

## Comandos de entrada e saída

Prof. Humberto Henriques de Arruda

### 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

Recomendamos que você instale o software Dev C++, pois esse será o ambiente de programação que utilizaremos. Além deste, existem outros compiladores que suportam a linguagem C, como o Code::Blocks.

## Objetivos

### Módulo 1

## Comando de atribuição

Utilizar o comando de atribuição.

### Módulo 2

## Comandos de saída de dados

Aplicar os comandos de saída de dados.

Módulo 3

## Comandos de entrada de dados

Executar os comandos de entrada de dados.



## Introdução

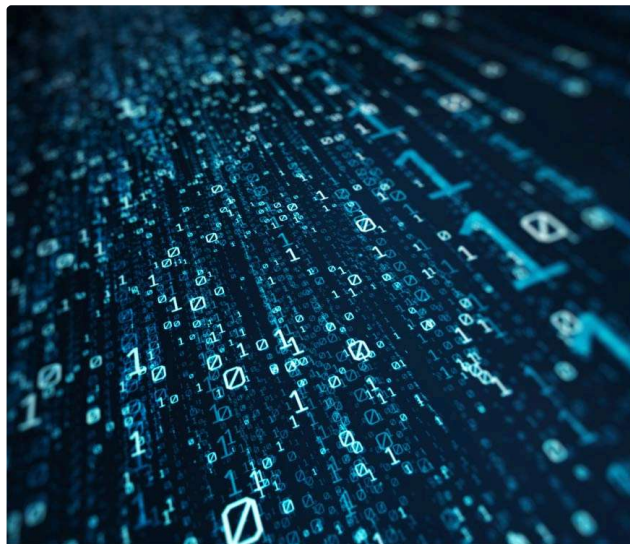
Você já ouviu falar ou realizou algum teste no site BuzzFeed? Esses tipos de teste são muito conhecidos e utilizados nas redes sociais para identificar perfis variados de usuários em diferentes contextos. Sugerimos que você faça o teste sobre os diferentes tipos de inteligência propostos pelo psicólogo Howard Gardner, que permitirá identificar as inteligências que você possui, e observe atentamente a dinâmica por trás dele. Ele mostra exatamente o que vamos tratar neste conteúdo: comandos de entrada e saída.

Ao realizá-lo, você precisará inserir dados para que um resultado seja exibido ao final.

Para ajudá-lo a entender melhor essa dinâmica, o professor Humberto Henriques explica a seguir os contextos nos quais são utilizados os comandos de entrada e saída. Confira!

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





## 1 - Comando de atribuição

Ao final deste módulo, você será capaz de utilizar o comando de atribuição.

# Comando de atribuição de valor

Neste vídeo, vamos explorar os comandos de atribuição de valores a variáveis em C. Aprenderemos como utilizar o operador de atribuição para armazenar e atualizar valores em variáveis. Além disso, discutiremos técnicas avançadas de atribuição, tais como atribuição composta e atribuição condicional.

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



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.



Todos os comandos apresentados aqui obedecem à sintaxe (conjunto de regras) da **linguagem C** e do **Portugol**. Antes de utilizar o comando de atribuição, você deve **inicializar** a variável.

Tomemos como exemplo a declaração da variável inteira chamada **a**. Vejamos como realizar esse procedimento:

C



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

C



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

C



C



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. Com a seguinte instrução, é dado o valor 2 às variáveis **a** e **b**.  
Vejamos um exemplo:

C



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. Observe:

C



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:

## Exemplo

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

$a \leftarrow a + 1$  (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:

$a \leftarrow 5$  (pseudocódigo) ou  $a = 5$  (Portugol e C).

$a \leftarrow a + 1$  (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:

```
C
```



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:

```
C
```



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](#).

## 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:

<code>\7</code>	<i>Bell</i> (sinal sonoro do computador)	<code>\</code>
<code>\a</code>	<i>Bell</i> (sinal sonoro do computador)	<code>\'</code>
<code>\b</code>	<i>BackSpace</i>	<code>\"</code>
<code>\n</code>	<i>New Line</i> (mudança de linha)	<code>\?</code>
<code>\r</code>	<i>Carriage Return</i>	<code>\000</code>
<code>\t</code>	Tabulação Horizontal	<code>\xyy</code>
<code>\v</code>	Tabulação Vertical	

Tabela: Caracteres especiais.  
Humberto Henriques de Arruda

O próximo exemplo ilustra essa relação. Veja:

c



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



# Atribuição de valor a uma variável

Entenda a seguir as principais dúvidas sobre atribuição de valor a uma variável.

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

## Vamos praticar

Você receberá agora uma série de práticas para realizar em seu ambiente de programação. Tente executá-las. Vamos lá!

### Prática 1

Vamos descobrir qual é o valor da variável `cont` após a execução das seguintes linhas:

Exercício 1

TUTORIAL COPIAR

`C`

null

null

O valor da variável é 2. A variável `cont` é inicializada com 1, mas a segunda linha acresce uma unidade a esse valor.

### Prática 2



Vamos descobrir qual é o valor da variável **escolha** após a execução das seguintes linhas:

Exercício 1

 TUTORIAL

 COPIAR

C

```

null

[ ]

null
```

▶

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.

Prática 3

Vamos descobrir qual é o valor da variável **c** após a execução das seguintes linhas:

Exercício 1

 TUTORIAL

 COPIAR

C

```

null

[ ]

null
```

▶

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

Prática 4

Vamos descobrir qual é o resultado da execução das seguintes linhas:

Exercício 1

 TUTORIAL  COPIAR

C

null

null



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**.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Qual é o valor armazenado na variável **a** após a execução destas linhas?

C



A 1

B 2

- C 3
- D Um valor aleatório.
- E Ocorrerá um erro de compilação.

Parabéns! 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.

## Questão 2

Qual é o valor armazenado na variável **ch** após a execução destas linhas?

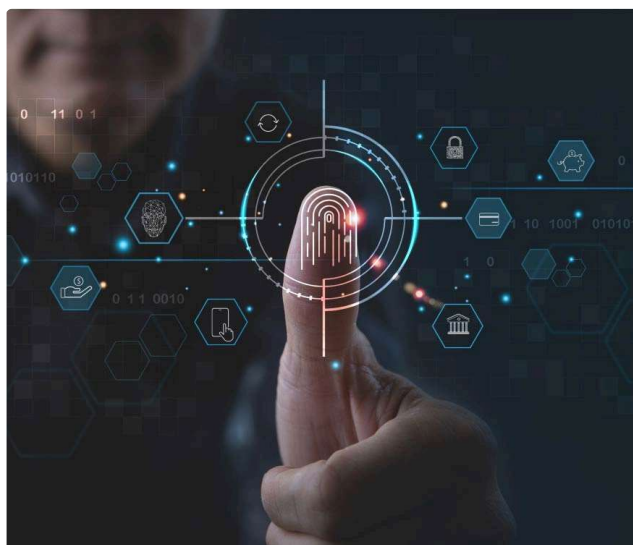
C



- A 1
- B 'A'
- C 'B'
- D Ocorrerá um erro de compilação.
- E 'A' + 'a'

Parabéns! 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'.



## 2 - Comandos de saída de dados

Ao final deste módulo, você será capaz de aplicar os comandos de saída de dados.

# Comandos de saída

## Contextualização

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 se lembra do teste estilo BuzzFeed indicado no início deste conteúdo?



## O teste a partir dos comandos de saída

Veja agora 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.



## Programa *Hello World!*

Você sabe qual a relação entre um programa e a linguagem C? Confira as suas respectivas definições:

## Programa

É uma sequência de instruções dadas para resolver um problema.


## 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!

C



Digite a sequência de comandos do código anterior no compilador de código abaixo:

Exercício 1


 TUTORIAL

 COPIAR

C

null

null



## Função Printf()

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. Mas você pode estar se perguntando:

## O que essa função exhibe para o usuário?

Ela exhibe 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:

C



Digite a sequência de comandos do código anterior no compilador de código abaixo e observe:


Exercício 1

 TUTORIAL  COPIAR

C

null

null



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:

C

Digite a sequência de comandos do código anterior no compilador de código abaixo e observe o resultado:

Exercício 1

TUTORIAL

COPIAR

C

null

null

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:

C

Digite a sequência de comandos do código anterior no compilador de código abaixo e observe o resultado de saída:

Exercício 1

TUTORIAL

COPIAR

C

---

null

null



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:

C



Agora, digite a sequência de comandos do código anterior no compilador de código abaixo:

Exercício 1

 TUTORIAL  COPIAR

C

---

null

null



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

C





Digite a sequência de comandos do código anterior no compilador de código abaixo:

Exercício 1

 TUTORIAL

 COPIAR

C

null

null

No próximo exemplo, utilizamos variáveis do tipo **char**. Confira:

C



Mais uma vez, digite a sequência de comandos do código anterior no compilador de código abaixo:

Exercício 1

 TUTORIAL

 COPIAR

C

null

null

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 decimal)
int	%o	Um inteiro (Base octal)
int	%x ou %X	Um inteiro (Base hexadecimal)
short int	%hd	Um short inteiro (Base decimal)
long int	%ld	Um long inteiro (Base decimal)
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	

Tabela: Formatos de escrita e leitura das variáveis usados com a função printf().  
Humberto Henriques de Arruda.

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

Observe:

```
c
```

Digite a sequência de comandos do código anterior no compilador de código abaixo:

Exercício 1

 TUTORIAL  COPIAR

C

```

null



```

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:

C 

O resultado seria:

C 

Viu só?

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.



# Comandos de saída de dados

Veja a seguir as principais dúvidas sobre os comandos de saída de dados.

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

## Vamos praticar

Você receberá agora uma série de práticas para realizar em seu ambiente de programação. Tente executá-las. Vamos lá!

### Prática 1

Vamos executar o seguinte trecho de código escrito em C e observe o que será exibido na tela:

Exercício 1

TUTORIAL

COPIAR

C

```

null


```

```

null


```

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

C

Prática 2

Vamos executar o seguinte código escrito na linguagem C e observe o que será exibido na tela:

Exercício 1

 TUTORIAL

 COPIAR

C

```


null


```

```

null


```



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

C



Prática 3

Vamos determinar qual é a função que as strings `"%d"`, `"%f"` e `"%s"` estão usualmente associadas na linguagem C.

A solução é que os símbolos `%d`, `%f` e `%s` são utilizados para compor a frase que a função `printf()` vai exibir na tela.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

## Questão 1

Considere o seguinte trecho de código escrito em C:

C



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`
- E O programa irá apresentar erro de execução.

Parabéns! 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.

## Questão 2

Considere o seguinte trecho de código escrito em C:

C

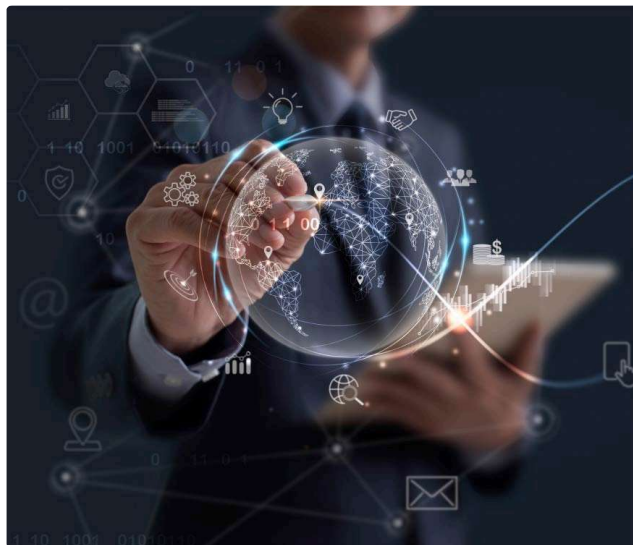


Assinale a alternativa que apresenta, corretamente, o conteúdo a ser exibido na tela quando o trecho for executado:

- A     `a = 10` e `letra = L`
- B     `a = 10` e `letra = M`
- C     `a = 0` e `letra = L`
- D     `a = 10` e `letra = K`
- E     O programa irá apresentar um erro de execução.

Parabéns! 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.



### 3 - Comandos de entrada de dados

Ao final deste módulo, você será capaz de executar os comandos de entrada de dados.

## Comandos de entrada

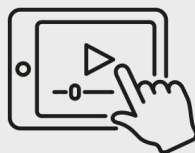
Já conhecemos os **comandos de saída**. Agora você vai conhecer os comandos de entrada, utilizados na programação para receber e processar as informações fornecidas pelo usuário. Mas, antes, vamos novamente retomar aquele teste estilo BuzzFeed do início do conteúdo.



## O teste a partir dos comandos de entrada

Veja a seguir 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.



## Função Scanf ()

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.

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

```
c
```



Veja a seguir o que consiste cada um desses parâmetros:

### O primeiro

É composto pela string que traz o formato de leitura, com **%d**, **%f** ou **%c** entre aspas.

### O segundo



Armazena o valor recebido, sendo o nome dessa variável precedido de **&**.

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**;
- Não vamos nos aprofundar, por enquanto, 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:

C



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:

C



Execute o programa no emulador:

Exercício 1

TUTORIAL COPIAR

C

---

null

null



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. Se o usuário digitar 10<enter>2, você verá o seguinte:

Exercício 1

 TUTORIAL  COPIAR

C

---

null

null



Se o usuário digitar duas letras, você verá o seguinte resultado no código abaixo:

Exercício 1

 TUTORIAL  COPIAR

C

---

null

null



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: A primeira é que, quando 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:

C



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:

C



Caso seja usuário do Linux, utilize a função:

C



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:

```
c
```



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



## Comandos de entrada de dados

Entenda a seguir as principais dúvidas sobre os comandos de entrada de dados.

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



### Vamos praticar

Você receberá agora uma série de práticas para realizar em seu ambiente de programação. Tente executá-las. Vamos lá!

#### Prática 1

Vamos considerar o seguinte código:

```
c
```



Digite esse código no compilador de código abaixo; entre com os seguinte valores:

30

H

Exercício 1

 TUTORIAL  COPIAR

C

null

null

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

C 

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

Prática 2

Vamos considerar o seguinte código:

C



Digite esse código no compilador de código abaixo; entre com os seguintes valores:

30

H

Exercício 1

 TUTORIAL  COPIAR

C

---

null

null

▶

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

Prática 3

Vamos considerar o seguinte código:

C



Digite esse código no compilador de código abaixo; entre com os seguintes valores:

30

H

Exercício 1

 TUTORIAL  COPIAR

C

null

null

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

C



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

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Considere o seguinte trecho de código escrito em C:

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.
- E Ocorrerá um erro de execução no printf.

Parabéns! 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.

## Questão 2

Considere o seguinte trecho de código escrito em C:

C



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.
- E Seu IMC será 23.148149.

Parabéns! 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.

## 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.



### Podcast

Para encerrar, ouça um resumo sobre a importância dos comandos de entrada e saída.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.



## Explore +

Para ter acesso a exercícios e desafios mais complexos, recomendamos visitar o site [Online Judge](#).

## Referências

ARAÚJO, I. **Howard Gardner**. Escola Educação. Consultado na internet 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**. Tech Tudo. Publicado em: 15 fev. 2015.

### Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

[Download material](#)

O que você achou do conteúdo?



[! Relatar problema](#)