

Operadores aritméticos, relacionais e lógicos

Linguagem C

Prof^a. Rose Yuri Shimizu

Roteiro

1 Operadores Aritméticos

2 Operadores Relacionais

3 Operadores Lógicos

Operadores Aritméticos em C

- Operador de Atribuição =

- ▶ Este é o operador usado para transferir o resultado de uma expressão para uma variável. Por exemplo:

- ★ soma = a + b;

- ★ pi = 3.1415;

- ★ letra = 'f';

- ★ Atenção: (soma = a + b) \neq (a + b = soma)

- Operadores Aritméticos:

Operador	Exemplo
+ (adição)	x = y + 1
- (subtração)	x = y - 1
/ (divisão)	x = y/z
* (multiplicação)	x = y*10
% (resto)	x = y%4
++ (mais um)	++x \rightarrow x = x + 1
-- (menos um)	--x \rightarrow x = x - 1

Expressão aritmética - Precedência dos operadores

- 1 $()$
- 2 $++$ $--$
- 3 $*$ $/$ $\%$
- 4 $+$ $-$
- 5 $=$

- Exemplo:

- ▶ $a * b / c + 30 - ++ e$
- ▶ $(((a * b) / c) + 30) - (++ e)$

Vamos praticar

- 1 Faça um algoritmo com duas variáveis inteiras, com os valores 12 e 33, imprima seus valores e a soma deles na tela .
 - 2 Faça um algoritmo que imprima a mensagem “Programar é fácil!”.
 - 3 Faça um algoritmo com uma variável do tipo caractere para cada letra do seu nome e imprima a mensagem: “Meu nome é Fulano”. Sendo que o seu nome deve ser impresso através do conteúdo das variáveis.
- Desenvolvimento de Programa
 - 1 Leia e interprete o problema;
 - 2 Separe:
 - ★ Dados e valores;
 - ★ Ações;
 - 3 Desenvolva o algoritmo: pense no passo a passo para a resolução;
 - 4 Implemente o programa.

Vamos praticar

- Imprima o resultado das expressões e, se necessário, declare as variáveis e atribua os valores indicados:

- ▶ $\frac{20 + 5 \times 9}{10}$

- ▶ $b \times \frac{a}{2}$ (a valendo -13 e b valendo 7)

- ▶ $\frac{a \times 4}{2} + p \times 6$ (a valendo 3 e p valendo 23)

Vamos praticar

- Imprima o resultado das expressões e, se necessário, declare as variáveis e atribua os valores indicados:

▶ $\frac{20 + 5 \times 9}{10}$

▶ $b \times \frac{a}{2}$ (a valendo -13 e b valendo 7)

▶ $\frac{a \times 4}{2} + p \times 6$ (a valendo 3 e p valendo 23)

```
1 #include <stdio.h>
2
3 int main(){
4     printf("%d\n", (20+5*9)/10);
5
6     int a=3;
7     int p=23;
8     printf("%d\n", (a*4)/2+p*6);
9
10    int a=-13;
11    int b=7;
12    printf("%d \n", b*(a/2));
13
14    return 0;
15 }
```

Vamos praticar

- Imprima o resultado das expressões e, se necessário, declare as variáveis e atribua os valores indicados:
 - ▶ $a = a + 1$ (a valendo 111)
 - ▶ $b = b/3$ (b valendo 9)
 - ▶ $b = b + 2$ (b valendo 9)

Vamos praticar

- Imprima o resultado das expressões e, se necessário, declare as variáveis e atribua os valores indicados:
 - ▶ $a = a + 1$ (a valendo 111)
 - ▶ $b = b/3$ (b valendo 9)
 - ▶ $b = b + 2$ (b valendo 9)

```
1 #include <stdio.h>
2
3 int main() {
4     int a=111;
5     a++; //a=a+1; ou a+=1;
6     printf("%d \n",a);
7
8     int b=9;
9     b/=3; //b=b/3;
10    printf("%d \n",b);
11
12    int b=9;
13    b+=2; //b=b+2;
14    printf("%d \n",b);
15    return 0;
16 }
```

Operações aritméticas com inteiros

- Divisão, não inteira, de inteiros

```
1 #include <stdio.h>
2
3 int main()
4 {
5     float x,y;
6     x=5/2;
7     y=2/5;
8     printf("%f %f\n",x,y);
9     return 0;
10 }
```

- Saída: 2.000000 0.000000
- Observe:
 - ▶ $5/2$ retornou 2 ao invés de 2.5
 - ▶ $2/5$ retornou 0 ao invés de 0.4
- Como 5 e 2 são constantes do tipo inteiro, a divisão é convertida para inteiro, ignorando a parte decimal.

Operações aritméticas com reais

- Divisão, não inteira, de inteiro e real

```
1 #include <stdio.h>
2
3 int main()
4 {
5     float x,y;
6     x=5.0/2;
7     y=2/5.0;
8     printf("%f %f\n",x,y);
9     return 0;
10 }
```

- Saída: 2.500000 0.400000
- Observe:
 - ▶ $5/2$ retornou 2.5
 - ▶ $2/5$ retornou 0.4
- Basta um dos números da expressão ser constante do tipo real, para a divisão resultar em um número real.

Operações aritméticas atribuídas a uma variável inteira

- Operações, não inteiras, atribuídas a uma variável inteira

```
1 #include <stdio.h>
2
3 int main() {
4     int x,y;
5     x=5.0/2.0;
6     printf("%d\n",x);
7     //printf("%f\n", (float)x);
8
9     x=5-2.1;
10    printf("%d\n",x);
11    //printf("%f\n", (float)y);
12
13    return 0;
14 }
```

- Todas as saídas serão igual a 2
- Observe:
 - ▶ 5.0/2.0 retorna 2 ao invés de 2.5
 - ▶ 5 - 2.1 retorna 2 ao invés de 2.9
- Como a variável x é do tipo inteiro, qualquer operação será convertida para inteiro, ignorando a parte decimal.

Área do Círculo

Adaptado por Neilor Tonin, URI  Brasil

Timelimit: 1

A fórmula para calcular a área de uma circunferência é: $\text{area} = \pi \cdot \text{raio}^2$. Considerando para este problema que $\pi = 3.14159$:

- Efetue o cálculo da área, elevando o valor de **Raio** ao quadrado e multiplicando por π .

Entrada

A entrada contém um valor de ponto flutuante (dupla precisão), no caso, a variável **raio**.

Saída

Apresentar a mensagem "A=" seguido pelo valor da variável **area**, conforme exemplo abaixo, com 4 casas após o ponto decimal. Utilize variáveis de dupla precisão (double). Como todos os problemas, não esqueça de imprimir o fim de linha após o resultado, caso contrário, você receberá "Presentation Error".

Exemplos de Entrada	Exemplos de Saída
2.00	A=12.5664
100.64	A=31819.3103
150.00	A=70685.7750

Vamos praticar - Área do círculo

```
1 #include <stdio.h>
2
3 int main() {
4     //Declaracao de variaveis
5     double raio;
6
7     //Inicializacao de variaveis pela entrada padrao
8     scanf("%lf",&raio);
9
10    //Saida
11    printf("A=%.4lf\n", raio*raio*3.14159);
12
13    return 0;
14 }
```

Vamos praticar - Média

Leia 2 valores de ponto flutuante de dupla precisão A e B, que correspondem a 2 notas de um aluno. A seguir, calcule a média do aluno, sabendo que a nota A tem peso 3.5 e a nota B tem peso 7.5 (A soma dos pesos portanto é 11). Assuma que cada nota pode ir de 0 até 10.0, sempre com uma casa decimal.

Entrada

O arquivo de entrada contém 2 valores com uma casa decimal cada um.

Saída

Calcule e imprima a variável **MEDIA** conforme exemplo abaixo, com 5 dígitos após o ponto decimal e com um espaço em branco antes e depois da igualdade. Utilize variáveis de dupla precisão (double) e como todos os problemas, não esqueça de imprimir o fim de linha após o resultado, caso contrário, você receberá "Presentation Error".

Exemplos de Entrada	Exemplos de Saída
5.0 7.1	MEDIA = 6.43182
0.0 7.1	MEDIA = 4.84091
10.0 10.0	MEDIA = 10.00000

Vamos praticar - Média

```
1 #include <stdio.h>
2
3 int main() {
4     //Declaracao de variaveis
5     double a, b;
6
7     //Inicializacao de variaveis pela entrada padrao
8     scanf("%lf",&a);
9     scanf("%lf",&b);
10
11     //Saida
12     printf("MEDIA = %.5lf\n", (a*3.5+7.5*b)/11);
13
14     return 0;
15 }
16
```

Vamos praticar - Volume

Faça um programa que calcule e mostre o volume de uma esfera sendo fornecido o valor de seu raio (R). A fórmula para calcular o volume é: $(4/3) * \pi * R^3$. Considere (atribua) para pi o valor 3.14159.

Dica: Ao utilizar a fórmula, procure usar (4/3.0) ou (4.0/3), pois algumas linguagens (dentre elas o C++), assumem que o resultado da divisão entre dois inteiros é outro inteiro.

Entrada

O arquivo de entrada contém um valor de ponto flutuante (dupla precisão), correspondente ao raio da esfera.

Saída

A saída deverá ser uma mensagem "VOLUME" conforme o exemplo fornecido abaixo, com um espaço antes e um espaço depois da igualdade. O valor deverá ser apresentado com 3 casas após o ponto.

Exemplos de Entrada	Exemplos de Saída
3	VOLUME = 113.097
15	VOLUME = 14137.155
1523	VOLUME = 14797486501.627

Vamos praticar - Volume

```
1 #include <stdio.h>
2
3 int main() {
4     //Declaracao de variaveis
5     int R;
6
7     //Inicializacao de variaveis pela entrada padrao
8     scanf("%d", &R);
9
10    //Saida
11    printf("VOLUME = %.31f\n", (4.0/3)*3.14159*R*R*R);
12
13    return 0;
14 }
15
```

Vamos praticar - Fórmula Maior

Faça um programa que leia três valores e apresente o maior dos três valores lidos seguido da mensagem "eh o maior". Utilize a fórmula:

$$\text{MaiorAB} = \frac{(a+b+abs(a-b))}{2}$$

Entrada

O arquivo de entrada contém três valores inteiros.

Saída

Imprima o maior dos três valores seguido por um espaço e a mensagem "eh o maior".

Exemplos de Entrada	Exemplos de Saída
7 14 106	106 eh o maior
217 14 6	217 eh o maior

Obs: Módulo de um número - função **abs()** - inclua o cabeçalho **stdlib.h**

Vamos praticar - Fórmula Maior

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     //Declaracao de variaveis
6     int A, B, C, maior;
7
8     //Inicializacao de variaveis pela entrada padrao
9     scanf("%d%d%d", &A, &B, &C);
10
11    //Inicializacao de variavel por atribuicao
12    maior = ((A+B)+abs(A-B))/2;
13    maior = ((maior+C)+abs(maior-C))/2;
14
15    //Saida
16    printf("%d eh o maior\n", maior);
17
18    return 0;
19 }
20
```

Vamos praticar - Minúsculo

- Faça o algoritmo que leia um caractere e imprima o seu minúsculo.
 - ▶ Dica:
 - ★ 'A' é o decimal 65 na tabela ASCII
 - ★ 'a' é o decimal 97 na tabela ASCII
 - ▶ Entrada:
 - ★ A entrada é composta por um caractere maiúsculo.
 - ▶ Saída:
 - ★ Seu programa deve imprimir uma linha contendo o correspondente minúsculo do caractere lido.
 - ▶ Salvar `minusculo.c`
 - ▶ Compilar: `gcc minusculo.c -o minusculo`
 - ▶ Executar: `./minusculo`

Vamos praticar - Minúsculo

```
1 #include <stdio.h>
2
3 int main()
4 {
5     //Declaracao de variaveis
6     char letra;
7
8     //Inicializacao de variaveis pela entrada padrao
9     scanf("%c", &letra);
10
11     //Saida
12     printf("%c\n", letra+32);
13
14     return 0;
15 }
```

Vamos praticar - Último algarismo

- Faça o algoritmo de achar o último algarismo dado um número inteiro.
 - ▶ Dica:
 - ★ Resto % $\rightarrow 3\%2 = 1$
 - ★ Divisão / $\rightarrow 33/10 = 3$
 - ▶ Entrada:
 - ★ A entrada é composta por um número inteiro.
 - ▶ Saída:
 - ★ Seu programa deve imprimir uma linha contendo um inteiro que representa o último algarismo do número de entrada.
 - ▶ Salvar ultimo.c
 - ▶ Compilar: `gcc ultimo.c -o ultimo`
 - ▶ Executar: `./ultimo`

Vamos praticar - Último algoritmo

```
1 #include <stdio.h>
2
3 int main()
4 {
5     //Declaracao de variaveis
6     int num;
7
8     //Inicializacao de variaveis pela entrada padrao
9     scanf("%d", &num);
10
11     //Saida
12     printf("%d\n", num%10);
13
14     return 0;
15 }
16
```

Roteiro

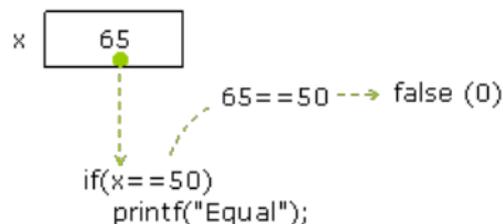
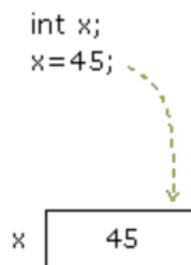
1 Operadores Aritméticos

2 Operadores Relacionais

3 Operadores Lógicos

Operadores Relacionais

- Compara a relação entre dois valores ou expressões;
- Tipos
 - ▶ < Menor
 - ▶ > Maior
 - ▶ <= Menor ou igual
 - ▶ >= Maior ou igual
 - ▶ == Igual
 - ▶ != Diferente
- Em C: == é diferente de =
 - ▶ = é atribuição, == é comparação



Expressão Relacional

- Combinação de constantes, variáveis e operadores;
- O resultado é 1 (verdade) ou 0 (falso);
- Precedência dos operadores:
 - 1 ()
 - 2 * / %
 - 3 + -
 - 4 < > <= >=
 - 5 == !=
 - 6 =

Expressão Relacional - Exemplos

```
1 #include <stdio.h>
2 int main()
3 {
4     int a, b, c, d;
5     a = 40 < 20; //atribui 0 (falso) para a
6     b = 40 != 20; //atribui 1 (verdade) para b
7     c = a != b; //atribui 1 (verdade) para c
8     d = b == c; //atribui 1 (verdade) para d
9
10    printf("%d, %d, %d, %d\n", a, b, c, d);
11
12    return 0;
13 }
14 //Saida: 0, 1, 1, 1
15
```

Expressão Relacional - Exemplos

```
1 #include <stdio.h>
2 int main()
3 {
4     int a=45, b=30, c=30, d;
5     d = ( b == c != a );
6     printf("%d\n",d);
7
8     return 0;
9 }
10 //Saida: 1
11
```

1 ()

2 * / %

3 + -

4 < > <= >=

5 == !=

6 =

Expressão Relacional - Exemplos

```
1 #include <stdio.h>
2 int main()
3 {
4     int a=45, b=30, c=30, d;
5     d = ( b == c != a );
6     printf("%d\n",d);
7
8     return 0;
9 }
10 //Saida: 1
11
```

- 1 ()
- 2 * / %
- 3 + -
- 4 < > <= >=
- 5 == !=
- 6 =

d = (b == c != a);

1

1

Expressão Relacional - Exemplos

```
1 #include <stdio.h>
2 int main()
3 {
4     int a=10, b=20, c=30, d;
5     d = c == a+b;
6     printf("%d\n",d);
7
8     return 0;
9 }
10 //Saida: ?
11
```

- 1 ()
- 2 * / %
- 3 + -
- 4 < > <= >=
- 5 == !=
- 6 =

Expressão Relacional - Exemplos

```
1 #include <stdio.h>
2 int main()
3 {
4     int a=10, b=20, c=30, d;
5     d = c == a+b;
6     printf("%d\n",d);
7
8     return 0;
9 }
10 //Saida: ?
11
```

- 1 ()
- 2 * / %
- 3 + -
- 4 < > <= >=
- 5 == !=
- 6 =

The diagram shows the expression `d = c == a + b;` with dashed lines and arrows indicating the order of evaluation. A bracket under `a + b` points to the value `30`. Another bracket under `c == 30` points to the value `1`. A final arrow points from `1` to `d`, showing that `d` is assigned the value `1`.

Expressão Relacional - Exemplos

```
1 #include <stdio.h>
2 int main()
3 {
4     int a=10, b=20, c=0, d;
5     d = c == a < b;
6     printf("%d\n",d);
7
8     return 0;
9 }
10 //Saida: ?
11
```

- 1 ()
- 2 * / %
- 3 + -
- 4 < > <= >=
- 5 == !=
- 6 =

Expressão Relacional - Exemplos

```
1 #include <stdio.h>
2 int main()
3 {
4     int a=10, b=20, c=0, d;
5     d = c == a < b;
6     printf("%d\n",d);
7
8     return 0;
9 }
10 //Saida: ?
11
```

- 1 ()
- 2 * / %
- 3 + -
- 4 < > <= >=
- 5 == !=
- 6 =

d = c == a < b;

1

0

Roteiro

1 Operadores Aritméticos

2 Operadores Relacionais

3 Operadores Lógicos

Lógica proposicional

- Lógica proposicional através da álgebra booleana (George Boole, matemático inglês)
- Formalizar matematicamente o pensamento humano através da lógica:
 - ▶ Definindo proposições (declarações afirmativas)
 - ★ A: Gato é mamífero
 - ★ B: Tom é gato
 - ★ A implica B: Tom ser gato implica que é mamífero
 - ▶ Formulando expressões através da combinação das proposições que conclui-se, como verdadeira ($\neq 0$) ou falsa (0)
 - ▶ Operações básicas: **OU**, **E** e **NÃO**
 - ▶ Outras: **OU EXCLUSIVO**, **IMPLICA**, **SSE (se e somente se)**
 - ★ Possuem equivalências lógicas utilizando os operadores básicos
- Objetivo: deduzir uma fórmula como verdadeira ou falsa
- Programação:
 - ▶ As proposições são associados simbolicamente aos valores de variáveis ou constantes
 - ▶ Valores representativos: se 0 (zero) é considerado falso; se diferente de 0 (zero) é considerado verdadeiro

Operação lógica: negação/inverso - NÃO

- Notação lógica: \neg ou \sim
 - ▶ P: Está chovendo
 - ▶ $\neg P$: Não está chovendo
- Operador lógico em C: !

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 int chuva = 3; //Leitura de um sensor, por exemplo
3
4 //Portanto, !chuva, significa "Não está chovendo"
5 printf("%d %d\n", !chuva, !!chuva); //Saídas?
6
7 //Variável "num" recebe um número do teclado
8 int num;
9 scanf("%d", &num);
10 printf("Par: %d\n", !(num%2)); //Se par, qual a saída?
11                               //Se ímpar, qual a saída?
```

- Tabela verdade (2^1 linhas - combinações de V e F):

P	$\neg P$
V	F
F	V

Operação lógica: conjunção - E

- Notação lógica: \wedge
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \wedge Q = V$, se e somente se $P=V$ e $Q=V$
 - ★ TODAS as condições tem que ser verdadeiras
- Operador lógico em C: **&&**

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 0;
5
6 //Irigar se não estiver chovendo e o solo estiver seco
7 printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
V	V	F	F	

Operação lógica: conjunção - E

- Notação lógica: \wedge
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \wedge Q = V$, se e somente se $P=V$ e $Q=V$
 - ★ TODAS as condições tem que ser verdadeiras
- Operador lógico em C: **&&**

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 0;
5
6 //Irigar se não estiver chovendo e o solo estiver seco
7 printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
V	V	F	F	F
V	F	F	V	

Operação lógica: conjunção - E

- Notação lógica: \wedge
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \wedge Q = V$, se e somente se $P=V$ e $Q=V$
 - ★ TODAS as condições tem que ser verdadeiras
- Operador lógico em C: **&&**

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 0;
5
6 //Irigar se não estiver chovendo e o solo estiver seco
7 printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
V	V	F	F	F
V	F	F	V	F
F	V	V	F	F
F	F	V	V	V

Operação lógica: conjunção - E

- Notação lógica: \wedge
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \wedge Q = V$, se e somente se $P=V$ e $Q=V$
 - ★ TODAS as condições tem que ser verdadeiras
- Operador lógico em C: **&&**

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 0;
5
6 //Irigar se não estiver chovendo e o solo estiver seco
7 printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
V	V	F	F	F
V	F	F	V	F
F	V	V	F	F
F	F	V	V	V

Operação lógica: conjunção - E

- Notação lógica: \wedge
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \wedge Q = V$, se e somente se $P=V$ e $Q=V$
 - ★ TODAS as condições tem que ser verdadeiras
- Operador lógico em C: **&&**

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 0;
5
6 //Irigar se não estiver chovendo e o solo estiver seco
7 printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
V	V	F	F	F
V	F	F	V	F
F	V	V	F	F
F	F	V	V	V

Operação lógica: disjunção - OU

- Notação lógica: \vee

- ▶ P: *Está chovendo* e Q: *Solo está úmido*

- ▶ $P \vee Q = V$, se $P=V$ ou $Q=V$

- ★ ALGUMA das condições tem que ser verdadeiras

- Operador lógico em C: `||`

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 2;
5
6 //Iniciar adubação somente com umidade
7 printf("%d\n", chuva || solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \vee Q$
V	V	

Operação lógica: disjunção - OU

- Notação lógica: \vee
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \vee Q = V$, se $P=V$ ou $Q=V$
 - ★ ALGUMA das condições tem que ser verdadeiras
- Operador lógico em C: `||`

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 2;
5
6 //Iniciar adubação somente com umidade
7 printf("%d\n", chuva || solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \vee Q$
V	V	V
V	F	

Operação lógica: disjunção - OU

- Notação lógica: \vee
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \vee Q = V$, se $P=V$ ou $Q=V$
 - ★ ALGUMA das condições tem que ser verdadeiras
- Operador lógico em C: `||`

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 2;
5
6 //Iniciar adubação somente com umidade
7 printf("%d\n", chuva || solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	F

Operação lógica: disjunção - OU

- Notação lógica: \vee
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \vee Q = V$, se $P=V$ ou $Q=V$
 - ★ ALGUMA das condições tem que ser verdadeiras
- Operador lógico em C: `||`

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 2;
5
6 //Iniciar adubação somente com umidade
7 printf("%d\n", chuva || solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	

Operação lógica: disjunção - OU

- Notação lógica: \vee
 - ▶ P: *Está chovendo* e Q: *Solo está úmido*
 - ▶ $P \vee Q = V$, se $P=V$ ou $Q=V$
 - ★ ALGUMA das condições tem que ser verdadeiras
- Operador lógico em C: `||`

```
1 //Suponha "chuva" > 0, significa que "Está chovendo"
2 //Suponha "solo" > 0, significa que "Solo está úmido"
3 int chuva = 1;
4 int solo = 2;
5
6 //Iniciar adubação somente com umidade
7 printf("%d\n", chuva || solo); //Saída?
```

- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	F

Expressão Lógica - Precedência dos operadores

- 1 ()
- 2 !
- 3 * / %
- 4 + -
- 5 < > <= >=
- 6 == !=
- 7 &&
- 8 ||
- 9 =

Expressão Lógica - Exemplo

- $a \parallel ! (b \&\& c)$
- Tabela verdade (2^3 linhas - combinações de V e F)

a	b	c	$b \&\& c$	$! (b \&\& c)$	$a \parallel ! (b \&\& c)$
V	V	V	V	F	V
V	V	F	F	V	V
V	F	V	F	V	V
V	F	F	F	V	V
F	V	V	V	F	F
F	V	F	F	V	V
F	F	V	F	V	V
F	F	F	F	F	F

- Outras operações lógicas:
 - ▶ OU EXCLUSIVO, IMPLICA, BI-IMPLICAÇÃO
 - ▶ Equivalências lógicas: duas expressões são logicamente equivalentes quando são compostas pelas mesmas proposições simples e os mesmos resultados de suas tabelas-verdade

Expressão Relacional e Lógica - Exemplos

```
1 #include <stdio.h>
2 int main() {
3     //Passou na disciplina
4     int nota, frequencia;
5     scanf("%d", &nota);
6     scanf("%d", &frequencia);
7
8     printf("%d\n", nota >= 50 && frequencia > 74);
9
10    return 0;
11 }
12 //Entrada: 45 80
13 //Saida: ?
```

Operadores lógicos bit a bit (bitwise)

Operador	Operação
&	E
	OU
^	OU EXCLUSIVO
~	não

bit a	bit b	a & b	a b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

```
i = 7; //0000 0111
j = 3; //0000 0011

k = i & j; //0000 0111 &
           //0000 0011
           //-----
           //???? ????

printf("%d\n", k);
```

```
k = i | j; //0000 0111 |
           //0000 0011
           //-----
           //???? ????

printf("%d\n", k);
```

```
k = i ^ j; //0000 0111 ^
           //0000 0011
           //-----
           //???? ????

printf("%d\n", k);

k = -1; //1000...0000 0001
printf("%d\n", ~k);
```

Operadores lógicos bit a bit (bitwise)

- $p \ll i$: deslocar de p , i bits para esquerda (left shift)
 - ▶ Equivale a multiplicar por 2^i
- $p \gg i$: deslocar de p , i bits para direita (right shift)
 - ▶ Equivale a dividir por 2^i

```
1 int i = 14; //00001110
2 int j = i >> 1; //14/2 : 00001110 : 00000111 = 7
3
4 i = 7; //0000 0111
5 j = 3; //0000 0011
6 int k = (i << j); //7*2^3 = 7*8 = 56
7 //0011 1000 = 0000 0111 << 0000 0011
```