

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

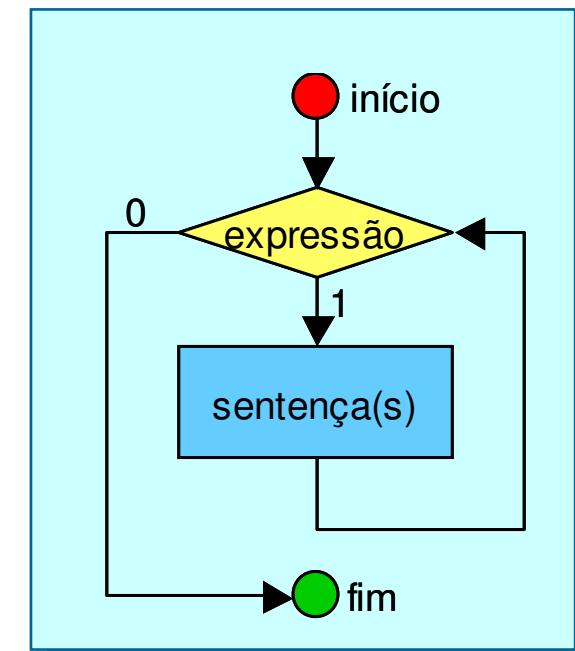
while

Estrutura while:

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

Sintaxe:

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



while

Exemplo while:

Código:

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

Resultado:

1 2 3 4 5 6 7 8 9 10

EstruturasRepeticao\while01\while01.vcproj

while

Controle das condições:

Inicializa valores usados na condição

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

while

Controle das condições:

Inicializa valores usados na condição

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

Condição que controla repetição

while

Controle das condições:

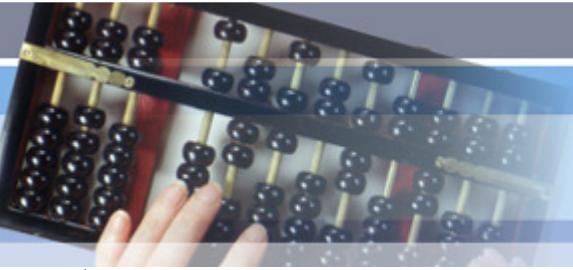
Inicializa valores usados na condição

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

Condição que controla repetição

}

Atualiza valores usados na condição



while

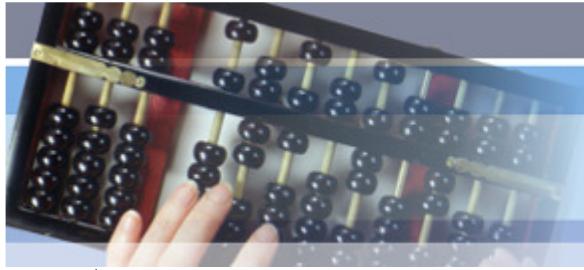
Exemplo while:

```
int main(int argc, char *argv[]) // num. de 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



while

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

while

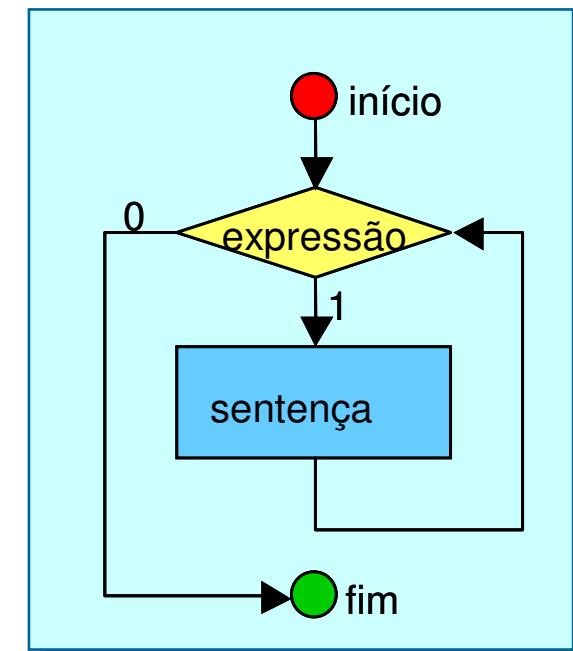
Estrutura while:

Sintaxe simplificada:

- Uma única sentença
- Sem bloco

Sintaxe:

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



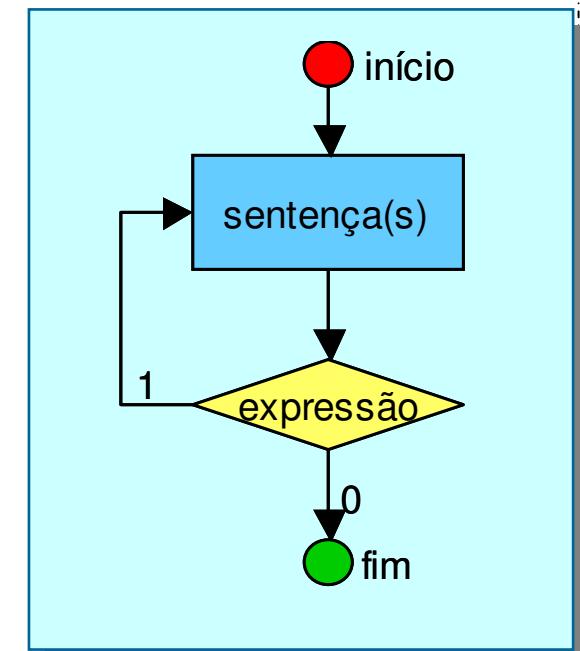
do . . . while

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

do . . . while

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;`

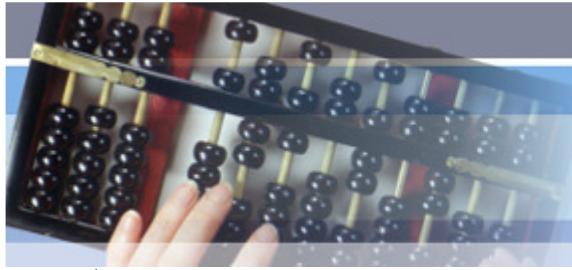
`numero = numero - 1;`

- Agora:

`++numero;`

`--numero;`

- Retornam valor da variável **após** a operação



Operadores de Incremento

Operadores de incremento:

Para:	Atalho:	Original:
Somar um	<code>++numero</code>	<code>numero=numero+1</code>
Subtrair um	<code>--numero</code>	<code>numero=numero-1</code>

Operadores de Incremento

Operadores de incremento:

- Antes:

`numero = numero + 1;`

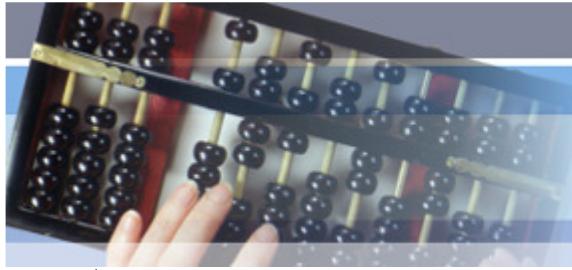
`numero = numero - 1;`

- Agora:

`numero++;`

`numero--;`

- Retornam valor da variável **antes** da operação



Operadores de Incremento

Operadores de incremento:

Para:	Atalho:	Original:
Somar uma unidade	<code>numero++</code>	<code>numero=numero+1</code>
Subtrair uma unidade	<code>numero--</code>	<code>numero=numero-1</code>

Operadores Aritméticos

Operadores aritméticos: notação simplificada

- Antes:

numero = numero * 10;

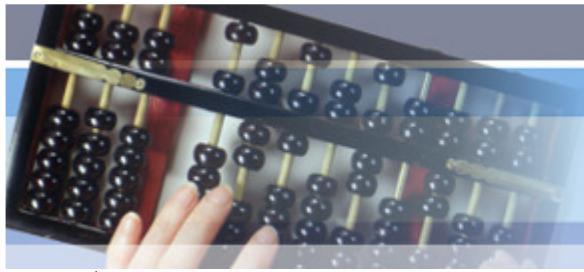
numero = numero + 3;

- Agora:

numero *= 10;

numero += 3;

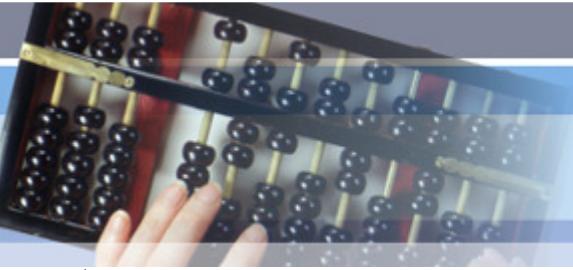
- Retornam valor da expressão



Operadores de Incremento

Operadores de incremento:

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



Operadores de Incremento

Exemplo:

Antes:

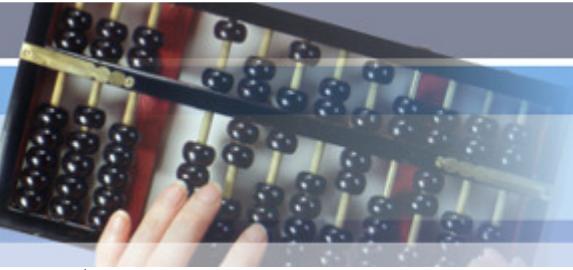
```
int numero = 1;
while (numero <= 10) {
    printf("%d ", numero);
    numero = numero + 1;
}
```

Depois:

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

EstruturasRepeticao\while02\while02.vcproj

EstruturasRepeticao\dowhile02\dowhile02.vcproj



for

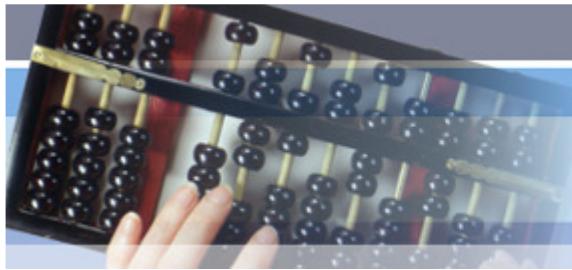
Controle das condições:

Uma estrutura de repetição tem 4 componentes:

- Inicialização
- Condição
- Sentenças
- Atualização

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

for



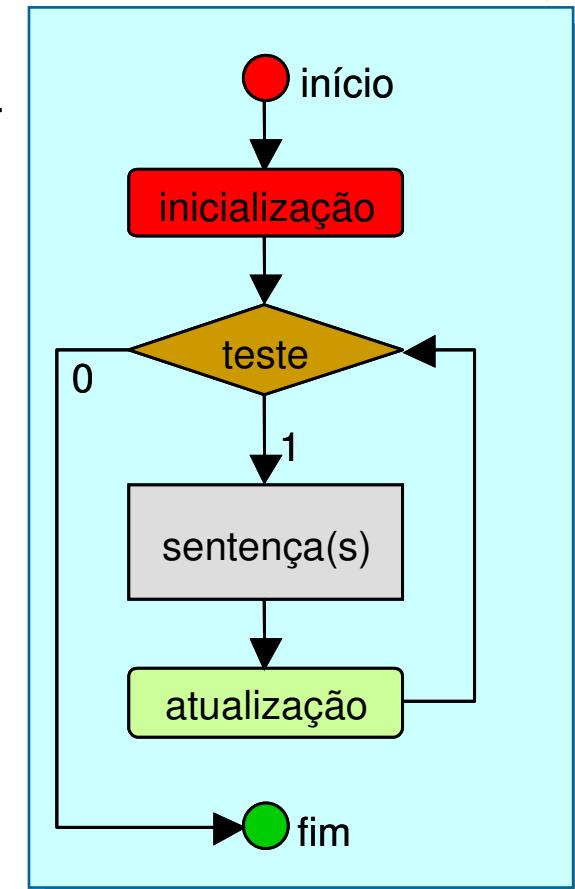
Estrutura for:

- Automatiza estrutura de repetição típica

Sintaxe:

```

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



for

Exemplo for:

Código:

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

Resultado:

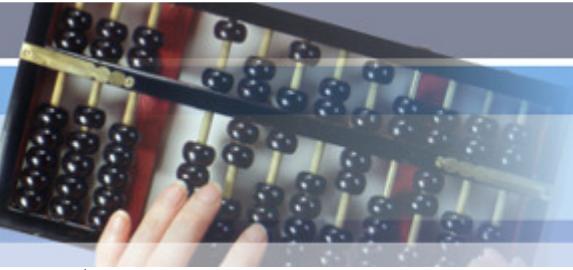
1 2 3 4 5 6 7 8 9 10

EstruturasRepeticao\for01\for01.vcproj

for

Por que usar `for`?

- Cabeçalho agrupa:
 - Inicialização
 - Condição
 - Atualização
 - Separa:
 - Controle (lógica) de repetição
 - Código a ser repetido
- Programador não
“esquece” nenhuma etapa
- Código
organizado

A photograph of a person's hand using a traditional abacus to perform calculations. The abacus has multiple rods with beads, and the hand is shown in the process of moving the beads.

for

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

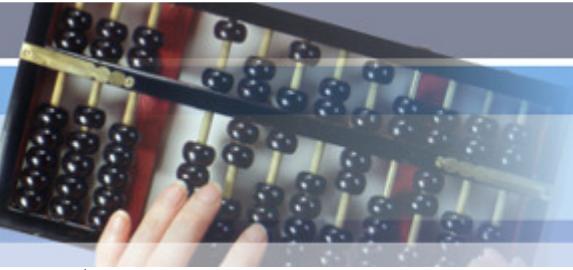
Casos de Uso

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

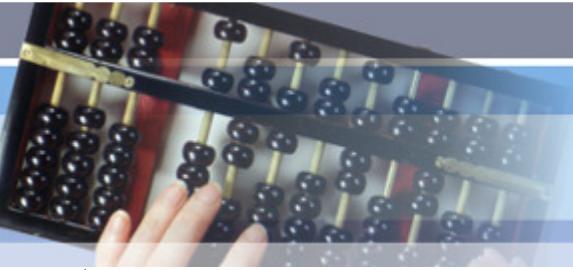
Caso 1: for:

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

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

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

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



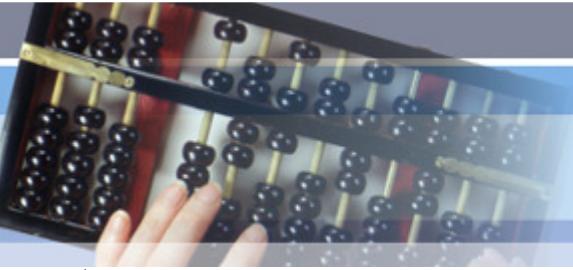
Exemplos

Caso 2: while:

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

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

EstruturasRepeticao\Caso02\Caso02.vcproj



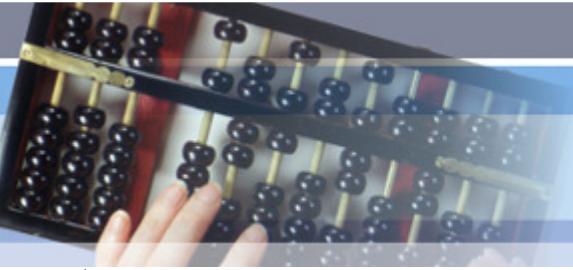
Exemplos

Caso 3: while:

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

    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;
}
```



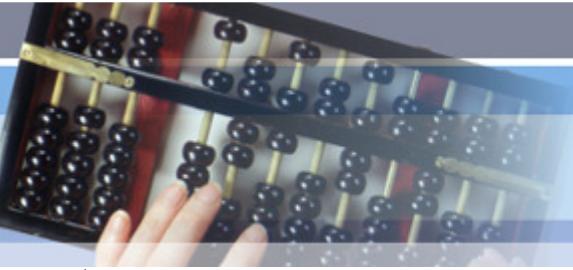
Exemplos

Caso 4: do...while:

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

    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, media;
    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



Exemplos

- SomaSerie
- PotBase2
- ContaVogais
- Tabuada
- ContaPalavras
- TrianguloFloyd
- OplIncr01
- OplIncr02

Curso de C

Controle de Execução

Controle de Execução

Roteiro:

- Comando break
- Comando continue
- Comando goto

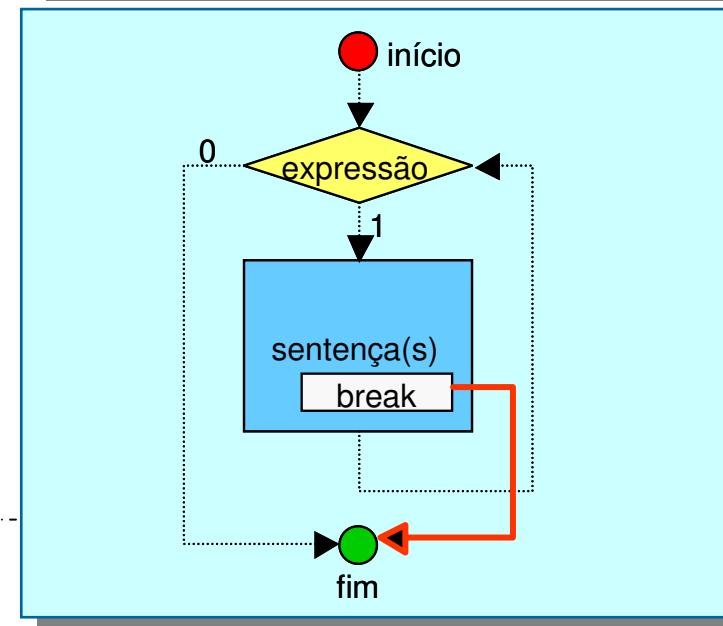
break

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

```

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

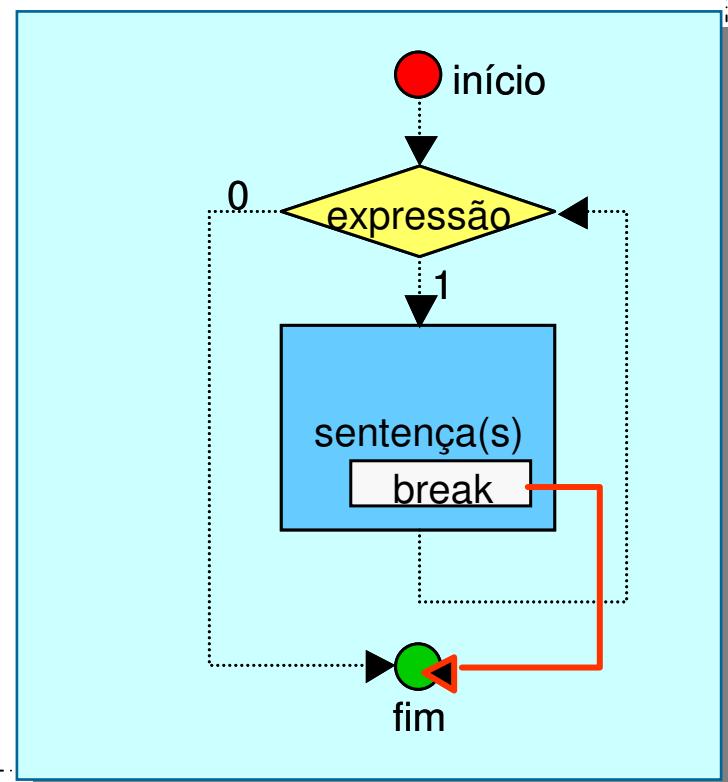


break

Sintaxe break com while

Sintaxe:

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

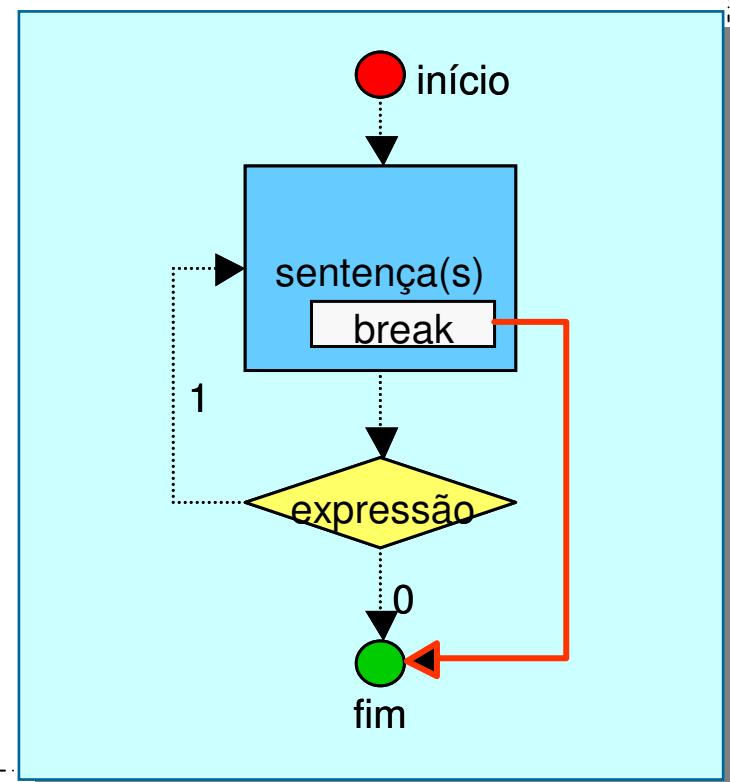


break

Sintaxe break com do...while

Sintaxe:

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

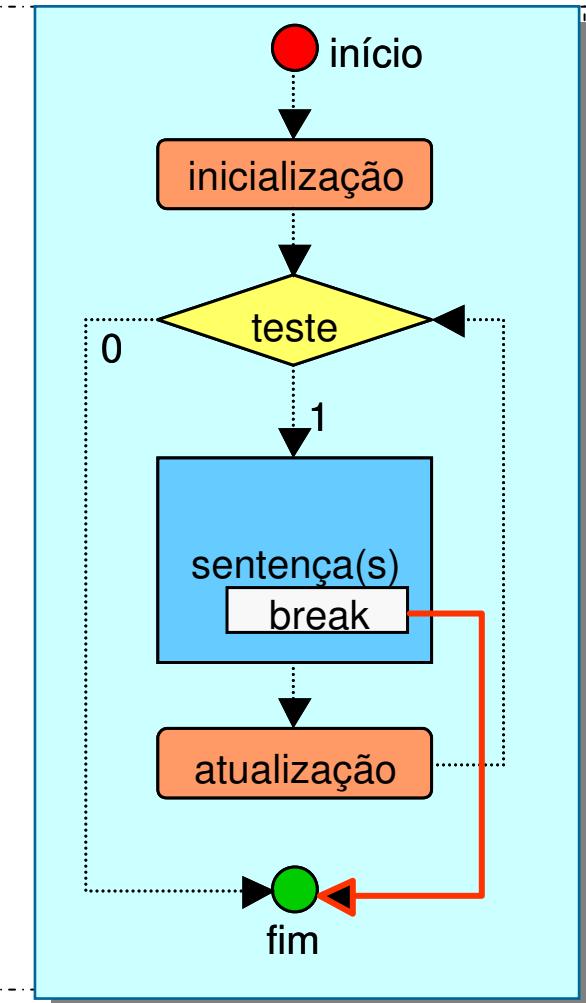


break

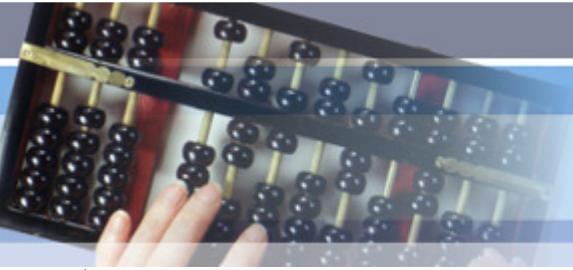
Sintaxe break com for ()

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);

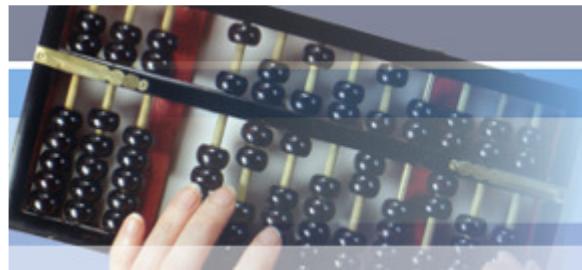
    numero_divisores = 0;
    for (divisor = 1; divisor <= numero; divisor++) {
        resto = numero % divisor;
        if (resto == 0) {
            numero_divisores++;
            if (numero_divisores >= 3) {
                break;
            }
        }
    }
    if (numero_divisores == 2) {
        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

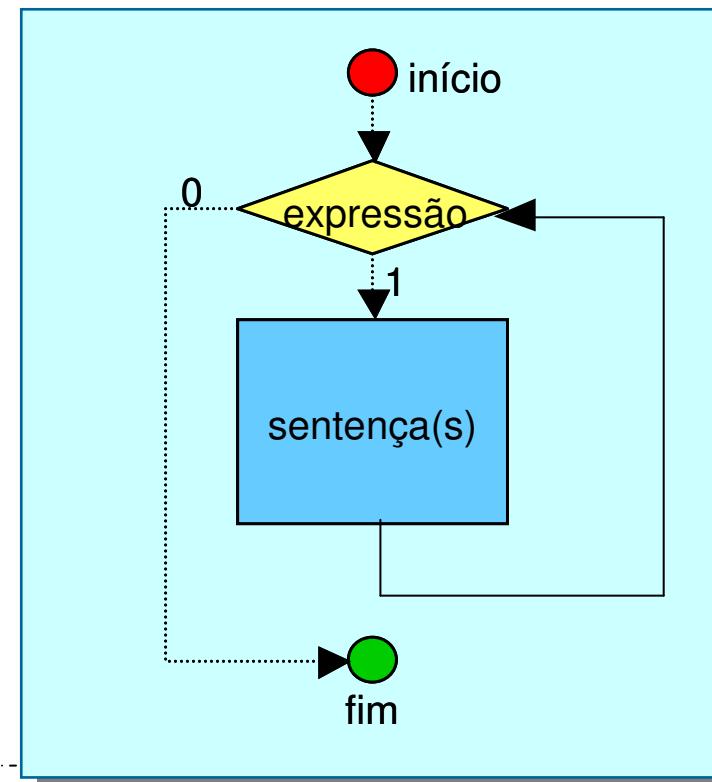


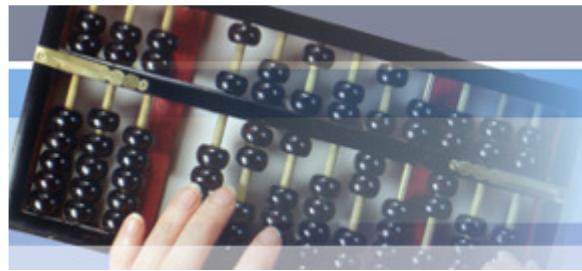
continue

Sintaxe continue com while

Sintaxe:

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



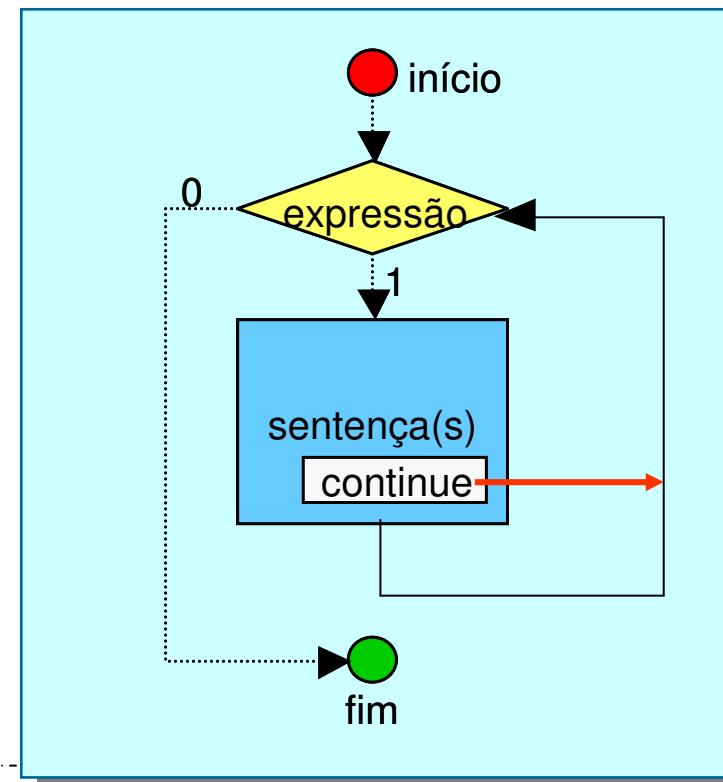


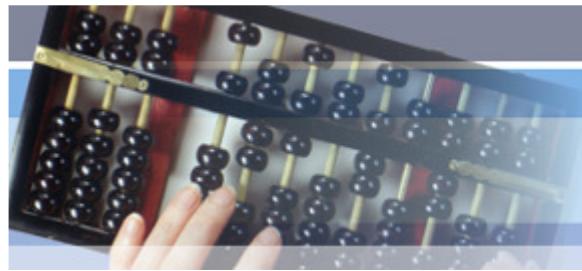
continue

Sintaxe continue com while

Sintaxe:

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



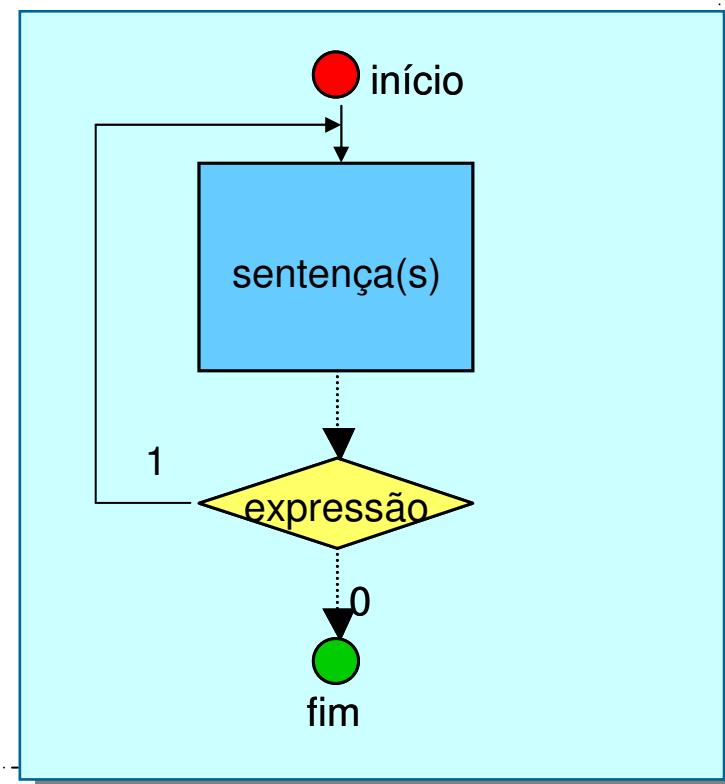


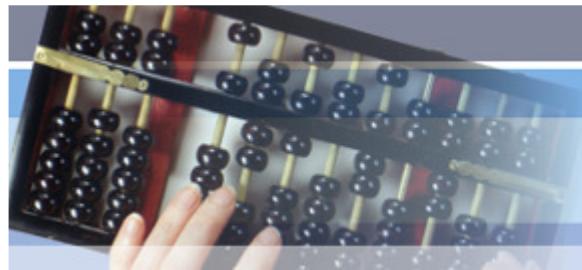
continue

Sintaxe continue com do...while

Sintaxe:

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



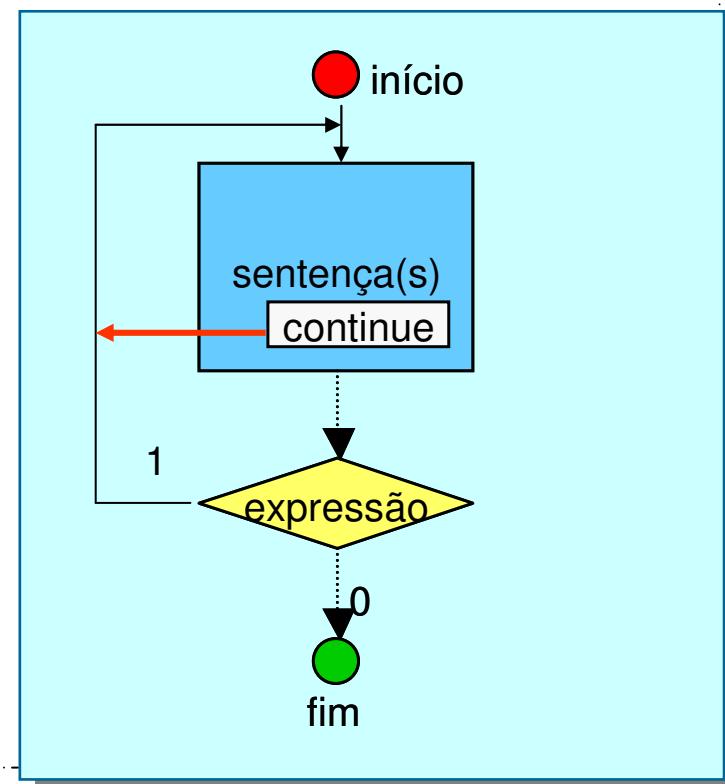


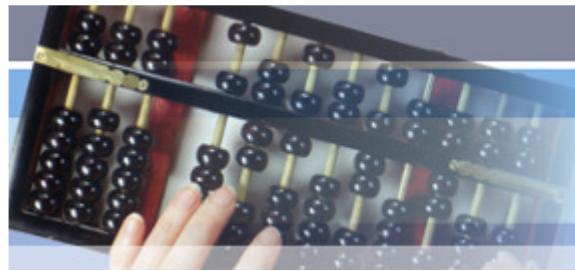
continue

Sintaxe continue com do...while

Sintaxe:

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



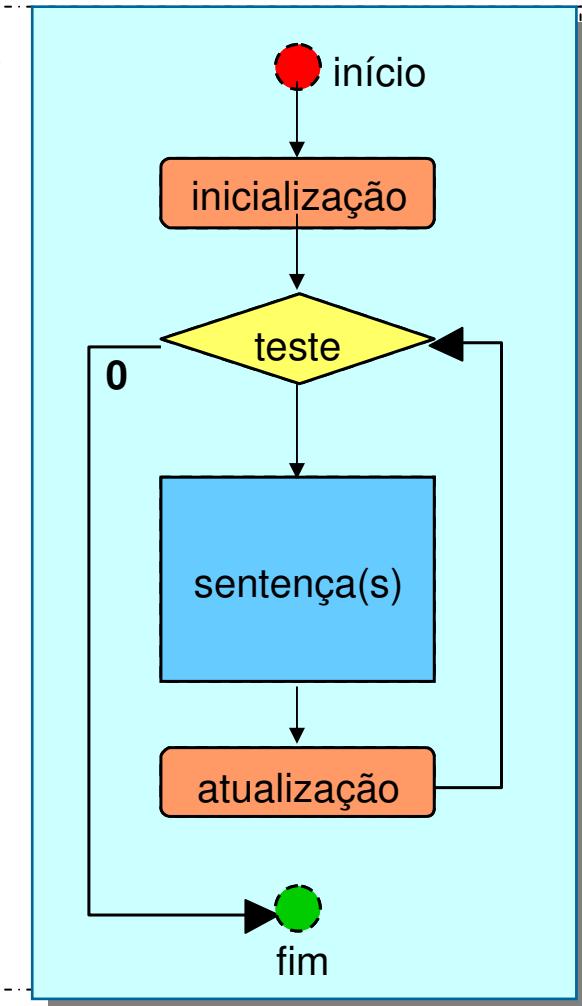


continue

Sintaxe continue com for

Sintaxe:

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



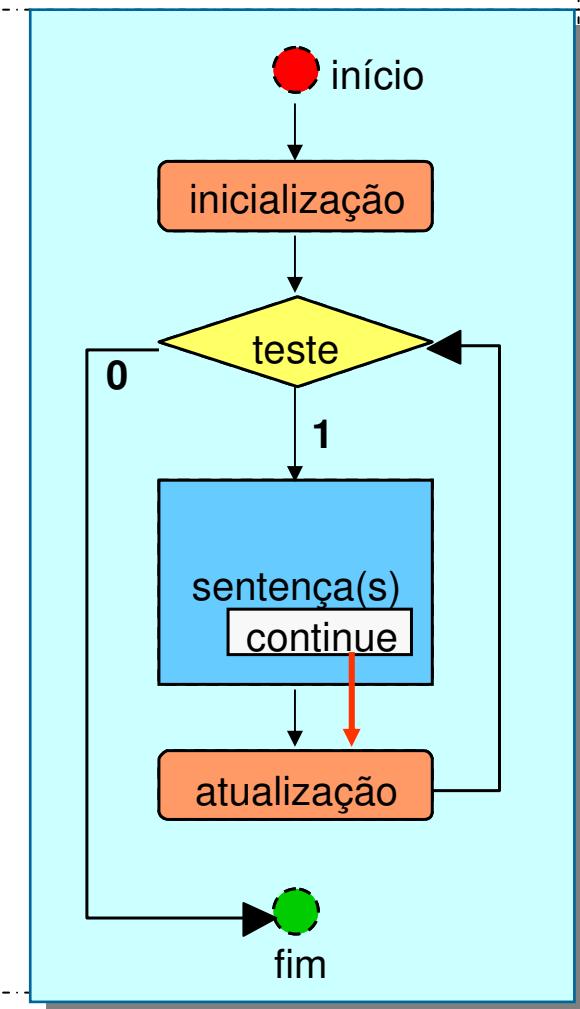


continue

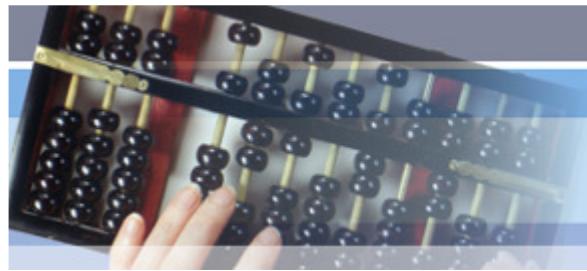
Sintaxe continue com for

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 = 3.1415926535897932384626433832795;

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

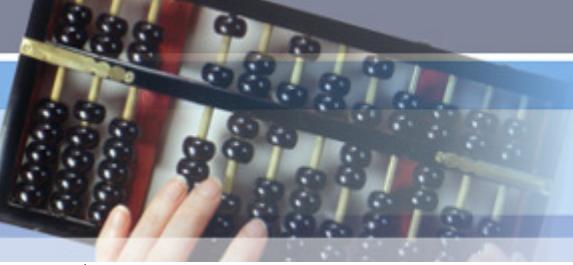
        tangente = tan((angulo/180)*pi);
        printf("tan(%8.2f)=%8.2f\n", angulo, tangente);
    }

    return 0;
}
```

ControleExecucao\Tangente01\Tangente01.vcproj

Controle de Execução

goto



goto

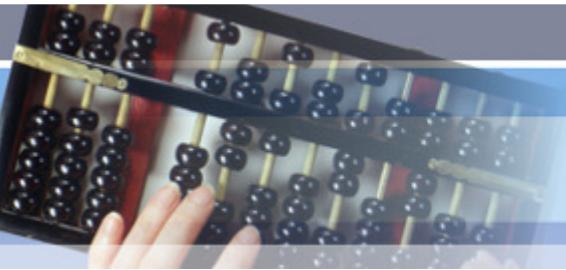
Objetivo do goto:

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

Exemplo:
Repetição infinita

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

goto



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

Estruturas de Repetição

Fim do Capítulo