Linguagem de Programação I

Aula 5 – Estruturas Condicionais

Prof Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Estruturas Condicionais

Controlando o fluxo do programa

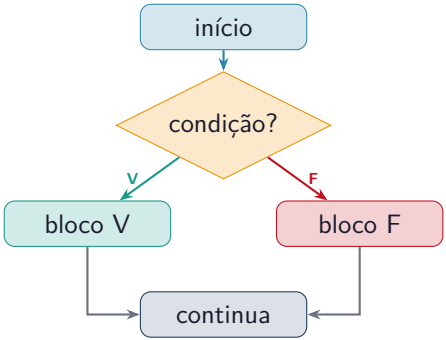
O que são Estruturas Condicionais?

Definição

Permitem realizar **testes nas variáveis** e alterar o **fluxo de execução** do programa conforme as condições testadas.

Tipos disponíveis em C:

1. `if` — decisão simples
2. `if-else` — dois caminhos
3. `? :` — operador ternário
4. `switch-case` — múltiplas opções



Estrutura IF

Decisão simples

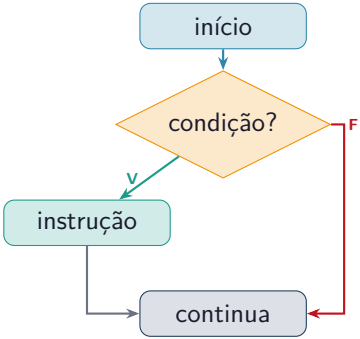
Estrutura Condicional `if`

Como funciona

Executa as instruções **somente se a condição for verdadeira**. Se falsa, nada é executado. [Schildt e Mayer 1997]

Sintaxe

```
if (condicao) {
  instrucoes;
}
```



Estrutura `if` — Exemplo 1

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
  int A, B, Soma;

  printf("Digite um numero inteiro: ");
  scanf("%d", &A);

  printf("Digite um numero inteiro: ");
  scanf("%d", &B);

  Soma = A + B;
  printf ("O Valor da soma = %d\n", Soma);

  if(Soma > 10){
    printf("O valor da soma eh maior que 10\n");
  }

  return(0);
}
```

Estrutura `if` — Exemplo 2

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
  int x, y;

  printf("Digite um numero inteiro: ");
  scanf("%d", &x);

  printf("Digite outro numero inteiro: ");
  scanf("%d", &y);

  if(x > y) printf("O primeiro número é maior!\n");

  if(y > x) printf("O segundo número é maior!\n");

  if(x==y) printf("Os números são iguais!\n");

  return(0);
}
```

Estrutura IF-ELSE

Decisão com dois caminhos

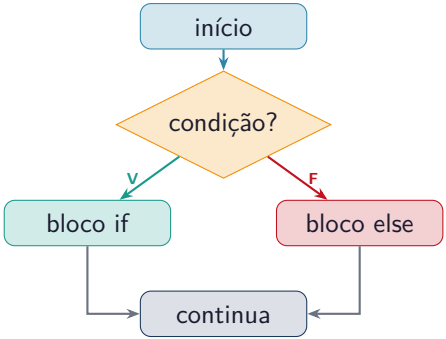
Estrutura Condicional `if-else`

Como funciona

Permite **dois caminhos**: `if` para condição verdadeira, `else` para condição falsa.

Sintaxe

```
if (condicao) {
  instr.  verdadeira;
} else {
  instr.  falsa;
}
```



Estrutura `if-else` — Exemplo 1

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
  int x, y;

  printf("Digite um numero inteiro: ");
  scanf("%d", &x);

  printf("Digite outro numero inteiro: ");
  scanf("%d", &y);

  if(x > y){ printf("O primeiro número é maior!\n")}
  else{printf("O segundo número é maior!\n");}

  return(0);
}
```

Estrutura `if-else` — Exemplo 2

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
  int x, y;

  printf("Digite um numero inteiro: ");
  scanf("%d", &x);

  printf("Digite outro numero inteiro: ");
  scanf("%d", &y);

  if(x > y){printf("O primeiro número é maior!\n");}
  else if(y>x){printf("O segundo número é maior!\n");}
       else{printf("Os números são iguais!\n");}

  return(0);
}
```

Operador Ternário `? :`

Forma compacta do if-else

Operador Ternário ?:

Sintaxe

```
(condicao) ? expressao1 :  expressao2
```

Com ternário

```
max = (num1>num2) ? num1 :  num2;

mod = (num<0) ? -num :  num;
```

Equivalente com if-else

```
if (num1 > num2)
  max = num1;
else
  max = num2;
```

Quando usar?

Ideal para expressões simples. Para lógicas complexas, prefira `if-else` para manter a legibilidade.

Operador Ternário — Exemplo 1

```c
#include <stdio.h>
#include <stdlib.h>
void main() {
    int a,b,c,d;
    printf("Digite o 1° Número: ");
    scanf("%d",&a);
    printf("\nDigite o 2° Número: ");
    scanf("%d",&b);
    printf("\nDigite o 3° Número: ");
    scanf("%d",&c);

    d = (a > b)? a : b;

    if (c > d) printf("\nO Maior é %d",c);
    else  printf("\nO Maior é %d",d);
}
```

Operador Ternário — Exemplo 2

```c
#include <stdio.h>
void main() {
    int a,b,c,d;
    printf("Digite três números: ");
    scanf("%d\n%d\n%d",&a,&b,&c);

    d = (a > b)? a : b;

    if (c > d) printf("\nO Maior é %d",c);
    else  printf("\nO Maior é %d",d);
}
```

Estrutura SWITCH-CASE

Múltiplas decisões

Estrutura `switch-case`

Para que serve?

Substitui sequências de `if-else` quando se testa o **mesmo valor** contra uma lista de constantes. Mais legível e eficiente nesses casos.

Restrições

- Variável: **inteiro** ou **char**
- Valores de `case`: **constantes**

Sintaxe

```
switch (variavel) {
  case c1:
    comandos;
    [break;]
  case c2:
    comandos;
    [break;]
  default:
    comandos;
}
```

O papel do `break`

Encerra o `case` atual. Se omitido, a execução continua para o próximo `case` (fall-through).

Exemplo: Dias da Semana

Problema

Leia um número (1–7) e informe o nome do dia da semana correspondente.

Número	Nome
1	Domingo
2	Segunda
3	Terça
4	Quarta
5	Quinta
6	Sexta
7	Sábado

```c
#include <stdio.h>
void main (){
  int dia;
  printf ("Digite o dia da semana de 1 a 7: ");
  scanf("%d", &dia);

  if (dia == 1) printf ("Domingo\n");
  else if (dia == 2) printf ("Segunda\n");
  else if (dia == 3) printf ("Terça\n");
  else if (dia == 4) printf ("Quarta\n");
  else if (dia == 5) printf ("Quinta\n");
  else if (dia == 6) printf ("Sexta\n");
  else if (dia == 7) printf ("Sabado\n");
  else printf ("Valor invalido!\n");
}
```

solução com `if-else`

Dias da Semana com `switch-case`

```c
#include <stdio.h>
void main (){
  int dia;
  printf ("Digite o dia da semana de 1 a 7: ");
  scanf("%d", &dia);

  switch (dia){
      case 1: printf ("Domingo\n"); break;
      case 2: printf ("Segunda\n"); break;
      case 3: printf ("Terça\n"); break;
      case 4: printf ("Quarta\n"); break;
      case 5: printf ("Quinta\n"); break;
      case 6: printf ("Sexta\n"); break;
      case 7: printf ("Sabado\n"); break;
      default: printf ("Valor invalido!\n");
  }
}
```

Exemplo: Calculadora com 4 Operações

▪ Problema

Implemente uma calculadora com +, -, *, /.

Entrada: *(número real) (operador) (número real)* **Saída:** $5 + 7 = 12$

▪ Por que `switch` é melhor aqui?

Testar um único operador (`char`) contra 4 constantes é o caso de uso ideal do `switch-case`: código mais limpo e direto que uma cadeia de `if-else`.

Calculadora com `if-else`

```c
#include <stdio.h>
void main(){
    float n1, n2;
    char op;

    printf("Digite a expressão matemática: ");
    scanf("%f\n%c\n%f",&n1,&op,&n2);

    if (op == '+') printf("%5.2f + %5.2f = %5.2f", n1, n2, n1+n2);
    else if (op == '-') printf("%5.2f - %5.2f = %5.2f", n1, n2, n1-n2);
    else if (op == '*') printf("%5.2f * %5.2f = %5.2f", n1, n2, n1*n2);
    else if (op == '/')
        if (n2 != 0) printf("%5.2f / %5.2f = %5.2f", n1, n2, n1/n2);
        else printf("Expressão inválida! Proibido divisão por zero!");
    else printf("Operador inválido!");
}
```

Calculadora com `switch-case`

```c
#include <stdio.h>
void main(){
    float n1, n2;
    char op;

    printf("Digite a expressão matemática: ");
    scanf("%f\n%c\n%f",&n1,&op,&n2);

    switch (op){
        case '+': printf("%5.2f + %5.2f = %5.2f", n1, n2, n1+n2);
            break;
        case '-': printf("%5.2f - %5.2f = %5.2f", n1, n2, n1-n2);
            break;
        case '*': printf("%5.2f * %5.2f = %5.2f", n1, n2, n1*n2);
            break;
        case '/': if (n2!=0) printf("%5.2f + %5.2f = %5.2f", n1, n2, n1+n2);
                  else printf("Expressão inválida! Proibido divisão por zero!");
            break;
        default: printf("Operador inválido!");
    }
}
```

`switch-case` sem `break` — Fall-through

Omitir o `break`: cuidado!

A execução **cai** automaticamente para o próximo `case`. Pode ser intencional quando dois casos devem executar o mesmo bloco.

Com `break` (usual)

```
case 1:  puts("um"); break;
case 2:  puts("dois"); break;
```

Sem `break` (intencional)

```
case 1:
case 2:
  puts("1 ou 2"); break;
```

Ambos os casos executam o mesmo bloco.

Referências I

SCHILDT, H.; MAYER, R. *C completo e total*. [S.l.]: Pearson University, 1997. ISBN 9788534605953.