



# ALGORITMOS

PROF<sup>ª</sup>: SIMONE DOMINICO

## ALGORITMO – VARIÁVEIS E CONSTANTES

## TIPOS DE DADOS

- Algoritmos são compostos por instruções e dados:
  - Instruções manipulam os dados para chegar ao resultado desejado.
  - Dados possuem tipos, que determinam o que se pode fazer com eles.
  - Trabalharemos com os tipos mais comuns de dados, presentes na grande maioria das linguagens.

## DADOS NUMÉRICOS

- Tornando ao aspecto computacional, os dados numéricos representáveis em um computador são divididos em apenas duas classes:
- **INTEIROS**
  - Os números inteiros são aqueles que não possuem componentes decimais ou fracionários, podendo ser positivos ou negativos.
- **REAIS**
  - Os dados de tipo REAL são aqueles que podem possuir componentes decimais ou fracionários, e podem também ser positivos ou negativos.

## DADOS LITERAIS

- Sequência de caracteres contendo qualquer letra, dígito ou símbolo que o computador puder representar;
- Também conhecido como alfanumérico, cadeia de caracteres ou, em inglês, string;
- Delimitado por aspas duplas ("..."), possui comprimento igual ao número de caracteres;
- Exemplos:
- "Asdfg", "PBC", "\$ 200", "123", "1-2+3="

## CARACTER (CHAR)

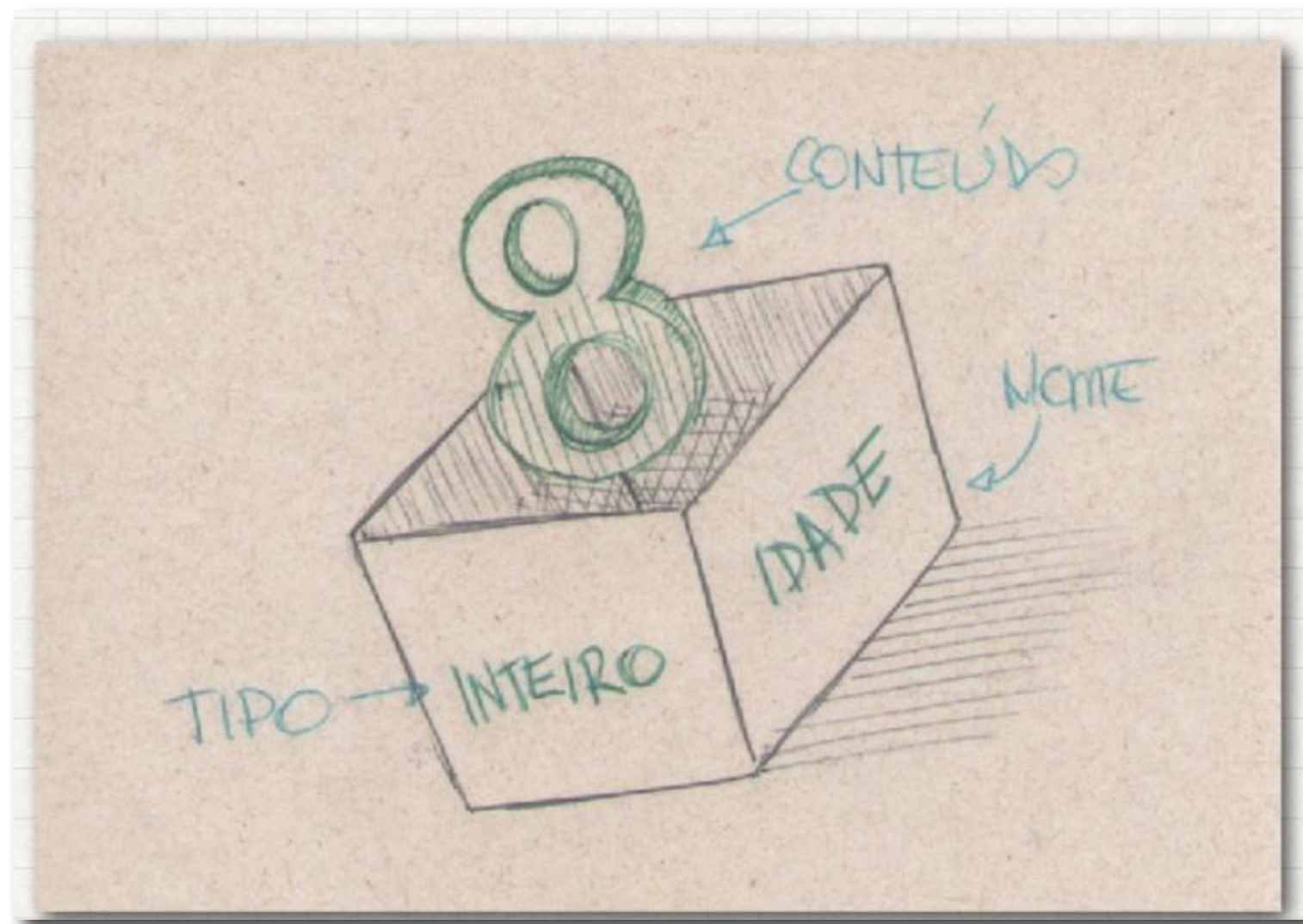
- Consiste de um único símbolo ou de uma concatenação de símbolos do alfabeto
- Os elementos do conjunto de valores do tipo caractere devem ser escritos, nos algoritmos, entre aspas duplas, como, por exemplo, "a".
- "Esta e uma frase formada por caracteres". Há um elemento especial, "", denominado de palavra vazia, pois não possui nenhum símbolo.

## DADOS LÓGICOS

- Representa os dois únicos valores lógicos possíveis: VERDADEIRO e FALSO;
- Também conhecido como booleano, graças ao matemático George Boole; Representaremos como .V. e .F..

## TIPOS DE DADOS EM LINGUAGEM C

TIPO	BIT	BYTES	ESCALA
char	8	1	-128 a 127
int	16	2	-32768 a 32767
float	32	4	3.4e-38 a 3.4e+38
double	64	8	1.7e-308 a 1.7e+308
void	0	0	Sem valor



# ALGORITMO – VARIÁVEIS E CONSTANTES



## VARIÁVEIS E CONSTANTES

- Um algoritmo manipula dados, que podem ser dados variáveis ou constantes.
- Dados variáveis são representados por **variáveis**, enquanto dados constantes são representados por **constantes**

## VARIÁVEIS

- Uma variável pode ser imaginada como um “caixa” para armazenar valores de dados.
- Esta caixa só pode armazenar um único valor por vez.
- O valor armazenado na caixa pode mudar inúmeras vezes durante a execução do algoritmo.
- Em um ambiente computacional de verdade, a caixa correspondente a uma variável é uma posição da memória do computador

## CONSTANTE

- Um valor pré definido que não é alterado ao longo da execução.
- Armazena um único valor.
- Uma constante é uma posição na memória do computador que nunca muda seu valor.

## VARIÁVEIS E CONSTANTES

- Possui um nome;
- Possui um tipo;
- Possui conteúdo

## VARIÁVEIS E CONSTANTES

- Possui um nome:
  - O **nome de uma variável** deve ser único, isto é, identificar, de forma única, a variável no algoritmo.
- Possui um tipo:
  - O **tipo de uma variável** define os valores que podem ser armazenados na variável.
- Possui conteúdo:
  - O **conteúdo de uma variável** é o valor que ela armazena.

## VARIÁVEIS E CONSTANTES

- O ato de criar uma variável/constante é chamado de declaração;
- Usando o pseudocódigo, declaramos uma **variável** usando uma sentença da seguinte forma:

tipo nome;

const tipo nome;

## VARIÁVEIS E CONSTANTES

- Exemplos de declaração:

real valor;

- declara uma variável de nome valor do tipo real.
- declarar mais de uma variável de um mesmo tipo:

real valor , quantidade, salario;

inteiro numero;

## VARIÁVEIS E CONSTANTES

```
AGORITMO EXEMPLO  
  REAL VALOR;  
  INTEIRO NUMERO;  
  CHARACTER NOME[7];  
INICIO  
  ESCREVA("HELLO WORD")  
FIM
```



## VARIÁVEIS E CONSTANTES EM C

- declaração:

float valor;

float valor , quantidade, salario;

int numero;

char letra;

char \*nome;

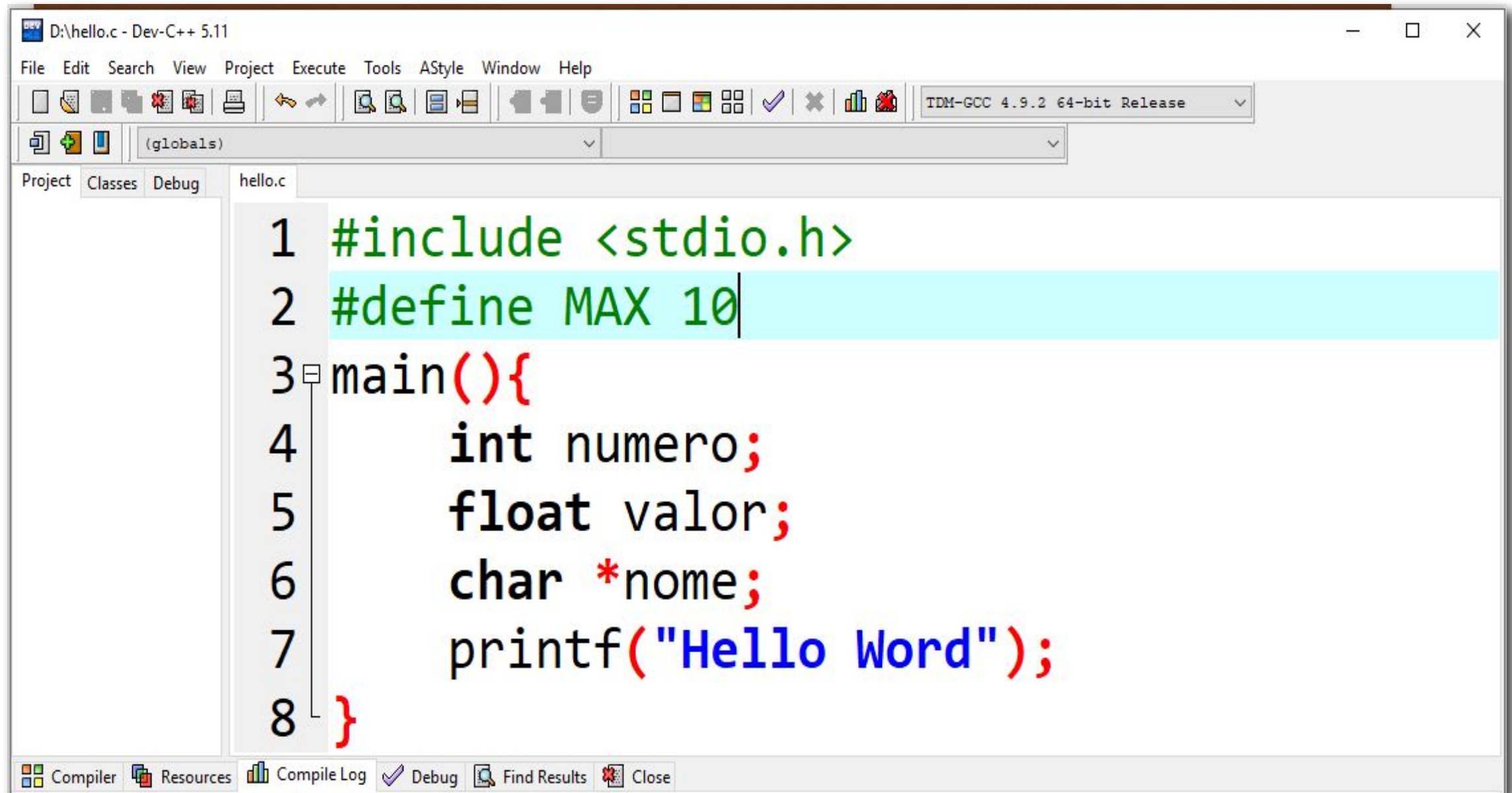
## VARIÁVEIS E CONSTANTES EM C

- Constante

```
#define MAX 10
```

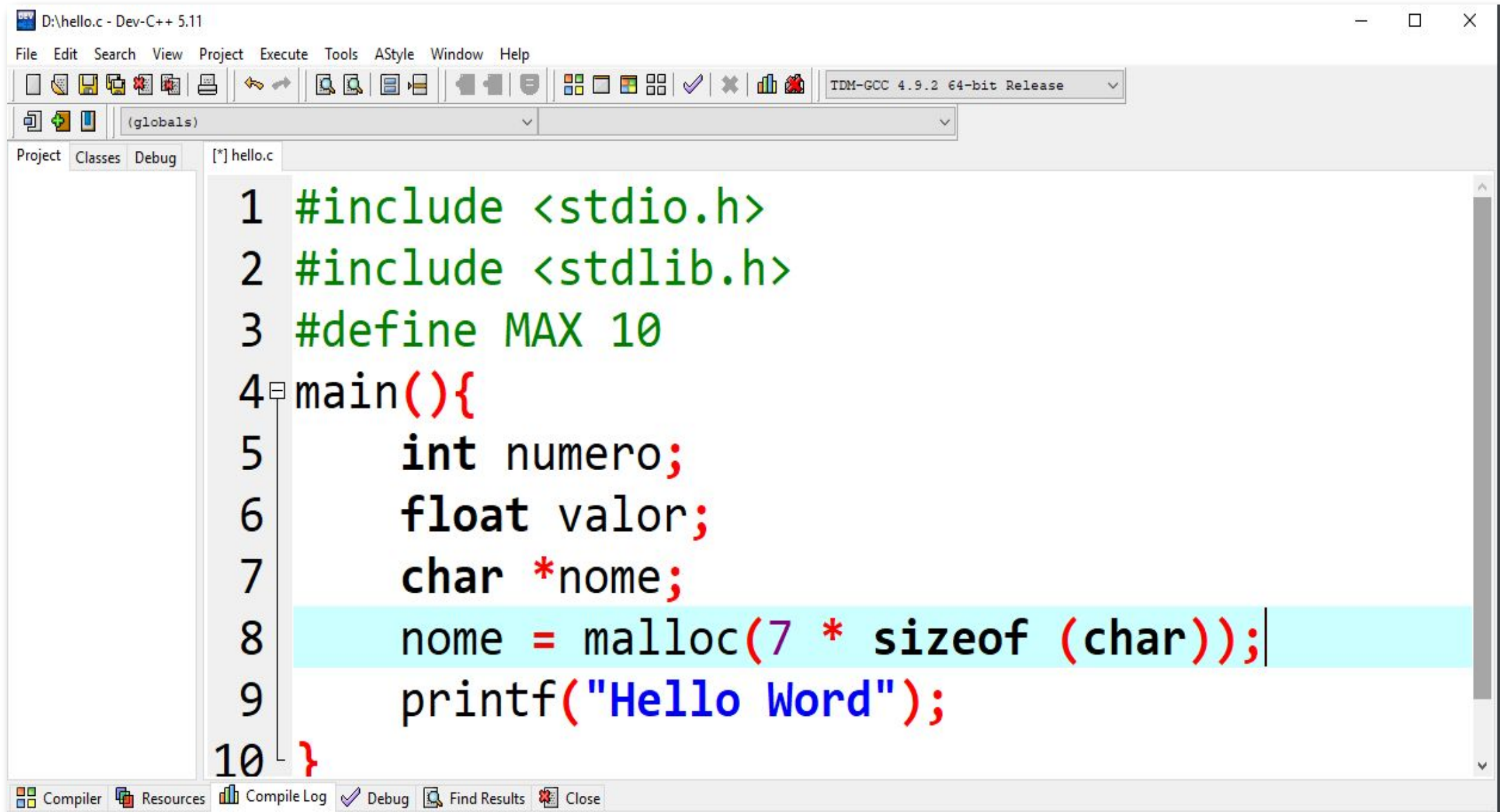
```
#define pi 3.14
```

## VARIÁVEIS E CONSTANTES EM C



```
D:\hello.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug hello.c
1 #include <stdio.h>
2 #define MAX 10
3 main(){
4     int numero;
5     float valor;
6     char *nome;
7     printf("Hello Word");
8 }
```

## VARIÁVEIS E CONSTANTES EM C



```
D:\hello.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug [*] hello.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX 10
4 main(){
5     int numero;
6     float valor;
7     char *nome;
8     nome = malloc(7 * sizeof(char));
9     printf("Hello Word");
10 }
```



# ATRIBUIÇÃO DE VALORES



## ATRIBUIÇÃO DE VALORES

- As variáveis que declaramos tem conteúdo?
- Em C normalmente lixo de memória;
- Em pseudocódigo nada.

## ATRIBUIÇÃO DE VALORES

- Podemos usar atribuição para adicionar um valor:
  - `numero = 3` (pseudocódigo)
  - `numero <- 3` (pseudocódigo)
  - `numero = 3;` (linguagem c)

## ATRIBUIÇÃO DE VALORES

AGORITMO EXEMPLO

REAL VALOR;

INTEIRO NUMERO;

CARACTER NOME[7];

INICIO

VALOR = 3.01

NUMERO = 3;

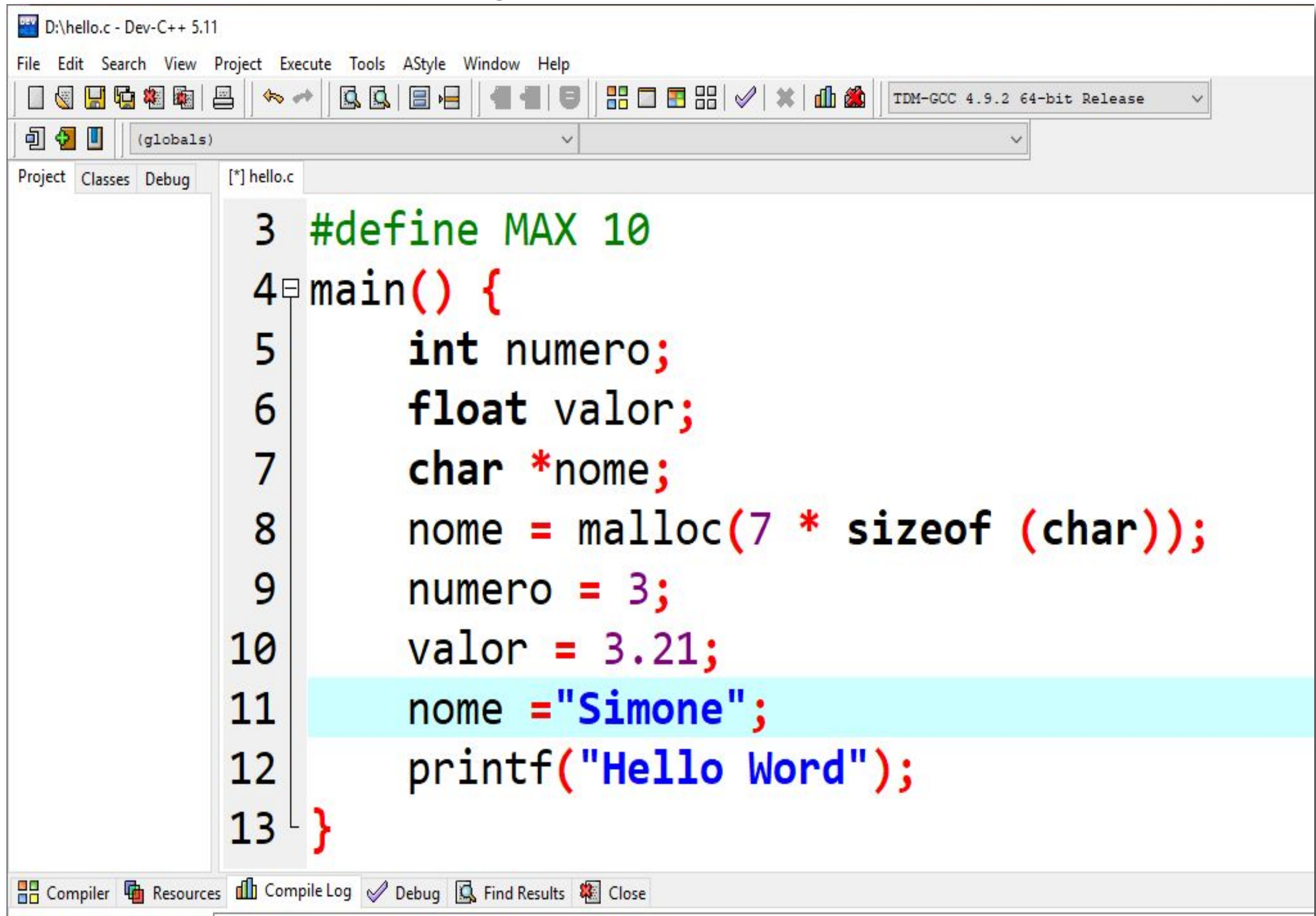
NOME = "SIMONE"

ESCREVA("HELLO WORD")

FIM



## ATRIBUIÇÃO DE VALORES



The screenshot shows the Dev-C++ 5.11 IDE with a C program named 'hello.c'. The code defines a constant 'MAX' as 10 and a 'main' function. Inside 'main', it declares 'int numero', 'float valor', and 'char \*nome'. It then allocates memory for 'nome' using 'malloc', assigns the value 3 to 'numero', 3.21 to 'valor', and the string 'Simone' to 'nome'. Finally, it prints 'Hello Word' using 'printf'.

```
3 #define MAX 10
4 main() {
5     int numero;
6     float valor;
7     char *nome;
8     nome = malloc(7 * sizeof (char));
9     numero = 3;
10    valor = 3.21;
11    nome = "Simone";
12    printf("Hello Word");
13 }
```



# LEITURA DE VALORES



## LEITURA DE VALORES

- Podemos usar leitura para adicionar um valor:
  - `leia(numero)` (pseudocódigo)
  - `scanf("%d", &numero);` (linguagem c)
  - `scanf("%d", nome);` //string

# LEITURA DE VALORES

ALGORITMO EXEMPLO

REAL VALOR;

INTEIRO NUMERO;

CARACTER NOME[7];

INICIO

ESCREVA("HELLO WORD, DIGITE UM  
NÚMERO")

LEIA(NUMERO)

FIM

## LEITURA DE VALORES

```
[*] hello.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define MAX 10
4  main() {
5      int numero;
6      float valor;
7      char *nome;
8      nome = malloc(7 * sizeof (char));
9      printf("Hello Word, digite uma valor inteiro:\n");
10     scanf("%d", &numero);
11 }
```

## LEITURA DE VALORES

[\*] hello.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define MAX 10
4  main() {
5      int numero;
6      float valor;
7      char *nome;
8      nome = malloc(7 * sizeof (char));
9      printf("Hello Word, digite uma valor inteiro:\n");
10     scanf("%d", &numero);
11     printf("Hello Word, digite uma valor real: \n");
12     scanf("%f", &valor);
13     printf("Hello Word, digite uma cadeia de caracter:\n");
14     scanf("%s", nome);
15
16 }
```



# EXPRESSÕES LÓGICAS, ARITMÉTICAS E RELACIONAIS





## EXPRESSÕES

- **Expressões:** É a combinação de variáveis, constantes e operadores, que quando avaliada, resulta em um valor.
- **Operadores:** São elementos que atuam sobre os operandos e produzem um determinado resultado.
- **Operandos:** Podem ser variáveis ou constantes.



## OPERADORES RELACIONAIS

Linguagem C	Operação
<	menor que
>	maior que
<=	menor ou igual a
>=	maior ou igual a
==	igual a
!=	diferente de

### Exemplo

```
int x = 2;  
int y = 5;  
int z;  
  
z = x+y;  
  
if(z >= y)  
    x = 0;
```

## EXPRESSÕES LÓGICAS

Linguagem C	Operação
&&	E lógico (ambos)
	Ou lógico (qualquer)
!	Negação

### Exemplo

```
int x = 2;  
int y = 5;  
int z;  
  
z = x+y;  
  
if(z >= y && y != 0)  
    x = 0;
```

- Em pseudocódigo:

PSEUDOCÓDIGO	Operação
E	E lógico (ambos)
OU	Ou lógico (qualquer)
!	Negação

## EXPRESSÕES LÓGICAS

&&	F	V
F	F	F
V	F	V

```
if (nota >= 6.0 && frequencia >= 0.75)  
    printf("aprovado");
```

	F	V
F	F	V
V	V	V

```
if (nota < 6.0 || frequencia < 0.75)  
    printf("reprovado");
```

AND (&&): a saída só é verdadeira quando as duas expressões de entrada são verdadeiras.  
OR (||): basta uma entrada ser verdadeira para a saída ser verdadeira.

## EXPRESSÕES ARITMÉTICAS

- Mais: +
- Menos: -
- Vezes: \*
- Dividir: /
- Módulo: %      resto da divisão



## EXPRESSÕES ARITMÉTICAS

Operador	Exemplo	Comentário
+	$x + y$	Soma $x$ e $y$
-	$x - y$	Subtrai $y$ de $x$
*	$x * y$	Multiplica $x$ e $y$
/	$x / y$	Divide $x$ por $y$
%	$x \% y$	Resto da divisão de $x$ por $y$
++	$x++$	Incrementa em 1 o valor de $x$
--	$x--$	Decrementa em 1 o valor de $x$

Equivalente:

→  $x = x + 1;$

→  $x = x - 1;$

Em C, o sinal = significa atribuição



## ENTRADA E SAÍDA DE DADOS

A função *printf*

```
printf("Hello World.");
```

← Aula 1

```
int valor = 5;  
float numero = 3.14;  
char sexo = 'M';
```


← Declarando variáveis  
e imprimindo seus  
conteúdos (abaixo)

```
printf("%d %f %c", valor, numero, sexo);
```

*Expressão de controle*

*Argumentos*


## ENTRADA E SAÍDA DE DADOS




```
printf("%d", valor);
```



```
printf("O valor tal = %d", valor);
```



```
printf("O valor tal = %d unidades", valor);
```



```
printf("\n O valor tal = %d unidades. \n", valor);
```

(Explorando a função *printf*)

## ENTRADA E SAÍDA DE DADOS

- função scanf: É o complemento de printf. Permite ler dados formatados de entrada padrão (teclado)
- Sua sintaxe é similar à de printf, isto é, uma expressão de controle seguida por uma lista de argumentos separados por vírgulas.

```
scanf("%d", &valor1);
```

expressão de controle      argumentos



## EXEMPLO

```
main()
{
    int a,b,c; ← Declaração de variáveis

    printf("Informe o valor 1: ");
    scanf("%d",&a); ← Solicitando o valor 1;
                    Lendo valor 1.

    printf("Informe o valor 2: ");
    scanf("%d",&b); ← Solicitando o valor 2;
                    Lendo valor 2.

    c = a+b; ← Efetuando a soma

    printf("%d + %d = %d \n", a,b,c); ← Imprimindo resultado

    system("pause");
}
```

## EXERCÍCIOS

- 1 - Repetir exatamente o que o usuário digitar. Pode receber valores reais um de cada vez.
- 2 - PROGRAMA CADASTRO: solicita NOME, ENDEREÇO e TELEFONE do usuário e depois mostra os dados digitados em uma única linha.
- 3 - PROGRAMA TROCA: o usuário digita um valor na variável X e outro valor na variável Y. O programa deve trocar os valores, passando Y a ter o valor de X e X o valor de Y.