Linguagem C
Operadores aritméticos, relacionais e lógicos

Profª. Rose Yuri Shimizu

Roteiro

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

Exemplos

- $\frac{20 + 5 \times 9}{10}$
- $b \times \frac{a}{2}$ (a valendo -13 e b valendo 7)
- $\frac{a \times 4}{2} + p \times 6$ (a valendo 3 e p valendo 23)

```c
#include <stdio.h>

int main()
{
    printf("%d\n",(20+5*9)/10);

    int a=3;
    int p=23;
    printf("%d\n",(a*4)/2+p*6);

    int a=-13;
    int b=7;
    printf("%d \n",b*(a/2));

    return 0;
}
```

Exemplos

- $a = a + 1$ (a valendo 111)
- $b = b/3$ (b valendo 9)
- $b = b + 2$ (b valendo 9)

```c
#include <stdio.h>

int main()
{
    int a=111;
    a++; //a=a+1; ou a+=1;
    printf("%d \n",a);

    int b=9;
    b/=3; //b=b/3;
    printf("%d \n",b);

    int b=9;
    b+=2; //b=b+2;
    printf("%d \n",b);
    return 0;
}
```

Operações aritméticas com inteiros

- Divisão, não inteira, de inteiros

```c
#include <stdio.h>

int main()
{
    float x,y;
    x=5/2;
    y=2/5;
    printf("%f %f\n",x,y);
    return 0;
}
```

- 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

```c
#include <stdio.h>

int main()
{
    float x,y;
    x=5.0/2;
    y=2/5.0;
    printf("%f %f\n",x,y);
    return 0;
}
```

- 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

```c
#include <stdio.h>

int main() {
    int x,y;
    x=5.0/2.0;
    printf("%d\n",x);
    //printf("%f\n", (float)x);

    x=5-2.1;
    printf("%d\n",x);
    //printf("%f\n", (float)y);

    return 0;
}
```

- 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.

Vamos praticar

Área do Círculo

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

A fórmula para calcular a área de uma circunferência é: **area = π . raio²**. Considerando para este problema que **π** = 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

```c
#include <stdio.h>

int main() {
    //Declaracao de variaveis
    double raio;

    //Inicializacao de variaveis pela entrada padrao
    scanf("%lf",&raio);

    //Saida
    printf("A=%.4lf\n", raio*raio*3.14159);

    return 0;
}
```

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

```c
#include <stdio.h>

int main() {
    //Declaracao de variaveis
    double a, b;

    //Inicializacao de variaveis pela entrada padrao
    scanf("%lf",&a);
    scanf("%lf",&b);

    //Saida
    printf("MEDIA = %.5lf\n", (a*3.5+7.5*b)/11);

    return 0;
}

```

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

```c
#include <stdio.h>

int main() {
    //Declaracao de variaveis
    int R;

    //Inicializacao de variaveis pela entrada padrao
    scanf("%d", &R);

    //Saida
    printf("VOLUME = %.3lf\n", (4.0/3)*3.14159*R*R*R);

    return 0;
}

```

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:

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

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

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

int main() {
    //Declaracao de variaveis
    int A, B, C, maior;

    //Inicializacao de variaveis pela entrada padrao
    scanf("%d%d%d", &A, &B, &C);

    //Inicializacao de variavel por atribuicao
    maior = ((A+B)+abs(A-B))/2;
    maior = ((maior+C)+abs(maior-C))/2;

    //Saida
    printf("%d eh o maior\n", maior);

    return 0;
}

```

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

```c
#include <stdio.h>

int main()
{
    //Declaracao de variaveis
    char letra;

    //Inicializacao de variaveis pela entrada padrao
    scanf("%c", &letra);

    //Saida
    printf("%c\n", letra+32);

    return 0;
}
```

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

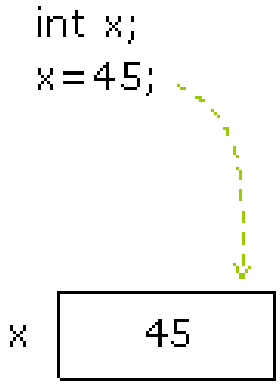
Vamos praticar - Último algarismo

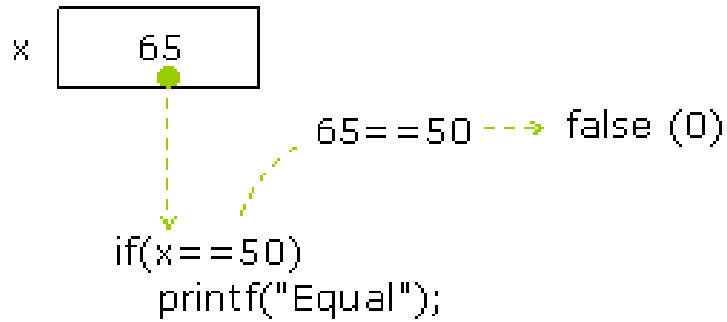
```c
#include <stdio.h>

int main()
{
    //Declaracao de variaveis
    int num;

    //Inicializacao de variaveis pela entrada padrao
    scanf("%d", &num);

    //Saida
    printf("%d\n", num%10);

    return 0;
}

```

Roteiro

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

```c
#include <stdio.h>
int main()
{
    int a, b, c, d;
    a = 40 < 20; //atribui 0 (falso) para a
    b = 40 != 20;//atribui 1 (verdade) para b
    c = a != b;  //atribui 1 (verdade) para c
    d = b == c;  //atribui 1 (verdade) para d

    printf("%d, %d, %d, %d\n",a,b,c,d);

    return 0;
}
//Saida: 0, 1, 1, 1

```

Expressão Relacional - Exemplos

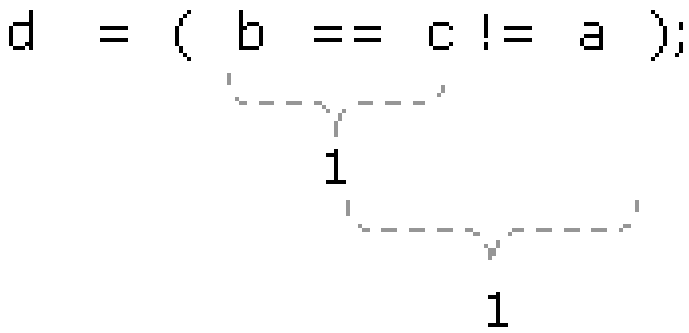
```c
#include<stdio.h>
int main()
{
    int a=45, b=30, c=30, d;
    d = ( b == c != a );
    printf("%d\n",d);

    return 0;
}
//Saida: 1

```

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

Expressão Relacional - Exemplos

```c
#include<stdio.h>
int main()
{
    int a=45, b=30, c=30, d;
    d = ( b == c != a );
    printf("%d\n",d);

    return 0;
}
//Saida: 1

```

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



Expressão Relacional - Exemplos

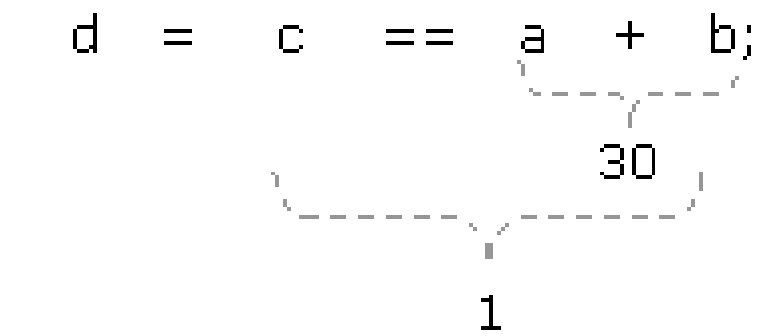
```c
#include<stdio.h>
int main()
{
    int a=10, b=20, c=30, d;
    d = c == a+b;
    printf("%d\n",d);

    return 0;
}
//Saida: ?

```

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

Expressão Relacional - Exemplos

```c
#include<stdio.h>
int main()
{
    int a=10, b=20, c=30, d;
    d = c == a+b;
    printf("%d\n",d);

    return 0;
}
//Saida: ?

```

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



Expressão Relacional - Exemplos

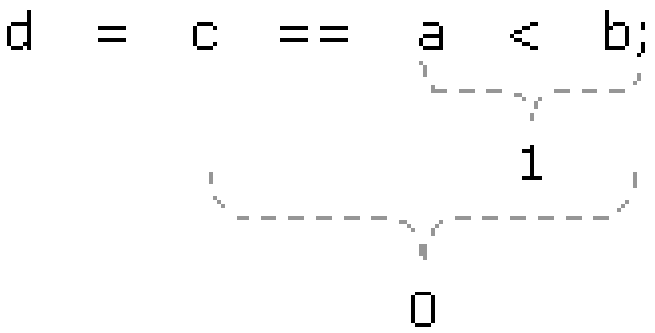
```c
#include<stdio.h>
int main()
{
    int a=10, b=20, c=0, d;
    d = c == a < b;
    printf("%d\n",d);

    return 0;
}
//Saida: ?

```

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

Expressão Relacional - Exemplos

```c
#include<stdio.h>
int main()
{
    int a=10, b=20, c=0, d;
    d = c == a < b;
    printf("%d\n",d);

    return 0;
}
//Saida: ?

```

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



Expressão Relacional - Exemplos

```c
#include <stdio.h>
int main()
{
    int a, b;

    //verificar intervalo?
    a = 40;
    b = 20 <= a <= 50;

    printf("%d\n", b); //saída?

    b = 20 <= a <= 10; //saída?

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

    return 0;
}

```

Roteiro

Lógica proposicional

- Operadores da lógica proposicional
- Álgebra booleana
  - Formula expressões através da combinação das proposições que conclui-se, como verdadeira (!= 0) ou falsa (0)
  - Operações básicas: **OU**, **E** e **NÃO**
- Objetivo: deduzir uma fórmula como verdadeira ou falsa
- Programação:
  - As proposições são associados simbolicamente aos valores de variáveis ou constantes
  - 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: **!**

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
int chuva = 3; //Leitura de um sensor, por exemplo

//Portanto, !chuva, significa "Não está chovendo"
printf("%d %d\n", !chuva, !!chuva); //Saídas?

//Variável "num" recebe um número do teclado
int num;
scanf("%d", &num);
printf("Par: %d\n", !(num%2)); //Se par, qual a saída?
                               //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: **&&**
- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \wedge Q$
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: **&&**
- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \wedge Q$
V	V	V
V	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: **&&**
- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \wedge Q$
V	V	V
V	F	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: **&&**
- Tabela verdade (2^2 linhas - combinações de V e F)

P	Q	$P \wedge Q$
V	V	V
V	F	F
F	V	F
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: **&&**
- Tabela verdade (2^2 linhas - combinações de V e F)

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

Operação lógica: conjunção - E

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 0;

//Irrigar se não estiver chovendo e o solo estiver seco
printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

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

Operação lógica: conjunção - E

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 0;

//Irrigar se não estiver chovendo e o solo estiver seco
printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

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

Operação lógica: conjunção - E

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 0;

//Irrigar se não estiver chovendo e o solo estiver seco
printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

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	

Operação lógica: conjunção - E

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 0;

//Irrigar se não estiver chovendo e o solo estiver seco
printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

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	

Operação lógica: conjunção - E

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 0;

//Irrigar se não estiver chovendo e o solo estiver seco
printf("Ativar: %d\n", !chuva && !solo); //Saída?
```

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: ||

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 2;

//Iniciar adubação somente com umidade
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: ||

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 2;

//Iniciar adubação somente com umidade
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: ||

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 2;

//Iniciar adubação somente com umidade
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	

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: ||

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 2;

//Iniciar adubação somente com umidade
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: ||

```c
//Suponha "chuva" > 0, signifca que "Está chovendo"
//Suponha "solo" > 0, signifca que "Solo está úmido"
int chuva = 1;
int solo = 2;

//Iniciar adubação somente com umidade
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 || ! (b && c)
- Tabela verdade (2^3 linhas - combinações de V e F)

a	b	c	b && c	! (b && c)	a || ! (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

Expressão Relacional e Lógica - Exemplos

```c
#include <stdio.h>
int main() {
    //Passou na disciplina
    int nota, frequencia;
    scanf("%d", &nota);
    scanf("%d", &frequencia);

    printf("%d\n", nota>=50 && frequencia>74);

    return 0;
}
//Entrada: 45 80
//Saida: ?
```

Expressão Relacional e Lógica - Exemplos

- Equivalências lógicas:
  - Duas expressões são logicamente equivalentes quando são compostas pelas mesmas proposições simples e os mesmos resultados de suas tabelas-verdade
  - "Não executar se `i==a && j==b`":
    - `!(i==a && i==b)`
  - "Executar se i==a || j==b":
    - `i!= a || i!=b`
    - `!(i==a) || !(i==b)`
  - Negação da conjunção: `!(A && B)` ≡ `(!A || !B)`

`i==a`	`i==b`	`!(i==a)`	`!(i==b)`	`!(i==a && i==b)`	`!(i==a) || !(i==b)`
V	V	F	F	F	F
V	F	F	V	V	V
F	V	V	F	V	V
F	F	V	V	V	V

Expressão Relacional e Lógica - Exemplos

- Equivalências lógicas:
  - Negação da disjunção: `!(A || B)` $\equiv$ `(!A && B)`

A	B	!A	!B	!(A || B)	!A && !B
V	V	F	F	F	F
V	F	F	V	F	F
F	V	V	F	F	F
F	F	V	V	V	V

Operadores lógicos bit a bit (bitwise)

Operador	Operação
&	E
|	OU
$\wedge$	OU EXCLUSIVO
$\sim$	não

bit a	bit b	a & b	a | b	$a \wedge b$
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 « i: deslocar de p, i bits para esquerda (left shift)
  - Equivale a multiplar por 2^i
- p » 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
```