("Deverá ser embrulhada para presente?");  
scanf("%c", &presente);
```



6. Funções Úteis

- As funções ***getchar()*** e ***putchar()***:

✓ Na verdade, a função ***getchar()*** serve para ler uma entrada do tipo ***char*** e pode, nesse exemplo, ser utilizada no lugar do ***scanf()***;

```
printf("Informe o valor da mercadoria:");
```

```
scanf("%f", &valor);
```

```
getchar();
```

 captura o *enter*

```
printf("Deverá ser embrulhada para presente?");
```

```
presente = getchar();
```



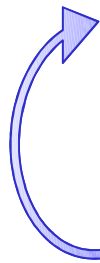
```
scanf("%c", &presente);
```



6. Funções Úteis

- As funções ***getchar()*** e ***putchar()***:

✓ Já a função ***putchar()*** serve para mostrar o conteúdo de uma variável do tipo ***char*** na tela do computador e pode, nesse exemplo, ser utilizada no lugar do ***printf()***;



```
putchar(letra);
```

```
printf("%c", letra);
```



6. Funções Úteis

- A função ***getch()***:

✓ Já a função ***getch()***, que requer a inclusão do arquivo `conio.h` (`#include <conio.h>`), dispensa a necessidade de se utilizar o *enter* após a entrada do caractere via teclado.

```
printf("Informe o valor da mercadoria:");  
scanf("%f", &valor);  
printf("Deverá ser embrulhada para presente?");  
presente = getch();
```

✓ Uma outra opção é a função ***getche()***, que além de ler o caractere dispensando o *enter*, faz um eco do caractere digitado na tela do computador.

6. Funções Úteis

• Entrada e saída de sequencias de caracteres (strings):

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

```
int main()
{
```

```
    char nome[50]; ← string
```

```
    printf("Digite seu nome: ");
```

```
    scanf("%[^\n]", nome);
```

```
    printf("O nome digitado foi: %s \n", nome);
```

```
    return 0;
```

```
}
```

6. Funções Úteis

- As funções *gets()* e *puts()*:

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

int main()
{
    char nome[50];

    printf("Digite seu nome: ");
    gets(nome);
    printf("O nome digitado foi: ");
    puts(nome);

    return 0;
}
```

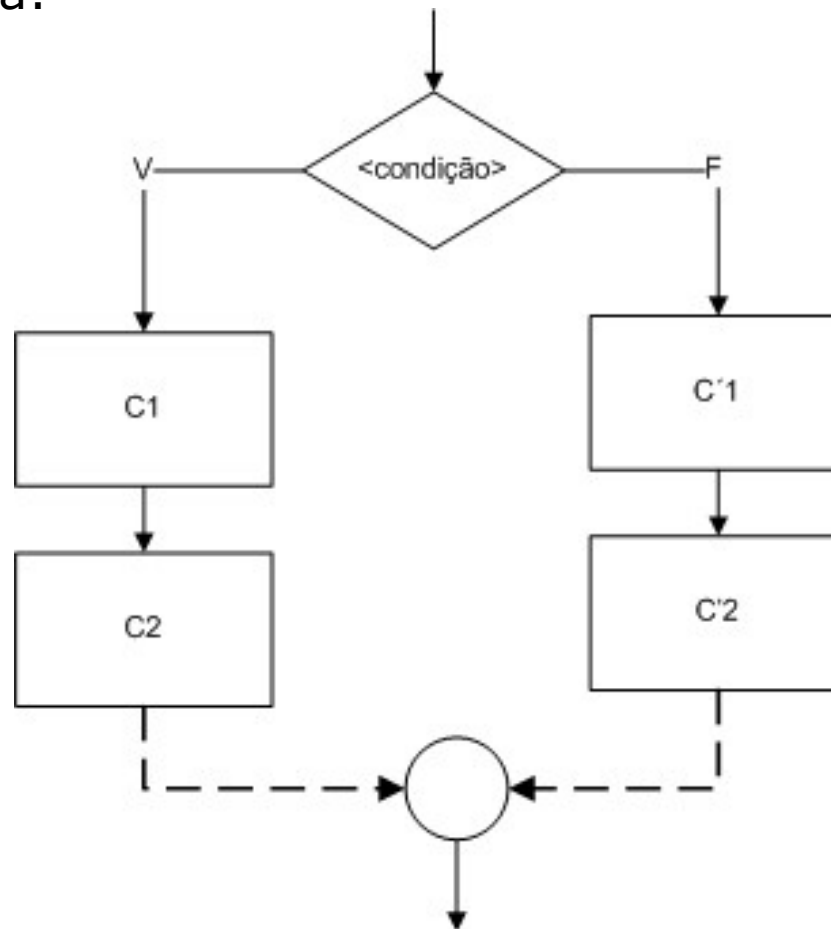
7. Alternativa Composta

- Forma geral em Portugol:

```
se <condição> entao  
    <ações (uma ou mais) a serem realizadas se a condição for verdadeira>  
senao  
    <ações (uma ou mais) a serem realizadas se a condição for falsa>  
fimse
```

7. Alternativa Composta

- Fluxograma:



7. Alternativa Composta

```
algoritmo "Area2"  
var  
real: Area, Raio:  
inicio  
    escreva ("Entre com o raio do círculo")  
    leia (Raio)  
    se Raio > 0 entao  
        Area <- 3.14*pow(Raio,2)  
        escreva ("A área é ", Area)  
    senao  
        escreva ("Raio não pode ser nulo ou negativo!")  
    fimse  
fimalgoritmo
```



7. Alternativa Composta

```
algoritmo "Area2"  
var  
real: Area, Raio:  
inicio  
    escreva ("Entre com o raio do círculo")  
    leia (Raio)  
    se Raio > 0 entao  
        Area <- 3.14*pow(Raio,2)  
        escreva ("A área é ", Area)  
    senao  
        escreva ("Raio não pode ser nulo ou negativo!")  
    fimse  
fimalgoritmo
```



7. Alternativa Composta

algoritmo "Area2"

var

real: Area, Raio:

inicio

escreva ("Entre com o raio do círculo")

leia (Raio)

se Raio > 0 entao

Area <- 3.14*pow(Raio,2)

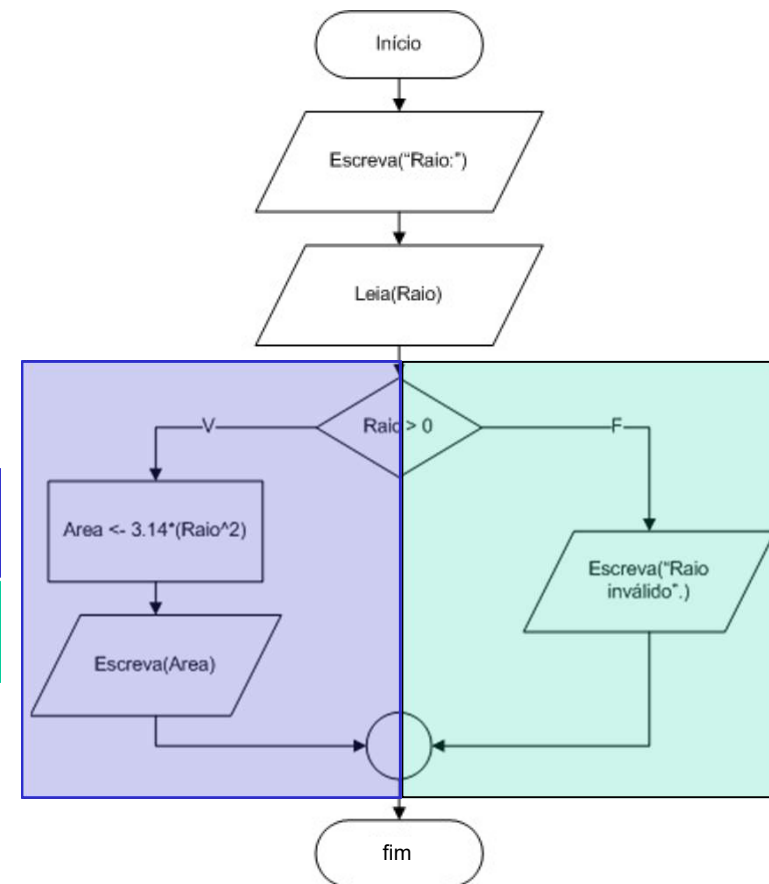
escreva ("A área é ", Area)

senao

escreva ("Raio não pode ser nulo ou negativo!")

fimse

fimalgoritmo



7. Alternativa Composta

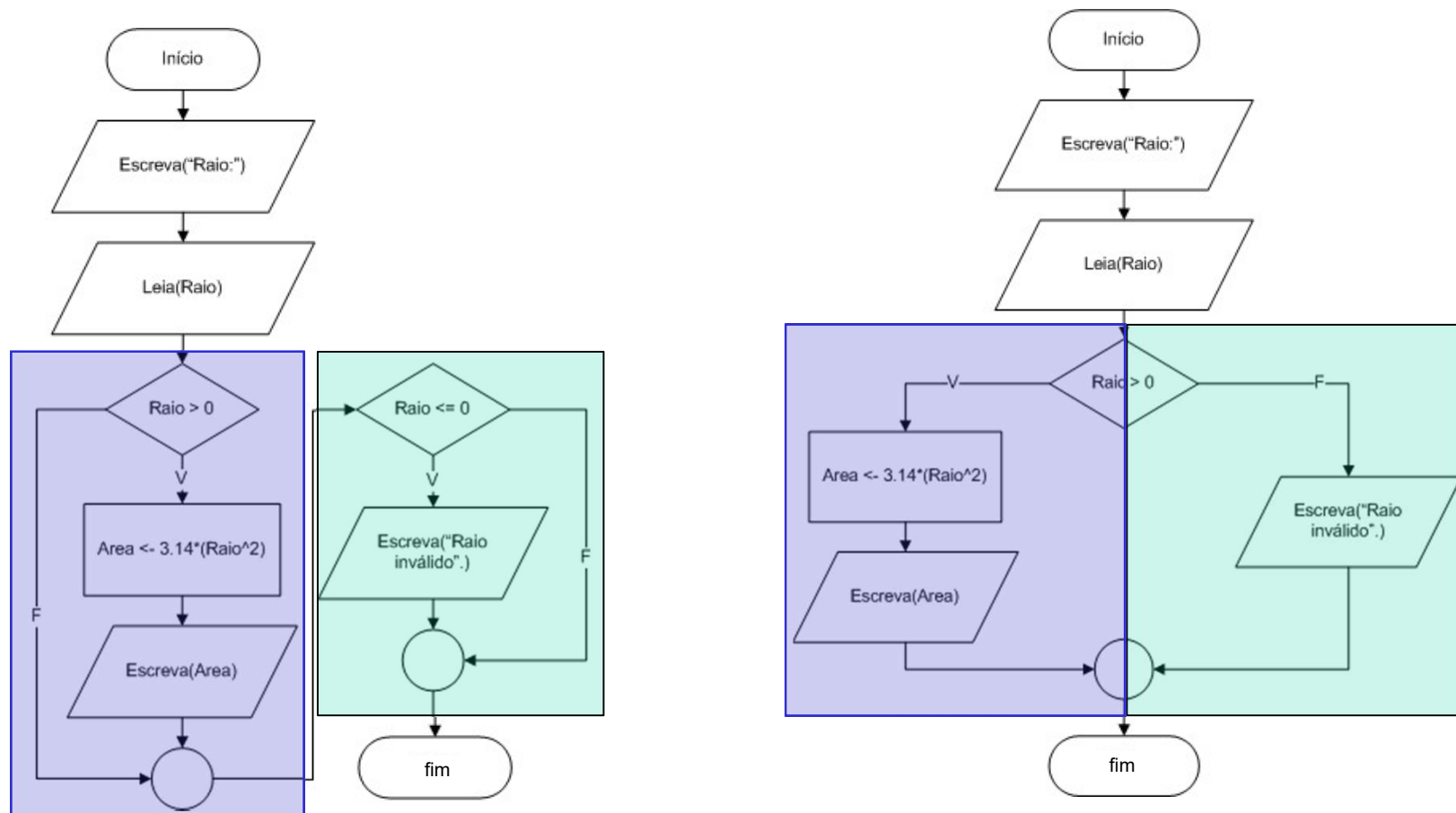
- Alternativa simples x composta:

```
algoritmo "Areal"  
var  
real: Area, Raio:  
inicio  
    escreva ("Entre com o raio do círculo")  
    leia (Raio)  
    se Raio > 0 entao  
        Area <- 3.14*pow(Raio, 2)  
        escreva ("A área é ", Area)  
    fimse  
    se Raio <= 0 entao  
        escreva ("Raio inválido!")  
    fimse  
fimalgoritmo
```

```
algoritmo "Area2"  
var  
real: Area, Raio:  
inicio  
    escreva ("Entre com o raio do círculo")  
    leia (Raio)  
    se Raio > 0 entao  
        Area <- 3.14*pow(Raio,2)  
        escreva ("A área é ", Area)  
    senao  
        escreva ("Raio inválido!")  
    fimse  
fimalgoritmo
```


7. Alternativa Composta

- Alternativa simples x composta:



7. Alternativa Composta

- Forma geral em C:

```
if (<expressão>) {  
    <comandos>  
} else {  
    <comandos>  
}
```



7. Alternativa Composta

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

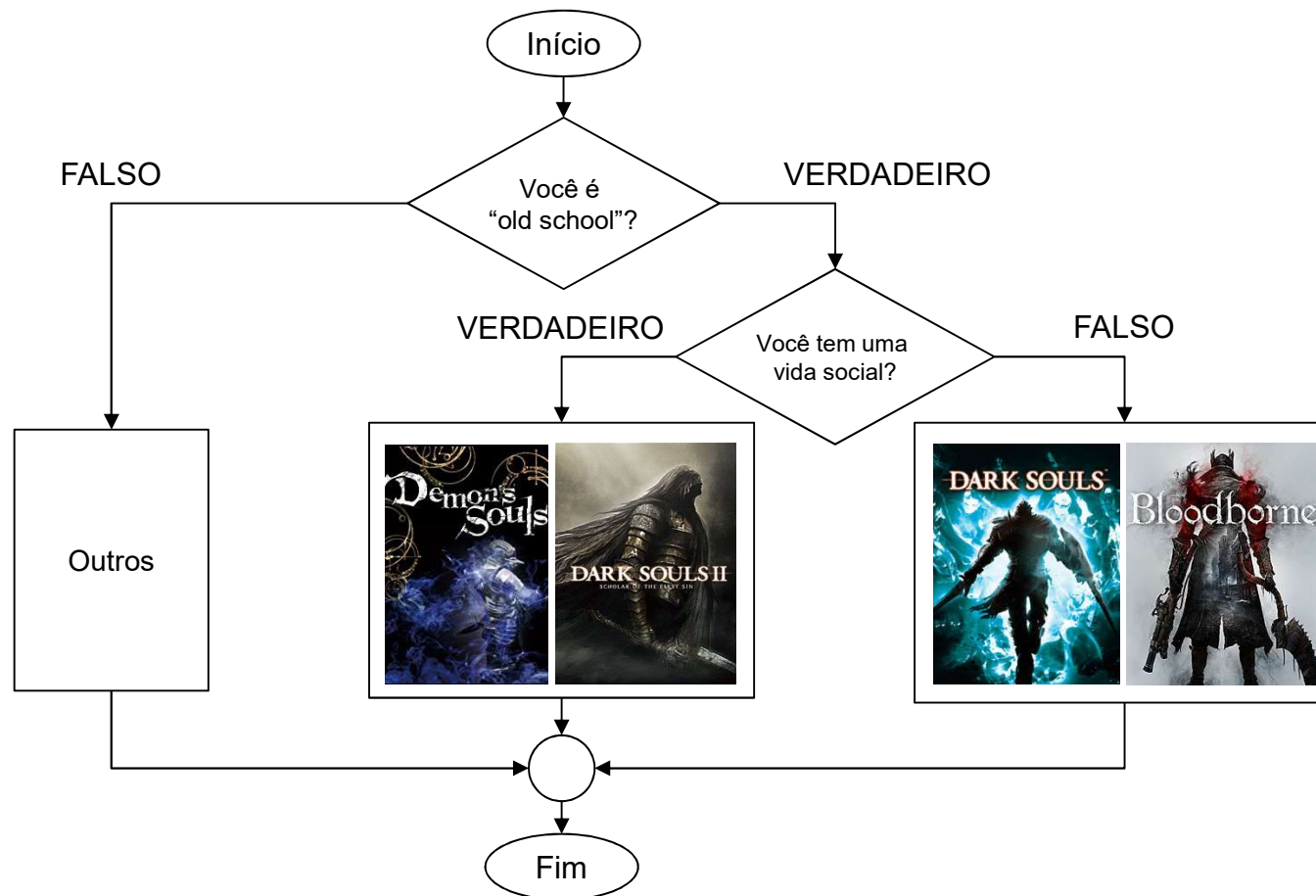
int main()
{
    float Area, Raio;

    printf("Informe o valor do raio:");
    scanf("%f", &Raio);

    if (Raio > 0) {
        Area = 3.14*pow(Raio,2);
        printf("A area eh: %.2f \n",Area);
    } else {
        printf("Raio não pode ser nulo ou negativo! \n");
    }

    return 0;
}
```

8. Alternativas Aninhadas



8. Alternativas Aninhadas

- Forma geral em Portugal:

```
se <condição> entao
    <ações (uma ou mais) a serem realizadas se a condição for verdadeira>
senao se <condição> entao
    <ações (uma ou mais) a serem realizadas se a condição for verdadeira>
senao se <condição> entao
    <ações (uma ou mais) a serem realizadas se a condição for verdadeira>
senao
    <ações (uma ou mais) a serem realizadas se a condição for falsa>
fimse
```

8. Alternativas Aninhadas

- Forma geral em C:

```
if (<expressão>) {  
|     <comandos>  
} else if (<expressão>) {  
|     <comandos>  
} else if (<expressão>) {  
|     <comandos>  
} else if (<expressão>) {  
|     <comandos>  
} else {  
|     <comandos>  
}
```



8. Alternativas Aninhadas

- Forma geral em C:

```
if (<expressão>) {  
    <comandos>  
    if (<expressão>) {  
        <comandos>  
    } else {  
        <comandos>  
    }  
    <comandos>  
} else {  
    <comandos>  
    if (<expressão>) {  
        <comandos>  
    } else {  
        <comandos>  
    }  
    <comandos>  
}
```

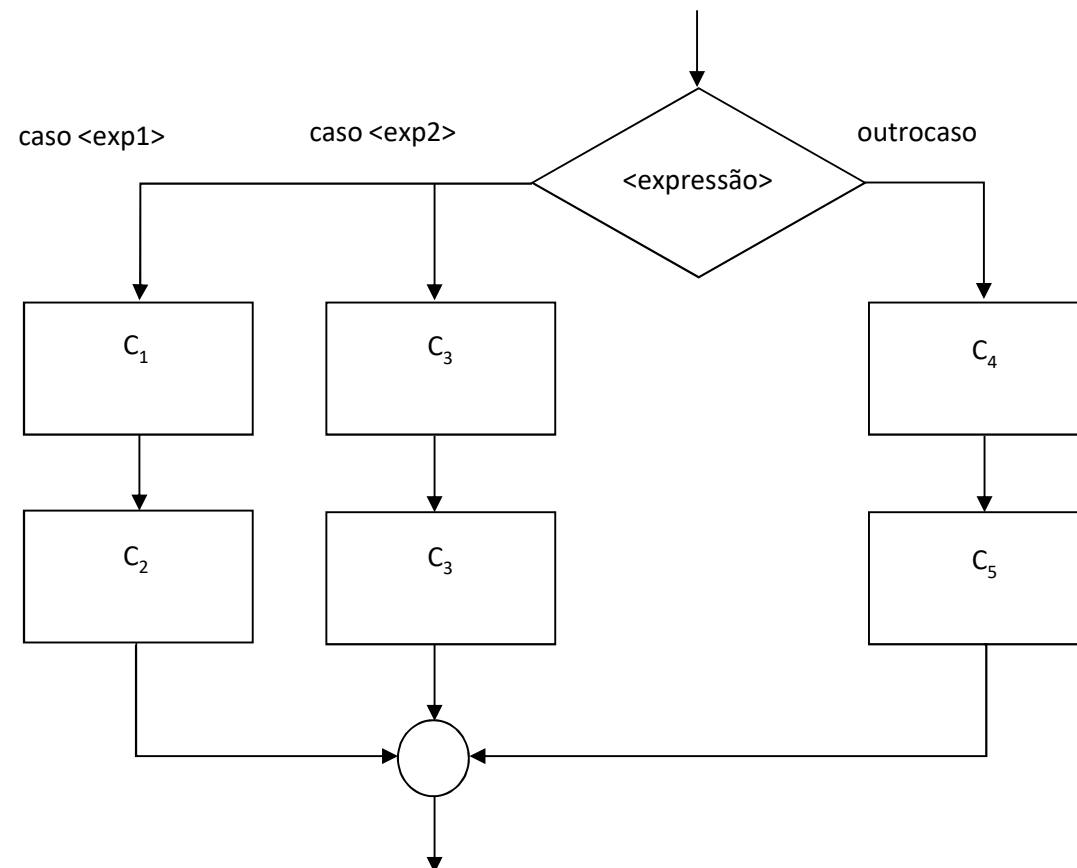
9. Alternativas de Múltiplas Escolhas

- Forma geral em Portugol:

```
escolha < expressão-de-seleção >  
  caso < exp 1 >  
    < lista-de-comandos-1 >  
  caso < exp 2 >  
    < lista-de-comandos-2 >  
  outrocaso  
    < lista-de-comandos-3 >  
fimescolha
```


9. Alternativas de Múltiplas Escolhas

- Fluxograma:



9. Alternativas de Múltiplas Escolhas

Exemplo 1: Escreva um algoritmo que solicita ao usuário dois operandos e um código de operação (1 - Soma, 2 - Subtração, 3 - Divisão ou 4 - Multiplicação) e realiza a operação correspondente sobre os operandos fornecidos pelo usuário.

9. Alternativas de Múltiplas Escolhas

```
algoritmo "calcula"  
var  
real: operando1, operando2, resultado;  
inteiro: operacao;  
inicio  
    escreva ("Entre com o primeiro operando:");  
    leia (operando1);  
    escreva ("Entre com o segundo operando:");  
    leia (operando2);  
    escreva ("*****");  
    escreva ("Menu");  
    escreva ("*****");  
    escreva ("1 - Soma");  
    escreva ("2 - Subtração");  
    escreva ("3 - Divisão");  
    escreva ("4 - Multiplicação");  
    escreva ("Entre com a opção:");  
    leia (operacao);
```



9. Alternativas de Múltiplas Escolhas

```
escolha operacao
  caso 1
    resultado = operando1 + operando2;
    escreva("O resultado da soma é:", resultado);
  caso 2
    resultado = operando1 - operando2;
    escreva("O resultado da subtração é:", resultado);
  caso 3
    resultado = operando1 / operando2;
    escreva("O resultado da divisão é:", resultado);
  caso 4
    resultado = operando1 * operando2;
    escreva("O resultado da multiplicação é:", resultado);
  outrocaso
    escreva("Operação inválida");
fimescolha
fimalgoritmo
```

9. Alternativas de Múltiplas Escolhas

- Forma geral em C:

```
switch (variavel) { → apenas int ou char  
  case valor1:  
    <comandos>  
    break;  
  case valor2:  
    <comandos>  
    break;  
  case valorN:  
    <comandos>  
    break;  
  default:  
    <comandos>  
}
```

9. Alternativas de Múltiplas Escolhas

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

int main()
{
    float operando1, operando2, resultado;
    int operacao;

    printf("Entre com o primeiro operando:");
    scanf("%f", &operando1);
    printf("Entre com o segundo operando:");
    scanf("%f", &operando2);
    printf("*****\n");
    printf("Menu\n");
    printf("*****\n");
    printf("1 - Soma\n");
    printf("2 - Subtração\n");
    printf("3 - Divisão\n");
    printf("4 - Multiplicação\n");
    printf("*****\n");
    printf("Entre com a opcao:");
    scanf("%d", &operacao);
```

9. Alternativas de Múltiplas Escolhas

```
switch (operacao) {  
    case 1:  
        resultado = operando1 + operando2;  
        printf("A soma eh: %f \n", resultado);  
        break;  
    case 2:  
        resultado = operando1 - operando2;  
        printf("A subtração eh: %f \n", resultado);  
        break;  
    case 3:  
        resultado = operando1 / operando2;  
        printf("A divisão eh: %f \n", resultado);  
        break;  
    case 4:  
        resultado = operando1 * operando2;  
        printf("A multiplicação eh: %f \n", resultado);  
        break;  
    default:  
        printf("Operação inválida!");  
}  
return 0;  
}
```

“(...) a novidade normalmente emerge apenas para aquele que, sabendo com precisão o que deveria esperar, é capaz de reconhecer que algo saiu errado.”

*Thomas S. Kuhn, Físico Teórico,
em seu livro A Estrutura das Revoluções Científicas.*