LINGUAGEM C:

COMANDOS BÁSICOS

Prof. Wesley Romão

Comando

- Um comando é uma **instrução** que o compilador C pode executar.

- Quando você digita uma instrução, o C a **compila**, executa e mostra o resultado, se houver um.

Instruções Primitivas

- Comando de **atribuição**
- Comando de saída
- Comando de entrada

Comando de Atribuição

- Comandos de **atribuição** não produzem um resultado visível;
- Permite que se atribua um valor a uma variável;
- Para se realizar uma atribuição em C, utiliza-se o sinal igual (=).

```
mensagem = “E aí galera?”;
idade = 19;
pi = 3.14159;
```

Comando de Atribuição

while = “Graduação“

SyntaxError: invalid syntax

O que ocorre com while?

É uma das **palavras reservadas** em C que definem as regras e a estrutura da linguagem e não podem ser usadas como nomes de variáveis.

Palavras Reservadas

A **linguagem C** tem 32 palavras reservadas:

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

Todas as palavras reservadas são escritas em minúsculo.

Comando de Atribuição

Linguagem Algorítmica

TOTAL = x^2 - $\sqrt{x}$

cor = “VERDE”

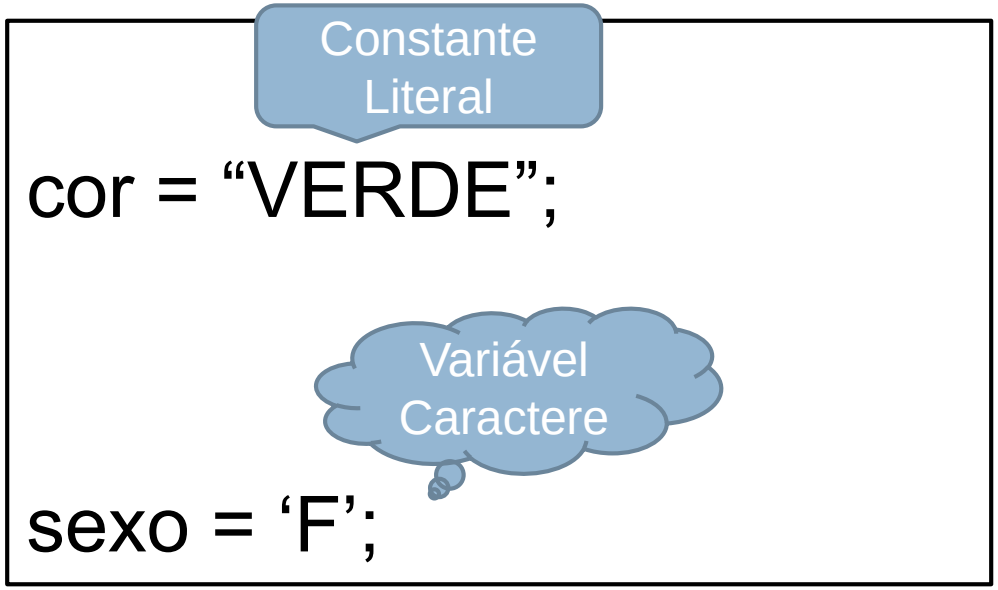
Nota = 10

sexo = “Masculino”

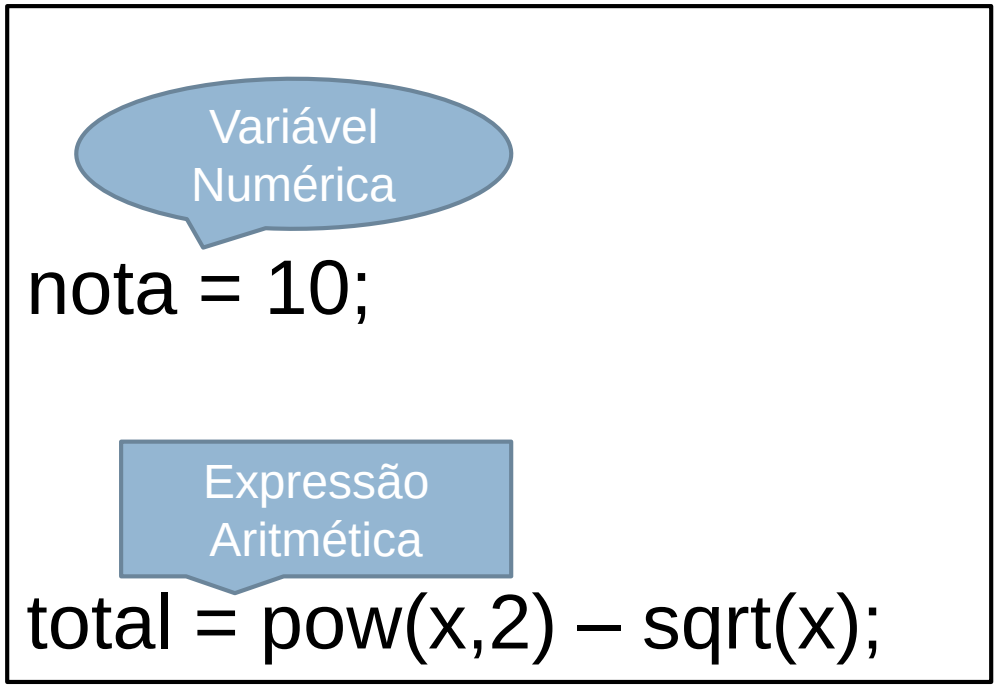
Linguagem C

```
total = pow(x,2) – sqrt(x);
cor = “VERDE”;
nota = 10;
sexo = ‘M’;
```

Comando de Atribuição



Comando de Atribuição



SCRIPT

- Temos que transformar nossos algoritmos em *scripts*;
- Um *script* normalmente contém uma sequência de comandos;
- Precisamos de uma IDE para escrever *scripts*;
- IDE (*Integrated Development Environment*), é um ambiente integrado para desenvolvimento de software.

Code Blocks

- Vamos utilizar uma IDE denominada **Code Blocks**.

- Conheça o Code::Blocks em:

www.codeblocks.org

Instruções Primitivas

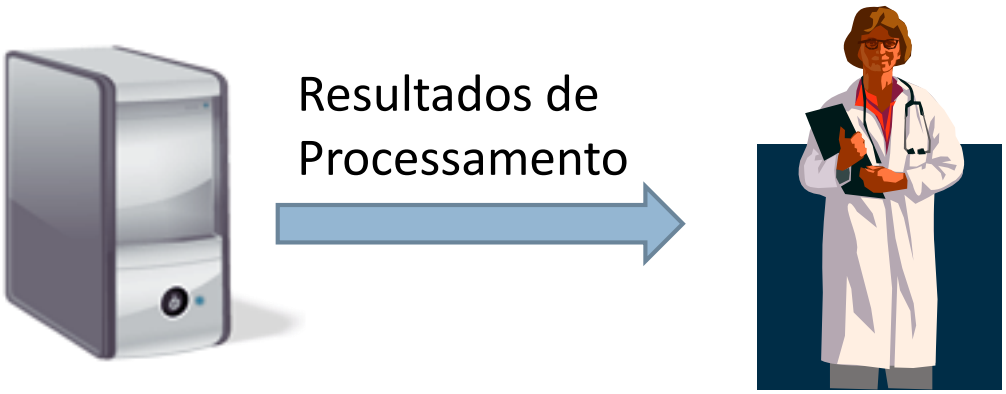
- Comando de atribuição
- Comando de **saída**
- Comando de entrada

Comando de Saída

Utilizado para que o sistema forneça, numa unidade de saída, os **resultados** do processamento e **mensagens**.

Saída: Resultados de Processamento

Fornecidos através de conteúdos de **variáveis**, conteúdos de **constantes** e resultados de **expressões** aritméticas e lógicas.



Utilizadas para que o programa informe ao usuário a respeito do processamento que está se realizando.

Comando de Saída: printf

Toda string deve ser escrita entre aspas duplas.

```
printf(“Olá Mundo!”);
```

Meu primeiro programa em C

Todo script deve iniciar com **main**.

```
int main(void)
{
    printf(“Olá Mundo!”);
}
```

Comando de Saída

Para mudar de linha devemos aplicar **\n**”:

```
int main(void)
{
    printf(“Olá Mundo! \n”);
}
```

Comandos de Saída

Para mostrar uma string **com aspas** devemos aplicar **\”**:

```
int main(void)
{
    printf(“Olá Mundo!”);
    printf(“\”Estou entre aspas duplas.\””);

}
```

Comando de Saída

printf(“%i”, 5/2);

- O resultado surge como um inteiro, neste caso 2.
- Isto acontece porque os operandos são inteiros.
- Como a saída é numérica requer a formatação %i.

OBS: Se pelo menos um operando for real o resultado será do tipo *float*.

Comando de Saída

Para obter resultados em números reais, utilize formato *float*:

```
printf("%f", 5/2.0);
```

Comando de Saída

Para calcular o **resto** da divisão entre dois inteiros:

```
printf(“%i”, 7%2);
```

% é conhecido como o operador de *modulo* ou *mod*.

Comando de Saída

Algoritmo

Escrever (sexo)

Escrever (“valor alto!”)

Formatação de caractere requer **%c**.

Linguagem C:

```
char sexo = ‘M”;
printf(“%c”, sexo);
printf(“valor alto!”);
```

Comando de Saída

Algoritmo

Escrever (idade)

Escrever (cor)

Formatação de *string* requer **%s**.

Linguagem C:

```
int idade = 21;
char cor[10] = “azul”;
printf(“%d”, idade);
printf(“Cor = %s”, cor);
```

Formatação de *decimal* requer **%d**.

Comando de Saída

- Agora vamos combinar *string* com números, separados por virgulas, utilizando formatações:

```
printf("A soma de %d com %d = %d\n", 7,18,7+18);
```

- As formatações ocorrem com o caracter %.
- A letra 'd' depois do % diz ao C que um número decimal deve ser posto naquele lugar.
- Os valores a serem usados são obtidos da lista no final.

Comando de Saída

- Lista de formatadores:

  %c – para caractere

  %s – para strings

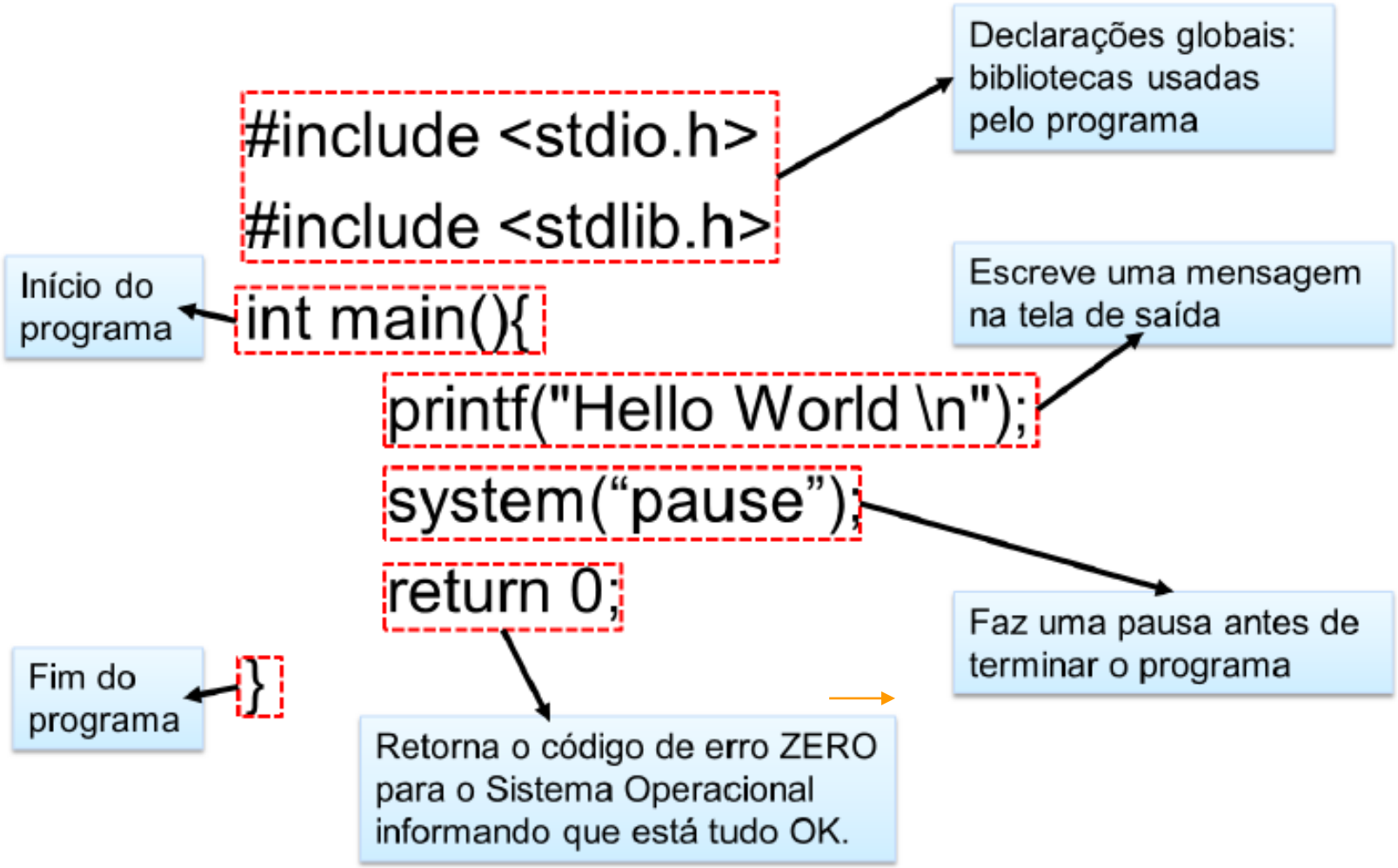
  %f – números reais

  %d – números inteiros (ou %i)

- Exemplo:

```
char palavra[10] = "ALGORITMO";
printf("Palavra = %s, Numero real = %f \n", palavra, 55.126);
printf("Numero inteiro = %d Outro numero inteiro = %i", 33, 44);
```

Programa Completo em C

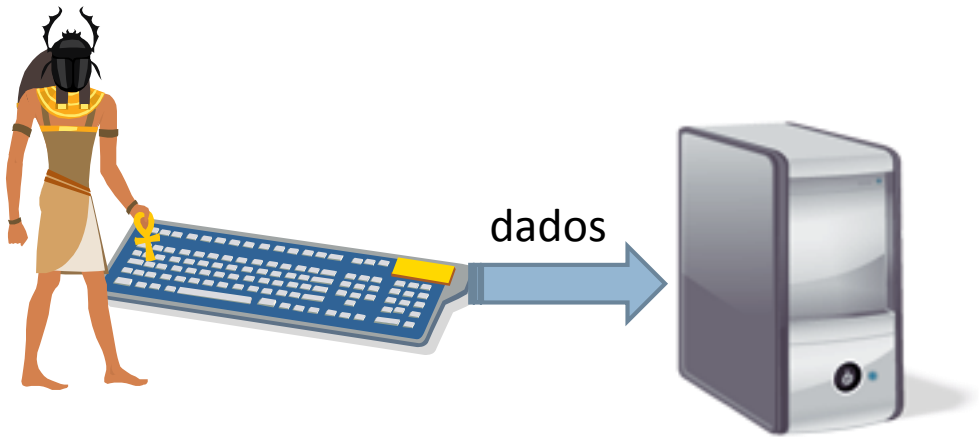


Instruções Primitivas

- Comando de atribuição
- Comando de saída
- Comando de **entrada**

Comando de Entrada

- Utilizado para receber dados digitados pelo usuário e armazená-los em variáveis;
- Os dados são fornecidos ao sistema através de uma unidade de entrada.



Comando de Entrada

Algoritmo

Ler (nome)

Linguagem C

```
char nome[30], sexo;
scanf("%s", &nome);
```

OBS: Utilize & antes da variável.

Comando de Entrada

Algoritmo

Ler (nome)

Ler (sexo)

Linguagem C

```
char nome[30], sexo;
scanf("%s", &nome);
scanf("  %c", &sexo);
```

Formatadores:
%s → string
%c → char
%d → decimal
%f → float
%i → inteiro

OBS: Utilize um **espaço** antes de %c.

Comando de Entrada

Algoritmo

Ler (nome)

Ler (sexo)

Ler (num)

Ler (valor)

Linguagem C

```
char nome[30], sexo;
int num; float valor;
scanf("%s", &nome);
scanf("%d", &num);
scanf("%f", &valor);
scanf(" %c", &sexo);
```

Formatadores:
%s → string
%c → char
%d → decimal
%f → float
%i → inteiro

OBS: Utilize um espaço antes de %c.

Comando de Entrada

Algoritmo

Ler (nome)

Ler (sexo)

Ler (num)

Ler (valor)

Formatadores:
%s → string
%c → char
%d → decimal
%f → float
%i → inteiro

Linguagem C

```
char nome[30], sexo;
int num; float valor;
//ENTRADA DE DADOS
printf("Nome: ”);
scanf("%s", &nome);
printf("Num: ");
scanf("%d", &num);
printf("Valor: “)
scanf("%f", &valor);
printf("Sexo: ");
scanf(" %c", &sexo);
```

Comando de Entrada

Algoritmo

Ler (nome)
Ler (sexo)
Ler (num)
Ler (valor)

Formatadores:
%s → string
%c → char
%d → decimal
%f → float
%i → inteiro

Linguagem C

```
char nome[30], sexo;
int num; float valor;

//ENTRADA DE DADOS
...

//SAÍDA DE DADOS
printf("Nome: %s\n", nome);
printf("Num: %i\n", num);
printf("Valor: %f\n", valor);
printf("Sexo: %c\n", sexo);
```

Comando de Entrada

- Na execução de um comando de entrada o processamento é interrompido até que sejam fornecidos, via unidade de entrada, valores para os dados de entrada;
- Os valores digitados pelo teclado são memorizados após a digitação da tecla ENTER;
- Os identificadores são separados por vírgula.

Concatenação de strings

Operador	*Descrição*
strcat(S1, S2)	Concatenação de S1 e S2

```
strcat(DESTINO, ORIGEM)
```

Exemplo:

```
char s1[20] = "Cidade";
char s2[10] = "Cancao";
```

Concatenação de strings

strcat(DESTINO, ORIGEM)

Exemplo:

```
#include <string.h>

    char s1[20] = "Cidade";
    char s2[10] = "Cancao";
    strcat(s1,s2);
    printf("%s", s1);
```

A função strcat() requer a inclusão da biblioteca string.h

EXERCÍCIO 1

Declarar, ler e escrever o cpf de uma pessoa.

```
char cpf[11];
 ...
```

EXERCÍCIO 1

Declarar, ler e escrever o cpf de uma pessoa.

```
char cpf[11];
printf("CPF = ");    scanf("%s", cpf);
printf("Escrevendo cpf: %s\n", cpf);
```

EXERCÍCIO 2

Calcular a **raiz cúbica** de um número.

Variável de entrada: numero = 90

Variável de saída: raiz

EXERCÍCIO 2

Calcular a raiz cúbica de um número.

```
#include <math.h>

    int numero = 90;
    float raiz = pow(90,1/3.);
```

A função pow() requer a inclusão da biblioteca math.h

EXERCÍCIO 2

Calcular a raiz cúbica de um número.

```
int numero = 90;
float raiz = pow(90,1/3.);
printf("Raiz = %f\n", raiz);
```

EXERCÍCIO 3

Atribuir 10 a variável A e 20 a variável B.

```
int a = 10;
int b = 20;
```

EXERCÍCIO 3

Atribuir 10 a variável A e 20 a variável B.

Em seguida **trocar** os valores entre A e B utilizando atribuição.

```
int a = 10, b = 20;
```

EXERCÍCIO 3

Atribuir 10 a variável A e 20 a variável B.

Em seguida **trocar** os valores entre A e B utilizando atribuição.

```
int a = 10, b = 20, aux;
aux = a;
a = b;
b = aux;
```

EXERCÍCIO 3

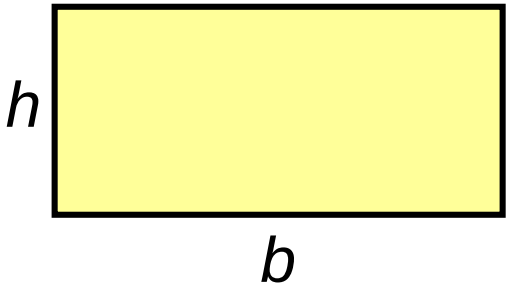
Atribuir 10 a variável A e 20 a variável B.

Em seguida trocar os valores entre A e B utilizando atribuição. **Mostrar** o resultado.

```
int a = 10, b = 20, aux;
aux = a;
a = b;
b = aux;
printf("a = %i  b = %i\n", a, b);
```

EXERCÍCIO 4

Considerando o retângulo abaixo, faça um algoritmo que calcule a área do mesmo.



Neste problema existem três valores (estados) a serem representados:

- Altura do retângulo (h);
- Largura (ou base) do retângulo (b);

Dados de Entrada

- Área do retângulo.

Dado de Saída

EXERCÍCIO 4

Solução em C:

```
#include<stdio.h>
int main(void)
{
        //algoritmo
}
```

EXERCÍCIO 4

Solução em C:

```
#include<stdio.h>
int main(void)
{
  float h, b, area;
  printf("Digite um valor para a altura do retangulo: ");
  scanf ("%f",&h);
  ...

}
```

EXERCÍCIO 4

Solução em C:

```
#include<stdio.h>
int main(void)
{
  float h, b, area;
  printf("Digite um valor para a altura do retangulo: ");
  scanf ("%f",&h);
  printf("Digite um valor para a base do retangulo: ");
  scanf ("%f",&b);
  area = h*b;
  printf("Valor da area: %f\n", area);
}
```

Indicação de Livro

TÍTULO: Algoritmos e Programação com exemplos em Pascal e C

AUTORES:

Nina **Edelweiss**

Maria Aparecida Castro Livi

EDITORA:

Bookman, 2014