Comandos de Repetição

Repetição por Condição

- Um conjunto de comandos de um algoritmo pode ser repetido quando subordinado a uma condição:

  enquanto *condição* faça

  comandos;

  fim enquanto

Repetição por Condição

- De acordo com a condição, os comandos serão repetidos zero (se falso) ou mais vezes (enquanto a condição for verdadeira).
- Essa estrutura normalmente é denominada ***laço*** ou ***loop***.

Repetição por Condição

- Condição
  - qualquer expressão que resulte em um valor do tipo lógico e pode envolver operadores aritméticos,lógicos, relacionais e resultados de funções.
  - Ex:

    x > 5

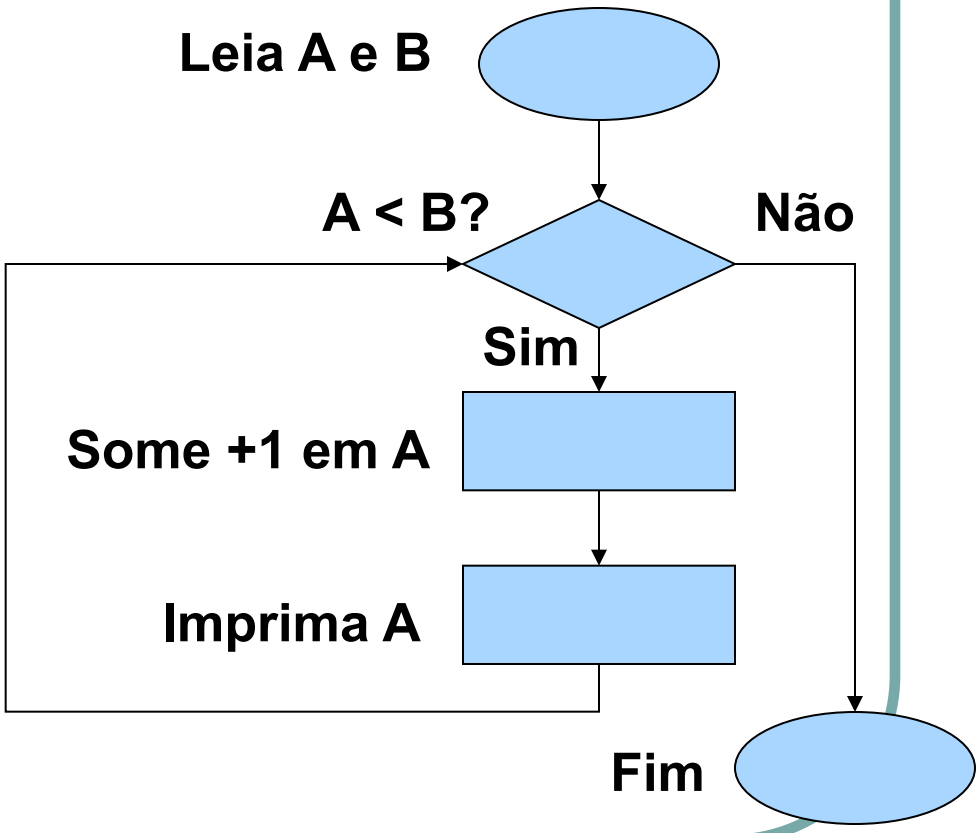
    (N < 60) && (N > 35)

Funcionamento

- A condição da cláusula ***enquanto*** é testada.
  - Se ela for verdadeira os comandos seguintes são executados em seqüência como em qualquer algoritmo, até a cláusula ***fim enquanto***.
  - O fluxo nesse ponto é desviado de volta para a cláusula ***enquanto*** e o processo se repete.
  - Se a condição for falsa (ou quando finalmente for), o fluxo do algoritmo é desviado para o primeiro comando após a cláusula ***fim enquanto***.

Repetição por Condição

- Relembrando em fluxogramas
  - Um processo pode ser repetido até atender ou não uma condição.



Exemplo – Pseudo-Código

```
Leia A;
Leia B;
Enquanto A < B
A recebe A + 1;
Imprima A;
Fim Enquanto
```

Loop Infinito

- Um loop ou laço infinito pode ocorrer quando cometemos algum erro
  - ao especificar a condição lógica que controla a repetição
  - ou por esquecer de algum comando dentro da iteração.

Loop Infinito

Exemplo: loop infinito (condição errônea)

```
01      X recebe 4;
02      enquanto (X < 5) faca
03          X recebe X - 1;
04          Imprima X;
05      fim enquanto
```

Exemplo: loop infinito (não muda valor)

```
01      X recebe 4;
02      enquanto (X < 5) faca
03         Imprima X;
04      fim enquanto
```

Exercício

- Escreva, em pseudo-código, o algoritmo para calcular a média de N números

Exercício

```
Leia n;
media recebe 0;
n1 recebe 0;
Enquanto (n1 < n)
    Leia x;
    media recebe media + x;
    n1 recebe n1 + 1;
Fim enquanto
Imprima media/n;
```

Comando while

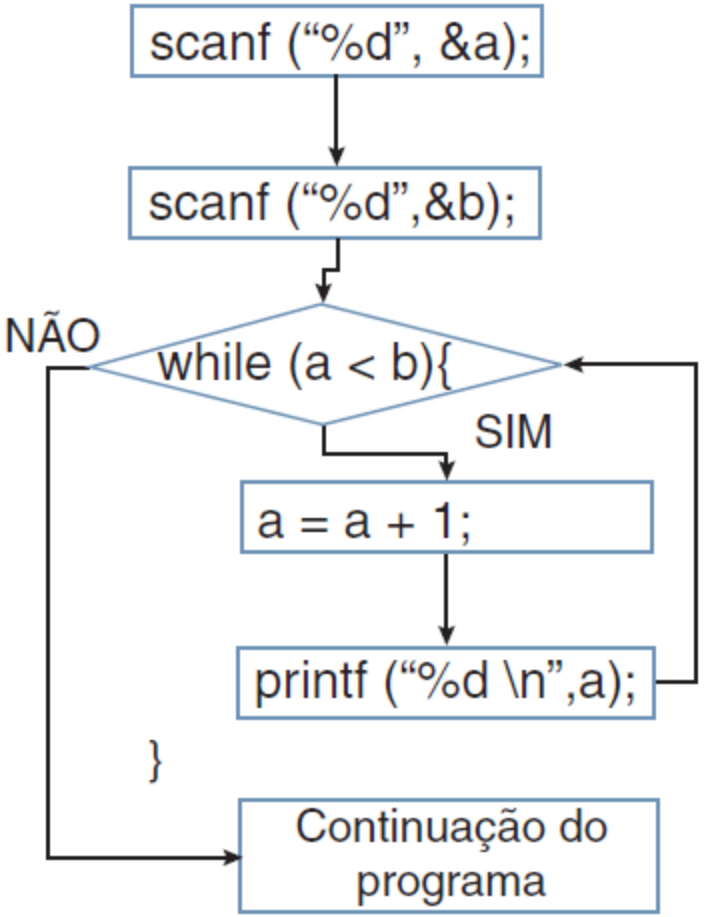
- Equivale ao comando “enquanto” utilizado nos pseudo-códigos.
  - Repete a seqüência de comandos enquanto a condição for verdadeira.
- Esse comando possui a seguinte forma geral:

```
while (condição) {
    seqüência de comandos;
}
```

Exemplo while

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int a,b;
05    printf("Digite o valor de a: ");
06    scanf("%d",&a);
07    printf("Digite o valor de b: ");
08    scanf("%d",&b);
09    while (a < b){
10       a = a + 1;
11       printf("%d \n",a);
12    }
13    system("pause");
14    return 0;
15  }
```

Exemplo while



Exercício

- Escreva, usando while, o algoritmo para calcular a média de N números

Exercício

```
int n,n1,x;
float media = 0;
printf(“Digite N:”);
scanf("%d",&n);
n1 = 0;
while (n1 < n){
    printf(“Digite X:”);
    scanf("%d",&x);
    media = media + x;
    n1 = n1 + 1;
}
printf("%f",media/n);
```

Comando for

- O loop ou laço for é usado para repetir um comando, ou bloco de comandos, diversas vezes
  - Maior controle sobre o loop.
- Sua forma geral é

```
for (inicialização; condição; incremento) {
    seqüência de comandos;
}
```

Comando for

1. inicialização: iniciar variáveis (contador).
2. condição: avalia a condição. Se verdadeiro, executa comandos do bloco, senão encerra laço.
3. incremento: ao término do bloco de comandos, incrementa o valor do contador
4. repete o processo até que a condição seja falsa.

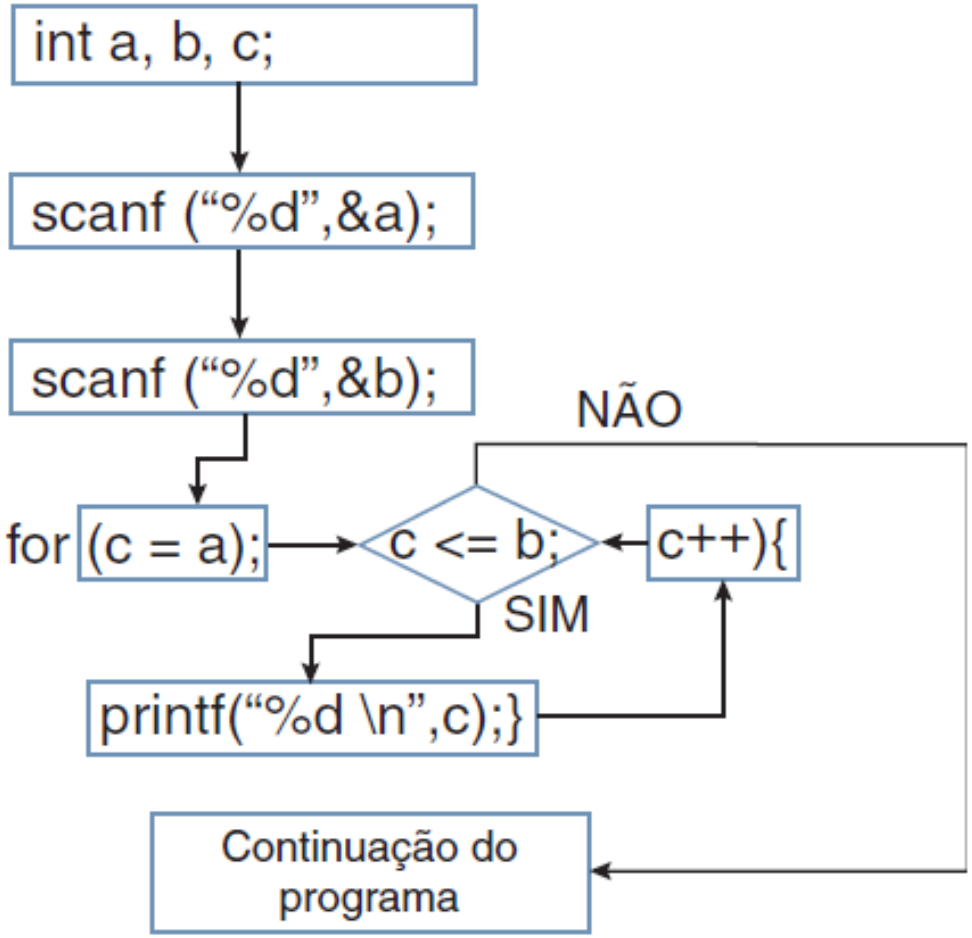
Comando for

- Comando while: repete uma seqüência de comandos enquanto uma condição for verdadeira.
- Comando for: repete uma seqüência de comandos “N vezes”.

Exemplo for

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int a,b,c;
05    printf("Digite o valor de a: ");
06    scanf("%d",&a);
07    printf("Digite o valor de b: ");
08    scanf("%d",&b);
09    for (c = a; c <= b; c++){
10       printf("%d \n",c);
11    }
12    system("pause");
13    return 0;
14  }
```

Exemplo for


int a, b, c;
scanf (“%d”,&a);
scanf (“%d”,&b);
NÃO
for (c = a);
c <= b;
c++){
SIM
printf(“%d \n”,c);}
Continuação do programa

for *versus* while

	for	while
01	`#include <stdio.h>`	`#include <stdio.h>`
02	`#include <stdlib.h>`	`#include <stdlib.h>`
03	`int main(){`	`int main(){`
04	`  int i, s = 0;`	`  int i, s = 0;`
05	`  for(i = 1; i <= 10; i++){`	`  i = 1`
06	`    s = s + i;`	`  while (i <= 10){`
07	`  }`	`    s = s + i;`
08	`  printf("Soma = %d \n",s);`	`    i++;`
09	`  system("pause");`	`  }`
10	`  return 0;`	`  printf("Soma = %d \n",s);`
11	`}`	`  system("pause");`
12		`  return 0;`
13		`}`

Comando for

- Podemos omitir qualquer um de seus elementos
  - inicialização, condição ou incremento.
- Ex.: **for** sem inicialização

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int a,b,c;
05    printf("Digite o valor de a: ");
06    scanf("%d",&a);
07    printf("Digite o valor de b: ");
08    scanf("%d",&b);
09    for (; a <= b; a++){
10       printf("%d \n",a);
11    }
12    system("pause");
13    return 0;
14  }
```

Comando for

- Cuidado: for sem condição
  - omitir a condição cria um laço infinito;
  - condição será sempre verdadeira.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int a,b,c;
05    printf("Digite o valor de a: ");
06    scanf("%d",&a);
07    printf("Digite o valor de b: ");
08    scanf("%d",&b);
09    //o comando for abaixo e um laco infinito
10    for (c = a; ; c++){
11       printf("%d \n",c);
12    }
13    system("pause");
14    return 0;
15  }
```

Comando for

- Cuidado: for sem incremento
  - omitir o incremento cria um laço infinito;
  - Incremento pode ser feito nos comandos.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int a,b,c;
05    printf("Digite o valor de a: ");
06    scanf("%d",&a);
07    printf("Digite o valor de b: ");
08    scanf("%d",&b);
09    for (c = a; c <= b; ){
10       printf("%d \n",c);
11       c++;
12    }
13    system("pause");
14    return 0;
15  }
```

Exercício

- Escreva, usando for, um algoritmo para calcular a soma dos elementos de 1 a 10.

Exercício

```
int n;
int soma = 0;
for (n = 1; n <= 10; n++){
   soma = soma + n;
}
printf("%d",soma);
```

Comando do-while

- Comando while: é utilizado para repetir um conjunto de comandos zero ou mais vezes.
- Comando do-while: é utilizado sempre que o bloco de comandos **deve ser executado ao menos uma vez.**

Comando do-while

- executa comandos
- avalia condição:
  - se verdadeiro, re-executa bloco de comandos
  - caso contrário, termina o laço

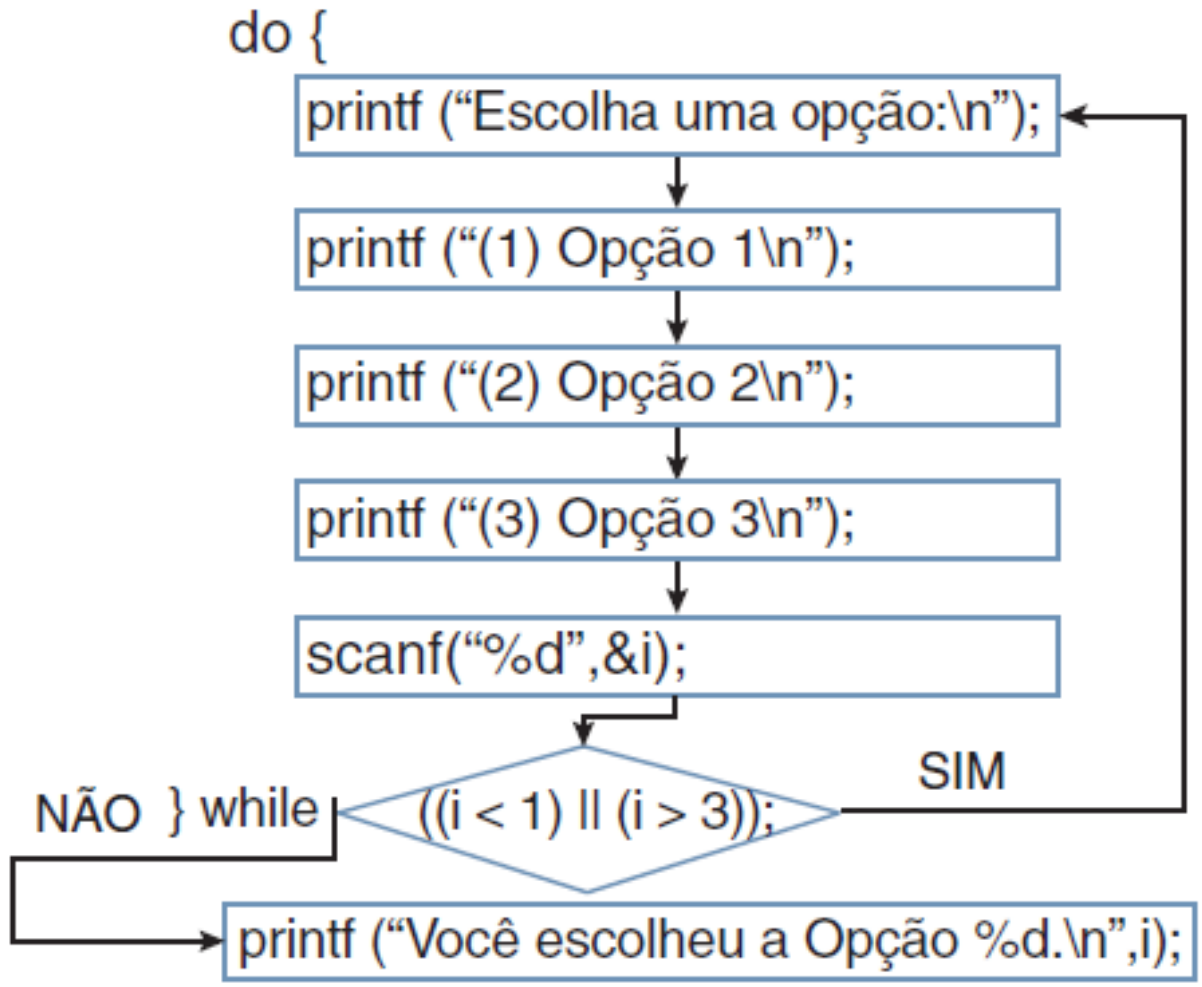
- Sua forma geral é (**sempre usa chaves!**)

```
do {
   comandos;
} while (condição);
```

Comando do-while

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int i;
05    do {
06        printf ("Escolha uma opcao:\n");
07        printf ("(1) Opção 1\n");
08        printf ("(2) Opção 2\n");
09        printf ("(3) Opção 3\n");
10        scanf("%d", &i);
11    } while ((i < 1) || (i > 3));
12    printf ("Voce escolheu a Opcao %d.\n",i);
13    system("pause");
14    return 0;
15  }
```

Comando do-while



Comando break

- Nós já vimos dois usos para o comando break: interrompendo os comandos switch. Ex.:

```
int num;
scanf(“%d”,&num);
switch(num) {
        case 0: printf(“Zero"); break;
        case 1: printf(“Um"); break;
}
```

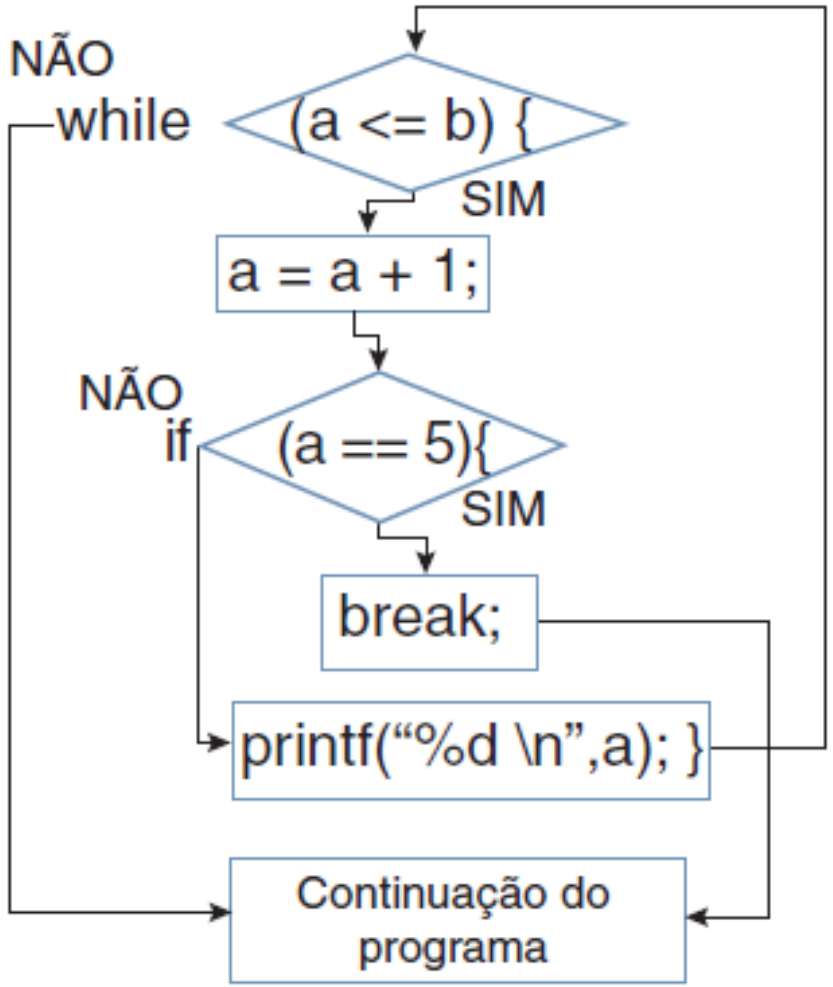
Comando break

- Na verdade, o comando **break** serve para quebrar a execução de um comando (como no caso do **switch**) ou interromper a execução de qualquer loop (**for**, **while** ou **do-while**).
- O **break** faz com que a execução do programa continue na primeira linha seguinte ao loop ou bloco que está sendo interrompido.

Comando break

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int a,b;
05    printf("Digite o valor de a: ");
06    scanf("%d",&a);
07    printf("Digite o valor de b: ");
08    scanf("%d",&b);
09    while (a <= b){
10       a = a + 1;
11       if(a == 5)
12            break;
13       printf("%d \n",a);
14    }
15    system("pause");
16    return 0;
17  }
```

Comando break



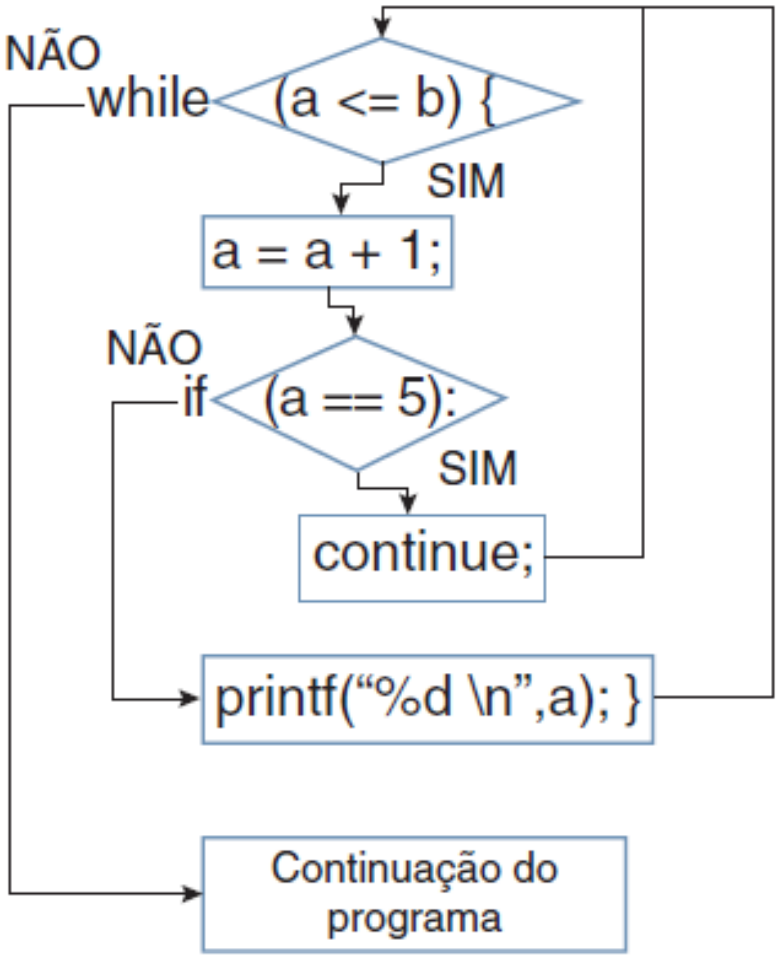
Comando continue

- Comando continue
  - Diferente do comando break, só funciona dentro do loop;
  - Pula essa iteração do loop;
  - Os comandos que sucedem o comando continue no bloco não são executados

Comando continue

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int a,b;
05    printf("Digite o valor de a: ");
06    scanf("%d",&a);
07    printf("Digite o valor de b: ");
08    scanf("%d",&b);
09    while (a <= b){
10       a = a + 1;
11       if(a == 5)
12            continue;
13       printf("%d \n",a);
14    }
15    system("pause");
16    return 0;
17  }
```

Comando continue



Goto e Label

- É um salto condicional para um local especificado.
- Este local é determinado por um rótulo (label).
  - Este local pode ser a frente ou atrás no programa, mas deve ser dentro da mesma função.

Goto e Label

- apesar de banido da prática de programação, *goto's* podem ser úteis em determinadas circunstâncias.
  - Ex: sair de dentro de laços aninhados.

- Forma geral:

  palavra_chave:

  goto palavra_chave;

Goto e Label

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int i,j,k;
05    for(i = 0; i < 5; i++)
06      for(j = 0; j < 5; j++)
07        for(k = 0; k < 5; k++)
08          if(i == 2 && j == 3 && k == 1)
09            goto fim;
10          else
11            printf("Posicao [%d,%d,%d]\n",i,j,k);
12
13    fim:
14    printf("Fim do programa\n");
15
16    system("pause");
17    return 0;
18  }
```

Exemplo - Teste de primalidade

- Definição: um número é primo se ele SOMENTE for divisível por 1 e por ele mesmo.
- Para identificar um número P como primo, desta forma, devemos verificar se os números de 2 a P-1 não geram resto na divisão

Exemplo - teste de primalidade

- Iremos iterar sobre os valores de 2 a p-1
- Precisamos guardar, de alguma forma, que o número é primo ou não
  - variável booleana – verdadeiro se é primo, falso senão
  - Se o resto der zero, marcar como falso.
- Como seria esse programa em C?

Exemplo - teste de primalidade

```
int primo(int p) {
1.  int ehprimo = 1;
2.  for(int i=2; i<p; i++) {
3.      if(p % i == 0) {
4.          ehprimo = 0;
5.      }
6.  }
7.  return ehprimo;
}
```

Exemplo – teste de primalidade

- Podemos interromper o laço e retornar indicando que o número não é primo ao achar o primeiro divisor

- Instrução **break**

Exemplo – teste de primalidade

```
int primo(int p) {
1.   int ehprimo = 1;
2.   for(int i=2; i<p; i++) {
3.      if(p % i == 0) {
4.         ehprimo = 0;
5.         break;
6.      }
7.   }
8.   return ehprimo;
}
```

Exemplo – teste de primalidade

- Também podemos mudar a condição de permanência do laço. Como seria esse algoritmo?

Exemplo – teste de primalidade

```
int primo(int p) {
1.   int ehprimo = 1;
2.   for(int i=2; i<p && ehprimo ==1; i++) {
3.       if(p % i == 0) {
4.           ehprimo = 0;
5.       }
6.   }
7.   return ehprimo;
}
```

Laços aninhados

- Podemos criar laços um dentro do outro.
- Exemplo: Imprimir todos os números primos de 2 a 100.

Exemplo 3: primos de 2 a 100

```
int primos() {
1.   for(int p=2; p<=100; p++)
2.   {
3.       ehprimo = 1;
4.       for(int i=2; i<p && ehprimo ==1; i++)
5.       {
6.           if(p % i == 0)
7.               ehprimo = 0;
8.       }
9.       if(ehprimo == 0) printf("%d nao eh primo\n",p);
10.      else printf("%d eh primo\n",p);
11.  }
}
```

Exemplo: máximo divisor comum (MDC)

- Algoritmo de Euclides
  - Dois números n1 e n2
  - Enquanto a divisão de n1 por n2 não for exata:
    - Novo n1 é o valor inteiro da divisão de n1 por n2
    - Novo n2 é o resto da divisão de n1 por n2
  - O divisor será o MDC

MDC entre 42 e 24:

Passo	x	y	r
1	42	24	18
2	24	18	6
3	18	6	0

x = 42, y = 24
Na etapa seguinte o y passa
a ser x e o resto passa a ser y.
O processo se repete até que o resto
da divisão seja 0. e o valor em y é o MDC desejado

MDC entre 42 e 23:

Passo	x	y	r
1	42	23	19
2	23	19	4
3	19	4	3
4	4	3	1
5	3	1	0

Exemplo: máximo divisor comum (MDC)

```
1. int mdc(int x, int y) {
2.    int r = x % y;
3.    while(r != 0) {
4.       x = y;
5.       y = r;
6.       r = x%y;
7.    }
8.    return y;
9. }
```

Primeiro primo depois de 11

- Queremos calcular o primeiro primo que vem em seguida de 11.
- Como fazer isso? ideias?

Primeiro primo depois de 11

```
1.  int main() {
2.      int valor = 12;
3.      primo = 0;
4.      while(primo == 0) {
5.          primo = 1;
6.          for(int x = 2; x<valor&&primo==1; x++)
7.              if(valor % x == 0)
8.                  primo = 0;
9.           if(primo == 0)
10.              valor++;
11.      }
12.      printf(“Primeiro primo: %d\n”,valor);
13. }
```

Primeiro primo depois de 11

Nâo coloquei abre e fecha colchetes para fechar bloco. Consequência: Bloco possui só uma instrução

```
1.  int main() {
2.      int valor = 12;
3.      primo = 0;
4.      while(primo == 0) {
5.          primo = 1;
6.          for(int x = 2; x<valor && primo==1; x++)
7.              if(valor % x == 0)
8.                  primo = 0;
9.           if(primo == 0)
10.              valor++;
11.      }
12.      printf("Primeiro primo: %d\n",valor);
13. }
```

Primeiro primo depois de 11

Nâo coloquei abre e fecha colchetes para fechar bloco. Consequência: Bloco possui só uma instrução. Código equivalente:

```
1.  int main() {
2.      int valor = 12;
3.      primo = 0;
4.      while(primo == 0) {
5.          primo = 1;
6.          for(int x = 2; x<valor && primo==1; x++) {
7.              if(valor % x == 0) {
8.                  primo = 0;
9.              }
10.         }
11.         if(primo == 0) {
12.             valor++;
13.         }
14.     }
15.     printf("Primeiro primo: %d\n,valor);
16. }
```