

# Linguagem de Programação I

**- Comandos de repetição e decisão -**

**Prof. Ulysses Santos Sousa**  
**[ulyssessousa@ifma.edu.br](mailto:ulyssessousa@ifma.edu.br)**

**Aula 03**

# Roteiro

- **Comandos de repetição**
  - while
  - do – while
  - for
- **Comandos de seleção**
  - if / else
  - switch

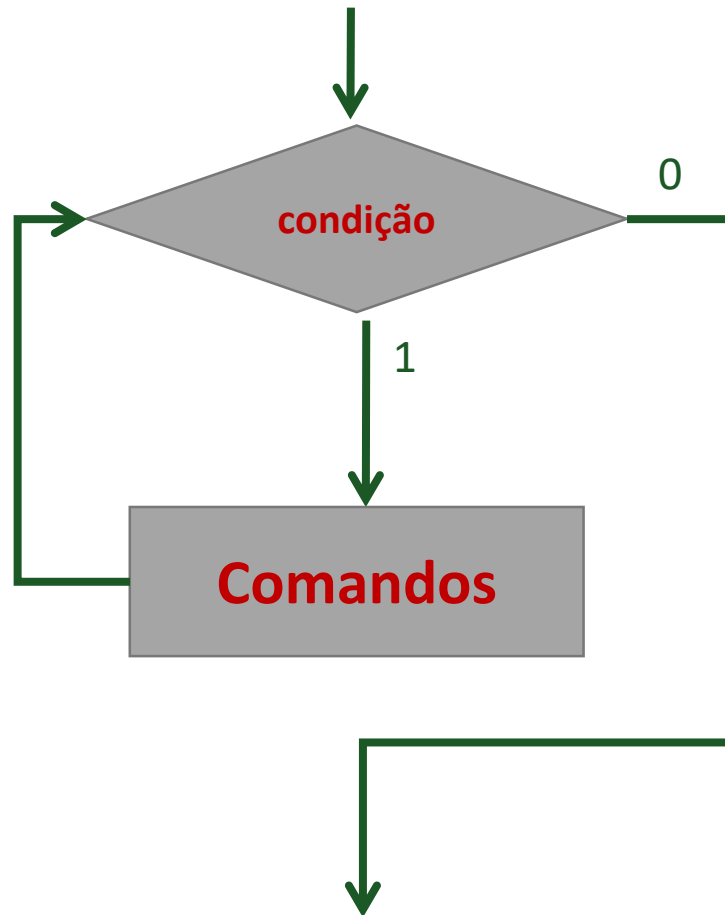
# Comandos de repetição

- Também chamados de laços ou *loop*.
- São utilizados sempre que uma ou mais instruções tiverem que ser repetidas enquanto uma determinada condição for verdadeira.
- Comandos:
  - for
  - while
  - do-while

# Comando *while*

- Significa *enquanto* e utiliza os mesmos elementos do laço *for* (inicialização, teste e incremento).
- Geralmente é utilizado quando o laço pode ser terminado inesperadamente, por condições desenvolvidas dentro do corpo do laço.

# Comando *while*



# Comando *while*

- Repetição controlada por contador

Essa técnica usa uma variável chamada **contador** para especificar o número de vezes que um conjunto de comandos será executado.

- Exemplo

Uma turma de 10 alunos realiza um teste. As notas (reais de 0 a 10) dadas aos alunos estão à disposição. Determine a média das notas da turma.

# Comando *while*

- Repetição controlada por contador
  - Sintaxe:

```
Inicialização;  
...  
while (teste){  
    ...  
    incremento;  
    ...  
}
```

# Comando *while*

- Exemplo:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int cont;
7      cont = 0;
8      while(cont < 10) {
9          printf("%d\n", cont);
10         cont++;
11     }
12     return 0;
13 }
```

Inicialização

Teste

Incremento



# Comando *while*

- Repetição controlada por contador

Variável  
contadora →

Operador de  
incremento →

O cálculo da  
média é feito após  
o encerramento  
do loop →

```
1  #include <stdio.h>
2
3  int main(){
4      int contador;
5      float nota, somaNotas, mediaTurma;
6
7      somaNotas = 0; ← Variável acumuladora
8      contador = 1;
9
10     while(contador <= 10){
11         printf("Digite a nota: ");
12         scanf("%f", &nota);
13         somaNotas += nota; ← Operador de
14         contador++;         atribuição aritmética
15     }
16
17     mediaTurma = somaNotas/10;
18
19     printf("\nA media da turma eh %.2f\n", mediaTurma);
20     return 0;
21 }
```

# Comando *while*

- Repetição controlada por contador

Este comando  
inicializa a  
variável  
contadora →

A variável  
contadora  
precisa ser  
incrementada →

```
1  #include <stdio.h>
2
3  int main(){
4      int contador;
5      float nota, somaNotas, mediaTurma;
6
7      somaNotas = 0;
8      contador = 1;
9
10     while(contador <= 10){
11         printf("Digite a nota: ");
12         scanf("%f", &nota);
13         somaNotas += nota;
14         contador++;
15     }
16
17     mediaTurma = somaNotas/10;
18
19     printf("\nA media da turma eh %.2f\n", mediaTurma);
20     return 0;
21 }
```

# Comando *while*

- Repetição controlada por contador

Variável  
acumuladora  
é usada para  
armazenar  
uma soma ou  
produto →

Deve sempre ser  
inicializada fora do  
bloco de repetição  
em que é utilizada

→  
Aqui é feita a  
soma das notas

```
1  #include <stdio.h>
2
3  int main(){
4      int contador;
5      float nota, somaNotas, mediaTurma;
6
7      somaNotas = 0;
8      contador = 1;
9
10     while(contador <= 10){
11         printf("Digite a nota: ");
12         scanf("%f", &nota);
13         somaNotas += nota;
14         contador++;
15     }
16
17     mediaTurma = somaNotas/10;
18
19     printf("\nA media da turma eh %.2f\n", mediaTurma);
20     return 0;
21 }
```

# Comando *while*

- Repetição controlada por contador

Após a  
execução do  
loop a variável  
acumuladora  
contém a  
soma das  
notas →

```
1  #include <stdio.h>
2
3  int main(){
4      int contador;
5      float nota, somaNotas, mediaTurma;
6
7      somaNotas = 0;
8      contador = 1;
9
10     while(contador <= 10){
11         printf("Digite a nota: ");
12         scanf("%f", &nota);
13         somaNotas += nota;
14         contador++;
15     }
16
17     mediaTurma = somaNotas/10;
18
19     printf("\nA media da turma eh %.2f\n", mediaTurma);
20     return 0;
21 }
```

# Comando *while*

- Exemplo de laço sem número de repetições definidas:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4
5  int main()
6  {
7      char x;
8      while((x = getch()) != 's'){
9          printf("%c\n", x);
10     }
11     return 0;
12 }
```

# Comando *while*

- Repetição controlada por sentinela

Essa técnica usa uma variável chamada **sentinela** ou **flag** para indicar o fim da inserção de dados.

- Este tipo de repetição é dito **indefinida** porque o número de repetições não é conhecido antes de o loop ser executado.

- Exemplo

Ler vários inteiros até que seja inserido o valor zero, calcular e mostrar a soma dos números digitados.

# Comando *while*

- Repetição controlada por sentinela

Podemos  
inicializar uma  
variável ao  
declará-la →

O loop é  
executado até  
que seja  
informado o →  
valor zero para  
a variável  
numero

```
1  #include <stdio.h>
2
3  int main() {
4      int numero, soma = 0;
5
6      printf("Informe um numero: ");
7      scanf("%d", &numero);
8
9      while(numero != 0) {
10         soma += numero;
11         printf("Informe um numero: ");
12         scanf("%d", &numero);
13     }
14
15     printf("\n A soma eh %d", soma);
16     return 0;
17 }
```

# Comando *while*

- Repetição controlada por sentinela

**Erro de Inicialização**

É necessário inicializar a variável da condição antes do laço while →

**Loop Infinito**

É necessário →  
modificar a variável da condição dentro do laço while

```
1  #include <stdio.h>
2
3  int main() {
4      int numero, soma = 0;
5
6      printf("Informe um numero: ");
7      scanf("%d", &numero);
8
9      while(numero != 0) {
10         soma += numero;
11         printf("Informe um numero: ");
12         scanf("%d", &numero);
13     }
14
15     printf("\n A soma eh %d", soma);
16     return 0;
17 }
```



# Comando *do-while*

- Bastante parecido com o laço *while*.
- É utilizado quando é necessário executar os comandos do laço uma primeira vez e depois avaliar a expressão de teste.
- Sintaxe:

```
Inicialização;  
...  
do{  
    ...  
    incremento;  
    ...  
} while (teste);
```

# Comando *do-while*

- Quando utilizar?
  - Quando os comandos do corpo do laço precisam ser executados pelo menos uma vez.

# Comando *for*

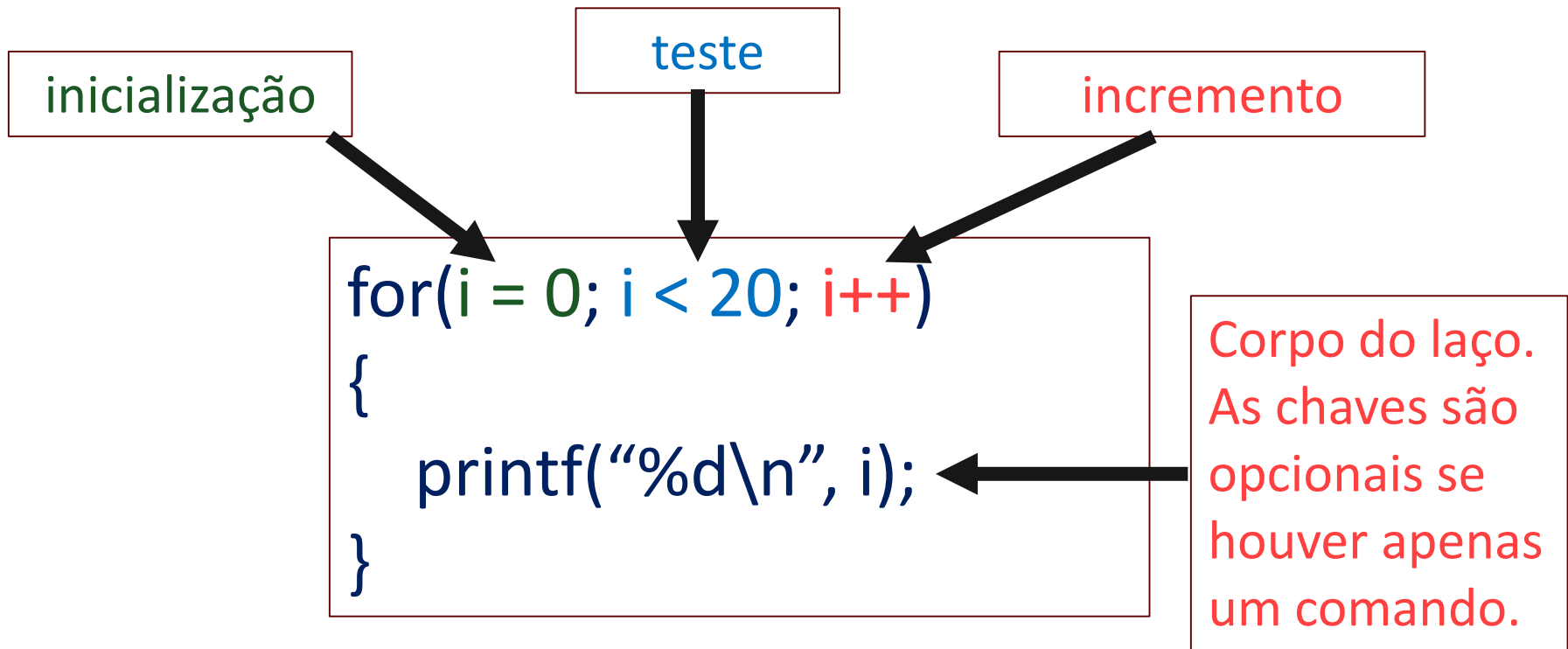
- Geralmente é utilizado quando já se sabe o número de vezes que o conjunto de instruções será repetido.

- Sintaxe:

```
for(inicialização; teste; incremento)
{
    <comandos>
}
```

# Comando *for*

- Exemplo 1:



# Comando *for*

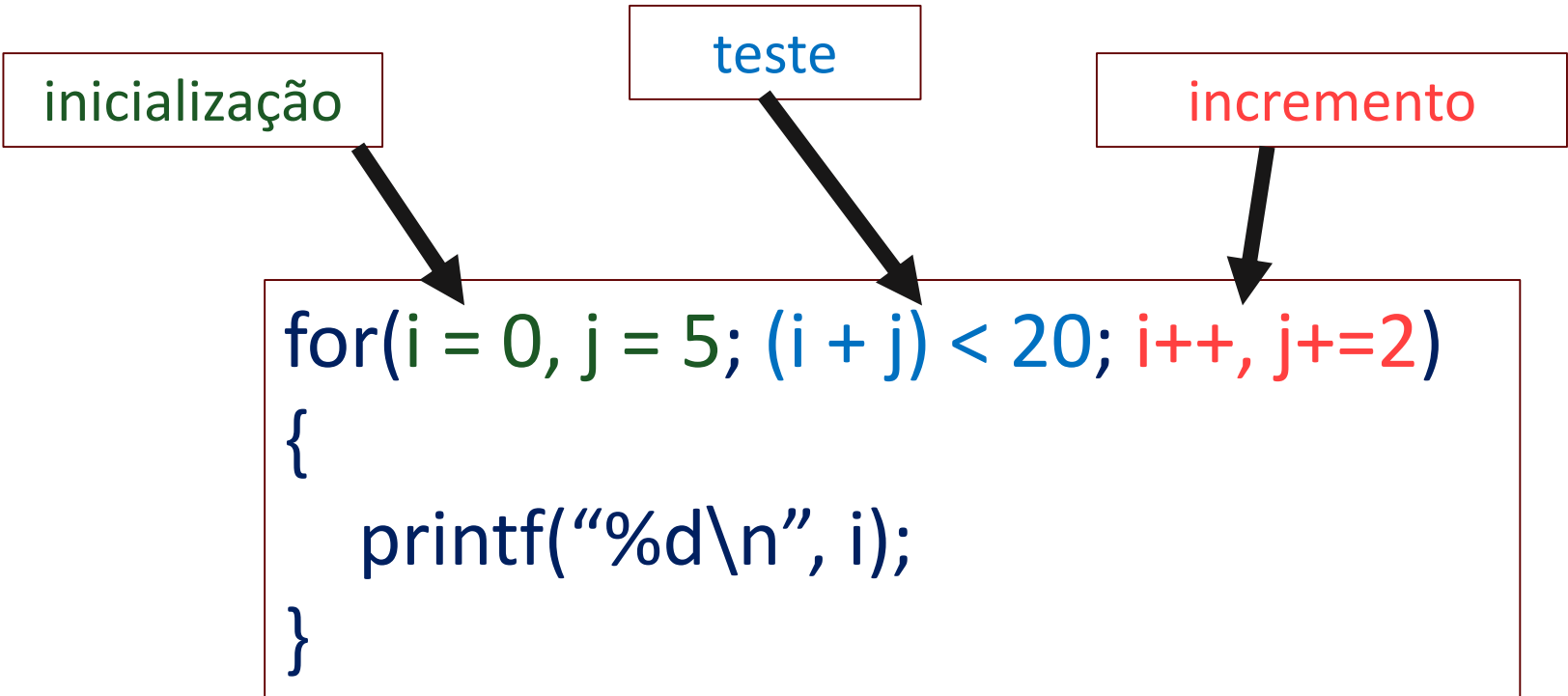
- Exemplo 2:

inicialização

teste

incremento

```
for(i = 0, j = 5; (i + j) < 20; i++, j+=2)
{
    printf("%d\n", i);
}
```



# Comando *for*

- **Omitindo expressões do laço *for***
  - Qualquer uma das três expressões pode ser omitidas, porém os ponto-e-vírgulas devem permanecer.
    - Se a expressão de **incremento** estiver omitida, será desconsiderada.
    - Se a expressão de **teste** não estiver presente, a condição será sempre verdadeira.

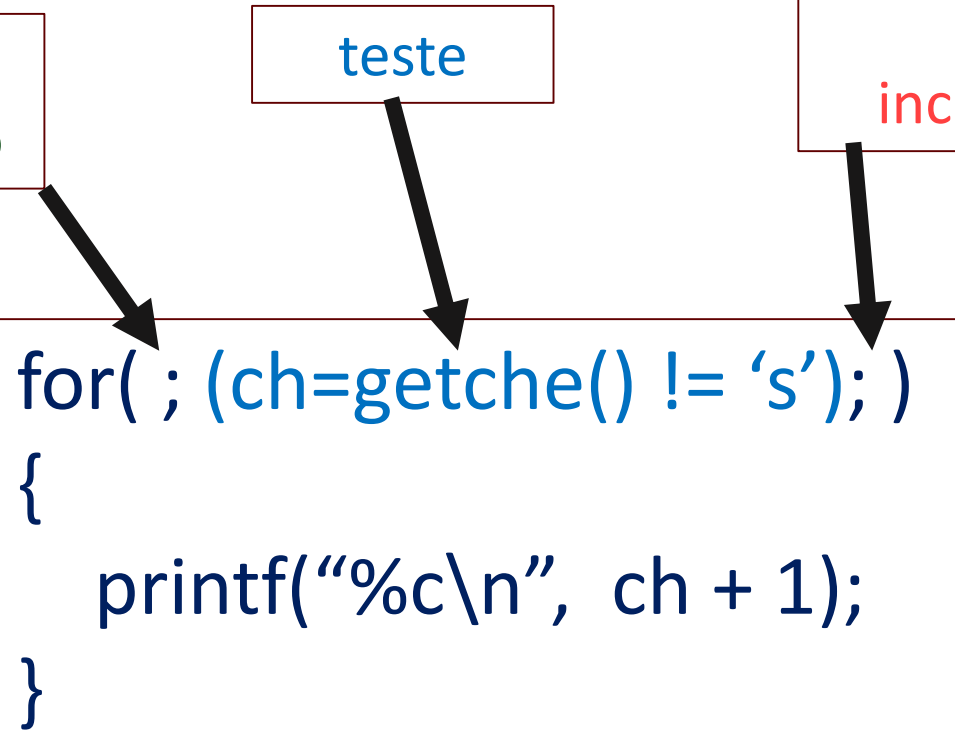
# Comando *for*

- Exemplo 3:

Sem  
inicialização

teste

Sem  
incremento



```
for( ; (ch=getche() != 's'); )  
{  
    printf("%c\n", ch + 1);  
}
```

# Comando *for*

- Laço infinito

```
for(;;)
{
    printf("loop infinito\n");
}
```



# Comando *for*

- **Comandos aninhados**

- Ocorre quando um laço for faz parte do corpo de outro laço for.

- **Exemplo:**

```
for(i = 0; i < 10; i++)  
{  
    for (j = 0; j < 5; j++){  
        printf("%d\n", i + j);  
    }  
}
```

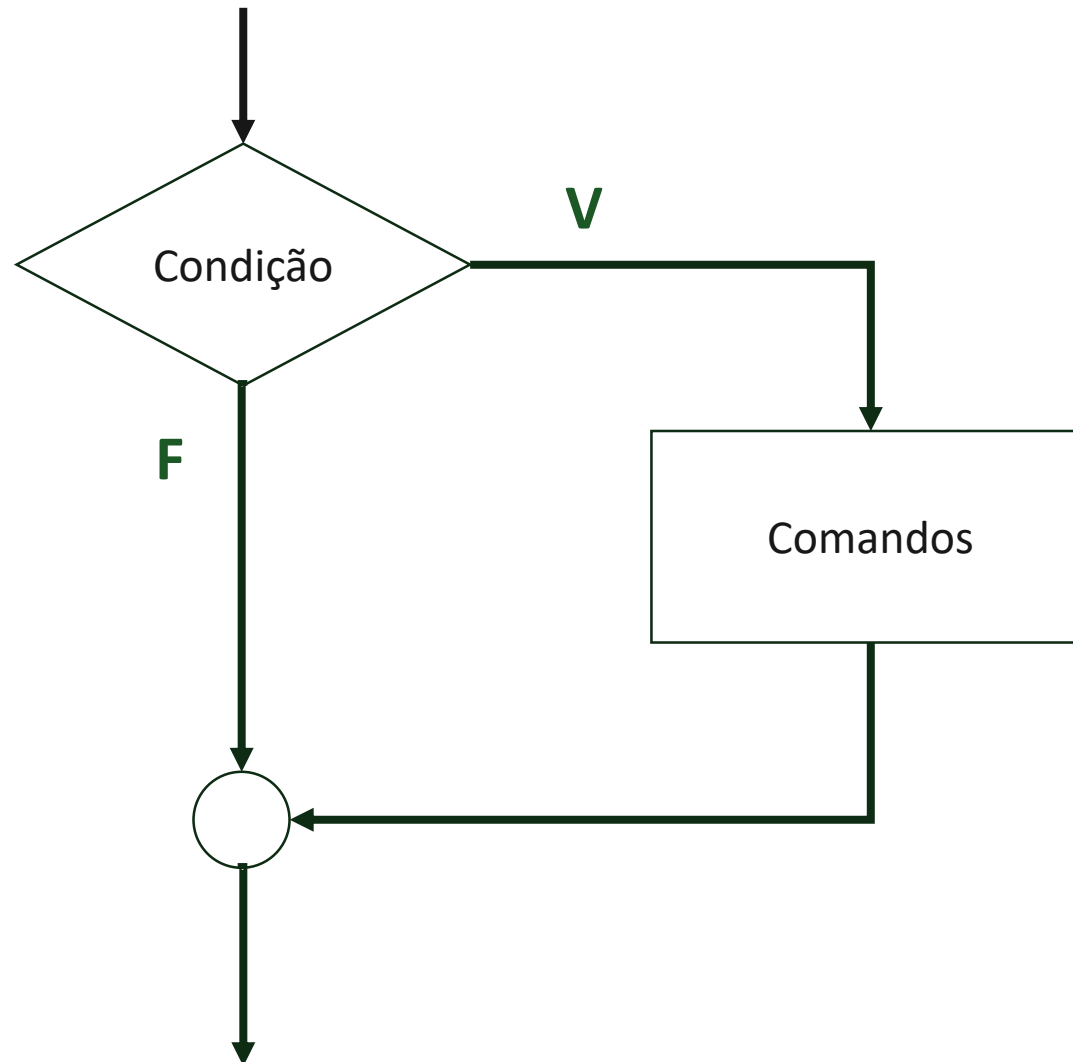
# Comandos de Decisão

- Permitem determinar qual é a ação a ser tomada com base no resultado de uma expressão condicional.
- Comando de decisão da Linguagem C:
  - if
  - if-else
  - switch

# Comando *if*

- Consiste da palavra *if* seguida de uma expressão de teste entre parênteses.
- Se a instrução for verdadeira, os comandos do corpo do *if* serão executados; caso contrário, nada será feito.

# Comando *if*



# Comando *if*

- Sintaxe:

```
if (expressão de teste)
{
    Instruções;
}
```

# Comando *if*

- Exemplo:

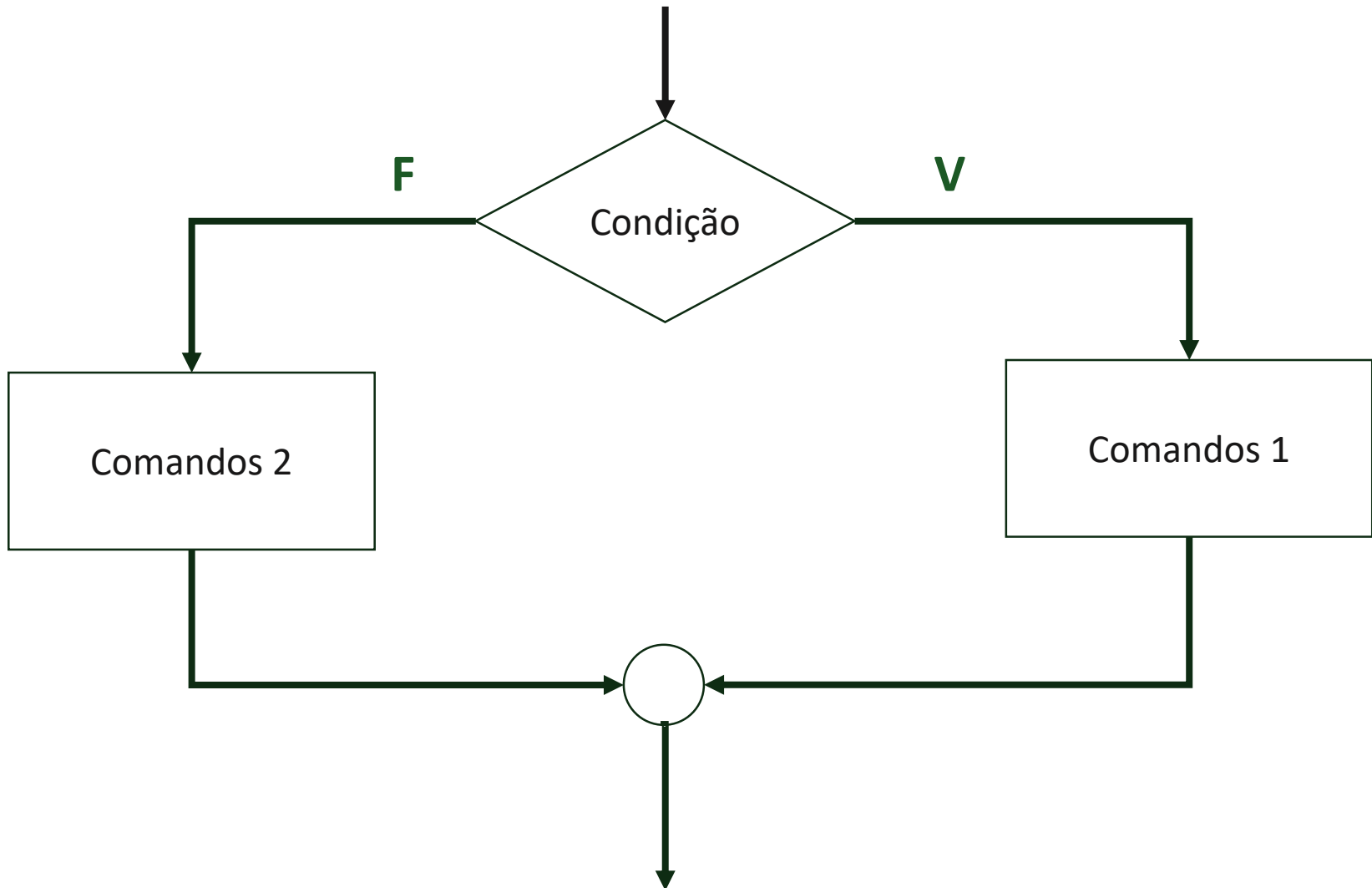
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  #include <conio.h>
5
6  int main()
7  {
8      char x;
9      int quantLetras = 0;
10     while((x = getche()) != '\r'){
11         if (isalpha(x))
12             quantLetras++;
13     }
14     printf("A quantidade de letras digitadas foi %d.\n", quantLetras);
15     return 0;
16 }
```

# Comando *if-else*

- Utilizado quando um conjunto de instruções deve ser executado se a expressão de teste do comando *if* for falsa.
- Sintaxe:

```
if (expressão de teste){  
    Conjunto de Instruções 1;  
}else{  
    Conjunto de Instruções 2;  
}
```

# Comando *if-else*





# Comando *if-else*

- Exemplo:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  #include <conio.h>
5
6  int main()
7  {
8      char x;
9      int quantLetras = 0;
10     while((x = getche()) != '\r'){
11         if (isalpha(x))
12             quantLetras++;
13     }
14     if (quantLetras > 0)
15         printf("Foram digitadas %d letras.\n", quantLetras);
16     else
17         printf("Nenhuma letra foi digitada.\n");
18     return 0;
19 }
```

# Comando *if/else/if*

- Permite a avaliação de várias condições.
- Os comandos inseridos imediatamente após cada condição só serão executados caso sua condição seja verdadeira.
- Com os comando if-else-if o código será composto de várias condições e para cada condição será estipulado um bloco de comando.

# Comando *if/else/if*

- Sintaxe:

```
if (condição1)
{
    <bloco de instruções>
}
else if (condição2)
{
    <bloco de instruções>
}
...
else
{
    <bloco de instruções>
}
```

# Comando *if/else/if*

- Exemplo:

```
if (media >= 7)
{
    printf("Aprovado\n");
}
else if (media >= 4)
{
    printf("Recuperação\n");
}
else
{
    printf("Reprovado\n");
}
```

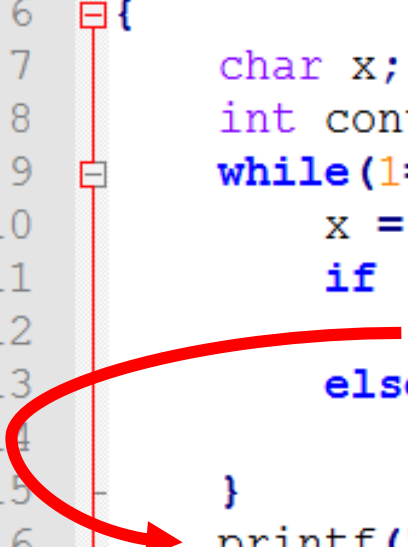
# Comandos *break* e *continue*

- São instruções que devem pertencer ao corpo do laço *for*, *while* ou *do-while*.
  - O comando *break* também pode ser utilizado com o comando *switch*.
- Comando *break*
  - Causa a saída imediata de um laço.
  - Se a instrução pertencer a um conjunto de laços aninhados, afetará somente o laço ao qual pertence e os laços internos a ele.

# Comandos *break* e *continue*

- Comando *break*

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4
5  int main()
6  {
7      char x;
8      int contador = 0;
9      while(1==1){
10         x = getche();
11         if (x == 's')
12             break;
13         else
14             contador++;
15     }
16     printf("\ncontador = %d\n", contador);
17     return 0;
18 }
```



# Comandos *break* e *continue*


- Comando *continue*

- Força a próxima iteração do laço e pula o código que estiver abaixo.
- Nos laços *while* e *do-while*, faz com que o controle do programa avalie imediatamente a expressão de teste e depois continua o processo do laço.
- No laço *for*, é executada a expressão de incremento e, em seguida, o teste.

# Comandos *break* e *continue*

- Comando *continue*

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int n, soma=0, contador = 0;
7      printf("Digite cinco numeros naturais: ");
8      do{
9          scanf("%d", &n);
10         if (n < 0){
11             printf("Numero invalido!\n");
12             continue;
13         }
14         soma += n;
15         contador++;
16     }while(contador < 5);
17     printf("\nSoma = %d\n", soma);
18     return 0;
19 }
```





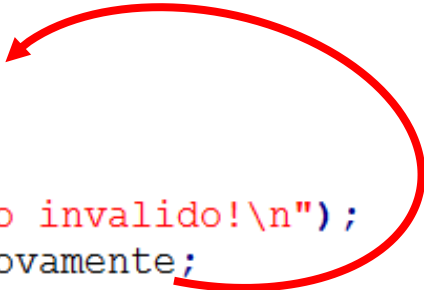
# Comando *goto*

- Está disponível em C para fornecer alguma compatibilidade com outras linguagens de programação, mas sua utilização é desaconselhada.
- Causa o desvio do controle do programa para a instrução seguinte ao rótulo com o nome indicado.
- Um rótulo é um nome seguido de dois pontos (:)

# Comando *goto*

- Exemplo:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int n, soma=0, contador = 0;
7      printf("Digite cinco numeros naturais: ");
8      do{
9          comeca_novamente:
10         scanf("%d", &n);
11         if (n < 0){
12             printf("Numero invalido!\n");
13             goto comeca_novamente;
14         }
15         soma += n;
16         contador++;
17     }while(contador < 5);
18     printf("\nSoma = %d\n", soma);
19     return 0;
20 }
```



# Comando *switch*

- Este comando é similar ao *if-else*, mas não poderá ser usado quando a condição a ser testada for uma expressão.

```
switch (variável)
{
    case constante1: <bloco de instruções> break;
    case constante2: <bloco de instruções> break;
    case constante3: <bloco de instruções> break;
    ...
    default <bloco de instruções>;
}
```

# Comando *switch*

- É frequentemente usado para processar uma entrada, via teclado, como uma seleção por menu.
- Comando *break*
  - É opcional
  - Termina a sequência de comandos associada com cada constante.
- Comando *default*
  - É opcional
  - É executado se nenhuma coincidência for detectada.
  - Se não estiver presente, nenhuma ação será realizada se todos os testes falharem.

# Comando *switch*

- Exemplo:

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

int main()
{
    char opcao;
    printf("1 - Saldo\n");
    printf("2 - Extrato\n");
    printf("3 - Saque\n");
    printf("4 - Finalizar\n");
    printf("Digite a opção: ");
    scanf("%c", &opcao);
    switch (opcao)
    {
        case '1': printf("Saldo selecionado\n");break;
        case '2': printf("Extrato selecionado\n");break;
        case '3': printf("Saque selecionado\n");break;
        case '4': printf("Opção Finalizar selecionada\n");break;
        default: printf("Opção inválida.\n");
    }
    system("pause");
    return 0;
}
```

# Comando *switch*

- **Três características importantes do *switch***
  1. Difere do comando *if*, pois só pode testar igualdade, enquanto o *if* pode avaliar uma expressão lógica ou relacional.
  2. Duas constantes *case* no mesmo *switch* não podem ter valores idênticos.
  3. Se constantes de caractere são usadas em um comando *switch*, elas são automaticamente convertidas para seus valores inteiros.

# Referências

- **MIZRAHI, V. V. Treinamento em Linguagem C. 2ª Edição. São Paulo: Person Prentice Hall, 2008.**
- **SCHILDT, H. C, Completo e Total. 3ª Edição revista e atualizada; Tradução e revisão técnica: Roberto Carlos Mayer. São Paulo: Makron Books, 1996.**