# LINGUAGEM DE PROGRAMAÇÃO C

André Feitoza de Mendonça

INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

# LINGUAGEM DE PROGRAMAÇÃO C

André Feitoza de Mendonça



INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

**Autor**

André Feitoza de Mendonça

Bacharel em Engenharia da Computação e mestre em Ciência da Computação pela Universidade Federal de Pernambuco. Atua como gerente de projetos de sistemas voltados à automação do processo de aplicação de exames nacionais (ENADE, por exemplo), no Instituto Nacional de Estudos e Pesquisas Anísio Teixeira, INEP/MEC. Foi analista de processos com vistas à automatização do Processo Decisório do Conselho Deliberativo Nacional do SEBRAE. Possui publicações em Congressos e em revistas especializadas em Ciência da Computação.

**Design Instrucional**
Pedro Meneses

**Projeto Gráfico**
NT Editora

**Revisão**
Filipe Lopes
Ricardo Moura

**Capa**
NT Editora

**Ilustração**
Thiago Ferreira

**Editoração Eletrônica**
Nathália Nunes
Daniel Lopes

**NT Editora, uma empresa do Grupo NT**
SCS Quadra 2 – Bl. C – 4° andar – Ed. Cedro II
CEP 70.302-914 – Brasília – DF
Fone: (61) 3421-9200
sac@grupont.com.br
www.nteditora.com.br e www.grupont.com.br

Mendonça, André Feitoza de.
Linguagem de programação C / André Feitoza de Mendonça – 1. ed. – Brasília: NT Editora, 2018.
190 p. il. ; 21,0 X 29,7 cm.

ISBN 978-85-8416-241-3

1. Programação. 2. Sistema.
I. Título

## ÍCONES

Prezado(a) aluno(a),
Ao longo dos seus estudos, você encontrará alguns ícones na coluna lateral do material didático. A presença desses ícones o(a) ajudará a compreender melhor o conteúdo abordado e a fazer os exercícios propostos. Conheça os ícones logo abaixo:

**Saiba mais**
Esse ícone apontará para informações complementares sobre o assunto que você está estudando. Serão curiosidades, temas afins ou exemplos do cotidiano que o ajudarão a fixar o conteúdo estudado.

**Importante**
O conteúdo indicado com esse ícone tem bastante importância para seus estudos. Leia com atenção e, tendo dúvida, pergunte ao seu tutor.

**Dicas**
Esse ícone apresenta dicas de estudo.

**Exercícios**
Toda vez que você vir o ícone de exercícios, responda às questões propostas.

**Exercícios**
Ao final das lições, você deverá responder aos exercícios no seu livro.

**Bons estudos!**

## Sumário

**1 CONCEITOS BÁSICOS DA LINGUAGEM C ..... 9**
1.1 Um pouco de história e características de C ..... 9
1.2 Elementos básicos da linguagem C ..... 13
1.3 Meus primeiros programas em C ..... 16

**2 SISTEMAS NUMÉRICOS E OPERADORES EM C ..... 25**
2.1 Sistemas numéricos ..... 25
2.2 Operadores ..... 30

**3 ESTRUTURAS DE CONTROLE ..... 46**
3.1 O que são estruturas de controle? ..... 46
3.2 Switch ..... 50
3.3 While ..... 54
3.4 Do-while ..... 57
3.5 For ..... 60

**4 FUNÇÕES EM C ..... 66**
4.1 O que são funções? ..... 66
4.2 Escopo de variáveis ..... 69
4.3 Pilha de execução de funções ..... 72
4.4 Funções recursivas ..... 77

**5 PONTEIROS ..... 85**
5.1 Declaração de ponteiros ..... 85
5.2 Como utilizar ponteiros ..... 87
5.3 Estruturas e ponteiros ..... 90
5.4 Passagem de ponteiros para funções ..... 96

***6 ARRAYS ..... 102***
6.1 O que são *arrays*? ..... 102
6.2 Aritmética de ponteiros ..... 106
6.3 *Strings* em C ..... 109
6.4 Matrizes ..... 113

**7 ALOCAÇÃO DE MEMÓRIA ..... 120**
7.1 O que é alocação estática e dinâmica? ..... 120
7.2 Alocando uma variável dinamicamente ..... 122

7.3 *Arrays* alocados em tempo de execução ..... 127
7.4 Alocação dinâmica de matrizes ..... 131

**8 INTERFACES EM C E MANIPULAÇÃO DE ARQUIVOS ..... 137**
8.1 Arquivos de interface e de implementação ..... 137
8.2 Um pouco mais sobre a biblioteca stdio ..... 143

**9 BIBLIOTECA STDLIB ..... 154**
9.1 Funções de conversão ..... 154
9.2 Controle de processos ..... 158
9.3 Geração de números aleatórios ..... 163
9.4 Ordenação e busca de elementos ..... 165

**10 CRIANDO UM PROJETO ..... 170**
10.1 Especificando o sistema em C ..... 170
10.2 Gerenciador ..... 174
10.3 Andar ..... 176
10.4 Vaga ..... 178
10.5 Carro ..... 180

**GLOSSÁRIO ..... 187**

**BIBLIOGRAFIA ..... 189**

# APRESENTAÇÃO

Caro(a) estudante,

Seja bem-vindo à **Linguagem de programação C**!

Atualmente, pode-se observar que os computadores estão cada vez mais presentes no nosso cotidiano. Já existem diversos eletrodomésticos inteligentes, como geladeiras que identificam produtos em falta e realizam compras *on-line* para suprir determinado alimento. Inúmeras soluções de automação de residências estão disponíveis no mercado, possibilitando que lâmpadas possam ser acionadas mediante comando de voz. Todas essas novidades necessitam de computadores programados para realizar tais atividades de maneira eficiente. Nesse contexto de profunda automação, a linguagem de programação C se encaixa perfeitamente.

A linguagem C surgiu no início da década de 1970 e foi concebida para substituir programas escritos diretamente na linguagem de máquina Assembly. Instruções em Assembly lidam diretamente com os registradores e a memória de um computador. Dessa forma, um programa nessa linguagem não pode ser executado em outra plataforma computacional sem ser devidamente adaptado. Assim, a linguagem C proporcionou que programas pudessem ser escritos em alto nível. Ou seja, se em um programa Assembly existe um instrução que atribui um valor numérico diretamente a um determinado registrador da CPU, na linguagem C, uma variável do programa recebe o valor numérico. O programa em C passa por um processo de compilação que gera instruções em Assembly. Ou seja, para esse caso em questão, é o compilador que determina qual registrador será utilizado para receber o valor numérico especificado. Portanto, C permitiu que o trabalho dos programadores fosse reduzido. Um código-fonte em C pode ser executado em várias plataformas computacionais distintas (desde que existam compiladores para tais processadores), não sendo necessária qualquer adaptação do programa escrito. A referida linguagem de programação também proporciona diversas estruturas de controle (IF, WHILE, FOR, etc) que facilitam a vida dos programadores, reduzindo o tamanho dos programas.

A linguagem C foi e ainda é muito utilizada para a escrita de sistemas operacionais (SOs). Podem-se citar como exemplo de tais sistemas o Microsoft Windows, o MAC OS X e o GNU/Linux. Os projetistas de SOs preferem C porque é uma linguagem extremamente confiável e rápida, além de possibilitar comunicação direta com o Kernel de computadores. C também é utilizado em microprocessadores, como 8051 e e ARM, os quais são empregados em diversos sistemas embarcados que podem ser encontrados de letreiros de ônibus a *smartphones*.

Cabe também destacar que a linguagem C foi a precursora de inúmeras outras linguagens de programação. Java, Javascript, Shell, PHP, C++, C# são linguagens que foram fortemente influenciadas por C. Assim, o aprendizado de C se faz muito importante para que um programador possa consolidar melhor o seu conhecimento em outras linguagens. Como C possui mecanismos de manipulação explícita de recursos do processador, uma base sólida nessa linguagem proporciona que o profissional seja capaz de tratar questão de perfomance computacional e uso racional do poder de processamento.

**Bons estudos!**

**André Feitoza de Mendonça**

# 1 CONCEITOS BÁSICOS DA LINGUAGEM C

## Objetivos

Ao finalizar esta lição, você deverá ser capaz de:

- entender as principais características da linguagem C;
- compreender o funcionamento dos principais elementos de C: variável e função;
- descrever o funcionamento dos primeiros programas C, em especial, como as funções da biblioteca stdio.h funcionam.

## 1.1 Um pouco de história e características de C

A linguagem C foi criada no início da década de 1970 e o seu pai foi o projetista Dennis Ritchie, da Bell Laboratories. C foi o resultado de um esforço para criar uma nova linguagem de programação para codificar o sistema operacional (SO) Unix.

Antes do advento de C, o Unix era totalmente codificado em Assembly. O principal problema dessa abordagem é a falta de portabilidade. Ou seja, para que o Unix pudesse executar em uma nova plataforma computacional, todo o código-fonte do SO teria de ser escrito novamente (o que, obviamente, é totalmente improdutivo). Dessa forma, C, por ser uma linguagem de programação de alto nível, foi empregada para codificar a sistema operacional Unix, permitindo que esse SO pudesse ser executado nos mais variados computadores.

A linguagem Assembly, que será mencionada no decorrer deste livro, permite que programadores possam escrever um conjunto de instruções para lidar diretamente com o processador em questão. Por exemplo, pode-se escrever a instrução MOV AX, 1Ah, ou seja, sse comando atribui ao registrador chamado de AX o valor hexadecimal 1A (na próxima lição, você aprenderá mais sobre o sistema hexadecimal). Perceba que a instrução exibida lida diretamente com um registrador do processador. Por ter esse tipo de característica, Assembly é considerada uma linguagem de baixo nível.

**Saiba mais**

Antes da linguagem C, o projetista Ken Thompson, também trabalhando para a Bell Laboratories, criou uma linguagem de programação muito parecida com C. Era uma linguagem que proporcionava a portabilidade de um programa para diversas plataformas computacionais, mas era considerada muito lenta e, por esse motivo, não prosperou. O nome dessa linguagem era B. Pelo próprio indicativo de seu nome, C foi uma evolução da linguagem B.

Até meados de década de 1980, toda a especificação da linguagem C era fornecida por publicações e livros sem uma padronização rígida. Esse fato fez com que a linguagem não avançasse como era o esperado. Entretanto, em 1985, foi criado um padrão mundialmente reconhecido para a linguagem C, chamado de ANSI C. Esse fato evitou problemas comuns da época em que programas C precisavam ser adaptados para que pudessem ser executados por determinados compiladores.

**Importante**

No mundo da tecnologia da informação, um compilador é um programa de computador que, a partir de um código-fonte de uma determinada linguagem de programação, gera um código objeto que, muitas vezes, é a própria linguagem da máquina. Dessa forma, geralmente, o compilador recebe como entrada um código-fonte de alto nível e gera código de baixo nível (mais perto da máquina). Em C, acontece o mesmo processo: para que um programa em C execute sem problemas em determinado computador, é requerido que exista um compilador C para a plataforma computacional em questão.

C é uma linguagem simples, mas extremamente rápida. Ela proporciona que os programadores deixem de lidar diretamente com registradores e posições de memória, fazendo com que os projetistas abstraiam esses detalhes. Assim, C é considerada uma linguagem de alto nível (atenção: alguns autores consideram C uma linguagem de médio nível). Entretanto, para programas muito grandes, C pode não ser adequada, uma vez que ela não implementa conceitos de orientação a objetos. Dessa forma, programas C com mais de 20.000 linhas de código tendem a se tornar desorganizados e, consequentemente, os custos de manutenção aumentam. Nesse sentido, na década de 1980, Bjarne Stroustrup foi o responsável por evoluir C e adicionar conceitos da orientação a objetos, criando a linguagem C++.

O escopo desse livro não abrangerá maiores detalhamentos com relação a essa última linguagem. Mas fique tranquilo, pois, ao finalizar os estudos neste livro, você estará mais do que preparado para aprender qualquer outra linguagem de programação que você queira.

## Programando o conhecimento

A linguagem de programação C foi inicialmente concebida para codificar qual sistema operacional?

a) Windows.

b) Unix.

c) Mac OS X

d) Linux.

**Comentário**: a linguagem C foi criada no início da década de 1970 e foi o resultado de um esforço para criar uma nova linguagem de programação para codificar o sistema operacional (SO) Unix. Logo, a alternativa correta a "b".

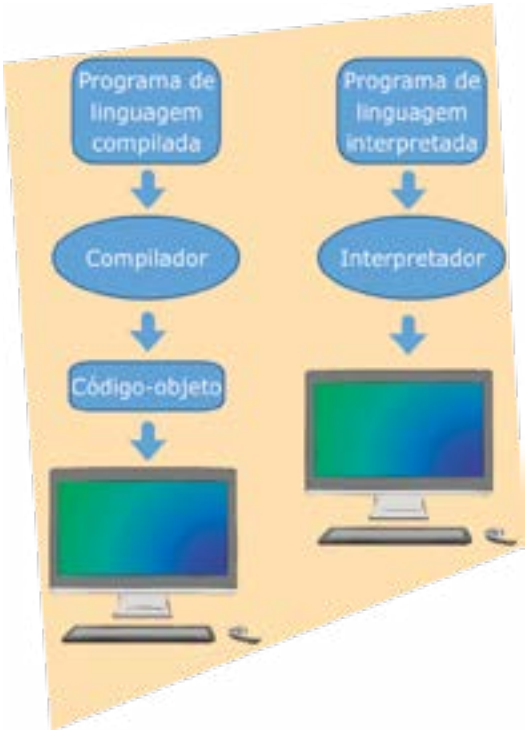
Agora, vamos detalhar um pouco as características da linguagem de programação C?

De uma forma geral, C é uma linguagem compilada, de propósito geral, estruturada e imperativa. A seguir, você poderá verificar com mais detalhes cada uma das características indicadas.

Como já indicado, C é uma linguagem compilada. As linguagens de programação podem ser compiladas ou interpretadas. Para o primeiro caso, para a efetiva execução de um programa, o código-fonte tem de, primeiramente, passar por um compilador que gera um código-objeto preparado para que a máquina o execute. Nesse caso, erros de sintaxe de linguagem são reportados pelo próprio

compilador. Por outro lado, códigos-fonte de linguagens interpretadas são executados na máquina diretamente, por meio de um programa chamado de interpretador. Nessa situação, erros de sintaxe são identificados apenas durante a execução de programas. De uma forma geral, as linguagens compiladas necessitam de um procedimento de criação de um código-objeto de baixo nível, enquanto os programas interpretados são analisados no momento de sua execução pelos chamados interpretadores. A figura ilustra a diferença entre os processos de compilação e interpretação.

**Processos de compilação e de interpretação**




A linguagem C é dita de propósito geral, ou seja, ela pode ser aplicada em uma vasta variedade de domínios diferentes. Por exemplo, C pode ser utilizado tanto para codificar sistemas operacionais, como para controlar dispositivos embarcados mais diversos. Por outro lado, existem diversas linguagens de programação classificadas como de propósito específico ou de domínio específico. Elas foram projetadas para resolver determinado problema particular. São exemplos de linguagens desse tipo o HTML e o Verilog (linguagem para descrição de hardwares).

Programas em C são construídos de forma estruturada. Assim, a linguagem de programação C apresenta duas principais características que sinalizam que uma linguagem é estruturada.

Primeiramente, C possui estruturas básicas de controle que permitem que as instruções de um programa sejam executadas de três formas distintas. Ou uma instrução é executada uma após a outra de forma sequencial ou determinado comando é selecionado para ser rodado por meio de uma condição que será testada. Por fim, uma instrução ou um conjunto delas pode ser executado repetidamente até que determinada condição ocorra.

Também garante que C seja estruturada o fato de ela ser uma linguagem que permite a criação de módulos. Tais módulos são subprogramas que possibilitam o reaproveitamento do código-fonte.

A linguagem de programação C obedece ao paradigma imperativo. Uma linguagem de programação imperativa baseia a sua execução em comandos de armazenam e processam dados. Dessa forma, quando um programa imperativo roda, ele armazena o seu estado de execução em variáveis de memória que podem ser alteradas por meio de atribuições. Ou seja, tais programas são orientados

por ações que manipulam valores de variáveis.

Ao contrário das linguagens imperativas, existem as chamadas funcionais. Tais linguagens se baseiam no conceito de função. Dessa forma, programas funcionais focam no objetivo de avaliar expressões. São exemplos desse tipo de linguagem de programação Lisp, Miranda e Haskell.

A orientação a objetos também é um paradigma bastante utilizado pelas linguagens de programação. Nesse tipo de paradigma, são definidas classes de objetos que representam elementos do mundo real que está sendo representado pelo programa. Cada classe possui atributos que indicam o seu estado atual e comportamentos definidos por seus métodos. São exemplos de linguagens baseadas nesse paradigma C++, C#, Java e PHP.

**Saiba mais**

O paradigma de uma linguagem de programação identifica um conjunto de características que são utilizadas para resolver determinados problemas. Dessa forma, os paradigmas de programação estabelecem uma visão para que o programador possa estruturar e executar seus códigos-fontes. Como já mencionado, a linguagem C possui o paradigma imperativo. Outros paradigmas atualmente existentes: funcional, lógico, declarativo, orientado a objetos, orientado a eventos e orientado a aspectos.

## Programando o conhecimento

Assinale a alternativa que indica quais são as características corretas da linguagem de programação C?

a) Compilada, funcional, estruturada.

b) Interpretada, imperativa, de propósito geral.

c) Compilada, funcional, de propósito específico.

d) Compilada, imperativa, estruturada.

**Comentário**: a linguagem C é compilada, estruturada, imperativa e de propósito geral. Dessa forma, a alternativa certa é a "d".

**Característica da linguagem C**

| Aspectos gerais | Características de C |
|---|---|
| Propósito de linguagem | Geral |
| Forma de execução | Compilada |
| Organização | Estruturada |

| Aspectos gerais | Características de C |
|---|---|
| Paradigma | Imperativo |

## 1.2 Elementos básicos da linguagem C

Antes que você possa escrever o seu primeiro programa em C, deve aprender quais são os elementos básicos da linguagem. Preparados para iniciarmos os estudos mais detalhados sobre C?

Os elementos básicos para a construção de qualquer programa C são os seguintes: variáveis e funções. Como é de seu conhecimento, a linguagem C é simples e o entendimento desses dois elementos é suficiente para que possa rodar o seu primeiro programa.

Uma variável nada mais é do que uma posição de memória identificada por meio de um identificador. Cada variável possui um tipo de dado, ou seja, quais são os valores que podem ser armazenados nessa porção de memória. Observe, na na figura abaixo, um exemplo de emprego de uma variável.

**Definição de variáveis**

```
int a;
a = 10;
char letra = 'b';
```

Você pode observar que a variável 'a' é declarada na primeira linha como do tipo inteiro. Tal qual você verá mais adiante, variáveis inteiras ocupam 2 *bytes* de memória e podem receber valores de -32.768 até 32.767. Na segunda linha, uma instrução que atribui o valor 10 à variável 'a' é executada. Por fim, outra variável chamada de 'letra' é, em uma única instrução, declarada como do tipo char e atribuída com o caractere 'b'.

Perceba que, quando estamos lindando com variáveis, podem existir comandos em um programa C que as declaram, definindo um tipo de dado e o seu identificador (ou nome). Também podem existir instruções que atribuem valores a essas variáveis previamente declaradas. Note que apenas valores compatíveis com o tipo de dado da variável podem ser atribuídos. Ainda existem comandos que, ao mesmo tempo, declaram uma variável e atribuem um valor a ela. Verifique agora, na tabela a seguir, todos os tipos de dados fornecidos pela linguagem C.

**Tipos de dado C**

| Tipo de dado | Tamanho (em bytes) | Intervalo de valores aceitos |
|---|---|---|
| Char | 1 | -128 a 127 |
| unsigned char | 1 | 0 a 255 |
| short int | 2 | -32.768 a 32.767 |
| unsigned short int | 2 | 0 a 65.535 |
| int | 2 (no processador de 16 bits)<br>4 (no processador de 32 bits) | -32.768 a 32.767<br>-2.147.483.648 a 2.147.483.647 |
| unsigned int | 2 (no processador de 16 bits)<br>4 (no processador de 32 bits) | 0 a 65.535<br>0 a 4.294.967.295 |
| long int | 4 | -2.147.483.648 a 2.147.483.647 |
| unsigned long int | 4 | 0 a 4.294.967.295 |
| Float | 4 | $3.4*10^{-38}$ a $3.4*10^{38}$ |
| Double | 8 | $1.7*10^{-308}$ a $1.7*10^{308}$ |
| long double | 10 | $3.4*10^{-4932}$ à $3.4*10^{4932}$ |

Na tabela vista anteriormente, note que o tipo de dado determina a quantidade de *bytes* de memória alocada quando uma variável é declarada. Por exemplo, uma variável do tipo *short int* requer um espaço de 2 *bytes*. Um *byte* é a menor posição de memória que pode ser alocada e representa um conjunto de 8 *bits*. Como cada bit pode apresentar apenas valores binários, um *byte* pode armazenar $2^8$ números.

Já uma função na linguagem de programação C é um conjunto de instruções que agrupados em um bloco que podem ser executado quando a função é chamada. Além do bloco de instruções, uma função possui o seu identificador (ou nome), uma lista de argumentos ou parâmetros e o seu retorno. Os parâmetros de uma função têm o intuito de tornar o seu uso mais flexível. Por meio deles, valores são passados para que o bloco de instruções seja executado de uma forma apropriada. Já o retorno indica o resultado do processamento da função.

Uma função é declarada pelo programador e pode ser chamada por meio de uma instrução de evocação dela própria. A figura a seguir apresenta a definição de uma função que realiza a soma de dois valores passados como argumento.

**Função que realiza soma de dois valores**

```
int somaValores(int operador1, int operador2)
{
    int resultadoSoma;
    resultadoSoma = operador1 + operador2;
    return resultadoSoma;
}
```

Na figura acima, perceba que o identificador da função é 'somaValores'. Essa função apresenta como argumentos dois inteiros chamados de 'operador1' e 'operador2'. No bloco de instruções do exemplo, é realizada a soma desses dois argumentos, e o resultado é armazenado em uma variável interna denominada de 'resultadoSoma'. Por fim, o valor da variável interna é retornado pelo comando 'return'.

Agora, verifique na figura a seguir, a forma pela qual uma função pode ser chamada. A última instrução apresentada na figura mostra a função 'resultadoSoma' sendo evocada. Assim, as instruções exibidas na figura acima são executadas sequencialmente. Perceba que são passados como argumentos os valores atribuídos nas variáveis numero1 e numero2. Nesse caso, são repassadas à função os valores 5 e 3, respectivamente. Ao fim de sua execução, o resultado é retornado para a variável 'soma'. Maiores detalhes relacionados a como definir e utilizar funções serão dados posteriormente.

**Trecho de código que evoca função definida previamente**

```
int numero1 = 5;
int numero2 = 3;
int soma = somaValores(numero1,numero2);
```

## Programando o conhecimento

Quantos *bytes* são necessários para armazenar o valor de uma variável do tipo unsigned char?

a) 1.

b) 2.

c) 4.

d) 8.

**Comentário**: uma variável do tipo unsigned char armazena valores entre 0 e 255. Dessa forma, apenas 1 *byte* é alocado quando essa variável é declarada. Perceba que, quando um programa C é executado, todas as posições de memória relacionadas às variáveis do código-fonte são alocadas previamente. Em algumas situações em que a memória é escassa, a escolha dos tipos de dados para as variáveis deve ser feita de forma que a quantidade de *bytes* alocada seja a menor possível.

## 1.3 Meus primeiros programas em C

Após as definições iniciais relativas à linguagem C terem sido feitas, agora, você terá a oportunidade de analisar o seu primeiro programa. Empolgados? Para os nossos primeiros programas, adotaremos a abordagem de, inicialmente, exibir o código-fonte e, logo em seguida, explicar os pontos mais relevantes. Para esse fim, visualize a figura a seguir.

**Primeiro programa C**

```
1   #include <stdio.h>
2   #include <locale.h>
3
4   int main()
5   {
6       setlocale(LC_ALL, "Portuguese");
7
8       printl("Olá Mundo");
9
10      return 0;
11  }
```

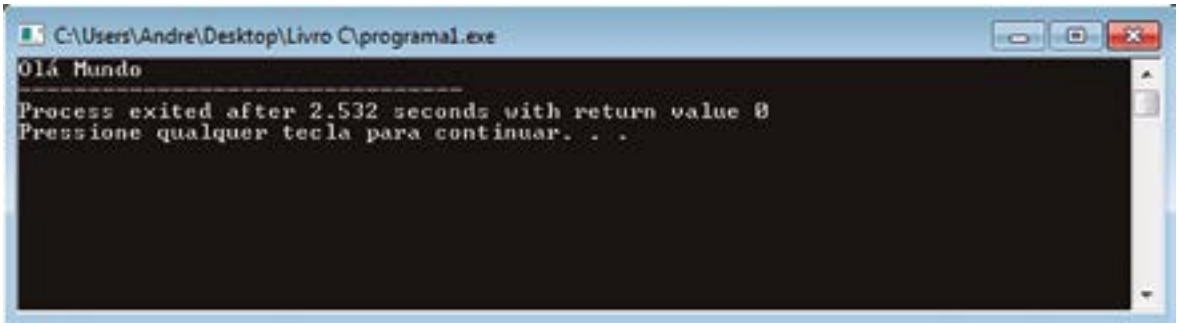
Antes de entrarmos nos detalhes do programa exibido pela figura acima, vale a pena ressaltar que, durante todas as lições, será empregado o *software* DevC++, o qual fornece uma plataforma para desenvolvimento de programas tanto em C como em C++. Embora códigos-fonte em C++ possam ser escritos nessa plataforma, apenas programas em C serão exibidos.

Agora, vamos analisar com detalhes todos os elementos da figura recém-observada. Primeiramente, observe que a linha 1 possui um diretiva para inclusão de uma biblioteca denominada de stdio.h. Uma biblioteca em C nada mais é do que um arquivo que possui diversas funções prontas para uso. No caso específico da biblioteca stdio.h, ela fornece diversas funcionalidades que permitem a entrada e a saída de dados. Por exemplo, existe a função printf (empregada na linha 8 do programa), que exibe, no console do programa, a *string* passada como parâmetro.

Já a linha 2 possui uma outra diretiva de inclusão de biblioteca padrão (locale.h). A linguagem de programação C, por padrão, não é configurada para a língua portuguesa e, por conta disso, não permite que acentos sejam exibidos como saída do programa. Para resolver esse problema, foi realizada a inclusão da biblioteca padrão locale.h (veja linha 2 da figura anterior). Com o uso dessa biblioteca, é possível o uso da função setlocale, a qual é chamada na linha 6 do programa em questão e fornece a informação ao compilador que a língua portuguesa deve se considerada e, consequentemente, acentos devem ser exibidos corretamente.

Outro ponto de extrema importância é a função main. Essa função é o ponto de início de execução de todo e qualquer programa C. Dessa forma, quando se coloca para rodar um código-fonte, o bloco de instruções presentes nessa função começa a ser executado. No nosso programa exibido na figura anterior, perceba que a função main é extremamente simples. Além da configuração realizada pela chamada à função setlocale, a única tarefa realizada é a exibição no console do programa a *string* "Olá Mundo", por meio da função padrão printf (verifique na figura a seguir como essa informação é exibida). Repare que, por definição, toda a função main retorna um inteiro. Assim, para que o compilador não reporte um erro, optou-se por retornar o valor zero.

**Informações exibidas pelo programa exibido na figura "Trecho de código que evoca função definida previamente".**




**Importante**

Para que um programa C rode, são envolvidas a compilação e a efetiva execução do código-fonte. Nesse processo, alguns erros podem ocorrer, sendo os mais comuns: compilação, linkagem e execução. Erros de compilação são gerados por erros sintáticos e, consequentemente, têm de ser corrigidos para que o programa seja compilado. Já os erros de linkagem são o resultado de problemas na inclusão de bibliotecas padrão. Na maior parte das vezes, ocorre quando uma biblioteca não está disponível para uso. Por fim, erros de execução relacionam-se com instruções de repetição, podendo resultar na execução indefinida de um programa ou, até mesmo, em problemas em operações aritméticas como a divisão por zero.

Terminamos de detalhar os principais pontos do programa apresentada na figura "Primeiro programa C". Que tal, agora, analisarmos um programa com um pouco mais de peculiaridades? Agora, verifique o código-fonte exibido na próxima figura.

**Programa C que realiza soma de dois números informados**

```
1   #include <stdio.h>
2   #include <locale.h>
3
4   int somaValores(int operador1, int operador2)
5   {
6       int resultadoSoma;
7
8       resultadoSoma = operador1, int operador2;
9
10      return resultadoSoma;
11  }
12
13  int main ()
14  {
15      setlocate(LC_ALL, "Portuguese");
16
17      int numero1;
18      int numero2;
19
20      printf("Digite o primeiro número");
21      scanf("%d",&numero1);
22      printf("Digite o segundo número");
23      scanf("%d",&numero2);
24
```

```
24
25      int = somaValores(numero1, numero2);
26      printf("A soma do primeiro número com o segundo número é igual a %d", soma);
27
28      return 0;
29  }
```

O programa exibido na figura acima realiza a soma de dois números informados pelo usuário do programa. A função scanf (da biblioteca padrão stdio.h), para a execução do programa, está a espera de um entrada de dado do usuário no console. Essa função é chamada duas vezes (linha 21 e linha 23) e os valores informados devem ser inteiros. Tais valores são armazenados nas variáveis 'numero1' e 'numero2' respectivamente.

**Saiba mais**

Observando o programa apresentado na figura acima, verifique que a função scanf foi chamada duas vezes nas linhas 21 e 23. Como já indicado, as variáveis informadas na função scanf recebem o valor digitado pelo usuário. Entretanto, perceba que o operador '&' é posicionado antes do nome do argumento. Esse operador indica que a posição de memória da variável é repassada para a função scanf. Ou seja, a função em questão, na verdade, recebe como argumento a posição de memória que deverá armazenar o valor digitado. Na linguagem C, o valor de posições de memória é chamado de ponteiros. Maiores detalhes sobre ponteiros serão disponibilizados no decorrer deste livro.

Os números digitados são repassados como argumento para a função 'somaValores', que, como já mencionado, realiza a operação de soma dos inteiros passados como parâmetros. Por fim, o resultado da função é retornado para a variável 'soma' e é, finalmente, exibido pela função printf na linha 26.

Perceba que a função printf recebe como argumento uma string. Note que existe dentro dessa sequência de caracteres uma *string* de conversão '%d'. Ela indica à referida função que o valor de uma variável inteira deve ser exibido no lugar dos caracteres '%d'. Assim, como se pode observar na linha 26, o valor da variável 'soma' foi também repassado como argumento na chamada da função printf. Esse valor deve ser exibido com representação decimal com sinal. Dessa forma, note que printf é uma função que pode apresentar uma quantidade variável de argumentos. Ou seja, para cada *string* de conversão, deve existir um argumento adicional. Observe na tabela a seguir quais são os caracteres que podem ser empregados na função printf para indicar diferentes formas de apresentação de dados.

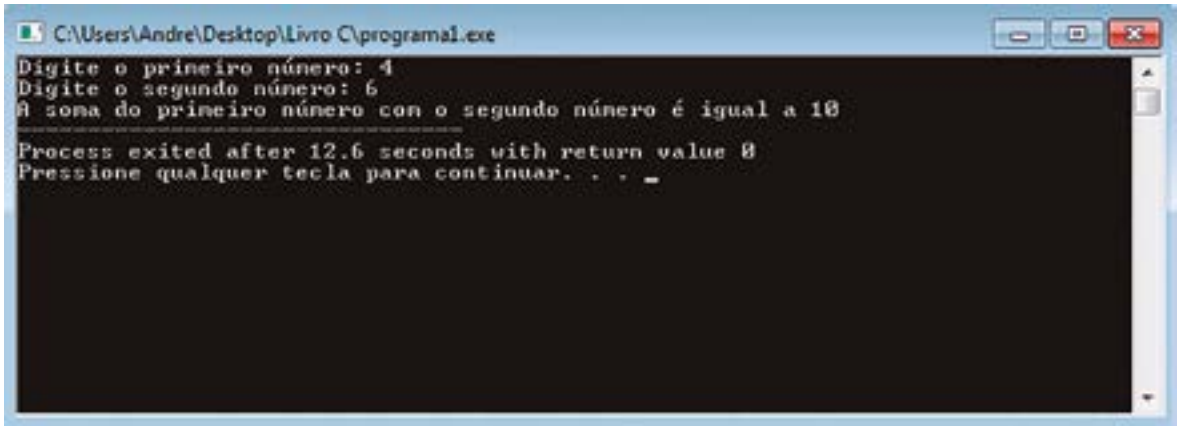
**Formas de exibição de dados pela função printf**

| ***String* de conversão** | **Forma de exibição do dado** |
|---|---|
| %c | Representação em ASCII |
| %d | Decimal com sinal |
| %u | Decimal sem sinal |
| %o | Octal sem sinal |
| %x | Hexadecimal sem sinal |

| *String* de conversão | Forma de exibição do dado |
|---|---|
| %f | Ponto flutuante com ponto decimal |
| %e | Ponto flutuante em notação científica |
| %g | Escolhe representação mais curta entre as indicadas por %f e %e |

Veja a figura abaixo, que apresenta qual é o resultado da execução do programa que está sendo analisado.

**Console da execução do programa exibido na figura da página 17**




Analisando a figura "Programa C que realiza soma de dois números informados", você pode observar que o usuário do programa informou os valores 4 e 6, sendo ambos atribuídos pela função scanf para as variáveis numero1 e numero2, respectivamente. Logo em seguida, a função 'somaValores' definida pelo programador foi chamada e retornou o valor 10 como resultado. Esse valor foi exibido na chamada da função printf na linha 26.

## Programando o conhecimento

Qual é o resultado impresso pela chamada à função printf exibida no trecho de código abaixo?

```
setlocale(LC_ALL, "Portuguese");

char arg1 = 'a';
int arg2 = -10;
int arg3 = 20;
double arg4 = 10.5;
double arg5 = 10.5;

printf("%c, %d, %x, %f, %e", arg1, arg2, arg3, arg4, arg5);
```

a) a, 10, 20, 10.5, 10.5.

b) a, -10, 14, 10.5, 10.5e+001.

c) a. -10, 20, 10.5e+001, 10.5.

d) a, -10, 14, 10.5, 10.5e+001.

**Comentário**: perceba que a função printf, além da *string* inicial, recebeu outros cinco argumentos (um para cada *string* de conversão). A *string* %c sinaliza que a variável deve ser exibida no formado ASCII, o que, de fato, ocorre em todas as alternativas. Já a *string* %d indica que a variável arg2 deve ser exibida no formato decimal com sinal. Já a *string* de conversão %x imprime o valor de arg3 no sistema hexadecimal. O número decimal 20 é representado nesse sistema distinto como 14. Na próxima lição, você poderá entender melhor os diversos tipos de sistema de representação numérica, como o hexadecimal. Por fim, as *strings* de conversão %f e %e sinalizam representação de número de ponto flutuante com ponto decimal e notação científica, respectivamente. Logo, a alternativa correta é a alternativa "d".

## Programando o conhecimento

Qual é o resultado que é impresso pela chamada à função printf exibida no trecho de código abaixo?

```
int letraA = 97;

printf("O código ASCII da letra %c é %d", letraA, letraA);
```

a) O código ASCII da letra 97 é 97.

b) O código ASCII da letra 97 é a.

c) O código ASCII da letra a é a.

d) O código ASCII da letra a é 97.

**Comentário**: cada caractere que pode ser exibido em tela possui um código numérico distinto. Esse tipo de codificação é denominado de ASCII. Por exemplo, o caractere 'a' pode também ser representado pelo número 97. Diante do exposto, observe que a variável 'letraA' é repassada à função printf duas vezes para ser exibida de maneiras distintas. A primeira forma de impressão desse valor é indicada (pela *string* de conversão %c) para ser a representação em ASCII. Já a segunda exibição da variável da 'letraA' considera que o valor deve ser impresso como número decimal. Dessa forma, a alternativa "d" é a correta. Veja na tabela a seguir alguns caracteres e respectivos códigos ASCII numéricos.

**Tabela ASCII**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | 65 | P | 80 | _ | 95 | n | 110 |
| B | 66 | Q | 81 | ` | 96 | o | 111 |
| C | 67 | R | 82 | a | 97 | p | 112 |
| D | 68 | S | 83 | b | 98 | q | 113 |
| E | 69 | T | 84 | c | 99 | r | 114 |
| F | 70 | U | 85 | d | 100 | s | 115 |
| G | 71 | V | 86 | e | 101 | t | 116 |
| H | 72 | W | 87 | f | 102 | u | 117 |
| I | 73 | X | 88 | g | 103 | v | 118 |
| J | 74 | Y | 89 | h | 104 | w | 119 |
| K | 75 | Z | 90 | i | 105 | x | 120 |
| L | 76 | [ | 91 | j | 106 | y | 121 |
| M | 77 | \ | 92 | k | 107 | z | 122 |
| N | 78 | ] | 93 | l | 108 | { | 123 |
| O | 79 | ^ | 94 | m | 109 | \| | 124 |

## Resumindo

Nesta lição foi apresentado um breve histórico sobre como surgiu a linguagem de programação C. Essa linguagem foi criada no início da década de 1970 e o seu pai foi o projetista Dennis Ritchie, da Bell Laboratories. C foi o resultado de um esforço para criar uma nova linguagem de programação para codificar o sistema operacional (SO) Unix. Os programas C substituíram os códigos-fonte escritos em linguagem de máquina, o que permitiu o aumento da portabilidade. Em seguida, foram especificadas as principais características de C. Em suma, a linguagem C é compilada, estruturada, imperativa e de propósito geral.

Os elementos básicos para o entendimento de como programas simples em C executam foram explicados. Assim, foram apresentados os conceitos de variáveis e funções. Uma variável nada mais é do que uma posição de memória que é identificada por meio de um identificador. Cada variável possui um tipo de dado, ou seja, quais são os valores que podem ser armazenados nessa porção de memória. Já uma função na linguagem de programação C é um conjunto de instruções que pode ser executado quando a função é chamada. Além do bloco de instruções, uma função possui o seu identificador (ou nome), uma lista de argumentos ou parâmetros e o seu retorno.

Por fim, os primeiros programas C foram exibidos. Inicialmente, foi apresentado um código-fonte

simples que apenas imprimia uma *string* no console do usuário, por meio da função printf. Em seguida, um programa que recebe dois números do usuário e realiza a soma deles foi mostrado.

Veja se você se sente apto a:

- entender as principais características da linguagem C;
- compreender o funcionamento dos principais elementos de C: variável e função;
- dominar o funcionamento dos primeiros programas C, em especial como as funções da biblioteca stdio.h funcionam.

Parabéns, você finalizou esta lição!

Agora responda às questões ao lado.

## Exercícios

**Questão 1 –** A linguagem de programa C foi concebida, inicialmente, para substituir o código-fonte escrito em linguagem de máquina de qual tipo de sistema?

a) Compiladores.

b) Sistemas operacionais.

c) Sistemas embarcados.

d) Sistemas de apoio a decisão.

**Questão 2 –** Com o advento de C, códigos-fonte puderam ser executados em diversas plataformas computacionais sem a necessidade de adaptações. Assinale a alternativa que nomeia esse tipo de vantagem da linguagem C.

a) Alto nível.

b) Imperatividade.

c) Portabilidade.

d) Executabilidade.

**Questão 3 –** Assinale a alternativa correta com relação à linguagem de programação Assembly.

a) Trata-se de uma linguagem de alto nível.

b) Um programa em C para ser executado precisa passar por um compilador que gerar um programa em Assembly.

c) As instruções dessa linguagem lidam com variáveis e não com registradores.

d) É uma linguagem altamente portável.

**Questão 4 –** Assinale a alternativa correta sobre a linguagem C.

a) É uma linguagem orientada a objetos e, por conta disso, é adequada para a construção de sistemas com mais de 20 mil linhas de código.

b) Não é uma linguagem orientada a objetos, mas apresenta conceitos de modularização, o que permite a construção de sistemas complexos sem incorrer no aumento de custos de manutenção.

c) Não é uma linguagem orientada a objetos e, por conta disso, não é recomendável à construção de sistemas com mais de 20 mil linhas de código. O custo de manutenção aumenta consideravelmente em sistemas desse tamanho.

d) É uma linguagem orientada a objetos e, por conta disso, não é adequada para a construção de sistemas com mais de 20 mil linhas de código.

**Questão 5 –** Assinale a alternativa que traz as características principais da linguagem C.

a) Compilada, funcional, estruturada.

b) Compilada, funcional, de propósito específico.

c) Interpretada, imperativa, de propósito geral.

d) Compilada, imperativa, estruturada.

**Questão 6 –** Marque a alternativa que traz a distinção entre linguagens compiladas e interpretadas.

a) Linguagens compiladas necessitam que seus programas sejam compilados para a geração de um código objeto de baixo nível. Por outro lado, linguagens interpretadas não necessitam desse procedimento.

b) O código-fonte de uma linguagem compilada executada diretamente sem a necessidade de geração de um código objeto de baixo nível. Por outro lado, linguagens interpretadas precisam passar por um processo de interpretação em que um código-objeto de baixo nível é gerado.

c) Em linguagens compiladas, erros de sintaxe são reportados durante a execução do programa. O inverso acontece em linguagens interpretadas.

d) Em sua maioria, linguagens compiladas executam de forma mais eficiente que linguagens interpretadas.

**Questão 7 –** Assinale a alternativa que indica a definição de uma variável em C.

a) É uma sequência de caracteres que pode ser manipulada por um programa.

b) É um conjunto de instruções que pode ser evocado por uma outra instrução.

c) É uma porção de memória que é alocada por meio de uma instrução de declaração de variável. Cada variável possui um tipo de dado que define quantos *bytes* serão alocados.

d) É uma porção de memória de 2 *bytes* que é alocada por meio de uma instrução de declaração de variável.

**Questão 8 –** Marque a instrução em C que realiza a declaração de uma variável.

a) int a;.

b) a = 10;.

c) int [] a;.

d) int a:.

**Questão 9 –** Assinale a alternativa que indica a quantidade correta de *bytes* alocados por uma variável do tipo char.

a) 1.

b) 2.

c) 3.

d) 4.

**Questão 10 –** Assinale a alternativa que traz o tipo de dado que pode representar número de 0 a 255.

a) int.

b) unsigned int.

c) char.

d) unsigned char.