INE5231 Computação Científica I

Prof. A. G. Silva

04 de abril de 2017

Conteúdo programático

- O computador - [3 horas-aula]
- Representação de algoritmos - [3 horas-aula]:
- Linguagens de programação estruturadas [3 horas-aula]
- Introdução à programação em C [6 horas-aula]
- Programas envolvendo processos de repetição e seleção [6 horas-aula]
- Variáveis estruturadas unidimensionais homogêneas [9 horas-aula]
- Variáveis estruturadas multidimensionais homogêneas [6 horas-aula]
- Variáveis estruturadas heterogêneas [6 horas-aula]
- Subdivisão de problemas e subprogramação [6 horas-aula]
- Programação utilizando uma linguagem de computação técnica numérica [6 horas-aula]

Curso de C

Estruturas Condicionais

Estruturas Condicionais

Roteiro:

- Exemplo
- Condições e operadores relacionais
- Comando `if`
- Comando `if ... else`
- Operadores lógicos
- Comandos `if ... else if` em cascata
- Comando `switch`

Estruturas Condicionais

Motivação:

```
int main(int argc, char* argv[]) {

        double pi = 3.141592;
        double raio, area, perimetro

        printf("Digite o raio: ");
        scanf("%lf", &raio);

        area = pi * (raio * raio);
        perimetro = 2.0 * pi * raio;

        printf("Raio: %f \n", raio);
        printf("Area: %f \n", area);
        printf("Perimetro: %f \n", perimetro);
        return 0;
}
```

Seqüencial:
⇒ Algoritmos Simples

Condições:
⇒ Como decidir se deve executar ou não um bloco

Estruturas Condicionais

Condições

Condições

O que são condições:

- Expressões matemáticas convencionais!
- Testam validade de uma afirmação
- Resultado: número inteiro
- Interpretação:
  - 0: significa *falso*
  - não 0: significa *verdadeiro*
- Operadores especiais:

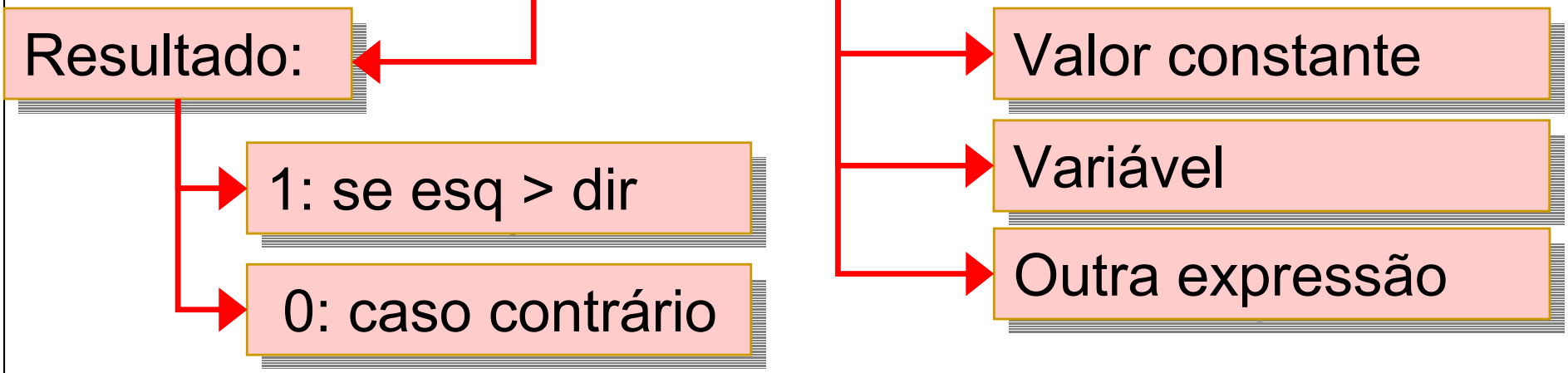
  `<` `<=` `>` `>=` `==` `!=`

Condições

Operador de Comparação:

Maior que:

```
esquerda  >  direita
```



Operadores de comparação:

Expressão	Condição	Resultado
a > b	se a > b	1 (verdadeiro)
	se a ≤ b	0 (falso)
a >= b	se a ≥ b	1 (verdadeiro)
	se a < b	0 (falso)
a < b	se a < b	1 (verdadeiro)
	se a ≥ b	0 (falso)
a <= b	se a ≤ b	1 (verdadeiro)
	se a > b	0 (falso)
a == b	se a = b	1 (verdadeiro)
	se a ≠ b	0 (falso)
a != b	se a ≠ b	1 (verdadeiro)
	se a = b	0 (falso)

Atenção

Cuidado!

Condições

Exemplo:

```
int a, b;
...
a = (1 < 2);
b = (3 <= 2);
```

a = ?
b = ?

a = 1 (verdade)
b = 0 (falso)

```
int c = 3;
int d = 10;
int e, f;
...
e = (c == d);
f = (c != d);
```

e = ?
f = ?

e = 0 (falso)
f = 1 (verdade)

Estruturas Condicionais

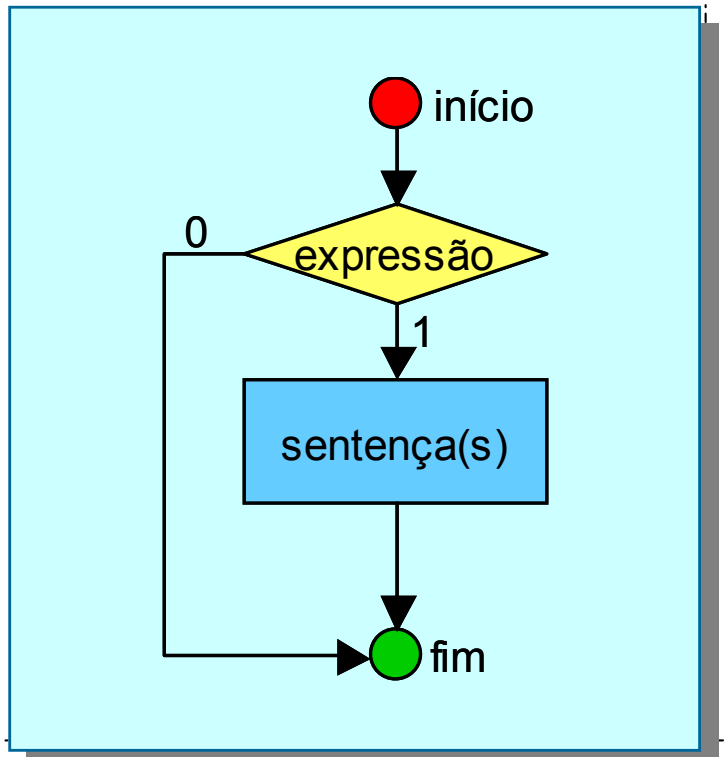
if...

Estrutura `if...`

Executa código somente se uma condição for verdadeira. (resultado da expressão diferente de zero)

Sintaxe:

```
início;
if (expressão) {
    sentença;
    sentença;
    ...
}
fim;
```



Exemplo if...

```
int main(int argc, char *argv[]) {
	int idade;

	printf("Digite sua idade: ");
	scanf("%d", &idade);

	if (idade >= 18) {
		printf("Já pode obter habilitação!");
	}

	return 0;
}
```

EstruturasCondicionais\Idade01\Idade01.vcproj

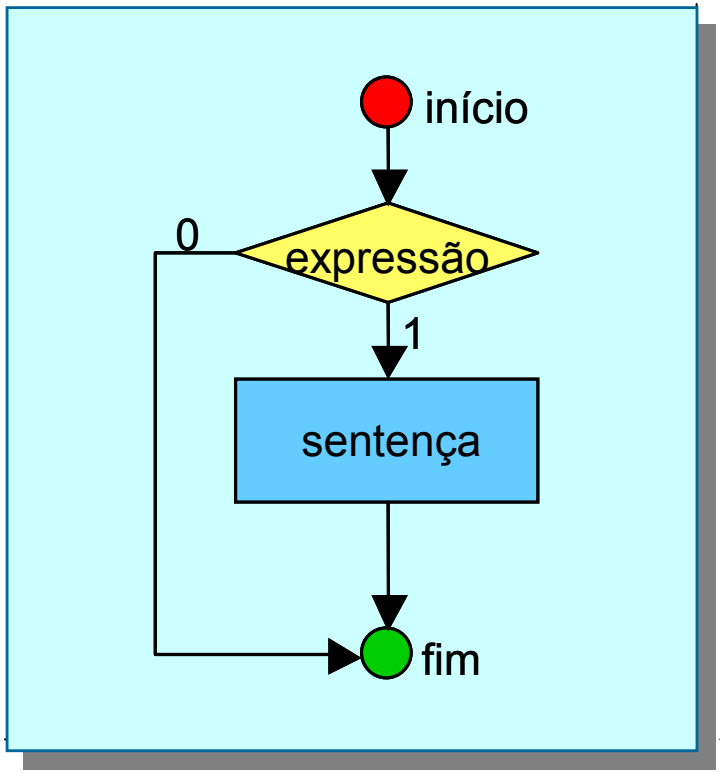
Estrutura if...

Sintaxe simplificada:

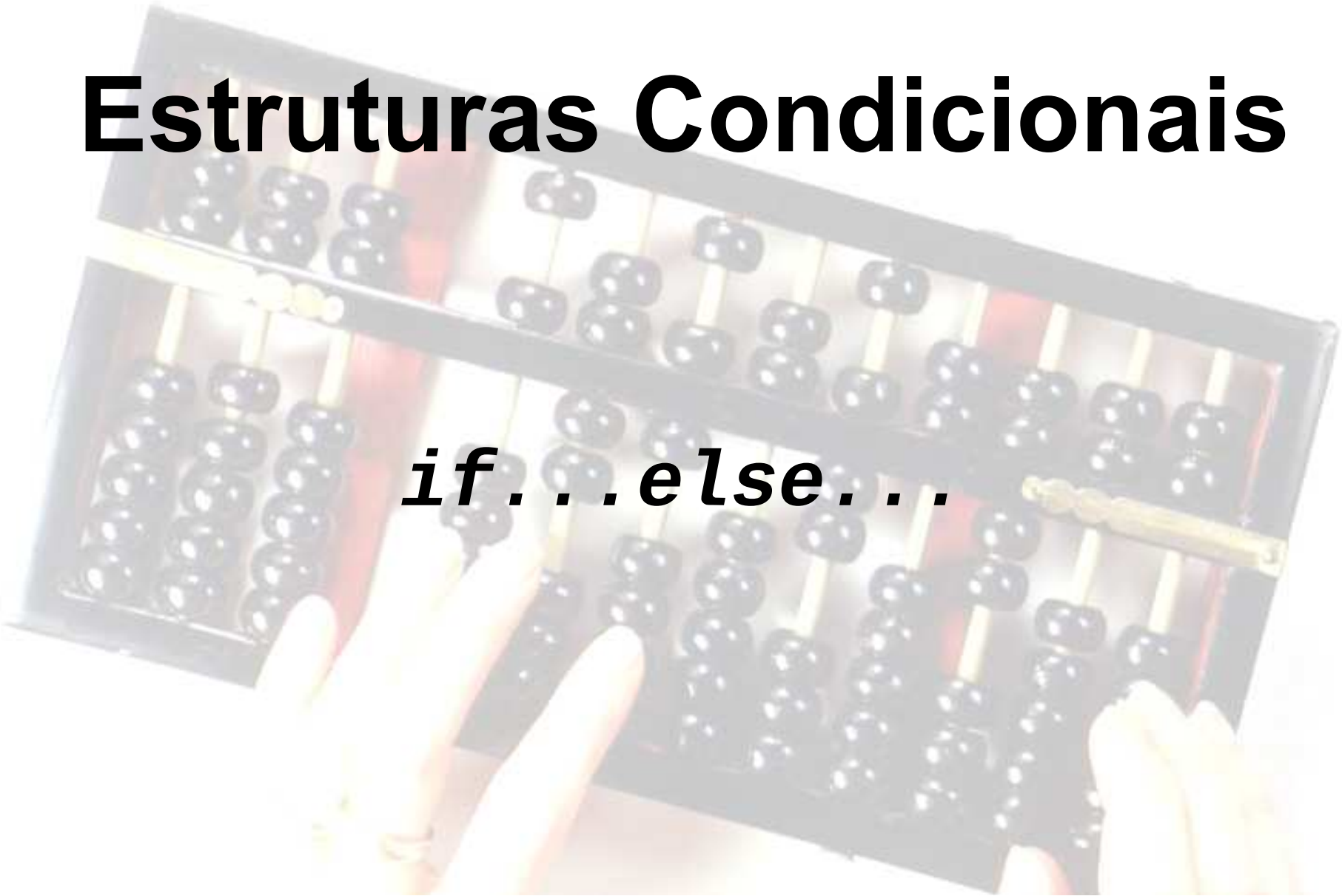
- Uma única sentença
- Sem bloco

Sintaxe:

```
início;
if (expressão)
      sentença;
fim;
```



Estruturas Condicionais



if...else...

if...else...

Exemplo:

```
int main(int argc, char *argv[]) {
        int idade;

        printf("Digite sua idade: ");
        scanf("%d", &idade);

        if (idade >= 18) {
                printf("Você já pode obter habilitação!");
        }
        if (idade <= 17) {
                printf("Espere mais alguns anos!");
        }
return 0;
```

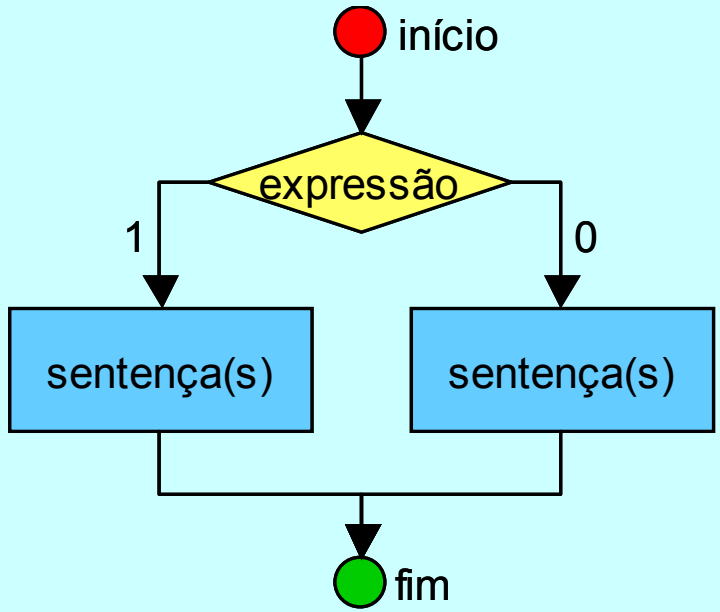
Condições mutuamente exclusivas

Estrutura `if...else...`

Condição verdadeira: executa o primeiro bloco.
Caso contrário: executa o segundo bloco.

Sintaxe:

```
início;
if (expressão) {
    sentença;
    ...
} else {
    sentença;
    ...
}
fim;
```



Exemplo if...else...

```
int main(int argc, char *argv[]) {
        int idade, diferenca_tempo;
        printf("Digite sua idade: ");
        scanf("%d", &idade);

        if (idade >= 18) {
                diferenca_tempo = idade - 18;
                printf("Voce tem habilitacao ha %d ano(s)",
                        diferenca_tempo);
        } else {
                diferenca_tempo = 18 - idade;
                printf("Espere mais %d ano(s)!\n",
                        diferenca_tempo);
        }
        return 0;
}
```

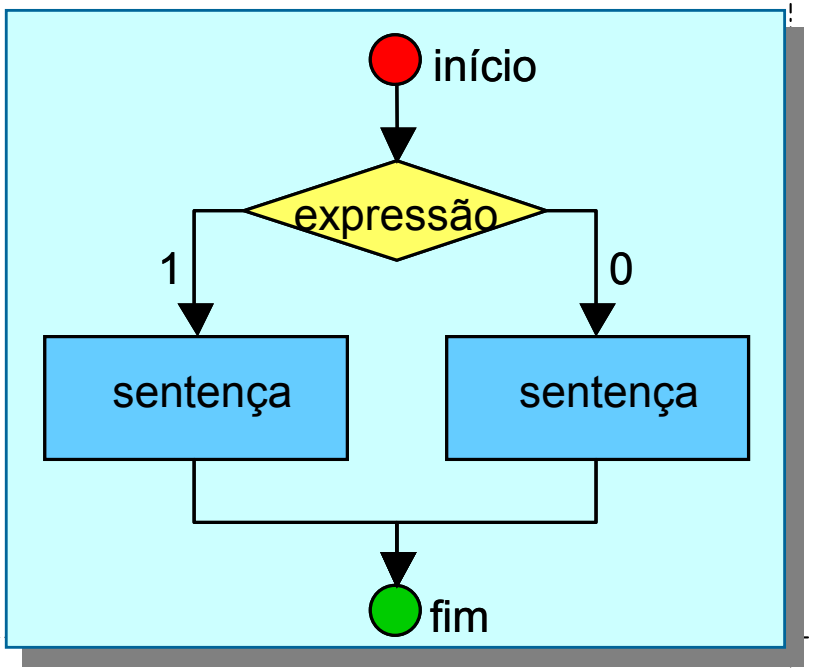
Estrutura `if...else...`

Sintaxe simplificada:

- Uma única sentença
- Sem bloco

Sintaxe:

```
início;
if (expressão)
    sentença;
else
    sentença;
fim;
```



Estruturas Condicionais

Operadores Lógicos

Operadores Lógicos

O que são Operadores Lógicos:

- Objetivo:
  - Criam condições com mais de um teste
- Combinação: (E / AND)
  - As duas expressões precisam ser verdadeiras
- Alternativas: (OU / OR)
  - Uma das duas expressões precisa ser verdadeira
- Negação: (NÃO / NOT)
  - A condição precisa ser falsa

Operadores Lógicos

Exemplos conceituais:

- Carteira de Habilitação:
  - Idade maior ou igual que 18 anos
  - Aprovação nos exames
  - Combinação E / AND
- Cargo de boa remuneração:
  - Excelente currículo profissional
  - Indicação de uma pessoa influente
  - Alternativa OU / OR

Operadores Lógicos:

Expressão	Condição	Resultado
a && b	se *a* não 0 e *b* não 0	1 (verdadeiro) (0 caso contrário)
a || b	se *a* não 0 ou *b* não 0	1 (verdadeiro) (0 caso contrário)
! a	se *a* é 0	1 (verdadeiro) (0 caso contrário)

Operadores Lógicos

Tabelas Verdade:

cond1 && cond2 (E / AND)		cond1	
		falso	*verdadeiro*
cond2	*falso*	*falso*	*falso*
	verdadeiro	*falso*	*verdadeiro*

cond1 || cond2 (OU / OR)		cond1	
		falso	*verdadeiro*
cond2	*falso*	*falso*	*verdadeiro*
	verdadeiro	*verdadeiro*	*verdadeiro*

cond	! cond
falso	*verdadeiro*
verdadeiro	*falso*

Estruturas Condicionais

if...else if...else

Operadores Lógicos

Exemplos:

Média pelo menos 7.0 e freqüência de pelo menos 40 aulas:

```
aprovado = (nota >= 7.0) && (frequencia >= 40)
```

```
nota = 5.0; frequencia = 30;  ⇒    aprovado = 0;
nota = 8.0; frequencia = 30;  ⇒    aprovado = 0;
nota = 5.0; frequencia = 50;  ⇒    aprovado = 0;
nota = 8.0; frequencia = 50;  ⇒    aprovado = 1;
```

Operadores Lógicos

Exemplos:

Média superior a 5.0 ou freqüência superior a 30 aulas:

```
aprovado = (nota > 5.0) || (frequencia > 30)
```

```
nota = 3.0; frequencia = 20;  ⇒    aprovado = 0;
nota = 8.0; frequencia = 20;  ⇒    aprovado = 1;
nota = 3.0; frequencia = 50;  ⇒    aprovado = 1;
nota = 8.0; frequencia = 50;  ⇒    aprovado = 1;
```

Operadores Lógicos

Exemplo:

```
int main(int argc, char *argv[]) {
        int idade;
        float media;
        printf("Digite sua idade: ");
        scanf("%d", &idade);
        printf("Digite sua media nos exames: ");
        scanf("%f", &media);

        if ( (idade >= 18) && (media >= 5.0) ) {
                printf("Voce esta aprovado!");
        } else {
                printf("Ainda nao aprovado!");
        }
        return 0;
}
```

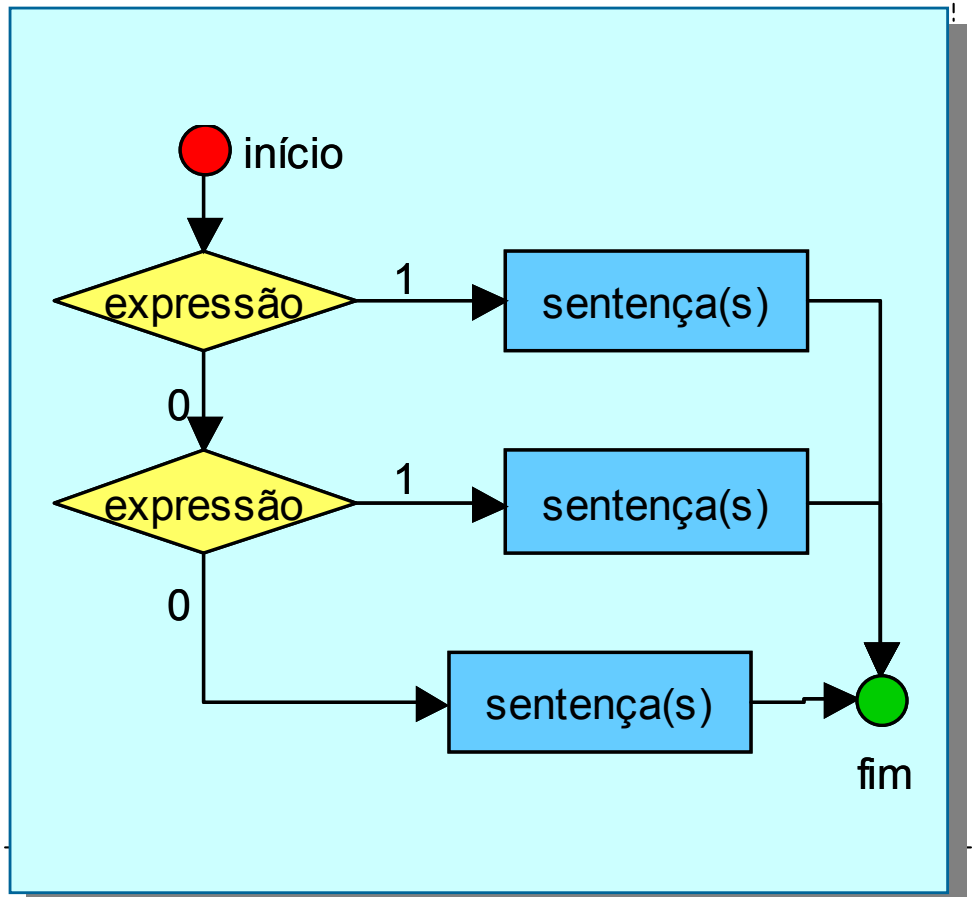
EstruturasCondicionais\Idade04\Idade04.vcproj

Estrutura if...else if...else...

Múltiplas decisões mutuamente exclusivas

Sintaxe:

```
início;
if (expressão) {
    sentença;
    ...
} else if (expressão) {
    sentença;
    ...
} else {
    sentença;
    ...
}
fim;
```



Exemplo if...else if...else...

```
int main(int argc, char *argv[]) {
        int idade;

        printf("Digite sua idade: ");
        scanf("%d", &idade);

        if ( (idade >= 0) && (idade < 18) ) {
                printf("Nao possui habilitacao.\n");
        } else if ( (idade >= 18) && (idade < 65) ) {
                printf("Renove exames a cada 5 anos.\n");
        } else if (idade >= 65) {
                printf("Renove exames a cada 3 anos.\n");
        }
        return 0;
}
```

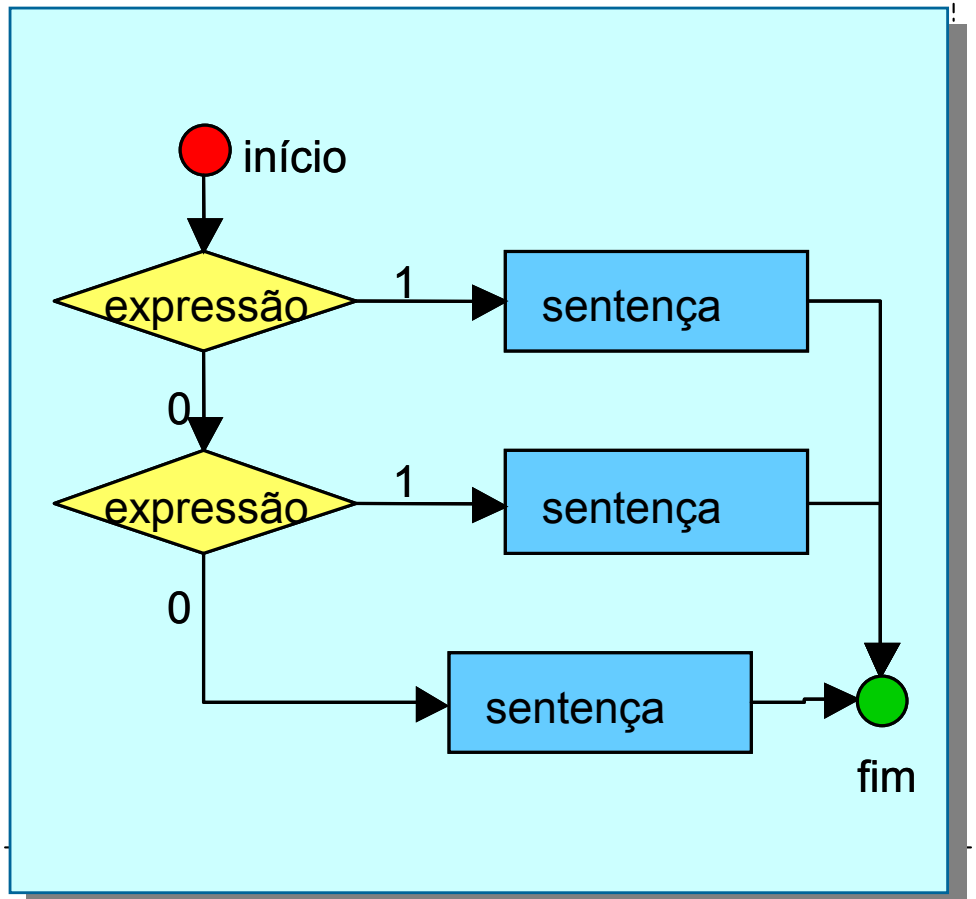
EstruturasCondicionais\Idade05\Idade05.vcproj

Estrutura if...else if...else...

Sintaxe simplificada:

Sintaxe:

```
início;
if (expressão)
    sentença;
else if (expressão)
    sentença;
else
    sentença;
fim;
```



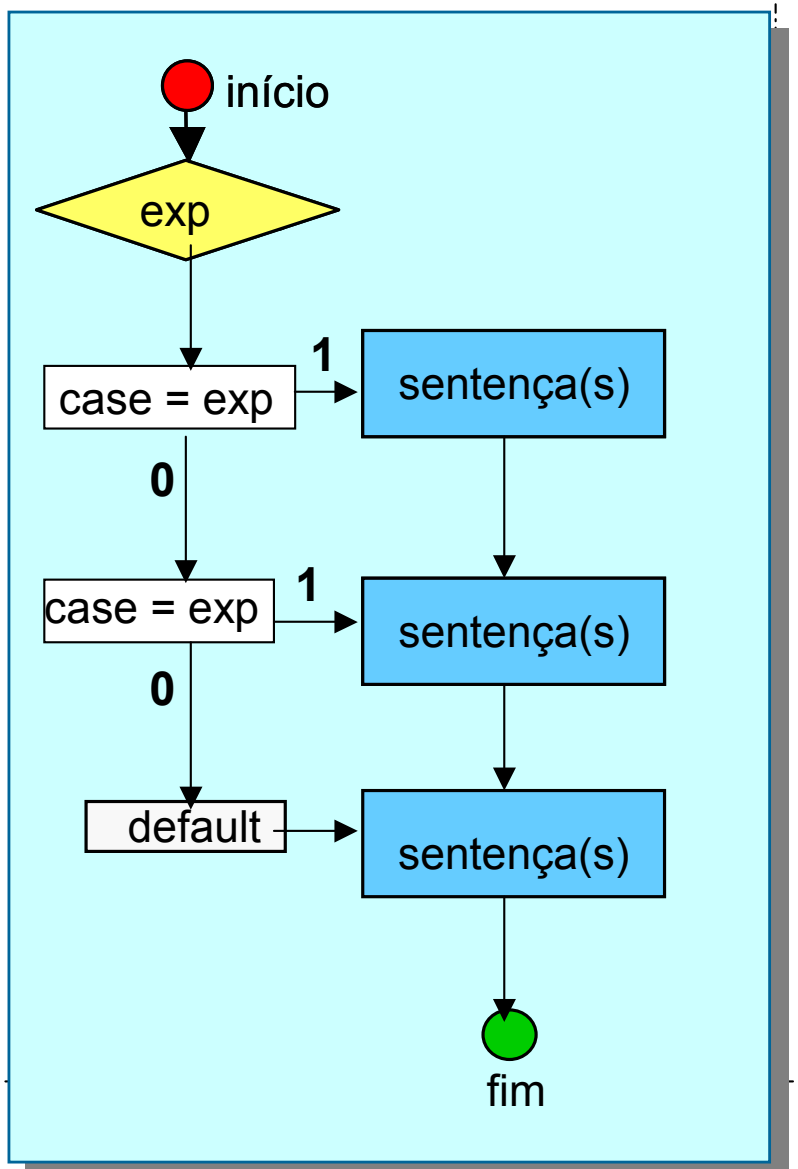
Estruturas Condicionais

`switch`

switch

Conceitos:

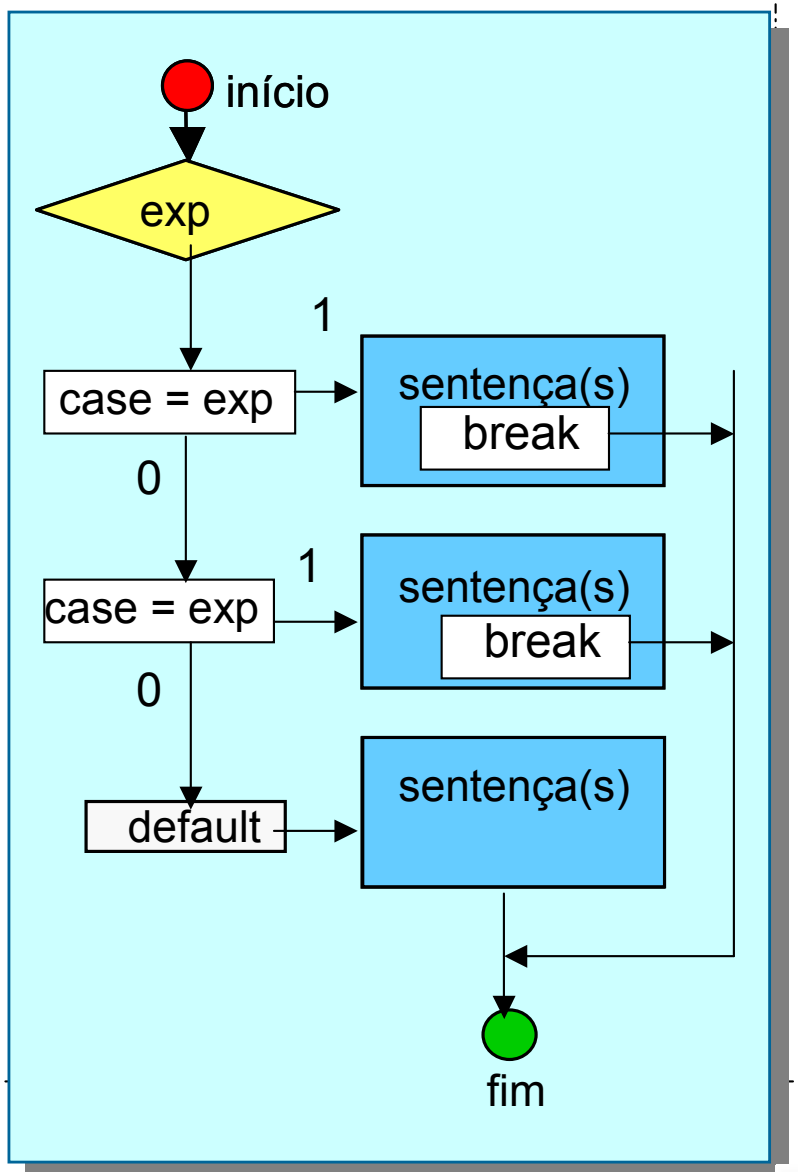
- Várias alternativas (`case`)
  - Valores constantes
- Avalia expressão
  - Compara com cada `case`
  - Entra no `case` correspondente
- `default`: se não encontra alternativa
- Execução prossegue através dos `case`s!



switch

Conceitos:

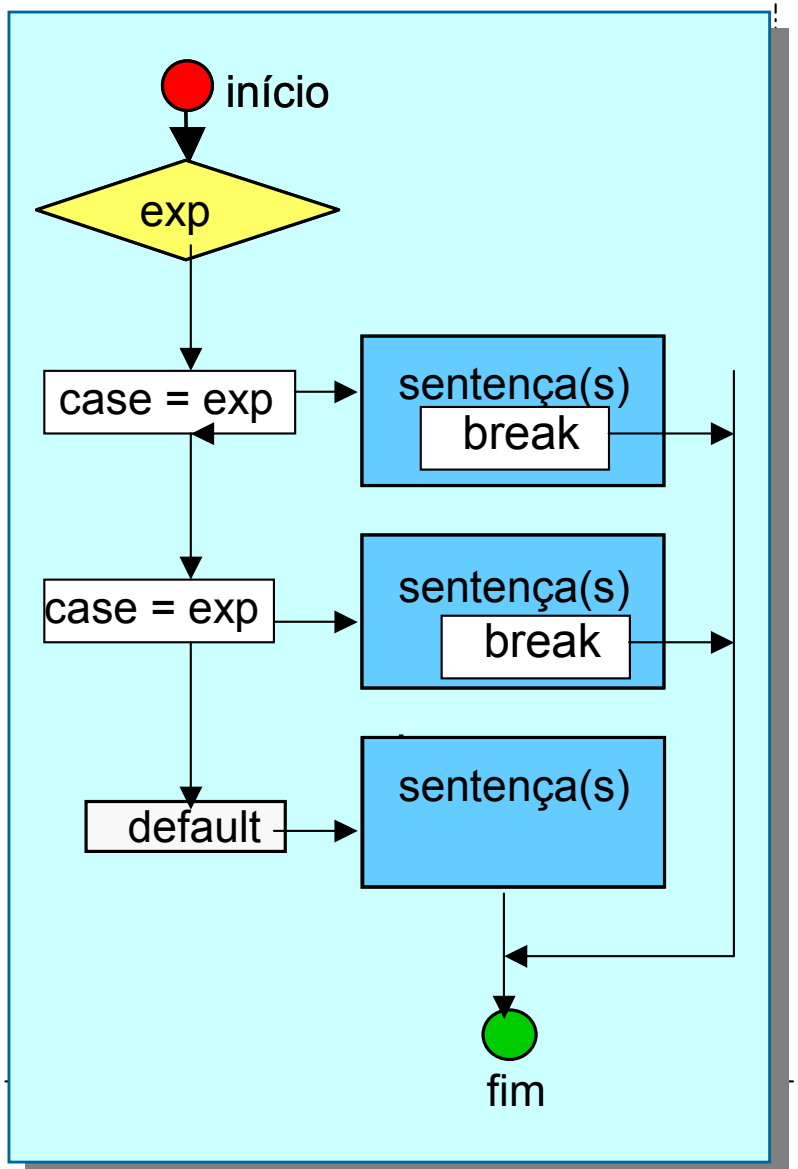
- Várias alternativas (`case`)
  - Valores constantes
- Avalia expressão
  - Compara com cada `case`
  - Entra no `case` correspondente
- `default`: se não encontra alternativa
- `break` – Finaliza switch



switch

Sintaxe:

```
início;
switch (expressão) {
    case valor1:
        sentença(s);
        break;
    case valor2:
        sentença(s);
        break;
    case valor3:
        sentenças;
        break;
    default:
        sentença(s);
        break;
}
fim;
```



Exemplo switch...

```
int main(int argc, char *argv[]) {
        float preco, preco_final;
        char categoria;

        printf("Digite o preço do ingresso: ");
        scanf("%f", &preco);
        printf("E - estudante,\n”);
        printf("A - aposentado,\n”);
        printf("N - normal\n");
        printf("Digite a categoria do cliente (E/A/N): ");
        scanf("%c", &categoria);

        switch (categoria) ...

        return 0;
}
```

Exemplo switch...

```
switch (categoria) {
        case 'e': case 'E':
                preco_final = preco * 0.50f;
                printf(“Preco: %f\n", preco_final);
                break;
        case 'a': case 'A':
                preco_final = preco * 0.70f;
                printf(“Preco: %f\n", preco_final);
                break;
        case 'n': case 'N':
                printf("Preco sem desconto: %f\n", preco);
                break;
        default:
                printf("Categoria invalida!\n");
                break;
}
```

EstruturasCondicionais\Cinema01\Cinema01.vcproj

Estruturas Condicionais

Casos de Uso

Casos de Uso

Quando usar cada estrutura?

- **`if`**
  - Execução condicional de um bloco
  - `if + return`: Para finalizar execução sob determinadas condições (ex: erros)
- **`if...else...`**
  - Execução condicional de um bloco ou outro
  - Condições mutuamente exclusivas
  - Aceitar um dado ou imprimir mensagem de erro

Casos de Uso

Quando usar cada estrutura?

- **`if...else if...else`**
  - Testar intervalos de valores
  - Várias condições mutuamente exclusivas
  - Condições com prioridade
- **`switch (...) ...`**
  - Expressão com alternativas discretas
  - Alternativas em grande número
  - Se processamento for igual para para várias alternativas

Curso de C

Estruturas de Repetição

Estruturas de Repetição

Roteiro:

- Introdução
- Comando `while`
- Comando `do...while`
- Op. de incremento; formas simplificadas
- Comando `for`

Estruturas de Repetição

Introdução:

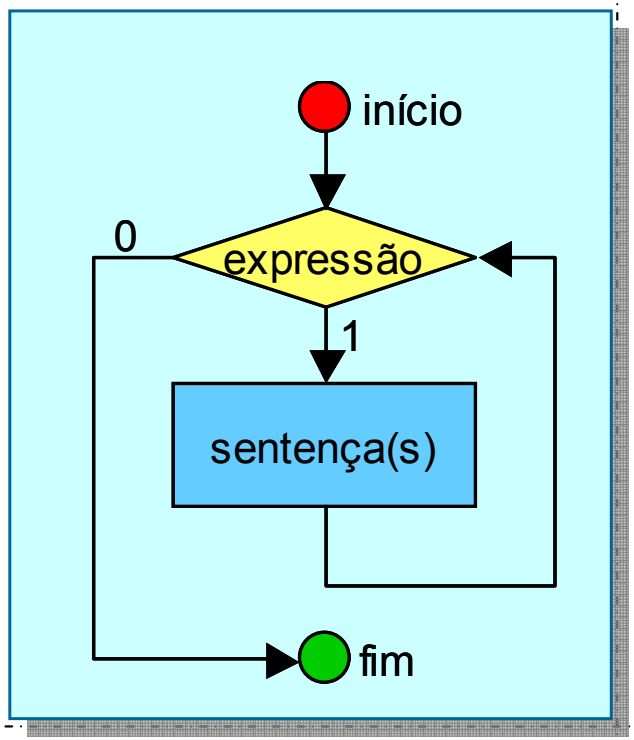
- Estruturas Condicionais:
  - Novidade: Execução condicional de um bloco
- Estruturas de Repetição:
  - Novidade: Repetir a execução de um bloco
  - Controlado por condições
- Exemplos:
  - Preencher uma tabela
  - Aplicar operação a todos elementos da lista
  - Testar vários números
  - Percorrer matrizes, vetores, listas

Estrutura while:

- Executa sentenças enquanto a condição for verdadeira.
- Condição é verificada antes do bloco.

Sintaxe:

```
início;
while (expressão) {
        sentença;
        sentença;
        ...
}
fim;
```



Controle das condições:

Inicializa valores usados na condição

```
int numero = 1;
while (numero <= 10) {
        printf("%d " , numero);
        numero = numero + 1;
}
```

Condição que controla repetição

Atualiza valores usados na condição

Exemplo while:

```
int main(int argc, char *argv[]) {// imprime divisores
        int numero, divisor, resto;

        printf("Digite o numero: ");
        scanf("%d", &numero);

        divisor = 1;
        while (divisor <= numero) {
                resto = numero % divisor;
                if (resto == 0) {
                        printf("Divisor: %d \n", divisor);
                }
                divisor = divisor + 1;
        }
        return 0;
}
```

EstruturasRepeticao\Divisores01\Divisores01.vcproj

Exemplo while:

```
int main(int argc, char *argv[]) {// MDC de positivos
        int numeroA, numeroB, resto;

        printf("Digite dois números (ordem crescente): ");
        scanf("%d %d", &numeroA, &numeroB);

        while (numeroA > 0) {
                resto = numeroB % numeroA;
                numeroB = numeroA;
                numeroA = resto;
        }

        printf("MDC: %d", numeroB);

        return 0;
}
```

EstruturasRepeticao\mdc01\mdc01.vcproj

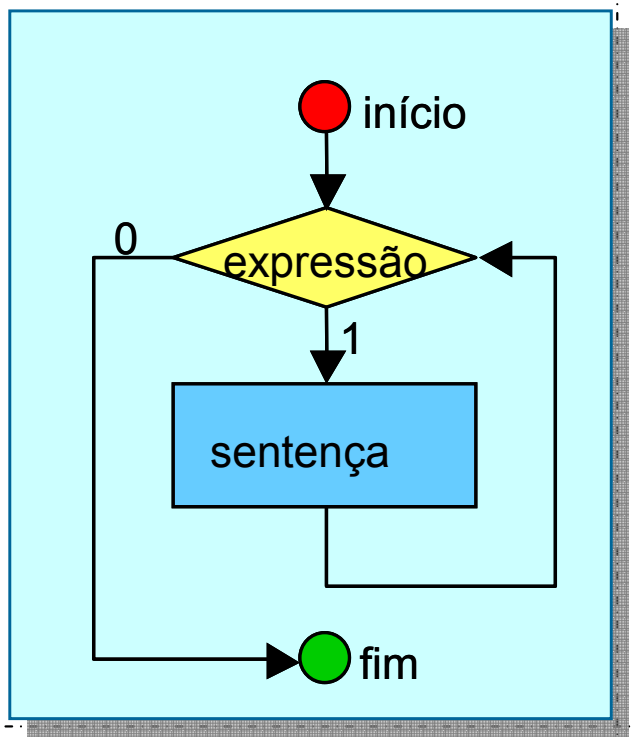
Estrutura while:

Sintaxe simplificada:

- Uma única sentença
- Sem bloco

Sintaxe:

```
início;
while (expressão)
    sentença;
fim;
```



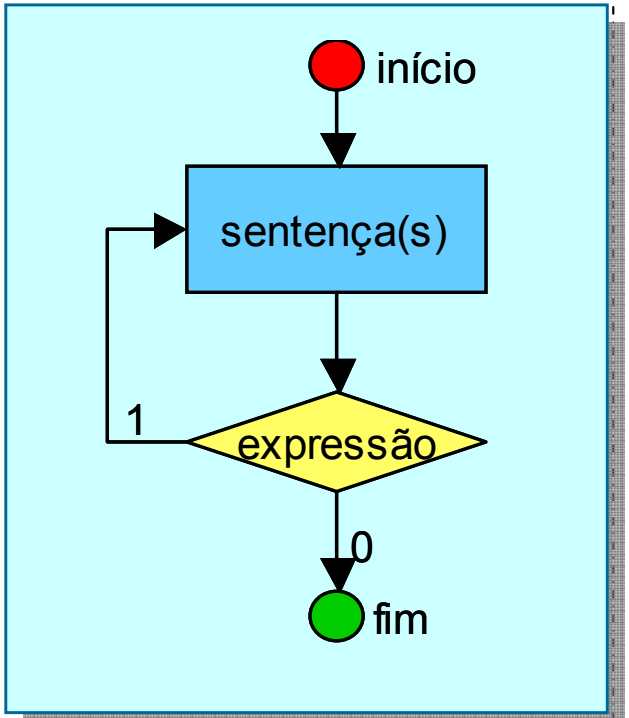
do...while

Estrutura do...while:

- Executa sentenças enquanto a condição for verdadeira.
- Condição é verificada depois do bloco

Sintaxe:

```
início;
do {
    sentença;
    sentença;
    ...
} while (expressão);
fim;
```



do...while

Exemplo do...while :

Código:

```
int numero = 1;
do {
    printf("%d " , numero);
    numero = numero + 1;
} while (numero <= 10);
```

Resultado:

```
1 2 3 4 5 6 7 8 9 10
```

EstruturasRepeticao\dowhile01\dowhile01.vcproj

Exemplo do...while:

```
int main(int argc, char *argv[]) {// MDC de positivos
        int numeroA, numeroB, resto;

        printf("Digite dois números (ordem crescente): ");
        scanf("%d %d", &numeroA, &numeroB);

        do {
                resto = numeroB % numeroA;
                numeroB = numeroA;
                numeroA = resto;
        } while (numeroA > 0);

        printf("MDC: %d", numeroB);
        return 0;
}
```

EstruturasRepeticao\mdc02\mdc02.vcproj

Operadores de Incremento

Operadores de incremento:

- Antes:

  numero = numero + 1;

  numero = numero - 1;

- Agora:

  ++numero;

  --numero;

- Retornam valor da variável após a operação

Operadores de incremento:

Para:	Atalho:	Original:
Somar um	`++numero`	`numero=numero+1`
Subtrair um	`--numero`	`numero=numero-1`

Operadores de Incremento

Operadores de incremento:

- Antes:

  numero = numero + 1;

  numero = numero - 1;

- Agora:

  numero++;

  numero--;

- Retornam valor da variável antes da operação

Operadores de incremento:

Para:	Atalho:	Original:
Somar uma unidade	`numero++`	`numero=numero+1`
Subtrair uma unidade	`numero--`	`numero=numero-1`

Operadores Aritméticos

Operadores aritméticos: notação simplificada

- Antes:

```
numero = numero * 10;
numero = numero + 3;
```

- Agora:

```
numero *= 10;
numero += 3;
```

- Retornam valor da expressão

Operadores de incremento:

Para:	Atalho:	Original:
Somar k unidades	`numero += k`	`numero=numero+k`
Subtrair k unidades	`numero -= k`	`numero=numero-k`
Multiplicar por k	`numero *= k`	`numero=numero*k`
Dividir por k	`numero /= k`	`numero=numero/k`

Operadores de Incremento

Exemplo:

Antes:

```
int numero = 1;
while (numero <= 10) {
        printf("%d " , numero);
        numero = numero + 1;
}
```

Depois:

```
int numero = 1;
while (numero <= 10) {
        printf("%d " , numero);
        numero++;
}
```

EstruturasRepeticao\while02\while02.vcproj

EstruturasRepeticao\dowhile02\dowhile02.vcproj

Controle das condições:

Uma estrutura de repetição tem 4 componentes:

- Inicialização
- Condição
- Sentenças
- Atualização

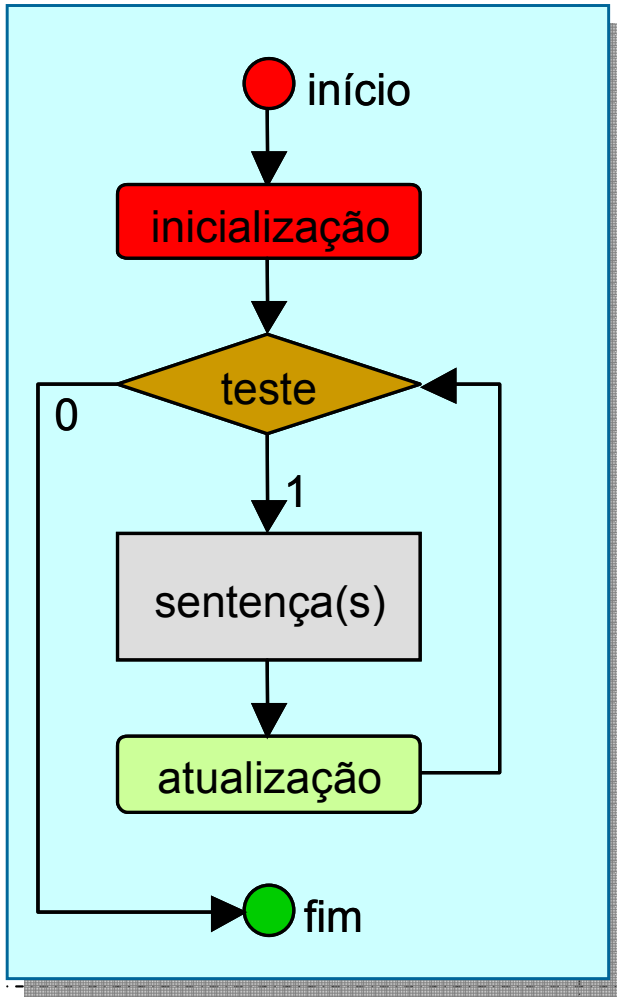
```
int numero = 1;
while (numero <= 10) {
        printf("%d " , numero);
        numero = numero + 1;
}
```

Estrutura for:

- Automatiza estrutura de repetição típica

Sintaxe:

```
inicio;
for (inicialização;
   teste;
   atualização) {
      sentença;
      sentença;
      ...
}
fim;
```



Exemplo for:

Código:

```
int numero;
for (numero = 1; numero <= 10; numero++) {
      printf("%d ", numero);
}
```

Resultado:

```
1 2 3 4 5 6 7 8 9 10
```

EstruturasRepeticao\for01\for01.vcproj

Por que usar `for`?

- Cabeçalho agrupa:
  - Inicialização
  - Condição
  - Atualização

  } Programador não "esquece" nenhuma etapa

- Separa:
  - Controle (lógica) de repetição
  - Código a ser repetido

  } Código organizado

Exemplo for:

```
int main(int argc, char *argv[]) {
    int numero, divisor, resto;

    printf("Digite o numero: ");
    scanf("%d", &numero);

    for (divisor = 1; divisor <= numero; divisor++) {
        resto = numero % divisor;
        if (resto == 0) {
            printf("Divisor: %d \n", divisor);
        }
    }

    return 0;
}
```

EstruturasRepeticao\Divisores02\Divisores02.vcproj

Casos de Uso

Casos de Uso:

- **`while (expressão) { ... }`**
  - Não há variável contadora
  - Inicialização, teste ou atualização complexos
  - Informações da condição obtidas na execução
- **`do { ... } while (expressão);`**
  - Executar um bloco pelo menos uma vez
  - Só é possível avaliar a condição depois de executar
  - Informações da condição obtidas após execução

Casos de Uso

Casos de Uso:

- `for (inicialização; teste; reinicialização) { ... }`
  - Há variável contadora de repetições
  - Inicialização, teste e atualização simples
  - Separa claramente as instruções de controle das instruções do bloco

Estruturas de Repetição

Exemplos

Exemplos

Caso 1: for:

```
int main(int argc, char *argv[]) {// acha media
   int quantidade, contador;
   double valor, soma = 0.0;

   printf("Quantidade de valores: ");
   scanf("%d", &quantidade);// quantidade >= 1

   for (contador = 1; contador <= quantidade; contador++) {
      scanf("%lf", &valor);
      soma += valor;
   }

   printf("Media: %f", soma / quantidade);
   return 0;
}
```

EstruturasRepeticao\Caso01\Caso01.vcproj

Caso 2: while:

```
int main(int argc, char *argv[]) {// acha media
   int quantidade, contador;
   double valor, soma = 0.0;

   printf("Quantidade de valores: ");
   scanf("%d", &quantidade); // >= 1
   contador = 1;
   while (contador <= quantidade) {
      scanf("%lf", &valor);
      soma += valor;
      contador++;
   }
   printf("Media: %f", soma / quantidade);
   return 0;
}
```

EstruturasRepeticao\Caso02\Caso02.vcproj

Exemplos

Caso 3: while:

```
int main(int argc, char *argv[]) {// acha media
   int quantidade = 0;
   double valor, soma = 0.0;

   printf("Escreva valores. -1 termina.\n");// >= 1
   scanf("%lf", &valor);
   while (valor >= 0.0) {
      soma += valor;
      quantidade++;
      scanf("%lf", &valor);
   }

   printf("Media: %f", soma / quantidade);
   return 0;
}
```

EstruturasRepeticao\Caso03\Caso03.vcproj

Exemplos

Caso 4: do...while:

```
int main(int argc, char *argv[]) {// acha media
   int quantidade = 0;
   double valor, soma = 0.0;

   printf("Escreva valores. -1 termina.\n"); // >= 1
   do {
      scanf("%lf", &valor);
      if (valor >= 0.0) {
        soma += valor;
        quantidade++;
      }
   } while (valor >= 0.0);

   printf("Media: %f", soma / quantidade);
   return 0;
}
```

Exemplos

Caso 5: do...while:

```
int main(int argc, char *argv[]) {// acha media; e repete
   int quantidade, contador;
   double valor, soma;
   char repetir;
   do {
      printf("Quantidade de valores: ");
      scanf("%d", &quantidade); // >=1
      soma = 0;
      for (contador = 1; contador <= quantidade; contador++) {
         scanf("%lf", &valor);
         soma += valor;
      }
      printf("Media: %f\n\n", soma / quantidade);
      printf("Deseja executar o programa novamente? (s/n) ");
      scanf(" %c", &repetir);// atencao p/ espaco
   } while (repetir == 's');
   return 0;
}
```

EstruturasRepeticao\Caso05\Caso05.vcproj

Curso de C

Controle de Execução

Controle de Execução

Roteiro:

- Comando `break`
- Comando `continue`
- Comando `goto`

Objetivo do `break`:

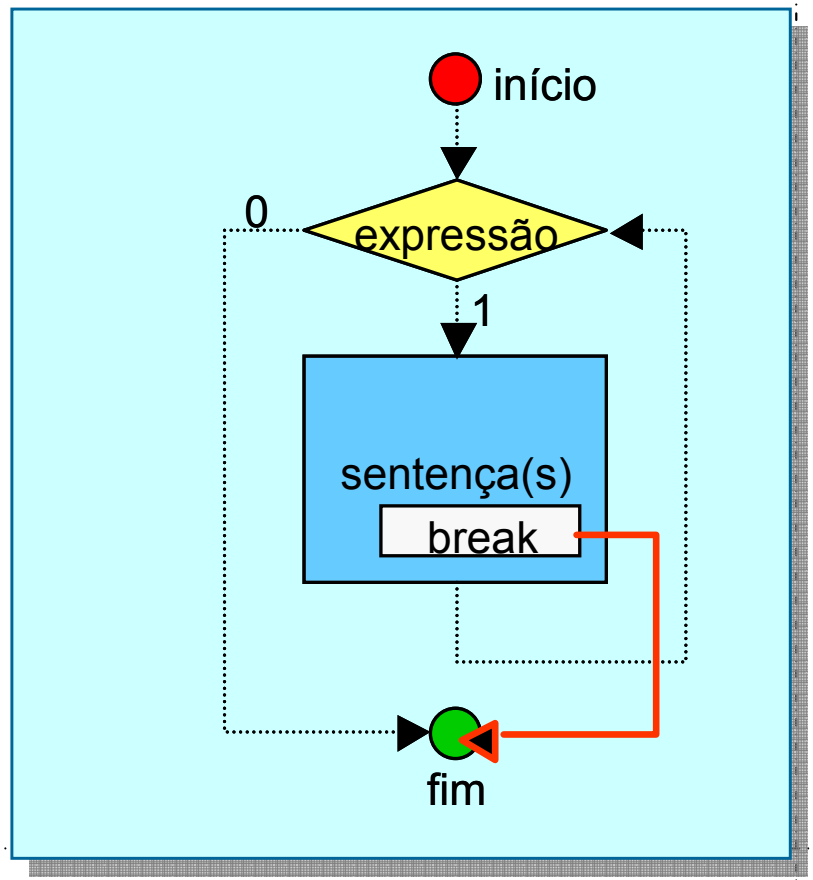
- Cancelar execução:
  - **`for` / `while` / `do...while`**
- Comportamento:
  - Termina imediatamente o bloco
  - Não executa restante do bloco
  - Continua logo após o bloco
- Exemplos:
  - Terminar uma busca
  - Situações de erro
  - Evitar repetições

break

Sintaxe break com while

Sintaxe:

```
while (expressão) {
    sentenças(s);
    if (condição) {
        break;
    }
    sentenças(s);
}
```

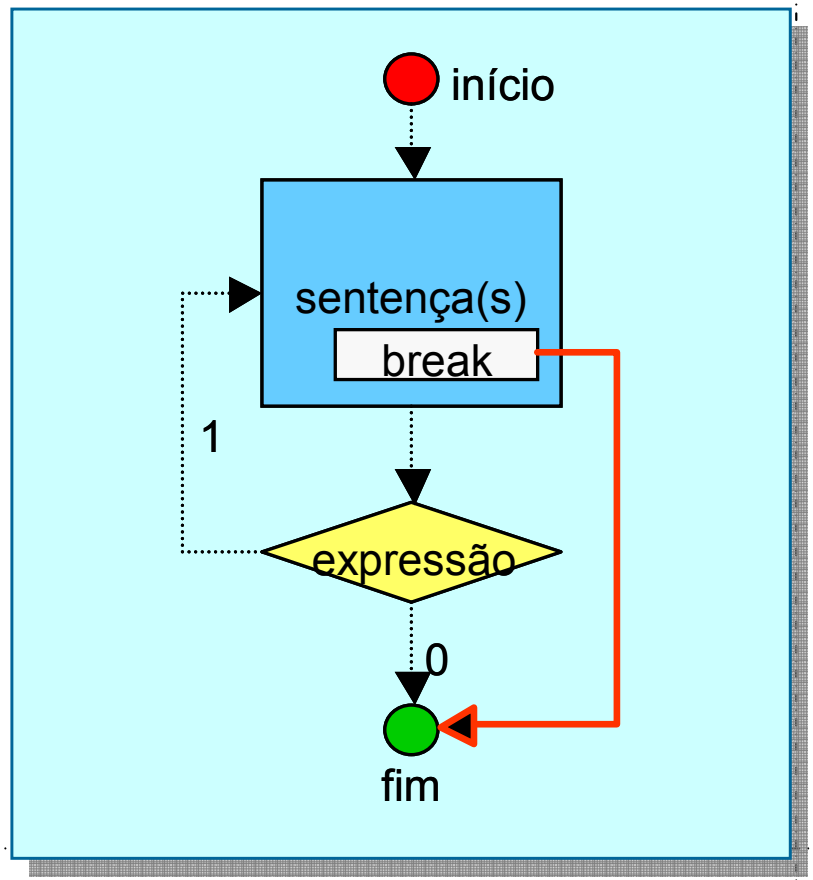


break

Sintaxe break com do...while

Sintaxe:

```
do {
    sentenças(s);
    if (condição) {
        break;
    }
    sentenças(s);
} while (expressão);
```

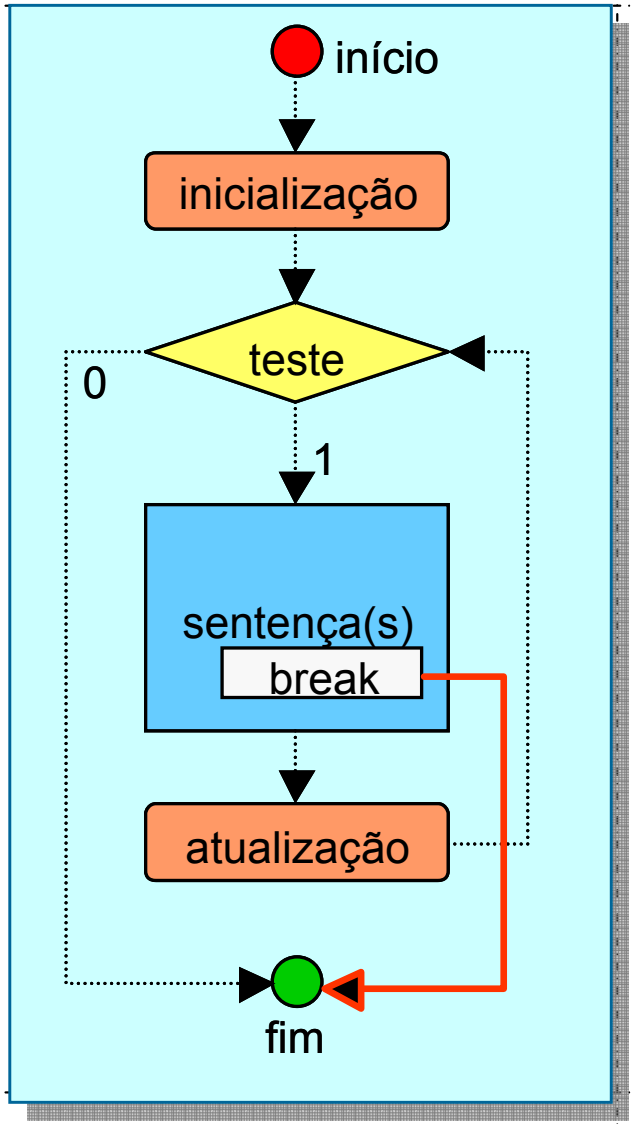


break

Sintaxe break com for()

Sintaxe:

```
for (inicialização;
     teste;
     atualização) {
    sentenças(s);
    if (condição) {
        break;
    }
    sentenças(s);
}
```



break

```
int main(int argc, char *argv[]) { // num. divisores
    int numero, divisor, resto, numero_divisores;

    printf("Digite o numero: ");
    scanf("%d", &numero);

    numero_divisores = 0;
    for (divisor = 1; divisor <= numero; divisor++) {
        resto = numero % divisor;
        if (resto == 0) {
            numero_divisores++;
            if (numero_divisores >= 3) {
                break;
            }
        }
    }
    if (numero_divisores == 2) {
        printf("O número %d é primo!\n", numero);
    }
    return 0;
}
```

ControleExecucao\Divisores03\Divisores03.vcproj

`continue`

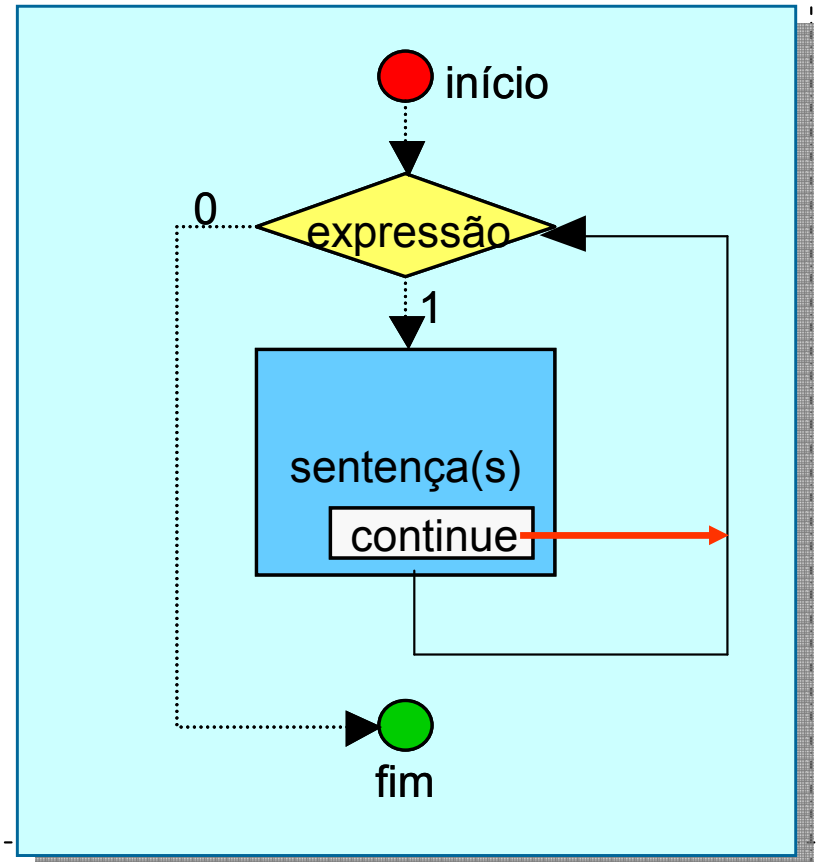
Objetivo do `continue`:

- Reiniciar execução:
  - **`for`** / **`while`** / **`do...while`**
- Comportamento:
  - Reinicia o bloco
  - Não executa resto do bloco
- Exemplos:
  - Pular valores inválidos
  - Evitar processamento

Sintaxe continue com while

Sintaxe:

```
while (expressão) {
    sentenças(s);
    if (condição) {
        continue;
    }
    sentenças(s);
}
```

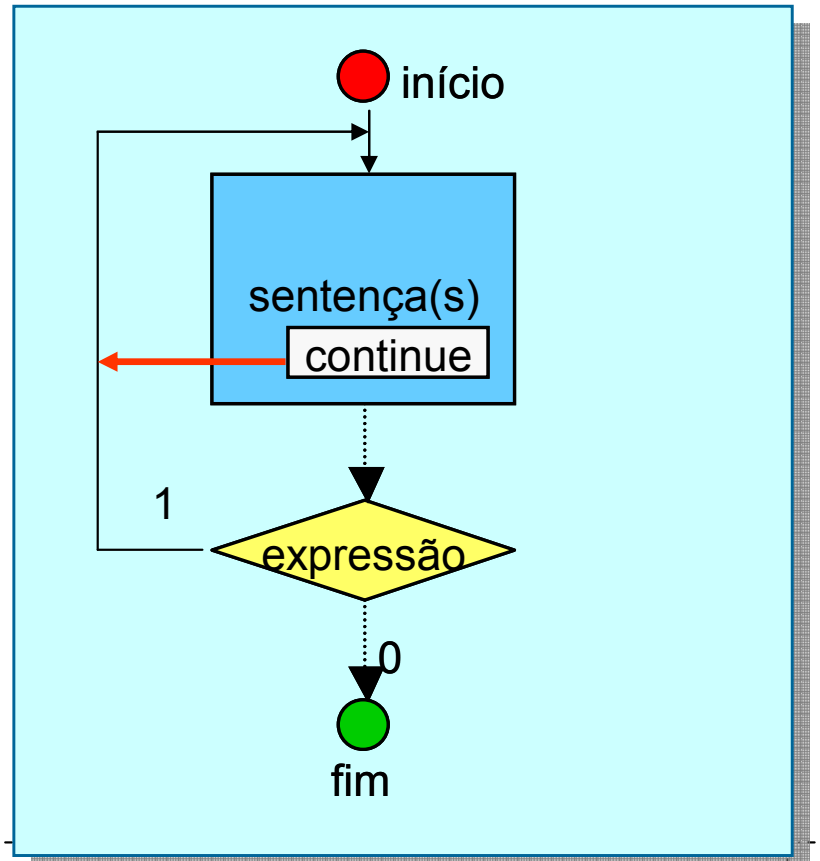


continue

Sintaxe continue com do...while

Sintaxe:

```
do {
    sentenças(s);
    if (condição) {
        continue;
    }
    sentenças(s);
} while (expressão);
```

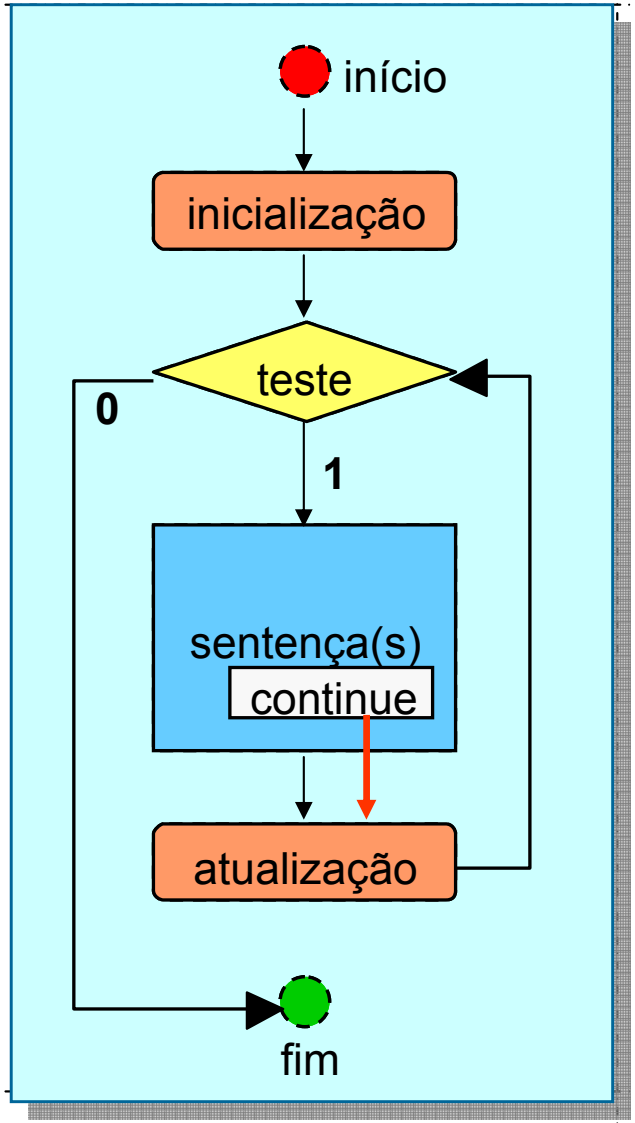


continue

Sintaxe continue com for

Sintaxe:

```
for (inicialização;
      teste;
      atualização) {
    sentenças(s);
    if (condição) {
       continue;
    }
    sentenças(s);
}
```



OBS: executa também a atualização!

continue

```
int main(int argc, char *argv[]) {
        double angulo, tangente;
        double pi = 3.14159265358979323846264338327795;

        for (angulo = 0;
               angulo <= 180;
               angulo += 10.0) {
                 if (angulo == 90.0) {
                       continue;
                 }

               tangente = tan((angulo/180)*pi);
               printf("tan(%8.2f)=%8.2f\n",angulo,tangente);
        }

        return 0;
}
```

ControleExecucao\Tangete01\Tangente01.vcproj

Controle de Execução

goto

goto

Objetivo do `goto`:

- Desviar execução para uma marca
- Saltos para pontos arbitrários
- Estrutura de repetição primitiva

Exemplo:
Repetição infinita

```
marca1:
...
sentença(s);
...
goto marca1;
```

Sintaxe: goto

Retrocesso de execução:

Sintaxe:

```
sentença(s);
...
marca1:
...
sentença(s);
...
goto marca1;
...
sentença(s);
```

Avanço de execução:

Sintaxe:

```
sentença(s);
...
goto marca2;
...
sentença(s);
...
marca2:
...
sentença(s);
```

goto

```
int main(int argc, char *argv[]) {
        int numero = 1;

inicio_repeticao:
        if (numero > 10) {
                goto fim_repeticao;
        }
        printf("%d " , numero);
        numero++;
        goto inicio_repeticao;

fim_repeticao:

        return 0;
}
```

ControleExecucao\Goto01\Goto01.vcproj

`goto`

Uso do `goto`:

- Difícil visualizar os destinos do **`goto`**
- Oculta lógica de execução
- Programas tornam-se incompreensíveis!
- **Dica:** não use goto

Referências

- Notas do Prof. Arnaldo V. Moura e Daniel F. Ferber – Curso C – IC/Unicamp