

# Estruturas condicionais

David Déharbe



## Objetivos da aula

- Revisar os primeiros elementos de linguagem C já vistos.
- Condicionais simples
- Condicionais compostas
- Operadores lógicos

## Objetivos da aula

- Revisar os primeiros elementos de linguagem C já vistos.
- Condicionais simples
- Condicionais compostas
- Resolução de problemas

# Revisão da linguagem C

- A estrutura do código fonte de um programa C é a seguinte:

## 1. inclusão de arquivo cabeçalho

```
#include <nome do arquivo cabeçalho>
```

## 2. método principal

```
int main ()  
{  
    declarações de variáveis  
    comandos  
    return 0;  
}
```

## 3. comentários podem ser usados para explicar o código ao programador

```
/* este eh um comentario inutil */
```

## Arquivos cabeçalhos

- **stdio.h**

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

- para imprimir (e ler)

- **limits.h**

```
#include <limits.h>
```

- para saber os limites do tipo int

# Declaração de variável

- Variável é:
  - um compartimento da memória
  - nome
  - valor
  - tipo
- com inicialização 

```
int v = 0;
```
- sem inicialização 

```
int v;
```

  - valor qualquer

# Sequência de comandos

- comando de escrita

```
printf(texto);
```

```
printf(texto, valores);
```

- comando de leitura

```
scanf(texto, &variável1, &variável2);
```

- atribuição

```
variável = valor;
```

## Exemplo básico

```
#include <stdio.h>

int main ()
{
    int n;

    printf("Digite um numero inteiro: ");
    scanf("%d", &n);
    printf("Voce digitou o numero %d.\n", n);

    return 0;
}
```

## Objetivos da aula

- Revisar os primeiros elementos de linguagem C já vistos.
- **Condicionais simples**
- Condicionais compostas
- Operadores lógicos

## Motivação

- O algoritmo a ser executado pode depender do valor dos dados lidos.
- Por exemplo, considere o seguinte problema.
  - *Escreva um programa que lê um número e que imprime "Bingo" se o valor lido é 42.*
- Algoritmo
  - ler um número.
  - se o número lido é 42, imprimir "Bingo".
- (até agora, só vimos execuções incondicionais).

## Comando condicional: sintaxe

- O comando **if** permite executar um ou mais comandos quando uma determinada condição é satisfeita.

- Sintaxe:

```
comando_if: if (expressão)
             comando1
```

- Semântica

- ✓ A *expressão* é avaliada. Seja  $v$  o resultado da avaliação.
- ✓ Se  $v$  for 0, termina a execução do *comando\_if*
- ✓ Se  $v$  não for 0, então *comando1* é executado.

## Solução do exemplo

- Para testar se dois valores são iguais, usa-se o operador == (igual-igual)

```
#include <stdio.h>

int main ()
{
    int n = 0;
    scanf("%d", &n);
    if (n == 42)
        printf("Bingo\n");
    return 0;
}
```

## Bloco de comandos

- Como fazer para executar vários comandos sujeitos a uma condição?

- Solução 1:

```
if (condição)
    comando 1
if (condição)
    comando 2
...
if (condição)
    comando n
```

- Solução 2: bloco de comandos

```
if (condição) {
    comando 1
    comando 2
    ...
    comando n
}
```

## Bloco de comandos

- **Bloco de comandos:** vários comandos são executados em sequência e formam uma unidade sintática comando.
- **Sintaxe:**

```
{  
    declaração de variáveis  
    comando 1  
    comando 2  
    ...  
    comando n  
}
```

- **Semântica**
  - cada declaração e cada comando é executado na ordem em que aparece
  - como no *main ( )*.

## Exercício: processamento de notas e frequência

- Escreva um programa que lê dois números: F (percentual de faltas) e MF (média final). Ambos números estão no intervalo de 0 a 100.
- Se F é maior que 25 então imprime "RF".
- Se F é menor que 25
  - Se MF é menor que 50 então imprime "RN".
  - Se MF é maior ou igual a 50 então imprime "AP".

# Teste de igualdade

- Sintaxe

- (expressão) ::= (expressão) "==" (expressão)

- Semântica de  $e1 == e2$

- $e1$  é avaliada, seja  $v1$  o resultado

- $e2$  é avaliada, seja  $v2$  o resultado

- se  $v1$  e  $v2$  são iguais, o valor é 1, caso contrário é 0.

# Outros operadores

## • Sintaxe

- (expressão) ::=
- | (expressão) "!=" (expressão)
- | (expressão) "<" (expressão)
- | (expressão) "<=" (expressão)
- | (expressão) ">" (expressão)
- | (expressão) ">=" (expressão)

- != diferente
- <= menor ou igual
- >= maior ou igual

## Exercício

- *Escreva um programa que lê a temperatura ambiente e que imprime uma mensagem se está muito quente (temperatura acima de 32°C).*

# Objetivos da aula

- Revisar os primeiros elementos de linguagem C já vistos.
- Condicionais simples
- **Condicionais compostas**
- Operadores lógicos

## Motivação

- **Problema:** *Escreva um programa que lê a média de um aluno e indica se é aprovado (média maior ou igual a cinco) ou reprovado (média menor que cinco).*
- **Algoritmo 1:**
  - imprimir mensagem solicitando a média;
  - ler a média e armazenar em memória;
  - se o número lido é maior ou igual a cinco, imprimir "Aprovado".
  - se o número lido é menor que cinco, imprimir "Reprovado".
    - ✓ são duas condições excludentes, podemos simplificar usando uma nova construção.

## Solução do exemplo

```
#include <stdio.h>

int main ()
{
    int n = 0;

    printf("Digite a media: ");
    scanf("%d", &n);

    if (n >= 5)
        printf("Aprovado\n");

    if (n < 5)
        printf("Reprovado\n");

    return 0;
}
```

## Estrutura condicional composta

- O comando `if` permite executar um ou mais comandos quando uma determinada condição é satisfeita.
- Sintaxe:

```
comando_if_else:  
if (expressão)  
    comando 1  
else  
    comando 2
```

- Semântica:
  - A *expressão* é avaliada. Seja  $v$  o resultado.
  - Se  $v$  for diferente de 0, *comando 1* é executado.
  - Se  $v$  for 0, *comando 2* é executado.

## Exemplo revisitado

- **Problema:** *Escreva um programa que lê a média de um aluno e indica se é aprovado (média maior ou igual a cinco) ou reprovado (média menor que cinco).*
- São duas condições excludentes, podemos simplificar usando uma estrutura condicional composta construção.
- **Algoritmo 2:**
  - imprimir mensagem solicitando a média;
  - ler a média e armazenar em memória;
  - se o número lido é maior ou igual a cinco, imprimir "Aprovado".
  - senão imprimir "Reprovado".

## Solução do exemplo

```
#include <stdio.h>

int main ()
{
    int n = 0;

    printf("Digite a media: ");
    scanf("%d", &n);

    if (n >= 5)
        printf("Aprovado\n");
    else
        printf("Reprovado\n");

    return 0;
}
```

## Exercício

- **Problema:** *Escreva um programa que lê dois números, digamos  $n$  e  $m$ , e imprime o menor de  $n$  e  $m$  e o maior de  $n$  e  $m$ .*

## Exercício

- **Problema:** *Escreva um programa que lê dois números, digamos  $n$  e  $m$ , e imprime o menor de  $n$  e  $m$  e o maior de  $n$  e  $m$ .*
- **Algoritmo:**
  - declarar duas variáveis  $n$  e  $m$ , que vão armazenar o valores lidos;
  - declarar duas variáveis "menor" e "maior", que vão armazenar o menor de  $n$  e  $m$ , e o maior de  $n$  e  $m$ , respectivamente.
  - imprimir mensagem solicitando dois números;
  - ler o primeiro número em  $n$  e o segundo número em  $m$ .
  - calcular "menor" e "maior":
    - se  $n$  for menor ou igual a  $m$ , atribuir "menor" com  $n$  e "maior" com  $m$ .
    - senão atribuir "menor" com  $m$  e "maior" com  $n$ .
  - imprimir o valor de "menor" seguido do valor de "maior".

# Solução

```
#include <stdio.h>

int main ()
{
    int n = 0;
    int m = 0;
    int maior = 0;
    int menor = 0;

    printf("Digite dois números: ");
    scanf("%d", &n);
    scanf("%d", &m);
    if (n <= m) {
        menor = n;
        maior = m;
    } else {
        menor = m;
        maior = n;
    }
    printf("Menor: %d\n", menor);
    printf("Maior: %d\n", maior);
    return 0;
}
```

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

```
int main ()
{
    int n = 0;
    int m = 0;
    int maior = 0;
    int menor = 0;

    printf("Digite dois números: ");
    scanf("%d", &n);
    scanf("%d", &m);
    if (n <= m) {
        menor = n;
        maior = m;
    } else {
        menor = m;
        maior = n;
    }
    printf("Menor: %d\n", menor);
    printf("Maior: %d\n", maior);
    return 0;
}
```

## Solução 2

```
#include <stdio.h>

int main ()
{
    int n = 0;
    int m = 0;

    printf("Digite dois números: ");
    scanf("%d", &n);
    scanf("%d", &m);
    if (n <= m) {
        printf("Menor: %d\n", n);
        printf("Maior: %d\n", m);
    } else {
        printf("Menor: %d\n", m);
        printf("Maior: %d\n", n);
    }
    return 0;
}
```

## Objetivos da aula

- Revisar os primeiros elementos de linguagem C já vistos.
- Condicionais simples
- Condicionais compostas
- Operadores lógicos
-

# Lógica

- A aritmética é a ciência do cálculo.
  - A lógica é a ciência do raciocínio.
- Há várias aritméticas (naturais, inteiros, reais, complexos, etc.)
  - Há várias lógicas.
- A lógica mais elementar é a lógica Booleana.
- Cada expressão pode ter um de dois valores:
  - ✓ verdadeiro (T) ou falso (F).
- Cada variável pode ter um de dois valores:
  - ✓ verdadeiro (T) ou falso (F).
- T e F são chamados valores booleanos.
- As operações são conjunção, disjunção e negação.

## Lógica e programação C

- A linguagem C usa o tipo `int` para representar valores booleanos.
  - O valor `int 0` representa F.
  - Os demais valores `int` representam V.
- Cada expressão `int C` pode ser interpretada como uma condição que é falsa (quando `= 0`) ou verdadeira (quando `≠ 0`).
  - `1 < 2`
  - `2 == 1 + 1`
  - `0 * 1`
- A linguagem C tem operadores que correspondem aos operadores lógicos: `&&` (conjunção), `||` (disjunção), `!` (negação).

## Definição da conjunção

- ▶ Seja  $P$  e  $Q$  duas condições.
- ▶ A conjunção de  $P$  e  $Q$  é denotada  $P \wedge Q$ ;
- ▶  $P \wedge Q = T$  se e somente se  $P = T$  e  $Q = T$ .

P	Q	$P \wedge Q$
F	F	F
F	T	F
T	F	F
T	T	T

## Leis da conjunção

- $P \wedge P = P$

- $P \wedge T = P$

- $P \wedge F = F$

- $P \wedge Q = Q \wedge P$

- $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

## Leis da conjunção

- $P \wedge P = P$
- $P \wedge T = P$
- $P \wedge F = F$
- $P \wedge Q = Q \wedge P$
- $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

P	$P \wedge P$
F	$F \wedge F = F$
T	$T \wedge T = T$

## Leis da conjunção

- $P \wedge P = P$
- $P \wedge T = P$
- $P \wedge F = F$
- $P \wedge Q = Q \wedge P$
- $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

P	$P \wedge T$
F	$F \wedge T = F$
T	$T \wedge T = T$

## Leis da conjunção

- $P \wedge P = P$
- $P \wedge T = P$
- $P \wedge F = F$
- $P \wedge Q = Q \wedge P$
- $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

P	$P \wedge F$
F	$F \wedge F = F$
T	$T \wedge F = F$

## Leis da conjunção

- $P \wedge P = P$
- $P \wedge T = P$
- $P \wedge F = F$
- $P \wedge Q = Q \wedge P$
- $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

P	Q	$P \wedge Q$	$Q \wedge P$
F	F	$F \wedge F = F$	$F \wedge F = F$
F	T	$F \wedge T = F$	$T \wedge F = F$
T	F	$T \wedge F = F$	$F \wedge T = F$
T	T	$T \wedge T = T$	$T \wedge T = T$

## Leis da conjunção

- $P \wedge P = P$
- $P \wedge T = P$
- $P \wedge F = F$
- $P \wedge Q = Q \wedge P$
- $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

P	Q	R	$P \wedge (Q \wedge R)$	$(P \wedge Q) \wedge R$
F	F	F	$F \wedge (F \wedge F) = F$	$(F \wedge F) \wedge F = F$
F	T	F	$F \wedge (T \wedge F) = F$	$(F \wedge T) \wedge F = F$
T	F	F	$T \wedge (F \wedge F) = F$	$(T \wedge F) \wedge F = F$
T	T	F	$T \wedge (T \wedge F) = F$	$(T \wedge T) \wedge F = F$
F	F	T	$F \wedge (F \wedge T) = F$	$(F \wedge F) \wedge T = F$
F	T	T	$F \wedge (T \wedge T) = F$	$(F \wedge T) \wedge T = F$
T	F	T	$T \wedge (F \wedge T) = F$	$(T \wedge F) \wedge T = F$
T	T	T	$T \wedge (T \wedge T) = T$	$T \wedge (T \wedge T) = T$

## Conjunção em C

- Sintaxe:

- (expressão) := (expressão) && (expressão)

- Semântica:  $e1 \ \&\& \ e2$

- A expressão  $e1$  é avaliada; seja  $v1$  o resultado.

- Se  $v1 = 0$ , então o resultado é  $0$ .

- Se  $v1 \neq 0$ , então a expressão  $e2$  é avaliada: seja  $v2$  o resultado.

- ✓ Se  $v2 = 0$ , então o resultado é  $0$ .

- ✓ Se  $v2 \neq 0$ , então o resultado é  $1$ .

- Atenção! Se a avaliação do primeiro operando resultar em  $0$ , o segundo operando não é avaliado.

## Algoritmo para avaliar uma conjunção C

- `e1 && e2`

```
▸ int v1 = e1;  
  int resultado;  
  if (v1 == 0)  
      resultado = 0;  
  else {  
      int v2 = e2;  
      if (e2 == 0)  
          resultado = 0;  
      else  
          resultado = 1;  
  }
```

## Algoritmo para avaliar uma conjunção C

- `e1 && e2`

```
▸ int v1 = e1;  
  int resultado;  
  if (v1 == 0)  
      resultado = 0;  
  else {  
      int v2 = e2;  
      resultado = (e2 != 0);  
  }
```

## Exemplo

```
#include <stdio.h>

int main ()
{
    int v1, v2, v3, m;
    int c1 = 4; /* coeficiente 1 */
    int c2 = 5;
    int c3 = 6;
    int pesos = c1 + c2 + c3;
    int corte1 = 7 * pesos;
    int corte2 = 3 * pesos;
    scanf("%d %d %d", &v1, &v2, &v3);
    m = c1 * v1 + c2 * v2 + c3 * v3;
    if (m < corte1 && m >= corte2)
        printf("4a prova\n"); /* corte2 <= m e m < corte1 */
    else if (m < corte2)
        printf("reprovado por nota\n"); /* m < corte2 */
    else
        printf("não reprovado por nota\n"); /* m >= corte1 */

    return 0;
}
```

## Definição da disjunção

- ▶ Seja  $P$  e  $Q$  duas condições.
- ▶ A conjunção de  $P$  e  $Q$  é denotada  $P \vee Q$ ;
- ▶  $P \vee Q = T$  se e somente se  $P = T$  ou  $Q = T$ .

P	Q	$P \vee Q$
F	F	F
F	T	T
T	F	T
T	T	T

## Leis da disjunção

- $P \vee P = P$
- $P \vee T = T$
- $P \vee F = P$
- $P \vee Q = Q \vee P$
- $P \vee (Q \vee R) = (P \vee Q) \vee R$

## Disjunção em C

- Sintaxe:

- (expressão) := (expressão)  $\|\|$  (expressão)

- Semântica:  $e1 \|\| e2$

- A expressão  $e1$  é avaliada; seja  $v1$  o resultado.

- Se  $v1 \neq 0$ , então o resultado é  $1$ .

- Se  $v1 = 0$ , então a expressão  $e2$  é avaliada: seja  $v2$  o resultado.

- ✓ Se  $v2 = 0$ , então o resultado é  $0$ .

- ✓ Se  $v2 \neq 0$ , então o resultado é  $1$ .

- Atenção! Se a avaliação do primeiro operando for diferente de 0, o segundo operando não é avaliado.

## Algoritmo para avaliar uma conjunção C

• `e1 || e2`

```
▸ int v1 = e1;  
  int resultado;  
  if (v1 != 0)  
      resultado = 1;  
  else {  
      int v2 = e2;  
      resultado = (v2 != 0);  
  }
```

## Algoritmo para avaliar uma conjunção C

•  $e1 \ || \ e2$

```
▸ int v1 = e1;
  int resultado;
  if (v1 != 0)
    resultado = 1;
  else {
    int v2 = e2;
    if (v2 != 0)
      resultado = 1;
    else
      resultado = 0;
  }
```

## Exemplo

```

#include <stdio.h>

int main ()
{
    int v1, v2, v3, faltas; /* dados do aluno */
    int ch = 60; /* carga horaria */
    int c1 = 4; /* pesos */
    int c2 = 5;
    int c3 = 6;
    int pesos = c1 + c2 + c3; /* soma dos pesos */
    int corte1 = 7 * pesos; /* pontos de corte */
    int corte2 = 3 * pesos;
    int mp; /* media parcial */
    scanf("%d %d %d", &v1, &v2, &v3);
    scanf("%d", &faltas);
    mp = c1 * v1 + c2 * v2 + c3 * v3;
    if (mp < corte2 || faltas * 4 > ch)
        printf("Reprovado");
    else if (m >= corte1)
        printf("Aprovado");
    else
        printf("4a prova\n");
    return 0;
}

```

## Definição da negação

- ▶ Seja  $P$  uma condição.
  - ▶ A negação de  $P$  é denotada  $\neg P$ ;
  - ▶  $\neg P = T$  se e somente se  $P = F$ .

$P$	$\neg P$
F	T
T	F

## Leis da negação

- $\neg\neg P = P$

- **Leis de De Morgan:**

- ▶  $\neg(P \vee Q) = \neg P \wedge \neg Q$

- ▶  $\neg(P \wedge Q) = \neg P \vee \neg Q$

# Negação em C

- Sintaxe:

- $(\text{expressão}) := ! (\text{expressão})$

- Semântica:  $!$  e

- A expressão  $e$  é avaliada; seja  $v$  o resultado.

- Se  $v = 0$ , então o resultado de  $!$  é  $1$ .

- Se  $v \neq 0$ , então o resultado é  $0$ .

# Algoritmo para avaliar uma negação C

- !e

```
▸ int v = e;  
  int resultado;  
  if (v == 0)  
      resultado = 1;  
  else  
      resultado = 0;
```

## Algoritmo para avaliar uma negação C

- !e

- `int resultado = (e == 0);`

## Precedência dos operadores

Operador	Tipo
+ -	Unário
* / %	Binário
+ -	Binário
<, >, <=, >=	Binário
==, !	Binário
!	Unário
&&	Binário
	Binário

• Como é avaliada a expressão seguinte?

2 < 3 && !4 > 5 || !3 == -1+4

## Exercício: processamento de notas e frequência

- Escreva um programa que lê dois números: CH (carga horária) e NF (número de faltas)
- Seja  $F = 100NF/CH$ .
- Se  $F$  é menor ou igual a 25 então
  - o programa lê três números entre 0 e 100: P1, P2 e P3.
  - seja  $MP = (4P1+5P2+6P3)/15$ .
  - se MP for maior ou igual a 30 e menor que 70 então:
    - ✓ o programa lê um número P4 entre 0 e 100.
    - ✓ seja  $MF = (MP+P4)/2$ .
  - se MP for maior que 70 ou menor que 3 então  $MF = MP$ .
- Se  $F$  é maior que 25 então imprime "RF".
- Se  $F$  é menor que 25 e MF é menor que 50 então imprime "RN".
- Se  $F$  é menor que 25 e MF é maior ou igual a 60 então imprime "AP".