1º Semestre

Técnicas de Programação A

Luiz Fernando Carvalho

luizfcarvalhoo@gmail.com

Estrutura de controle e repetição

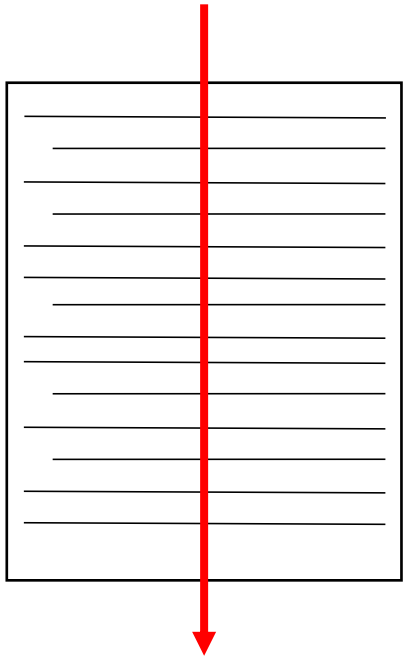
- Nesta aula
  - **Estrutura Sequencial**;
  - **Estruturas de Condição;**
  - Estruturas de Repetição;
- Objetivo
  - Apresentar o conceito de estrutura sequencial de fluxo de execução;
  - Explicar a aplicabilidade das estruturas de condição, suas variações e combinações;
  - Apresentar as estruturas de repetição e suas particularidades.

Execução de Instruções

- Programa de computador
  - Conjunto de instruções organizadas de forma a produzir a solução de um determinado problema.

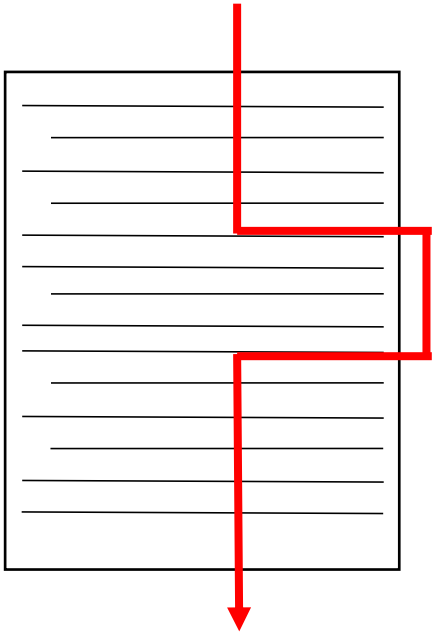
- Fluxo de execução
  - Começa na primeira linha e avança sequencialmente
    - de cima para baixo;
  - Em muitas circunstâncias é necessário executar instruções em uma ordem diferente;
  - Necessidade de decisão entre fluxos alternativos de execução ou da repetição de determinadas instruções
    - Pode haver bifurcações, repetição de código e tomada de decisão

Execução de Instruções



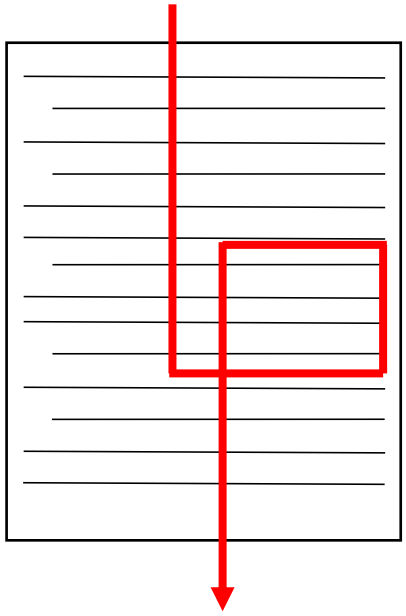
Fluxo de execução Sequencial

Comandos são executados um após o outro



Fluxo de execução com desvio

Comandos são executados dependendo do valor de uma condição



Fluxo de execução repetitivo

Comandos são executados de forma repetida

Estrutura de controle

- As estruturas de controle dividem-se em:
  - Estruturas de decisão *(condicionais)*;
  - Estruturas de repetição *(loops de repetição)*.
- As estruturas de controle estão vinculadas às **condições** que determinam se instruções serão ou não executadas;
- Uma **condição de controle** está relacionada aos **operadores relacionais e lógicos**;

Estrutura de decisão

- Uma estrutura de decisão permite decidir se um conjunto de instruções **será** ou **não** executado de acordo com determinadas condições;
- A decisão é feita com base no resultado de um teste lógico que determina a condição.
- Estruturas de decisão em C/C++:
  - `if`;
  - `if-else`;
  - `switch`;

Execução de Instruções

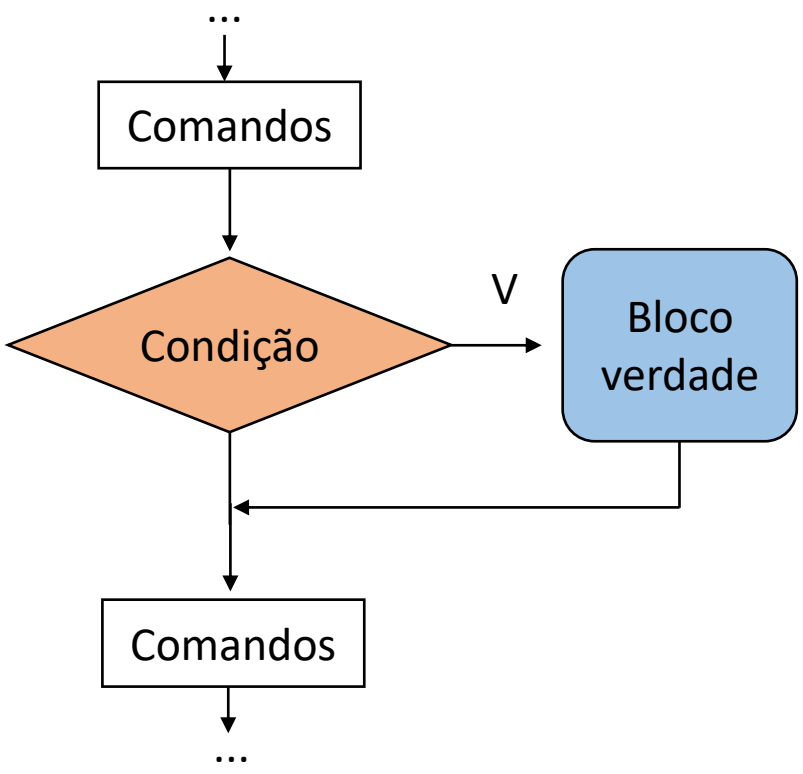
- Os operadores relacionais comparam dois valores e retornam um valor booleano
  - Verdadeiro (*true*);
  - Falso (*false*);

Operador	Descrição	X	Y	Lógico	Resultado
==	Igual a	2	3	X == Y	Falso
!=	Diferente de	2	3	X != Y	Verdadeiro
>	Maior que	2	3	X > Y	Falso
>=	Maior ou Igual	2	3	X >=Y	Falso
<	Menor que	2	3	X <Y	Verdadeiro
<=	Menor ou igual	2	3	X <=Y	Verdadeiro

A declaração if

Estrutura condicional simples

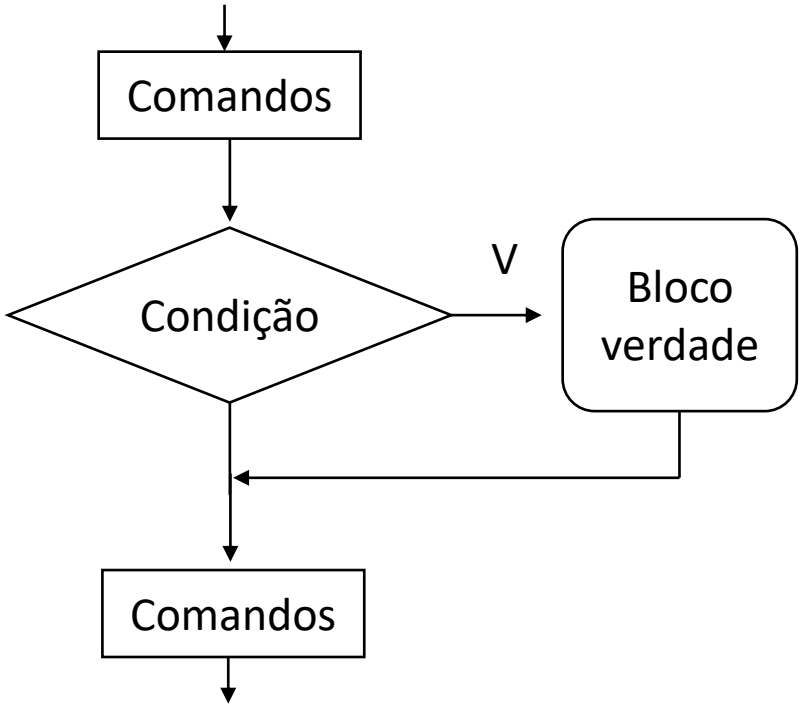
```
//comandos

if(condição)
{
    //bloco verdade
}

//comandos
```



A declaração if

Estrutura condicional Simples

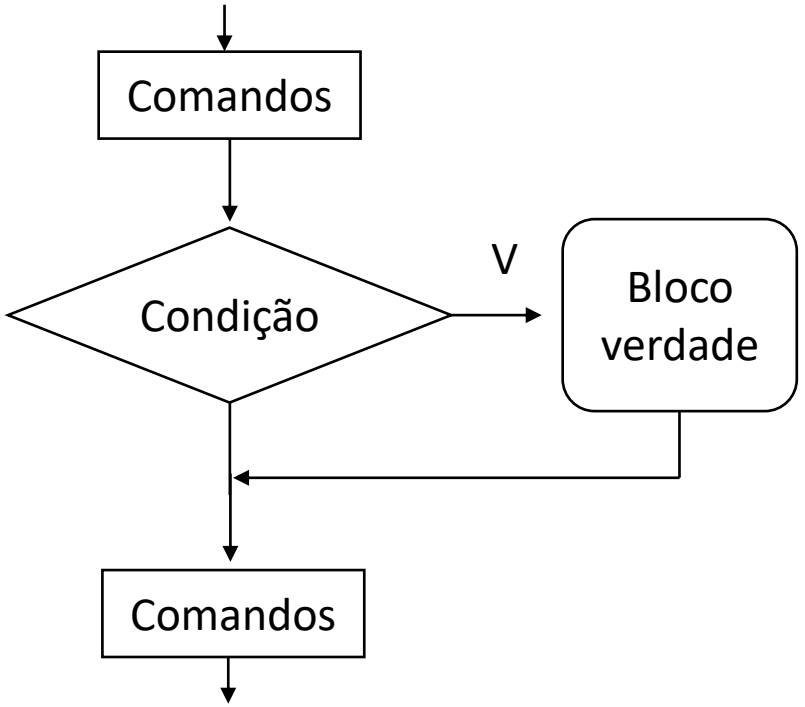
```
int main(){

    float nota = 7.6;

    if(nota >= 7.0)
       printf("APROVADO!");

    ...
    ...

    return 0;
}
```



A declaração if

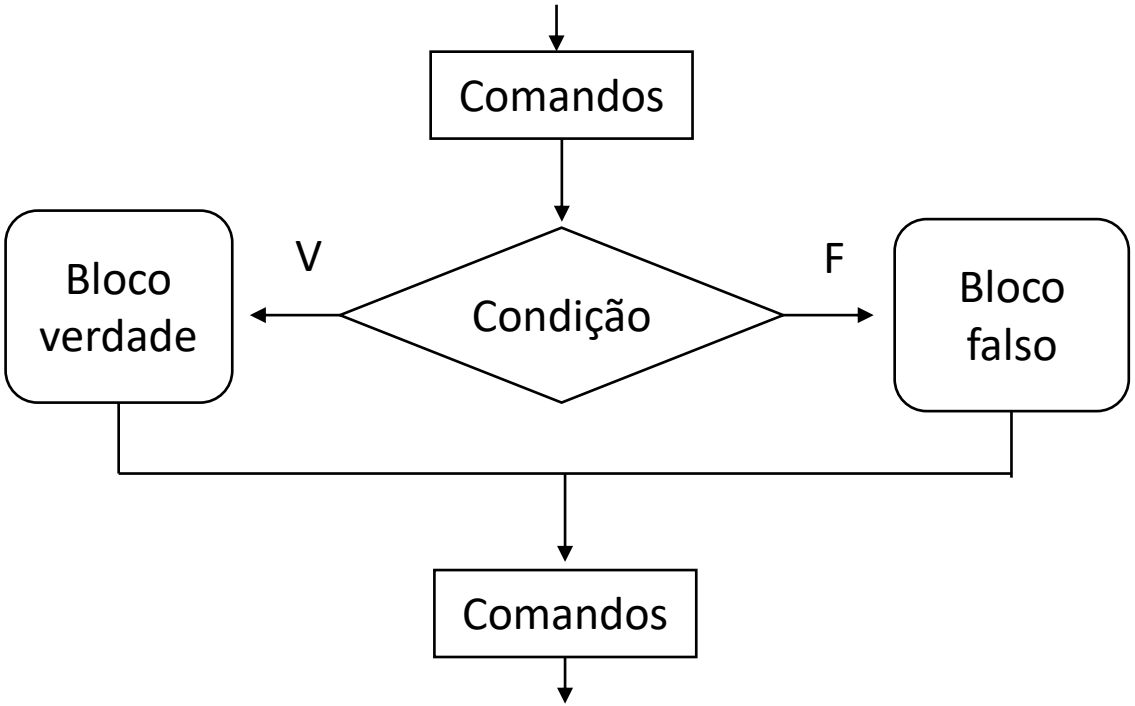
Estrutura condicional Simples

```
int main(){

    float nota1 = 7.6;
    float nota2 = 5.0;
    float media;

    if(nota1 == 10.0)
       printf("PARABÉNS!");

    media = (nota1 + nota2)/2;
    printf("Sua media e': %f", media);
    ...
    ...

    return 0;
}
```



A declaração if-else

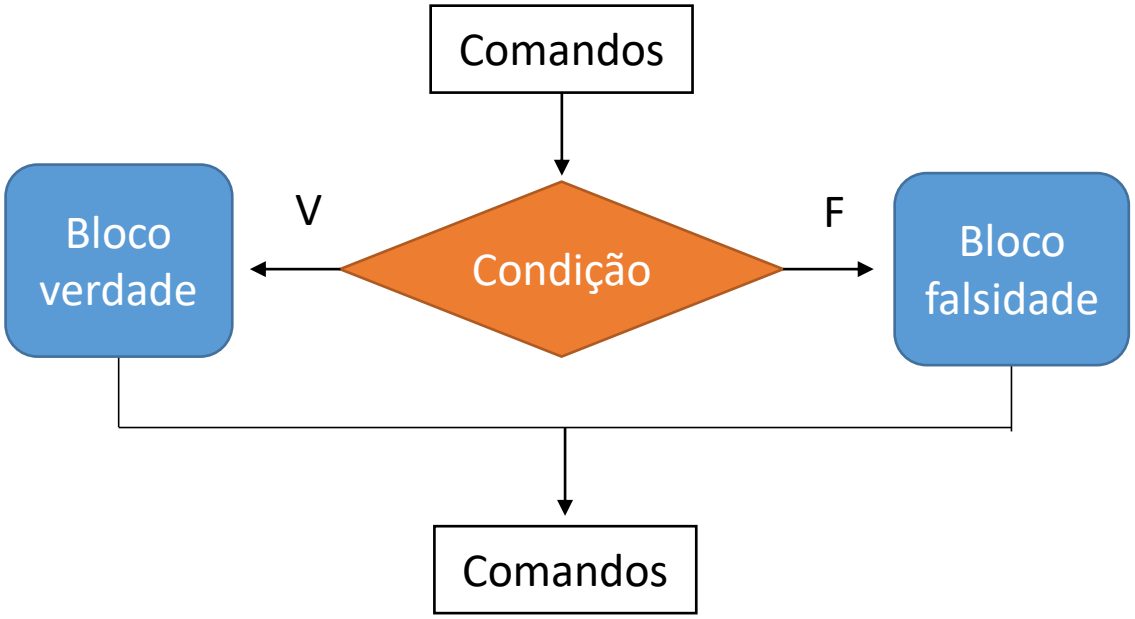
```
//comandos

if(condição)
{
   //bloco verdade
}
else
{
   //bloco falso
}

//comandos
```



A declaração *else* é opcional: pode-se utilizá-la para determinar um conjunto de comandos que serão executados caso a condição testada seja falsa

A declaração if-else

```
int a = 8;

if(a == 5)
   printf("a vale 5");
else
    printf("a não vale 5");


a = 3;
```



Somente um dos conjuntos de comandos (*if* ou *else*) será executado, nunca os dois!

A declaração if-else

```
#include<stdbool.h>
```
→ Biblioteca usada para tratar valores booleanos em C.

```
bool chovendo = true;
bool frio = false;

if(chovendo == true && frio == true)
   printf("Não vou para UEL");
else
    printf("Vamos para UEL =( ");

printf("O programa vai continuando...");
```

Equivale à

```
if(chovendo && frio)
```

V	V	V
V	F	F
F	V	F
F	F	F

Somente um dos conjuntos de comandos (*if* ou *else*) será executado, nunca os dois!

Blocos de instruções

- Com a adição de estruturas de controle segue um novo conceito: ***bloco de instruções***;
- **Bloco:** Conjunto de instruções agrupadas e delimitadas por chaves { }
- Se o bloco for de apenas uma instrução, pode-se omitir as { }
- Exemplos:

```
if(x > 100)
   printf("x maior que 100");
```

```
if(x > 100)
{
     printf("x maior que 100\n");
     printf("x vale: %d", x);
}
else
     printf("x não maior que 100");
```

Blocos de instruções – Erros comuns I

```
int main(){
    float saldo = 150.0;
    float saque = 200.0;

    if(saldo - saque >= 0)
        saldo = saldo - saque;
        printf(“Saque realizado com sucesso. Saldo atual = %f”, saldo);
    else
        printf(“Impossivel realizar o saque. Saldo insuficiente”);

    return 0;
}
```

Onde está o erro?

O compilador acusará que existe um ‘else’ sem um ‘if’

Blocos de instruções – Erros comuns II

```
int main(){
    float saldo = 150.0;
    float saque = 50.0;

    if(saldo - saque >= 0){
        saldo = saldo - saque;
        printf(“Saque realizado com sucesso. Saldo atual = %f”, saldo);
    }
    else
        printf(“Impossivel realizar o saque.”);
        printf(“Informe um valor menor ou igual a %f”, saldo);

    return 0;
}
```

Onde está o erro?

Estruturas de Condição Aninhadas

- É possível também aninhar comandos `if`, ou seja, fazer uma declaração `if` dentro de outra declaração `if` anterior.
- Exemplo: examinar se um número é positivo. Em caso afirmativo, verificar se o mesmo é divisível por 2.

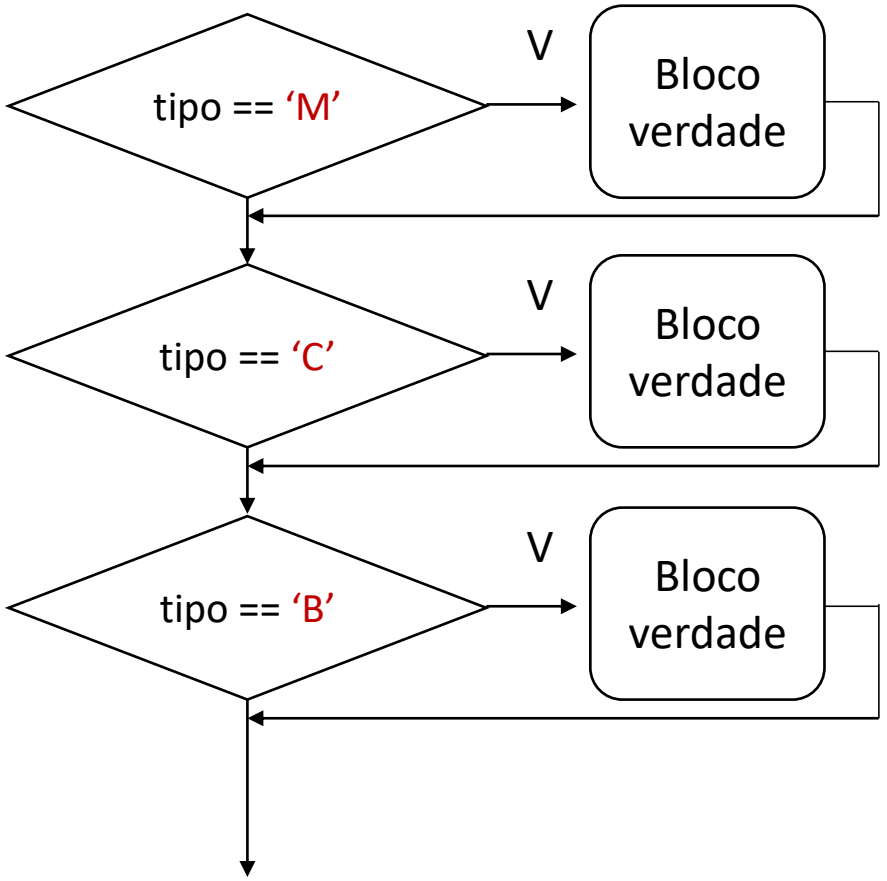
indentação

```
if(num >= 0) {
   if(num % 2 == 0) //resto da divisão por 2
      printf("O numero é par e positivo");
   else
        printf("O numero é impar e positivo");
}
else
    printf("O numero é negativo");
```

Um **if** aninhado é um comando **if** que está dentro de um outro comando **if**.

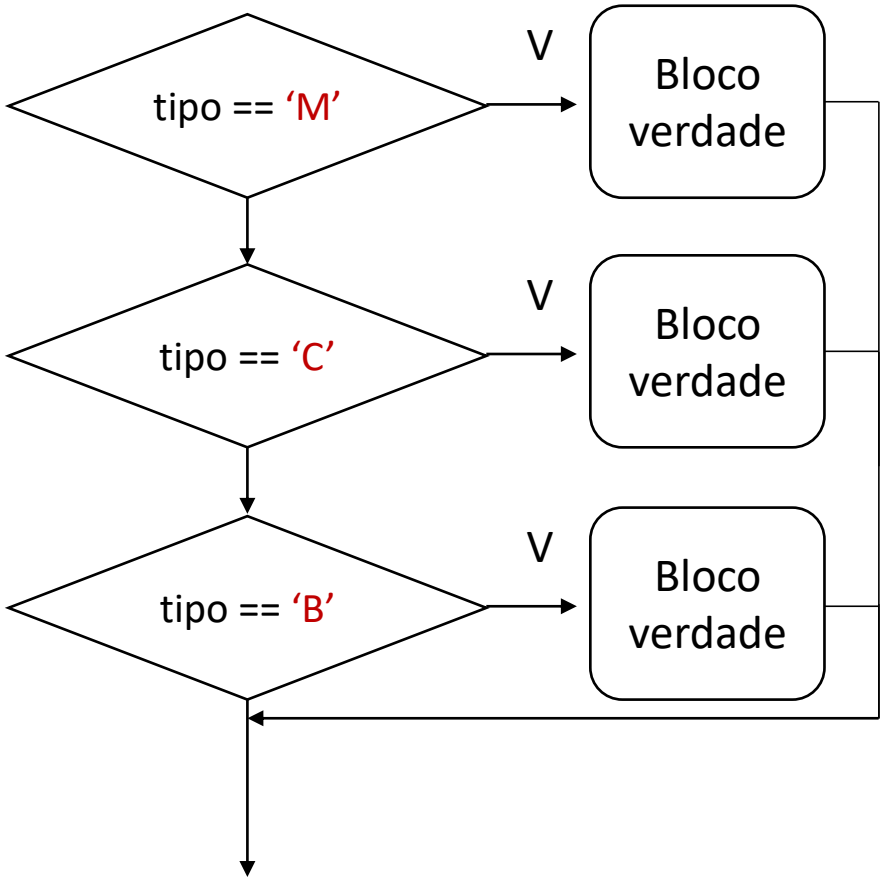
Estrutura condicional composta

```
int main(){
    char tipo;
    scanf("%c", &tipo);

    if(tipo == 'M')
       printf("Mucarela");
    if(tipo == 'C')
       printf("Calabresa");
    if(tipo == 'B')
       printf("Bacon");

    return 0;
}
```



Estrutura condicional composta

```
int main(){
    char tipo;
    scanf("%c", &tipo);

    if(tipo == 'M')
       printf("Mucarela");
    else if(tipo == 'C')
       printf("Calabresa");
    else
       printf("Bacon");

    return 0;
}
```



A declaração if-else-if

```
int num = 10;

if(num < 100)
     printf("Menor que 100");
if(num < 1000)
     printf("Menor que 1000");
if(num < 10000)
     printf("Menor que 10000");
if(num >= 10000)
     printf("Maior ou igual a 10000");
```

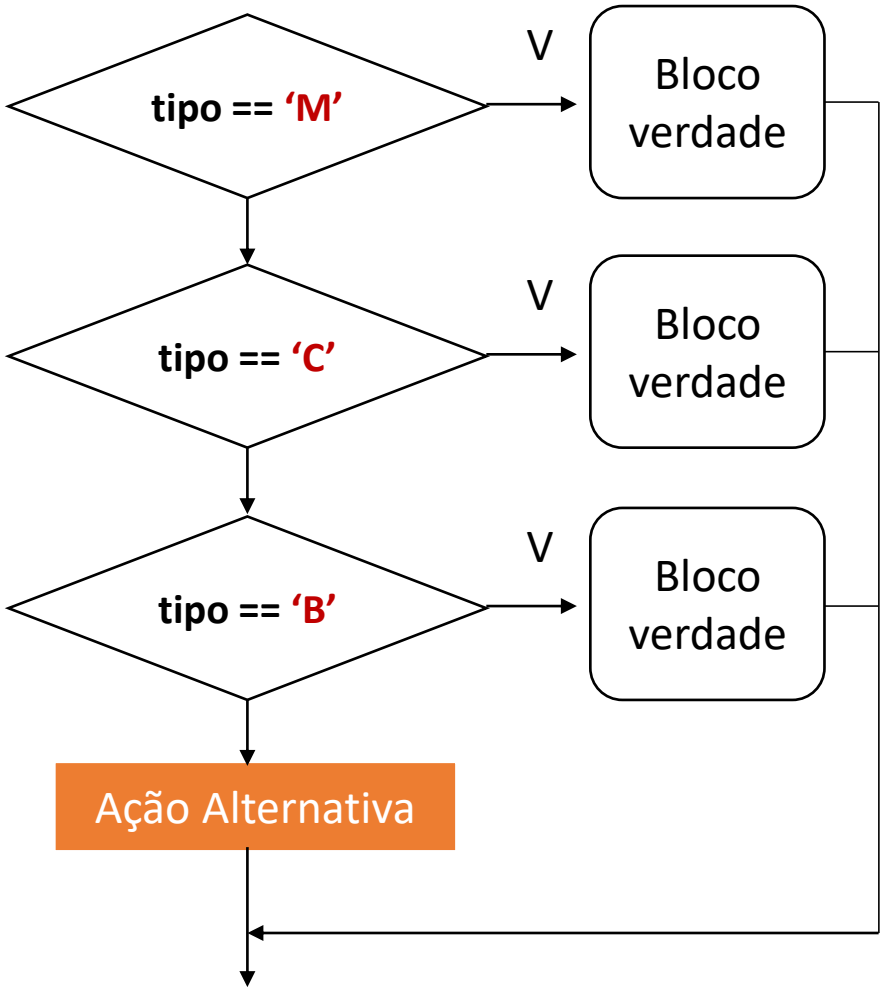
```
int num = 10;

if(num < 100)
     printf("Menor que 100");
else if(num < 1000)
     printf("Menor que 1000");
else if(num < 10000)
     printf("Menor que 10000");
else
     printf("Maior ou igual a 10000");
```

```
int num = 10;

if(num < 100)
     printf("Menor que 100");
else
    if(num < 1000)
       printf("Menor que 1000");
    else
       if(num < 10000)
          printf("Menor que 10000");
       else
          printf("Maior ou igual a 10000");
```

- As condições são avaliadas de cima para baixo;
- Assim que uma condição verdadeira é encontrada, o comando associado a ela é executado;
- O restante das condições não são executadas;
- Se nenhuma das condições for verdadeira, o último **else** é executado.

Estrutura condicional composta

```
int main()
{
  char tipo;
  scanf("%c", &tipo);

  switch(tipo)
  {
     case 'M':
          printf("Mucarela");
          break;
     case 'C':
          printf("Calabresa");
          break;
     case 'B':
          printf("Bacon");
          break;
     default:
          printf("opção invalida");
   }
return 0;
}
```

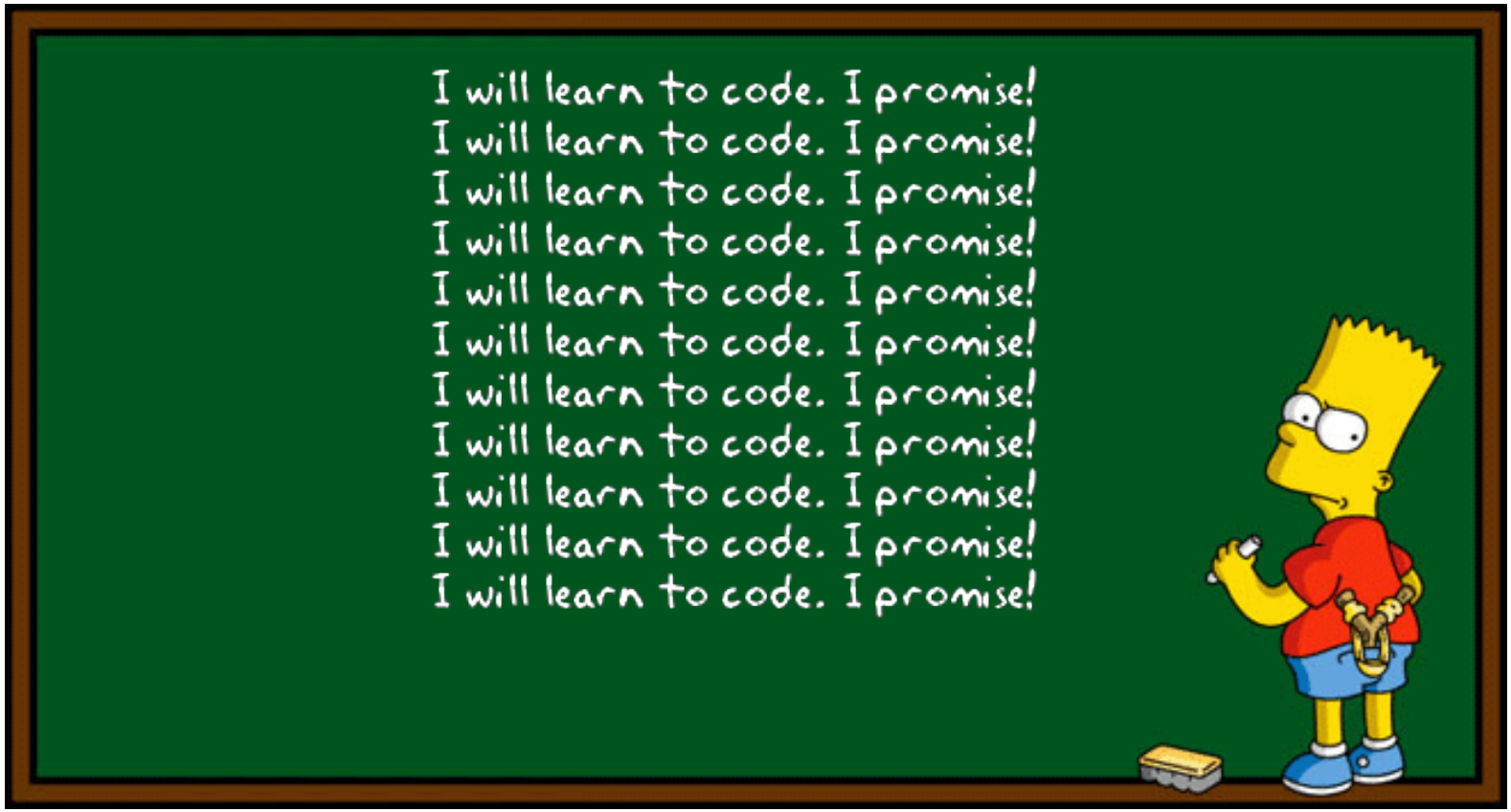


Estrutura condicional composta

```
int main()
{
  char tipo;
  scanf("%c", &tipo);

  switch(tipo)
  {
     case 'M':
         printf("Mucarela");
         break;
     case 'C':
         printf("Calabresa");
         break;
     case 'B':
         printf("Bacon");
         break;
     default:
         printf("opcao invalida");
   }
return 0;
}
```

```
Variáveis
   char tipo;
fim-variáveis
Início
   leia(tipo);

   escolha(tipo)
      caso 'M':
          imprima("Mucarela");
      caso 'C':
          imprima("Calabresa");
      caso 'B':
          imprima("Bacon");
      outrocaso:
          imprima("Opcao invalida");
fim
```

Atividades Propostas



Exercícios I

- Faça um programa que leia três números quaisquer e imprima o maior deles.

- Faça um programa que receba duas notas de um aluno. Calcule e mostre a média aritmética das notas e uma mensagem conforme a tabela a seguir. Obs.: Não existe nota negativa.

Média	Mensagem
[0, 3.9)	Reprovado
[4, 5.9)	Exame
[6, 10]	Aprovado

Exercício II

- Desenvolver a lógica para um programa que efetue o cálculo do reajuste de salário de um funcionário.
  1. Salario <= 500, reajuste de 15%
  2. Salario > 500, mas Salario <= 1000, reajuste será de 10%
  3. Salario > 1000, reajuste será de 5%

- Faça um programa que recebe a idade de um nadador e classifique-o numa das seguintes categorias:
  - Adulto (idade >=18);
  - Juvenil (idade >=14 e idade < 18);
  - Infantil (idade >=9 e idade < 14);
  - Mirim (Idade < 9).

Exercício III

- Crie um programa onde:
  - O usuário deve fornecer um valor;
  - O programa deve responder com o nome do dia da semana correspondente.
  - O programa não deve aceitar valores fora da faixa convencional, e deve apresentar uma mensagem de erro.

- 1 → Domingo
- 2 → Segunda-feira
- 3 → Terça-feira
- 4 → Quarta-feira
- 5 → Quinta-feira
- 6 → Sexta-feira
- 7 → Sábado

Estrutura de Repetição

- Um estrutura de repetição permite executar um conjunto de instruções um determinado número de vezes.

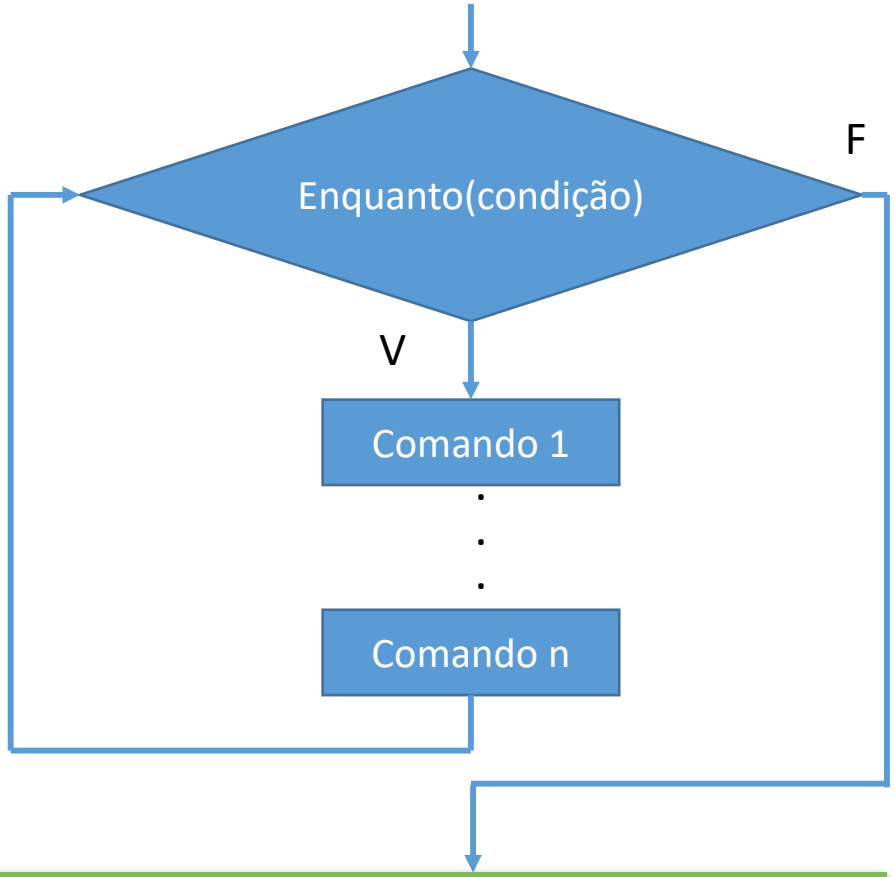
- A finalização da execução pode ser previamente determinada ou ser decorrente de uma ou mais condições se tornarem verdadeiras durante a execução do programa.

- Estruturas de repetição da linguagem C:
  - **while** → enquanto
  - **do** ... **while** → faça ... Enquanto
  - **for** → para

Repetição – Teste no Início

- Calculando a tabuada de algum número
- Enquanto... faça

```
int multiplicador = 0, resultado, num;

printf("Tabuada de qual numero: ");
scanf("%d", &num);

while(multiplicador <= 10)
{
     resultado = num * multiplicador;
     printf("%d", resultado);
     multiplicador = multiplicador + 1;
}
```



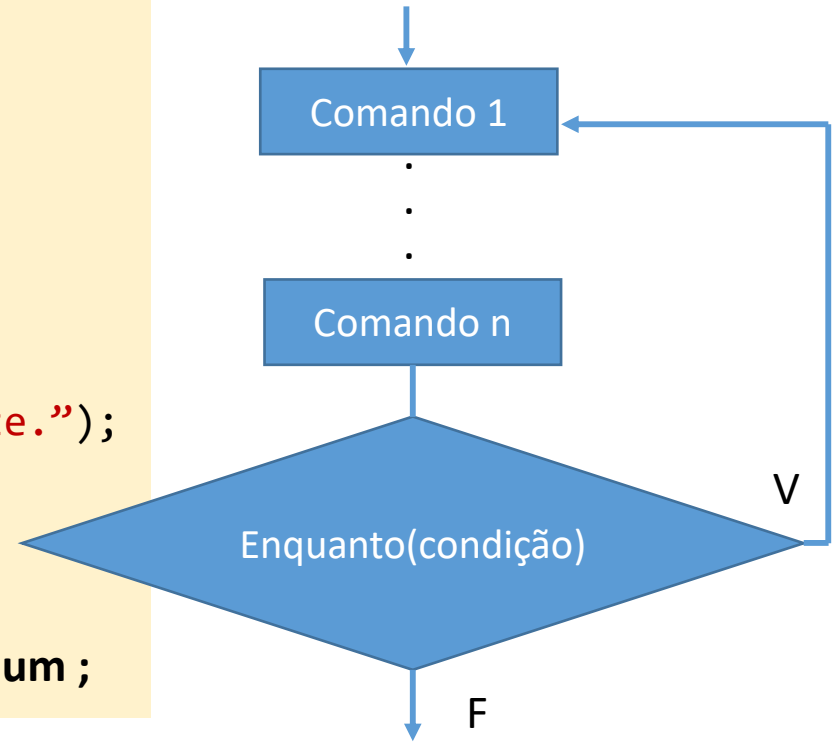
Em alguns programas os comandos dentro da instrução *while* podem não ser executados, uma vez que a condição é verificada antes que eles possam ser executados.

Repetição – Teste no Final

- Verificar se o usuário digitou um valor positivo
- Faça... Enquanto

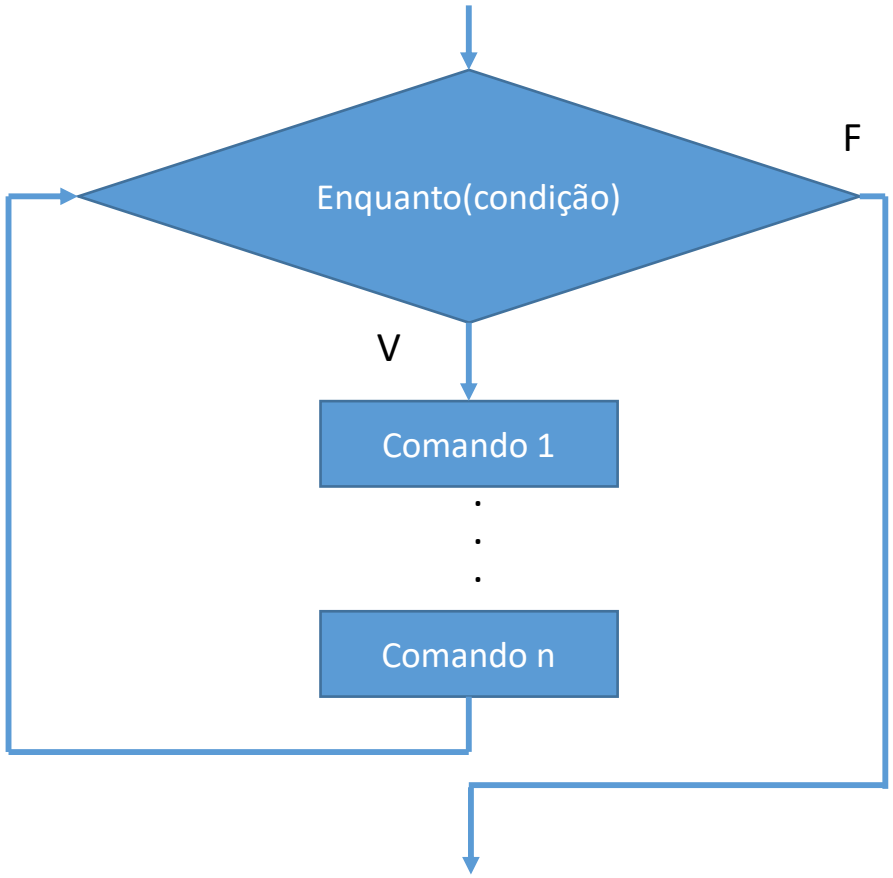
```
int num;

do
{
   printf("Forneca um inteiro positivo: ");
   scanf("%d", &num);

   if(num < 0)
     printf("Valor invalido. Tente novamente.");
}
while(num < 0);
```

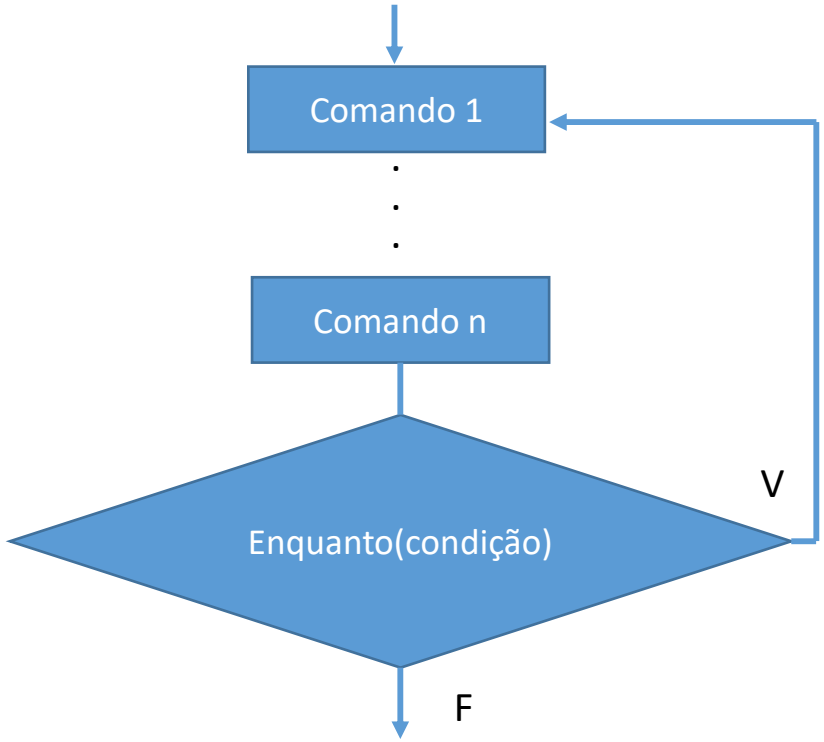
Esse while tem um ;



Os comandos dentro da instrução **do** são executadas pelo menos uma vez, uma vez que o teste só é realizado no final.

Repetição – Comparação






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

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

Repetição – Comparação

```
Enquanto(condição) faça
      comandos;

Fim-enquanto
```

```
Faça
    comandos;

Enquanto(condição);
```

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

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

Repetição – for

Calculando a Tabuada com o **para-faça**:

```
int multiplicador, resultado, num;

printf("Tabuada de qual numero: ");
scanf("%d", &num);

for(multiplicador=0; multiplicador <= 10; multiplicador++){
        resultado = multiplicador * num;
        printf("%d", resultado);
}
```

Inicialização

Condição

Incremento/
Decremento

```
for(inicialização; condição; incremento)
    comandos;
```

Atalhos de incremento e decremento

```
i++
++i
```

```
i = i + 1
```

```
i = 5;
a = i++;
```

```
a = 5, i = 6
```

```
i = 5;
a = ++i;
```

```
a = 6, i = 6
```

Tomar cuidado com atalhos de incremento em operações de atribuição

```
i--
--i
```

```
i = i - 1
```

```
i = 5;
a = i--;
```

```
a = 5, i = 4
```

```
i = 5;
a = --i;
```

```
a = 4, i = 4
```

Atalhos para operações aritméticas

Equivale a ...

```
i += 5;
```

```
i = i + 5;
```

```
i -= 5;
```

```
i = i - 5;
```

```
i *= 5;
```

```
i = i * 5;
```

```
i /= 5;
```

```
i = i / 5;
```

Repetição – Quebrando ou continuando o laço de repetição

- Break
  - Utilizado para sair abruptamente da estrutura de controle;
- Continue
  - Ignora o resto do bloco de dados de uma iteração, mas continua executando a estrutura de controle;

```
for(int i=1; i <= 100; i++)
{
   if(i % 10 == 0)
      continue;
   else
      printf("%d", i);
}
```

1 2 3 4 5 6 7 8 9 11 12 ... 19 21 ... 29 31 99

```
for(int i=1; i <= 100; i++)
{
   if(i % 10 == 0)
      break;
   else
      printf("%d", i);
}
```

1 2 3 4 5 6 7 8 9

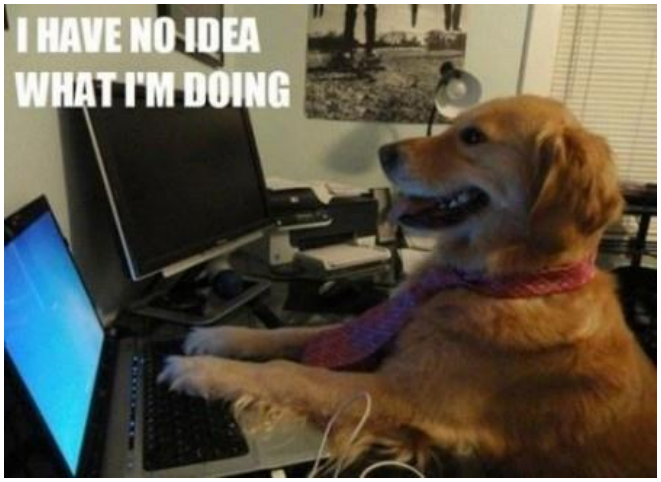
Repetição – Quebrando ou continuando o laço de repetição

- Break
  - Quando está dentro de um laço de repetição, o laço é imediatamente terminado e o programa passa a executar comandos logo após o laço;
- Continue
  - Em vez de terminar a execução do laço, ***continue*** força que ocorra a próxima iteração do laço, pulando qualquer código intermediário. Para os laços ***while*** e ***do-while***, o controle do programa passa para o teste condicional, assim que encontra o comando continue. Para um ***for***, o comando ***continue*** faz com que o teste condicional e a porção de incremento do laço sejam executados;

Repetição – Loop infinito

- Podem ocorrer por erros durante a programação:

```
int multiplicador = 0, resultado;

printf(“Tabuada de qual numero: ”);
scanf(“%d”, &num);

while(multiplicador <= 10)
{
     resultado = num * multiplicador;
     printf(“%d”, resultado);
}
```



O que está errado?

Qual o valor de multiplicador ao longo das iterações?

Quando o loop irá acabar?

Repetição – Loop infinito

- Podem ocorrer de propósito:

```
for(;;)
{
   printf(“Digite um numero inteiro: “);
   scanf(“%d”, &n);
   if (n == 7)
   {
      printf(“Saindo do loop...\n”);
      break; //força a saída do loop
   }
   printf(“Numero: %d\n”, n);
}
printf(“Fim de programa”);
```

```
while(true)
{
   printf(“Digite um numero inteiro: “);
   scanf(“%d”, &n);
   if (n == 7)
   {
      printf(“Saindo do loop...\n”);
      break; //força a saída do loop
   }
   printf(“Numero: %d\n”, n);
}
printf(“Fim de programa”);
```

Exercícios

1. Faça um Programa que leia 10 valores inteiros e escreva no final a soma dos valores lidos;

2. Faça um programa que receba a idade de dez pessoas, calcule e mostre a quantidade de pessoas com idade maior ou igual a 18 anos.

3. Faça um programa que receba dez números e mostre a quantidade de números entre 30 e 90.

4. Faça um programa para calcular $n!$

   Lembrando que: $n! = n * (n-1) * (n-2) * \ \ldots * 1$

   $0! = 1, \text{por definição}$

5. Calcule o resultado da série:

   - $S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \cdots + \frac{99}{50}$