



DESCRIÇÃO

Inicialização de variáveis, apresentação de seus formatos de escrita e de leitura e reconhecimento de funções para armazenamento e exibição de dados.

PROPÓSITO

Criar programas utilizando os comandos de entrada e saída, habilidade fundamental à formação de um programador.

PREPARAÇÃO

Para este tema, recomendamos que você instale o software **Dev C++**, pois esse será o ambiente de programação que utilizaremos. Além desse, existem outros compiladores que suportam a linguagem C, como o **Code::Blocks**.

OBJETIVOS

MÓDULO 1

Utilizar o comando de atribuição

MÓDULO 2

Aplicar os comandos de saída de dados

MÓDULO 3

Executar os comandos de entrada de dados

INTRODUÇÃO

VOCÊ JÁ OUVIU FALAR OU REALIZOU ALGUM TESTE NO SITE BUZZFEED?

Esse tipo de teste é muito conhecido e utilizado nas redes sociais para identificar perfis variados de usuários em diferentes contextos. Para começar este tema, preparamos um teste no estilo BuzzFeed sobre os diferentes tipos de inteligência propostos pelo psicólogo **Howard Gardner** que permitirá identificar as inteligências que você possui.



Fonte: Site Planeta de libros

HOWARD GARDNER

Nasceu em 1943 na Pensilvânia (EUA). Iniciou os estudos em Direito e História na Universidade de Harvard, mudando, depois, para Psicologia e Educação. Pesquisou sobre os sistemas simbólicos, vindo a criar a chamada Teoria das Inteligências Múltiplas.

Fonte: Escola e educação.



Responda as perguntas abaixo. Você pode escolher mais de uma opção.

Ao final do teste, clique nas legendas para ver as definições.

Você deve estar se perguntando:

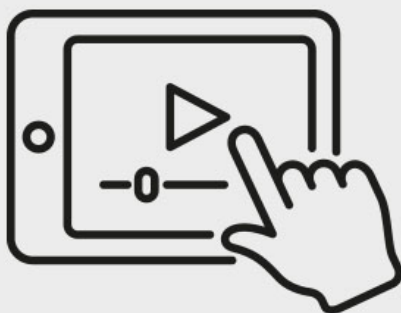
**SE ESTE TEMA É SOBRE PROGRAMAÇÃO, POR QUE
PRECISEI FAZER UM TESTE SOBRE AS
INTELIGÊNCIAS MÚLTIPLAS?**

Primeiro: o teste permite um autoconhecimento. Você saberá identificar suas inteligências com mais potencial e desenvolvê-las.

Segundo: a dinâmica do teste mostra exatamente o que vamos tratar neste tema: **comandos de entrada e saída**. Ao realizar o teste, você precisou inserir dados para que um resultado fosse exibido ao final.

Para ajudá-lo a entender melhor essa dinâmica, o professor **Humberto Henriques** explica, no vídeo a seguir, os contextos nos quais são utilizados os comandos de entrada e saída.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



HUMBERTO HENRIQUES

Possui graduação em Engenharia de Computação pelo Instituto Militar de Engenharia (2003) e mestrado em Engenharia de Produção pela Universidade Federal do Rio de Janeiro (2009). Tem experiência na área de Engenharia de Produção e trabalha com metodologias de aprendizagem ativa e CDIO no Instituto Militar de Engenharia.

Fonte: Currículo Lattes

MÓDULO 1

🕒 Utilizar o comando de atribuição

COMANDO DE ATRIBUIÇÃO

Ao **declarar uma variável**, o compilador reserva espaço na memória para o **armazenamento de valor**. Como a memória do computador é composta por *bytes*, formados a partir de *bits*, a variável pode assumir um valor aleatório, uma vez que não temos controle sobre eles. A inicialização só ocorre quando se **atribui valor** por meio de um **comando de atribuição**. A seguir, vamos ver um pouco mais sobre o uso dessa ferramenta na programação.



Imagem: Artram / Shutterstock.com

CONCEITOS

Todos os comandos apresentados neste tema obedecem à sintaxe (conjunto de regras) da **linguagem C** e do **Portugol**. Antes de utilizar o comando de atribuição, você deve **inicializar a variável**. Vejamos como realizar esse procedimento:

Tomemos como exemplo a declaração da variável inteira chamada *a*.

```
int a;
```

Na linguagem C e no Portugol, esse comando é representado pelo sinal de igual =, conforme se observa no formato geral da estrutura:

```
nome_da_variável = valor_atribuído;
```

Após a declaração de `a`, existem duas maneiras de atribuir o valor 10 a essa variável:

1

De forma separada

```
int a;  
a = 10;
```

2

Na mesma linha da declaração

```
int a = 10;
```

O nome da variável deve ajudar a entender seu significado. O uso de iniciais maiúsculas, a partir da segunda palavra, ou do símbolo ***underscore*** `_` permite a criação de nomes mais complexos, como: `idCliente`, `id_cliente`, `cpf_usuario`, `cpfUsuario`, entre outros.

Na linguagem C, ainda é possível **atribuir o mesmo valor a mais de uma variável**. Vejamos um exemplo:

Com a seguinte instrução, é dado o valor 2 às variáveis `a` e `b`:

```
a = b = 2;
```

Observe que não há como guardar o histórico de valores de uma variável. A atribuição de outro valor faz com que o anterior seja perdido. Para evitar que isso aconteça, deve-se usar outra variável. Na sequência de instruções a seguir, a variável `a` vale 3, sem que 1 e 2 sejam guardados.

```
int a;  
a = 1;  
a = 2;  
a = 3;
```

ATENÇÃO

Em **pseudocódigo**, o comando de atribuição é representado pela seta (\leftarrow), mas não simboliza a igualdade; ele atribui à variável do lado esquerdo o valor que está à direita. Vejamos alguns exemplos:

a ← 10 (pseudocódigo) ou **a = 10** (Portugol e C) atribui o valor 10 à variável a.

a ← a + 1 (pseudocódigo) ou **a = a + 1** (Portugol e C) acresce uma unidade à variável a, resultando no valor 11.

O mesmo ocorre na próxima sequência, em que a teria o valor 6 ao final da execução das instruções:

a ← 5 (pseudocódigo) ou **a = 5** (Portugol e C).

a ← a + 1 (pseudocódigo) ou **a = a + 1** (Portugol e C).

O comando de atribuição pode ser usado para variáveis dos tipos int, double e float da mesma forma que vimos anteriormente. Por outro lado, o tipo char deve ser usado com cautela para que não haja confusão entre o uso de caractere e variável, conforme é mostrado a seguir:

Para declarar uma variável do tipo char chamada escolha, usamos:

```
char escolha;
```

Como é do tipo char, espera-se receber **caracteres**. Para atribuir b à escolha, utilizaremos as aspas simples a fim de indicar que se trata do caractere **b**, e não da variável b, sendo o comando correto:

```
escolha = 'b';
```

Caso seja feito sem as aspas simples, o programa apontará erro, já que o compilador irá procurar a variável b, não declarada, para atribuir o seu valor à escolha.

A linguagem C também permite **operações aritméticas** com variáveis do tipo char, relacionando o valor dos caracteres armazenados nelas aos inteiros correspondentes na **tabela ASCII**, conforme representado no próximo exemplo:

TABELA ASCII

Criada em 1960 por Robert W. Bemer, cientista da computação norte-americano conhecido pelo seu trabalho na IBM entre os anos de 1950 a 1960, a tabela ASCII uniformizou a representação de caracteres entre as máquinas.

A sigla ASCII, do inglês *American Standard Code for Information Interchange*, significa **Código Padrão Americano para o Intercâmbio de Informação**. É baseado no alfabeto

romano e sua função é **padronizar** a forma como os computadores **representam letras, números, acentos, sinais diversos e alguns códigos de controle**.

No ASCII, existem apenas 95 caracteres que podem ser impressos. Eles são numerados de 32 a 126, sendo os caracteres de 0 a 31 reservados para funções de controle. Veja alguns caracteres especiais:

\7	<i>Bell</i> (sinal sonoro do computador)
\a	<i>Bell</i> (sinal sonoro do computador)
\b	<i>BackSpace</i>
\n	<i>New Line</i> (mudança de linha)
\r	<i>Carriage Return</i>
\t	Tabulação Horizontal
\v	Tabulação Vertical
\\	Caractere \ (forma de representar o próprio caractere especial \)
\'	Caractere ' (aspas simples)
\"	Caractere " (aspas)
\?	Caractere ? (ponto de interrogação)
\000	Caractere cujo código ASCII em Octal é 000
\xyy	Caractere cujo código ASCII em Hexadecimal é yy



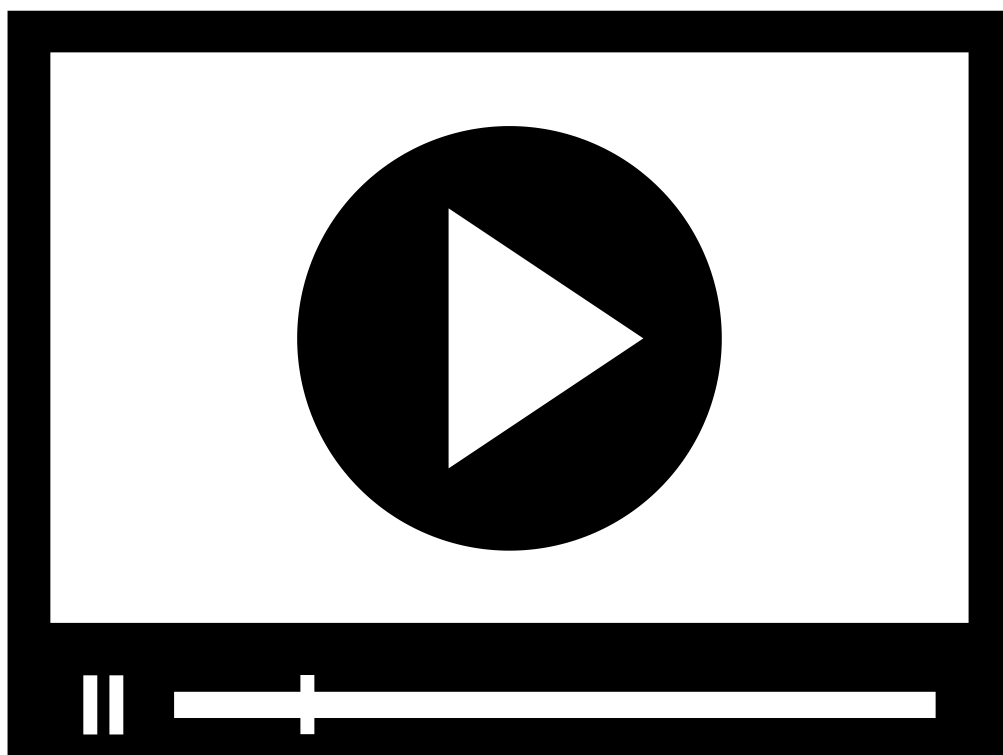
Atenção! Para visualização completa da tabela utilize a rolagem horizontal

```
char escolha;
```

```
escolha = 'b';
```

```
escolha = escolha + 1;
```

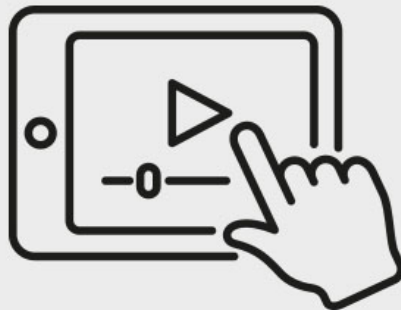
Ao final da execução dessas linhas, a variável `escolha` armazenará o caractere **'c'**.



E AÍ, ENTENDEU TUDO SOBRE COMO ATRIBUIR VALOR A UMA VARIÁVEL?

Para que não haja mais dúvidas, o professor Humberto Henriques responde, no vídeo a seguir, às principais dúvidas sobre **atribuição de valor a uma variável**.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



TEORIA NA PRÁTICA

A seguir você receberá uma série de atividades para realizar em seu ambiente de programação. Tente executá-las e, caso tenha alguma dúvida quanto ao resultado, basta clicar no botão “**FEEDBACK**” para obter a resposta.

1. Descubra qual é o valor da variável `cont` após a execução das seguintes linhas:

```
int cont = 0;  
cont = cont + 1;
```

FEEDBACK

O valor da variável é **1**. A variável `cont` é inicializada com 0, mas a segunda linha acresce uma unidade a esse valor.

2. Descubra qual é o valor da variável `escolha` após a execução das seguintes linhas:

```
char escolha;  
escolha = 'D';  
escolha = escolha - 2;
```

FEEDBACK

O valor da variável é **'B'**. Por se tratar de um caractere, ao realizar a operação aritmética para diminuir duas unidades da variável `escolha`, ficará aquele que estiver duas posições antes na tabela ASCII (nesse caso, no alfabeto). Vale lembrar que a linguagem C é *case sensitive*, ou seja, diferencia letras maiúsculas de minúsculas.

3. Descubra qual é o valor da variável `c` após a execução das seguintes linhas:

```
int a, b, c, d;  
a = 10;  
b = a + 1;  
c = b + 1;  
d = c + 1;  
a = b = c = d = 20;
```

FEEDBACK

O valor da variável é **20**. A última linha atribui valor 20 a todas as variáveis, não importando o valor que tinham previamente.

4. Descubra qual é o resultado da execução das seguintes linhas:

```
char escolha;  
escolha = a;  
escolha = escolha + 1;
```

FEEDBACK

Ocorrerá erro de compilação na segunda linha por não haver variável declarada com o nome a. Lembre-se sempre de não confundir caractere '**a**' com variável a.

VEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Teoria na Prática – Exemplo de comando de atribuição na linguagem Java

Teoria na Prática – Exemplo de atribuição na linguagem Python

VERIFICANDO O APRENDIZADO

1. QUAL É O VALOR ARMAZENADO NA VARIÁVEL A APÓS A EXECUÇÃO DESTAS LINHAS?

INT A, B, C;

A = B + C;

B = 1;

C = B + 1;

A) 1.

B) 2.

C) 3.

D) Um valor aleatório.

2. QUAL É O VALOR ARMAZENADO NA VARIÁVEL CH APÓS A EXECUÇÃO DESTAS LINHAS?

INT A = 1;

CHAR CH = 'A';

CH = CH + A;

A) 1.

B) 'A'.

C) 'B'.

D) Ocorrerá um erro de compilação.

1. Qual é o valor armazenado na variável `a` após a execução destas linhas?

```
int a, b, c;  
a = b + c;  
b = 1;  
c = b + 1;
```

A alternativa **"D "** está correta.

A variável `a` recebe a soma das variáveis `b` e `c`, porém, na segunda linha, elas ainda não têm valor atribuído. O resultado é um valor aleatório, visto que os bits são compostos por 0 e 1.

2. Qual é o valor armazenado na variável `ch` após a execução destas linhas?

```
int a = 1;  
char ch = 'A';  
ch = ch + a;
```

A alternativa **"C "** está correta.

A terceira linha somará uma unidade ao valor da variável `ch` e, com isso, ela passará a armazenar o caractere `'B'`.

MÓDULO 2

⦿ Aplicar os comandos de saída de dados

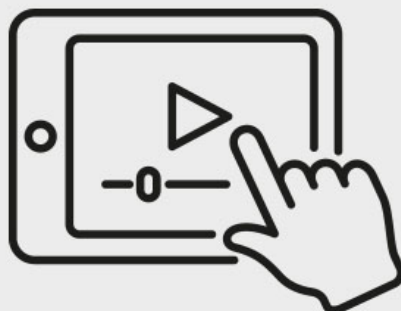
COMANDOS DE SAÍDA

A partir de agora vamos conhecer os **comandos de saída**, utilizados na programação para permitir a **exibição de informações ao usuário**. Além disso, construiremos nosso primeiro programa em C.

VOCÊ LEMBRA DO TESTE ESTILO BUZZFEED QUE REALIZOU NO INÍCIO DESTE TEMA?

No vídeo a seguir, o professor Humberto Henriques retoma o teste e mostra como os resultados são exibidos na tela a partir dos **comandos de saída**.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



CONCEITOS

VOCÊ SABE QUAL A RELAÇÃO ENTRE UM PROGRAMA E A LINGUAGEM C?

Um **programa** é uma **sequência de instruções** dadas para **resolver um problema**.





Imagem: Titov Nikolai / Shutterstock.com



A **linguagem C** é a **forma** de dar essas **orientações ao computador**.

O nosso primeiro programa em C será o mais conhecido no mundo da programação: o *Hello World*. Vamos começar!

No seu ambiente de programação, digite a seguinte sequência:

```
#include <stdio.h>

void main(){
printf("Hello World");
}
```

Salve o arquivo com o nome de sua preferência e execute-o. Você verá este resultado:

Hello World

Dentro da função `main()`, inserimos as instruções que serão executadas. Usam-se as chaves `{ }` para delimitar o que está incluso no corpo dessa função.

A primeira linha `#include <stdio.h>` é uma diretiva de pré-compilação e não uma instrução, por isso não é seguida por ponto e vírgula. A diretiva serve para incluir funções que estejam na biblioteca por meio das tags `< >`. Entende-se, então, que a biblioteca `stdio.h` tem funções que serão usadas em `main()`.

Em `main()`, nota-se uma única função, representada por `printf()`, que faz parte da biblioteca `stdio.h`. Por esse motivo, é preciso incluir a biblioteca no início do arquivo. **Printf**, traduzido do inglês como **escrever formatado** (*print + format*), tem como principal objetivo realizar a escrita na tela. Você pode estar se perguntando:

MAS O QUE ESSA FUNÇÃO EXIBE PARA O USUÁRIO?

Ela exibe o **parâmetro recebido dentro dos parênteses**! No exemplo anterior, `printf()` recebeu `Hello World` como parâmetro. Perceba que a string (cadeia de caracteres) está entre aspas, uma vez que servem para delimitá-la.

Para testar os conhecimentos adquiridos até aqui, tente fazer sozinho um programa que escreva o seu nome completo na tela.

Observe estas instruções:

```
#include <stdio.h>

void main(){
printf("Primeira linha");
printf("Segunda linha");
}
```

Ao compilar esse programa, você verá na tela:

Primeira linhaSegunda linha

Observe que a função `printf()` não faz a quebra de linha automática ao final da string. Em função disso, devemos inserir o caractere especial `'\n'`, ajustando o programa anterior para:

```
#include <stdio.h>

void main(){
printf("Primeira linha\n");
printf("Segunda linha");
}
```

Com isso, teremos:

Primeira linha
Segunda linha

A função `printf()` também permite a utilização de variáveis para compor o que será escrito na tela. Para indicar a posição de entrada de conteúdo de variáveis dos tipos `int` e `char` utilizam-se, respectivamente, os símbolos `%d` e `%c`. Vejamos, a seguir, a utilização dessas variáveis.

Observe o exemplo a seguir:

```
#include <stdio.h>

void main(){
```



```
int a = 10;
char ch = 'Z';
printf("Atualmente, temos a = %d e ch = %c.\n", a, ch);
}
```

Após a execução dessas instruções, o resultado será:

Atualmente, temos a = 10 e ch = Z.

Também podemos utilizar mais de uma variável do mesmo tipo, desde que sejam passadas, corretamente, quais delas preencherão a frase. Será seguida, então, a ordem invocada em `printf()`, com os conteúdos das variáveis acompanhando a sequência de uso dos símbolos `%d` ou `%c` e a correspondente passagem de parâmetros. Vejamos como aplicar essas variáveis:

Observe o exemplo a seguir:

```
#include <stdio.h>

void main(){
int a, b, c;
a = 10;
b = a + 1;
c = b + 2;
printf("Atualmente, temos a = %d, b = %d e c = %d.\n", a, b, c);
}
```

Depois de compilar esses códigos, você verá em sua tela:

Atualmente, temos a = 10, b = 11 e c = 13.

Você também pode escrever uma expressão matemática como parâmetro da função `printf()` por meio destas linhas:

```
#include <stdio.h>

void main(){
int a;
a = 10;
printf("A variavel a vale %d. Seu sucessor e o %d.\n", a, a + 1);
}
```

Em resposta a esses comandos, o computador exibirá:

A variavel a vale 10. Seu sucessor e o 11.

— No próximo exemplo, utilizamos variáveis do tipo `char`:

```
#include <stdio.h>

void main(){
char ch1, ch2, ch3;
ch1 = 'H';
ch2 = 'o';
ch3 = 'W';
printf("%cell%c %corld.\n", ch1, ch2, ch3);
}
```

Este será o resultado:

Hello World.

SAIBA MAIS

Para ampliar seus conhecimentos, listamos os principais formatos de escrita e leitura das variáveis, usados com a função printf():

Tipo	Formato	Observações
char	%c	Um único caractere
int	%d ou %i	Um inteiro (Base d ecimal)
int	%o	Um inteiro (Base o ctal)
int	%x ou %X	Um inteiro (Base hex adecimal)
short int	%hd	Um short inteiro (Base d ecimal)
long int	%ld	Um long inteiro (Base d ecimal)
unsigned short int	%hu	Short inteiro positivo

unsigned int	%u	Inteiro positivo
unsigned long int	%lu	Long inteiro positivo
float	%f ou %e ou %E	
double	%f ou %e ou %E	



Atenção! Para visualizaçãocompleta da tabela utilize a rolagem horizontal

O próximo exemplo mostra o uso de printf com variável do tipo float.

Observe o exemplo a seguir:

```
#include <stdio.h>

void main(){
float a;
a = 12.5;
printf("a = %f\n", a);
}
```

Na tela, será exibido:

```
a = 12.500000
```

Repare que a variável do tipo float é armazenada com seis casas decimais. Para reduzir esse número, utiliza-se %.1f, %.2f, entre outros. O número entre “.” e “f” indica as casas decimais exibidas. É importante lembrar que o conteúdo da variável permanece inalterado, visto que a mudança afeta apenas a forma como será feita a escrita na tela. Vamos fazer um teste!

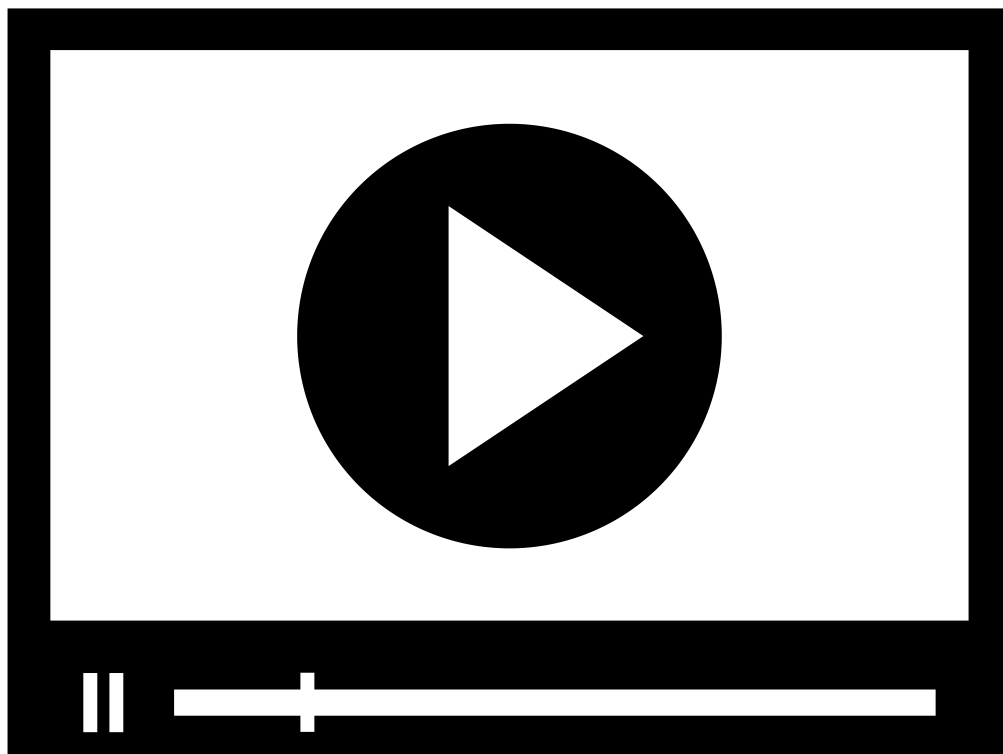
No exemplo anterior, caso alterássemos a última linha para:

```
printf("a = %.1f\n", a);
```

O resultado seria:

```
a = 12.5
```

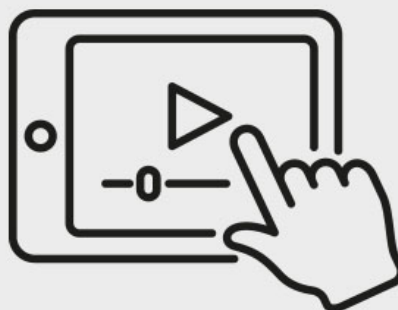
Outra função que pode ser usada para a escrita na tela é a puts(), traduzida do inglês como **colocar caractere** (*put* + *string*). Tanto puts (“Hello World”); quanto printf(“Hello World”); terão o mesmo efeito.



E AÍ, ALGUMA DÚVIDA SOBRE OS COMANDOS DE SAÍDA DE DADOS?

Então assista ao vídeo a seguir, onde o professor Humberto Henriques vai responder às principais dúvidas sobre os **comandos de saída**.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



TEORIA NA PRÁTICA

5. Execute o seguinte trecho de código escrito em C e observe o que será exibido na tela.

```
#include <stdio.h>

void main(){
int a, b, c;

a = 1;
b = a + 3;
c = a;

printf("b = %d e c = %d.\n", b, c);
}
```

FEEDBACK

Ao usar o símbolo %d, o conteúdo das variáveis b e c será colocado na frase e será exibido o seguinte resultado:

b = 4 e c = 1

6. Execute o seguinte código escrito na linguagem C e observe o que será exibido na tela.

```
#include<stdio.h>

int main()
{
printf("Valor total: %.1f\n", 9.1415169265);
return 0;
}
```

FEEDBACK

Ao usar o símbolo %.1f, o conteúdo da variável será exibido com apenas uma casa decimal:

Valor total: 9.1

7. Determine qual é a função que as strings “%d”, “%f” e “%s” estão usualmente associadas na linguagem C.

FEEDBACK

Os símbolos %d, %f e %s são utilizados para compor a frase que a função printf() vai exibir na tela.

DEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Teoria na Prática – Exemplo de comando de saída na linguagem Java

Teoria na Prática – Exemplo de comando de saída na linguagem Python

VERIFICANDO O APRENDIZADO

1. CONSIDERE O SEGUINTE TRECHO DE CÓDIGO ESCRITO EM C:

```
#INCLUDE <STDIO.H>

VOID MAIN(){
INT A, B;
A = 5;
B = A%2;
A = A + 1;
PRINTF("A = %D E B = %D.\N", A, B);
}
```

ASSINALE A ALTERNATIVA QUE APRESENTA, CORRETAMENTE, O CONTEÚDO A SER EXIBIDO NA TELA QUANDO O TRECHO FOR EXECUTADO.

A) a = %d e b = %d.\n.

B) a = 5 e b = 1.\n.

C) a = 6 e b = 1.

D) $a = 6$ e $b = 0$.

2. CONSIDERE O SEGUINTE TRECHO DE CÓDIGO ESCRITO EM C:

```
#INCLUDE <STDIO.H>

VOID MAIN(){
  CHAR LETRA;
  INT A;
  A = 10;
  LETRA = 'L';
  LETRA = LETRA + A%2;
  PRINTF("A = %D E LETRA = %C.\N", A, LETRA);
}
```

ASSINALE A ALTERNATIVA QUE APRESENTA, CORRETAMENTE, O CONTEÚDO A SER EXIBIDO NA TELA QUANDO O TRECHO FOR EXECUTADO:

- A) $a = 10$ e letra = L.
- B) $a = 10$ e letra = M.
- C) $a = 0$ e letra = L.
- D) $a = 10$ e letra = K.

GABARITO

1. Considere o seguinte trecho de código escrito em C:

```
#include <stdio.h>

void main(){
  int a, b;
  a = 5;
```

```
b = a%2;  
a = a + 1;  
printf("a = %d e b = %d.\n", a, b);  
}
```

Assinale a alternativa que apresenta, corretamente, o conteúdo a ser exibido na tela quando o trecho for executado.

A alternativa **"C "** está correta.

A variável b recebe o resto de a dividido por 2. Como a, nesse momento, tem valor 5, o resto da divisão por 2 é 1. A variável a, após a atribuição de valor de b, é incrementada em uma unidade.

2. Considere o seguinte trecho de código escrito em C:

```
#include <stdio.h>  
void main(){  
char letra;  
int a;  
a = 10;  
letra = 'L';  
letra = letra + a%2;  
printf("a = %d e letra = %c.\n", a, letra);  
}
```

Assinale a alternativa que apresenta, corretamente, o conteúdo a ser exibido na tela quando o trecho for executado:

A alternativa **"A "** está correta.

Como o resto da divisão de a por 2 é igual a 0, o valor da variável a não é alterado.

MÓDULO 3

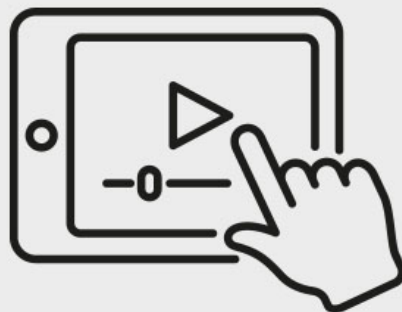
⦿ Executar os comandos de entrada de dados

COMANDOS DE ENTRADA DE DADOS

Já conhecemos os comandos de saída. Agora você vai conhecer os **comandos de entrada**, utilizados na programação para **receber e processar as informações fornecidas pelo usuário**. Mas antes, vamos novamente retomar aquele teste estilo BuzzFeed que você respondeu no início do tema.

No vídeo a seguir, o professor Humberto Henriques explica a relação do teste BuzzFeed com os **comandos de entrada de dados**.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



CONCEITOS

No cotidiano do programador, além de exibir a escrita formatada na tela, é preciso ler os dados informados pelo usuário. Para isso, utilizam-se **comandos de entrada**, permitindo a leitura formatada, principalmente, a partir do teclado, como é o caso do `scanf()`.

A função `scanf()` permite que o valor informado pelo usuário seja **armazenado em uma variável** e, posteriormente, usado para diversos cálculos.



Imagem: monkographic / Shutterstock.com

Para chamar essa função, basta passar dois parâmetros entre os parênteses. Observe:

```
#include <stdio.h>

void main(){
int numero;
printf("Entre com um número inteiro:\n");
scanf("%d", &numero);
}
```

1

O primeiro é composto pela string que traz o formato de leitura, com %d, %f ou %c entre aspas.

2

O segundo armazena o valor recebido, sendo o nome dessa variável precedido de &.

ATENÇÃO

É importante que você observe alguns detalhes:

O **formato de leitura** se mantém **igual ao da escrita na tela**: %d para as variáveis do tipo int, %f para as do tipo float e %c para as do tipo char.

Por enquanto, não vamos nos aprofundar no porquê do uso do & antes do nome da variável.

Saiba que não seguir essa recomendação pode causar consequências inesperadas.

Não confunda o símbolo & (comercial) com o operador lógico &&.

Não inclua o caractere especial '\n' na string parâmetro da função scanf().

VAMOS ENTENDER MELHOR COMO USAR A FUNÇÃO SCANF() ?

Observe o código a seguir:

```
#include <stdio.h>

void main(){
int numero;

printf("Entre com um número inteiro:\n");
scanf("%d", &numero);
}
```

Ao término de sua execução, a variável numero armazenará o valor informado pelo usuário via teclado. Poderíamos incluir mais uma linha, após a função scanf(), para escrever na tela a confirmação do número armazenado.

Vale a pena você testar essa inclusão. Escreva a linha a seguir e execute o programa:

```
printf("O valor informado pelo usuário foi %d.\n", numero);
```

A função scanf() também pode **ler mais de uma variável** simultaneamente. Para isso, você precisa **colocar os símbolos de formato de leitura na quantidade desejada e indicar as variáveis** correspondentes, que vão armazenar os valores recebidos. Vejamos a aplicação dessa função.

Observe o código a seguir:

```
#include <stdio.h>

void main(){

float dividendo, divisor;

printf("Entre com dois numeros reais:\n");
```

```
scanf("%f %d", &dividendo, &divisor);  
printf("A divisao de %.2f por %.2f vale %.2f", dividendo, divisor, dividendo/divisor);  
}
```

Se o usuário digitar os valores 10 e 2, teremos este resultado:

Entre com dois numeros reais:

10

2

A divisao de 10.00 por 2.00 vale 5.00

No código seguinte, há uma pequena alteração em relação ao anterior:

```
#include <stdio.h>  
void main(){  
float dividendo;  
int divisor;  
printf("Entre com dois numeros reais:\n");  
scanf("%f %d", &dividendo, &divisor);  
printf("A divisao de %.2f por %d vale %.2f", dividendo, divisor, dividendo/divisor);  
}
```

Se o usuário digitar novamente os valores 10 e 2, ao executar o programa, teremos:

Entre com dois numeros reais:


10

2

A divisao de 10.00 por 2 vale 5.00

Sugerimos agora que você digite o seguinte código no seu ambiente de programação e execute-o.

```
#include <stdio.h>  
void main(){  
char ch1, ch2;  
printf("Entre com duas letras:\n");  
scanf("%c", &ch1);  
scanf("%c", &ch2);  
printf("As letras inseridas foram %c e %c.\n", ch1, ch2);  
}
```

 Verifique se ocorreu algo semelhante ao que se segue:

Entre com duas letras:

a

As letras inseridas foram a e

.

VOCÊ SABE O QUE ACONTECEU? POR QUE NÃO FOI POSSÍVEL INSERIR A SEGUNDA LETRA?

POR CAUSA DO TECLADO!

Ele armazena temporariamente tudo o que digitamos, mas não repassa instantaneamente para o sistema. Podemos digitar alguma letra e apagá-la com a tecla **backspace** (←), mas quando apertamos a tecla **enter**, o sistema recebe a letra que digitamos e o **enter**.

Esse armazenamento temporário ocorre no chamado *buffer* do teclado. Como as variáveis do exemplo anterior recebem caracteres, a **letra** e o **enter** são armazenados, respectivamente, em `ch1` e `ch2`. Por isso, ocorre esse comportamento inesperado. Existem duas formas de evitar que isso aconteça:

1

Antes do símbolo de formato de leitura, você pode utilizar a função `scanf()` com um espaço na string. Isso fará com que sejam ignorados caracteres especiais, como o **enter**. Assim, o código seria alterado para:

```
scanf(" %c", &ch2);
```

2

Após a primeira chamada da função `scanf()`, efetue a limpeza do *buffer* do teclado com a seguinte instrução, caso seu sistema operacional seja o Windows:

```
fflush(stdin);
```

— Caso seja usuário do Linux, utilize a função:

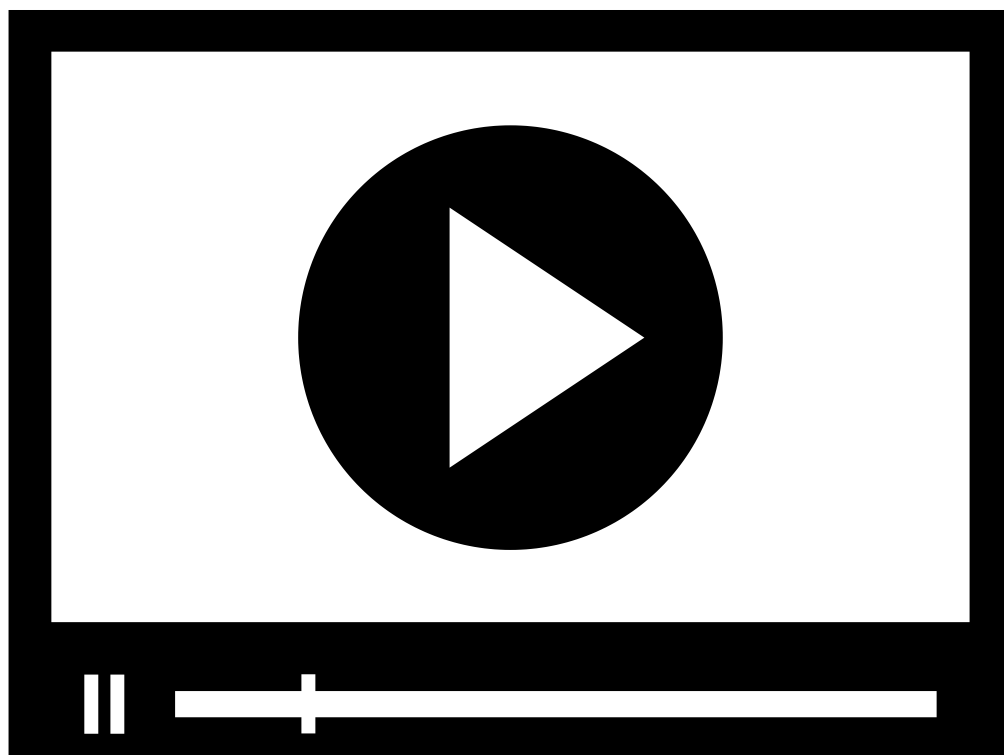
ATENÇÃO

Temos usado a função `scanf()` com os nomes das variáveis precedidos de `&`. Esse operador deve ser lido como o **endereço de**. Assim, quando passamos o parâmetro `&numero` para a função `scanf`, estamos informando o endereço na memória da variável `numero`. Por essa razão, todas as variáveis dos tipos `char`, `int`, `float` e `double` devem ser precedidas de `&`.

Outra função que pode ser usada para a leitura de `char`, a partir do teclado, é a `getc`, traduzida do inglês como **pegar o caractere**. Dessa forma, se declararmos a variável:

```
char ch1;
```

Tanto `getc (ch1);` quanto `scanf("%c", ch1);` terão o mesmo efeito.



Antes de finalizarmos, assista ao vídeo a seguir, no qual o professor Humberto Henriques resolve as principais dúvidas sobre os **comandos de entrada de dados**.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



TEORIA NA PRÁTICA

8. Execute os códigos a seguir escritos em C. Em seguida, como usuário, entre com os valores:

30

H

```
#include <stdio.h>

void main(){
    char inicial;
    int idade;

    printf("Entre com a sua idade e a sua inicial:\n");
    scanf("%d %c", &idade, &inicial);
    printf("Voce tem %d anos e seu nome começa com %c\n", idade, inicial);
}
```

FEEDBACK

Após a execução dos códigos, o conteúdo exibido na tela será:

Voce tem 30 anos e seu nome começa com H.

O valor **30** será armazenado na variável `idade`, enquanto a variável `inicial` guardará o caractere **'H'**.

9. Execute os códigos escritos em C a seguir. Em seguida, como usuário, entre com os valores:

30

H

```
#include <stdio.h>

void main(){
char inicial;
int idade;

printf("Entre com a sua idade e a sua inicial:\n");
scanf("%d %c", idade, inicial);
printf("Voce tem %d anos e seu nome começa com %c\n", idade, inicial);
}
```

FEEDBACK

Após a execução dos códigos, ocorrerá um erro e nada será exibido na tela. Isso aconteceu pois a função `scanf()` apresenta variáveis sem o operador `&`.

10. Execute os códigos escritos em C a seguir. Em seguida, como usuário, entre com os valores:

30

H

```
#include <stdio.h>

void main(){
char inicial;
int idade;

printf("Entre com a sua idade e a sua inicial:\n");
scanf("%d", &idade);
scanf("%c", &inicial);
printf("Voce tem %d anos e seu nome começa com %c\n", idade, inicial);
}
```

FEEDBACK

Após a execução dos códigos, o conteúdo exibido na tela será:

Voce tem 30 anos e seu nome começa com.

Ao apertar **30** e **enter**, a variável `idade` armazenará o valor 30 e a variável `inicial`, o *enter*.

VEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Teoria na Prática – Exemplo de comando de entrada na linguagem Java

Teoria na Prática – Exemplo de comando de entrada na linguagem Python

VERIFICANDO O APRENDIZADO

1. CONSIDERE O SEGUINTE TRECHO DE CÓDIGO ESCRITO EM C:

```
#INCLUDE <STDIO.H>

VOID MAIN(){
INT A, B, C;

C = A-B;

PRINTF("ENTRE COM DOIS INTEIROS:\N");
SCANF("%D %D", &A, &B);
PRINTF("A DIFERENÇA ENTRE %D E %D VALE %D\N", A, B, C);
}
```

SUPONHA QUE O USUÁRIO TENHA ENTRADO COM OS VALORES:

15

6

ASSINALE A ALTERNATIVA QUE APRESENTA, CORRETAMENTE, O RESULTADO DA EXECUÇÃO DESSE TRECHO.

A) A diferença entre 15 e 6 vale 9.

B) Ocorrerá um erro porque a variável c não está precedida de & na atribuição.

C) Ocorrerá um erro porque as variáveis a e b não estão precedidas de & na instrução de escrita do resultado.

D) A variável c terá um valor aleatório.

2. CONSIDERE O SEGUINTE TRECHO DE CÓDIGO ESCRITO EM C:

```
#INCLUDE  
VOID MAIN(){  
FLOAT ALTURA, PESO, IMC;  
PRINTF("ENTRE COM A SUA ALTURA E O SEU PESO:\N");  
SCANF("%F %F", &ALTURA, &PESO);  
IMC = (PESO/ALTURA)/ALTURA;  
PRINTF("SEU IMC VALE %F\N", IMC);  
}
```

SUPONHA QUE O USUÁRIO TENHA ENTRADO COM OS VALORES:

1.80

75

ASSINALE A ALTERNATIVA QUE APRESENTA, CORRETAMENTE, O RESULTADO DA EXECUÇÃO DESSE TRECHO.

A) Seu IMC vale 23.14.

B) Seu IMC vale 23.

C) Seu IMC vale 23.148149.

D) Ocorrerá um erro porque a variável imc não está precedida de & na atribuição.

1. Considere o seguinte trecho de código escrito em C:

```
#include <stdio.h>

void main(){
int a, b, c;
c = a-b;
printf("Entre com dois inteiros:\n");
scanf("%d %d", &a, &b);
printf("A diferença entre %d e %d vale %d\n", a, b, c);
}
```

Suponha que o usuário tenha entrado com os valores:

15

6

Assinale a alternativa que apresenta, corretamente, o resultado da execução desse trecho.

A alternativa "D " está correta.

A atribuição $c = a - b$ é feita antes que as variáveis a e b recebam os valores informados pelo usuário. Logo, não se sabe o valor delas.

2. Considere o seguinte trecho de código escrito em C:

```
#include
void main(){
float altura, peso, imc;
printf("Entre com a sua altura e o seu peso:\n");
scanf("%f %f", &altura, &peso);
imc = (peso/altura)/altura;
printf("Seu IMC vale %f\n", imc);
}
```

Suponha que o usuário tenha entrado com os valores:

1.80

75

Assinale a alternativa que apresenta, corretamente, o resultado da execução desse trecho.

A alternativa **"C "** está correta.

A impressão na tela de uma variável do tipo float é feita com 6 casas decimais. A entrada de dados é feita corretamente, com as variáveis altura e peso armazenando os valores 1.80 e 75, respectivamente.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

Você aprendeu as principais formas de interagir com o usuário. Os comandos de entrada e saída de dados são essenciais na sua jornada de formação como programador. Por isso, fique atento aos detalhes e procure sempre programar de forma organizada. Isso vai evitar erros bobos e tornar sua experiência mais agradável.

Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.



VERSÃO DO TEXTO

REFERÊNCIAS

ARAÚJO, I. **Howard Gardner**. *In*: Escola Educação. Consultado em meio eletrônico em: 16 mar. 2020.

DAMAS, L. **Linguagem C**. Grupo Gen-LTC, 2016.

SCHILDT, H. **C completo e total**. São Paulo: Makron, 1997.

SUGAI, A. **O que é o código ASCII e para que serve? Descubra**. *In*: Tech Tudo. Publicado em: 15 fev. 2015.

WIKIPÉDIA. **Robert Berner**. Consultado em meio eletrônico em: 16 mar. 2020.

EXPLORE+

Para ter acesso a exercícios e desafios mais complexos, recomendamos a visita ao site *URI Online Judge*.

CONTEUDISTA

Humberto Henriques de Arruda

 CURRÍCULO LATTES