

# C++ Para Leigos

Folha  
de Cola

Entender e executar programação em C++, que é o padrão para linguagens orientadas a objetos, é mais fácil quando você conhece as expressões, as declarações e os operadores para efetuar cálculos.

## Expressões e Declarações em Programação C++

Para efetuar um cálculo no programa C++, você precisa de uma expressão. Uma expressão é uma instrução que possui um valor e um tipo. No programa C++, uma declaração é uma instrução que define uma variável ou é o "tanque de segurança" para algum tipo de valor, como um número ou caractere.

### Expressões

As expressões tomam uma das seguintes formas:

objName	// para um objeto simples
operator expression	// para operadores unários
expr1 operator expr2	// para operadores binários
expr1 ? expr2 : expr3	// para o operador ternário
funcName([argument list]);	// para chamadas de função

### Expressões Literais

Uma literal é uma forma de expressão constante. Os vários tipos de literais são definidos na seguinte tabela literal.

Exemplo	Tipo
1	int
1L	long int
1LL	long long int
1.0	double
1.0F	float
'1'	char
"a string"	char* (automaticamente terminado por um caractere nulo)
L "a string"	wchar_t*
u8 "this is a UTF-8 string with a UTF-8 character: \u2018"	char8_t*
u "this is a UTF-16 string with a UTF-16 character: \u2018"	char16_t*
U "this is a UTF-32 string with a UTF-32 character: \U00002018"	char32_t*
true, false	bool
0b101	binary (padrão C++ 2014)

**Para Leigos: A série de livros para iniciantes que mais vende no mundo.**

## Declarações

Declarações são tipos intrínsecos e definidos pelo usuário. Os tipos intrínsecos são:

```
[<signed | unsigned >]char  
[<signed | unsigned >]wchar_t  
[<signed | unsigned>] [<short | long | long long>] int  
float  
[long] double  
bool
```

As declarações têm um destes formatos:

```
[<extern|static>][const] type var[=expression];           // variável  
[<extern|static>][const] type array[size][={list}];       // array  
[const] type object[(argument list)];                    // objeto  
[const] type object [= {argument list}];                 // alternativo  
[const] type * [const] ptr[=pointer expression];         // ponteiro  
type& refName = object;                                  // referência  
type fnName([argument list]);                             // função
```

A palavra-chave `auto` pode ser usada se C++ determinar o tipo de variável sozinho.

```
auto var = 1L;                                           // o tipo de var é long int
```

A palavra-chave `decltype` extrai o tipo de uma expressão. Esse tipo pode ser usado quando um nome de tipo for usado. Por exemplo, o seguinte exemplo usa `decltype` para declarar a segunda variável com o mesmo tipo de uma variável existente.

```
decltype(var1) var2; // o tipo de var2 é o mesmo de var1
```

Uma definição de função tem o seguinte formato:

```
// função simples  
[<inline|constexpr>] type fnName(argument list) {...}  
// função de membro definida fora da classe  
[inline] type Class::func(argument list) [const] {...}  
// construtores/destrutores também podem ser definidos fora da classe  
Class::Class([argument list]) {...}  
Class::~~Class() {...}  
//construtores/destrutores podem ser excluídos ou padronizados  
// no lugar de definição  
Class::Class([argument list]) = <delete|default>;  
Class::~~Class() = <delete|default>;
```



**por Stephen R. Davis**



ALTA BOOKS  
E D I T O R A  
Rio de Janeiro, 2016

## *Sobre o autor*

**Stephen R. Davis, CISSP** (também conhecido como “Randy”) vive com sua esposa e dois cachorros em Corpus Christi, Texas. Randy tem três filhos e dois netos, com mais um a caminho (netos, não filhos). Randy desenvolve aplicações baseadas em navegadores para a Agency Consulting Group.

## *Dedicatória*

Para Janet, o amor da minha vida.

## *Reconhecimentos do Autor*

Eu acho muito estranho que apenas um nome apareça na capa do livro, especialmente um livro como este. Na verdade, muitas pessoas contribuem para a criação de um livro *Para Leigos*. Desde o início, o editor de aquisições Constance Santisteban, o editor de projeto Pat O’Brien e minha agente, Claudette More, estiveram envolvidos em guiar e modelar o conteúdo do livro. Durante o desenvolvimento das sete edições deste livro, eu me vi envolvido em edições, correções e sugestões de um grupo de editores, revisores e revisores técnicos — este livro teria sido um trabalho pobre se não fosse pelo envolvimento deles. E nada teria feito ele ser impresso sem a ajuda de Suzanne Thomas, que coordenou a primeira e a segunda edição do livro, Susan Pink, que trabalhou na terceira e na sexta edição, Katie Feltman que trabalhou na sexta edição e Danny Kalev, que fez a revisão técnica da sexta e da sétima edição. Contudo, um nome aparece na capa e aquele nome deve assumir responsabilidade por imprecisões no texto.

Finalmente, um resumo da atividade animal na minha casa. Para aqueles que não leram meus outros livros, eu deveria avisá-los que isso se tornou uma característica frequente em meus livros *Para Leigos*.

Eu me mudei para a “cidade grande” em 2005, o que significou ter que dar meus cachorros Chester e Sadie. Eu tentei manter nossos dois grandes Dogues, Monty e Bonnie, mas eles eram demais para o quintal. Nós fomos obrigados a dá-los também. Eu me casei com minha namorada da escola em 2011 e me mudei de Dallas para minha cidade natal, Corpus Christi, o que me levou a adotar um novo par de cachorros (ou melhor, eles me adotaram). Jack é um cachorro preto e teimoso de raça indefinida. Dizem que Scruffy era um dachshund muito peludo mas você não consegue notar pela sua aparência já que ele fica tosado a maior parte do tempo.

# Sumário Resumido

<b><i>Introdução .....</i></b>	<b><i>1</i></b>
<b><i>Parte I: Começando com Programação em C++ .....</i></b>	<b><i>7</i></b>
Capítulo 1: Escrevendo Seu Primeiro Programa em C++.....	9
Capítulo 2: Declarando Variáveis Constantemente .....	33
Capítulo 3: Efetuando Operações Matemáticas .....	49
Capítulo 4: Efetuando Operações Lógicas .....	57
Capítulo 5: Controlando o Fluxo do Programa.....	71
<b><i>Parte II: Como se Tornar um Programador Funcional.....</i></b>	<b><i>89</i></b>
Capítulo 6: Criando Funções .....	91
Capítulo 7: Armazenando Sequências em Arrays .....	109
Capítulo 8: Uma Primeira Olhada nos Ponteiros em C++.....	127
Capítulo 9: Uma Segunda Olhada nos Ponteiros em C++ .....	143
Capítulo 10: O Pré-processador C++ .....	161
<b><i>Parte III: Introdução às Classes .....</i></b>	<b><i>175</i></b>
Capítulo 11: Examinando a Programação Orientada a Objeto .....	177
Capítulo 12: Adicionando Classes em C++.....	183
Capítulo 13: Aponte e Encare os Objetos.....	199
Capítulo 14: Protegendo Membros: Não Perturbe.....	217
Capítulo 15: Amor, Por Que Você me Constrói, Apenas Para me Destruir?.....	225
Capítulo 16: Fazendo Argumentos Construtivos.....	237
Capítulo 17: Os Construtores de Cópia e de Movimento.....	259
Capítulo 18: Membros Estáticos: Será que Amaciante de Roupas Pode Ajudar? .....	273
<b><i>Parte IV: Herança .....</i></b>	<b><i>283</i></b>
Capítulo 19: Herdando uma Classe.....	285
Capítulo 20: Examinando Funções de Membro Virtuais: Elas São Reais?.....	295
Capítulo 21: Fatorando Classes.....	305
<b><i>Parte V: Segurança .....</i></b>	<b><i>315</i></b>
Capítulo 22: Um Novo Operador de Atribuição, Caso Você Decida Aceitá-lo .....	317

Capítulo 23: Usando o Fluxo de I/O .....	329
Capítulo 24: Tratamento de Erros — Exceções .....	351
Capítulo 25: Herdando Heranças Múltiplas .....	361
Capítulo 26: Templates Tentadores em C++.....	373
Capítulo 27: Padronizando na Biblioteca Padrão de Gabaritos (STL) .....	383
Capítulo 28: Escrevendo Código a Prova de Hackers.....	395
<b><i>Parte VI: A Parte dos Dez</i> .....</b>	<b>421</b>
Capítulo 29: 10 Maneiras de Evitar Adicionar Bugs ao Seu Programa.....	423
Capítulo 30: 10 Maneiras de Proteger Seus Programas dos Hackers.....	431
<b><i>Índice</i> .....</b>	<b>447</b>

# Sumário



<b>Introdução .....</b>	<b>1</b>
Sobre Este Livro .....	1
Ícones Usados Neste Livro .....	4
Além do Livro .....	4
De Lá Para Cá, Daqui Para Lá .....	5
<b>Parte 1: Começando com Programação em C++ .....</b>	<b>7</b>
<b>Capítulo 1: Escrevendo Seu Primeiro Programa em C++ .....</b>	<b>9</b>
Dominando Conceitos de C++ .....	10
Instalando o Code::Blocks .....	11
Windows .....	12
Linux Ubuntu .....	14
Macintosh .....	16
Criando Seu Primeiro Programa em C++ .....	20
Criando um projeto .....	20
Inserindo código em C++ .....	22
Trapaceando .....	23
Construindo seu programa .....	25
Executando Seu Programa .....	26
Analisando o Programa Escrito .....	27
Examinando a estrutura de todos os programas C++ .....	27
Explicando o código-fonte com comentários .....	28
Baseando os programas em instruções C++ .....	29
Escrevendo declarações .....	29
Produzindo uma saída .....	30
Calculando Expressões .....	30
Armazenando os resultados de uma expressão .....	31
Examinando o restante do programa Conversion .....	31
<b>Capítulo 2: Declarando Variáveis Constantemente .....</b>	<b>33</b>
Declarando Variáveis .....	34
Declarando Tipos Diferentes de Variáveis .....	34
Analisando as limitações dos inteiros em C++ .....	35
Resolvendo o problema do arredondamento .....	36
Olhando para os limites dos números de ponto flutuante .....	37
Declarando Tipos de Variáveis .....	39
Tipos de constantes .....	40
A Variação dos Tipos Numéricos .....	41
Caracteres especiais .....	42
Grandes Carregamentos na Estrada do Char .....	43
Estes Cálculos São Lógicos Mesmo? .....	44



Expressões em Modo Misto .....	45
Declarações Automáticas .....	46
<b>Capítulo 3: Efetuando Operações Matemáticas .....</b>	<b>49</b>
Efetuando Aritmética Binária Simples .....	50
Decompondo as Expressões .....	51
Determinando a Ordem das Operações .....	51
Efetuando Operações Unárias .....	52
Utilizando os Operadores de Atribuição .....	54
<b>Capítulo 4: Efetuando Operações Lógicas .....</b>	<b>57</b>
Por Que Mexer com Operações Lógicas? .....	58
Usando os Operadores Lógicos Simples .....	58
Armazenando valores lógicos .....	59
Usando variáveis lógicas int .....	61
Tome cuidado ao efetuar operações lógicas em variáveis de ponto flutuante .....	61
Expressando Números Binários .....	63
O sistema numérico decimal .....	63
Outros sistemas numéricos .....	64
O sistema numérico binário .....	64
Efetuando Operações Lógicas Bitwise .....	66
Os operadores bit a bit .....	67
Usando os operadores bitwise .....	68
Um teste simples .....	69
<b>Capítulo 5: Controlando o Fluxo do Programa .....</b>	<b>71</b>
Controlando o Fluxo de Programa com os Comandos de Seleção (Branch) .....	72
Executando Loops em um Programa .....	74
Fazendo o loop enquanto a condição for verdadeira .....	74
Usando o recurso autoincremento/autodecremento .....	76
Usando o loop for .....	77
Evitando o temido loop infinito .....	80
Cada um com seu for .....	81
Aplicando controles de loop especiais .....	82
Comandos de Controle Aninhados .....	85
Mudando para um Assunto Diferente? .....	86
<b><i>Parte II: Como se Tornar um Programador Funcional.....</i></b>	<b>89</b>
<b>Capítulo 6: Criando Funções .....</b>	<b>91</b>
Escrevendo e Usando uma Função .....	92
Definindo nossa primeira função .....	94
Definindo a função sumSequence() .....	94
Chamando a função sumSequence() .....	95
Dividir e conquistar .....	95
Entendendo os Detalhes das Funções .....	95
Entendendo funções simples .....	96



Entendendo funções com argumentos .....	97
Sobrecarregando Nomes de Funções .....	100
Definindo Protótipos de Função .....	102
Estabelecendo Argumentos .....	103
Passagem por Valor e Passagem por Referência .....	105
Tipos de Armazenamento da Variável .....	106
<b>Capítulo 7: Armazenando Sequências em Arrays .....</b>	<b>109</b>
Formando os Argumentos para os Arrays .....	109
Usando um array .....	111
Inicializando um array .....	114
Indo longe demais no array .....	115
Formando loops for baseados em intervalo .....	116
Definindo e usando arrays de arrays .....	116
Usando Arrays de Caracteres .....	117
Criando um array de caracteres .....	117
Criando uma string de caracteres .....	118
Manipulando Strings com Caractere .....	120
Adicionando Funções de Biblioteca .....	122
Abrindo Espaço para Strings Grandes .....	124
<b>Capítulo 8: Uma Primeira Olhada nos Ponteiros em C++ .....</b>	<b>127</b>
Tamanho da Variável .....	127
O Que É um Endereço? .....	129
Operadores de Endereço .....	129
Usando Variáveis Ponteiro .....	131
Usando tipos diferentes de ponteiros .....	132
Passando Ponteiros para as Funções .....	133
Passagem por valor .....	133
Passagem por ponteiro .....	134
Passagem por referência .....	134
A Constante Irritação de const .....	135
Utilizando o bloco de memória chamado Heap .....	137
Escopo limitado .....	137
Examinando o problema de escopo .....	139
Gerando uma solução usando o heap .....	139
<b>Capítulo 9: Uma Segunda Olhada nos Ponteiros em C++ .....</b>	<b>143</b>
Definindo Operações em	
Variáveis Ponteiro .....	143
Reexaminando arrays do ponto de vista das variáveis ponteiro .....	144
Aplicando operadores ao endereço de um array .....	146
Expandindo operações de ponteiro para uma string .....	147
Justificando a manipulação de string baseada em ponteiro .....	149
Aplicando operadores aos tipos de ponteiro além de char .....	150
E Quando Não É um Ponteiro? .....	152
Declarando e Usando Arrays de Ponteiros .....	153
Utilizando arrays de strings de caractere .....	154
Acessando os argumentos para main() .....	156

**Capítulo 10: O Pré-processador C++ ..... 161**

O Que É um Pré-processador? .....	161
Incluindo Arquivos.....	162
#Definindo Coisas.....	165
Ok, que tal não #definir coisas? .....	168
Enumerando outras opções .....	169
Incluindo Itens #Se Eu Disser Sim .....	171
Objetos Definidos Intrinsecamente .....	172
Typedef .....	174

**Parte III: Introdução às Classes ..... 175****Capítulo 11: Examinando a Programação Orientada a Objeto .... 177**

Abstraindo o Forno de Micro-ondas.....	177
Preparando nachos funcionais .....	178
Preparando nachos orientados a objeto .....	179
Classificando os Fornos de Micro-ondas .....	179
Por Que Classificar?.....	180

**Capítulo 12: Adicionando Classes em C++..... 183**

Apresentando a Classe .....	183
O Formato de uma Classe .....	184
Acessando os Membros de uma Classe .....	185
Ativando Nossos Objetos .....	185
Simulando objetos do mundo real .....	186
Por que se preocupar com	
funções membro? .....	186
Adicionando uma Função membro.....	187
Chamando uma Função Membro .....	188
Acessando outros membros a partir de uma função membro.....	190
Resolução do Escopo (E Eu Não Quero Saber se Seu	
Telescópio Funciona Bem) .....	191
Definindo uma Função	
Membro na Classe .....	193
Mantendo uma Função Membro depois da Classe .....	195
Sobrecarregando Funções Membro .....	196

**Capítulo 13: Aponte e Encare os Objetos..... 199**

Declarando Arrays de Objetos .....	199
Declarando Ponteiros para Objetos.....	200
Derreferenciando um ponteiro objeto .....	201
Apontando com o operador seta.....	202
Passando Objetos para as Funções .....	202
Chamando uma função com um valor objeto .....	203
Chamando uma função com um ponteiro para objeto .....	204
Chamando uma função com o uso do operador de referência.....	206
Por Que se Incomodar com Ponteiros ou Referências? .....	207
Voltando para o Heap .....	207
Alocando Arrays de Objetos no heap .....	208

Quando a memória está alocada para você.....	209
Conexão com Listas Ligadas .....	209
Efetuando outras operações em uma lista ligada.....	211
Conectando com um programa LinkedListData .....	211
Um Fio de Esperança: Uma Lista de Contêineres	
Ligados à Biblioteca C++ .....	215
<b>Capítulo 14: Protegendo Membros: Não Perturbe .....</b>	<b>217</b>
Protegendo Membros.....	217
Por que você precisa de membros protegidos.....	218
Descobrimos como os membros protegidos funcionam .....	218
Criando um Argumento para Uso com Membros Protegidos .....	220
Protegendo o estado interno da classe .....	220
Usando uma classe com uma interface limitada.....	221
Dando a Funções Não-Membros Acesso aos Membros Protegidos.....	221
<b>Capítulo 15: Amor, Por Que Você me Constrói,</b>	
<b>Apenas Para me Destruir? .....</b>	<b>225</b>
Criando Objetos .....	225
Usando Construtores .....	226
Construindo um objeto único .....	227
Construindo objetos múltiplos .....	228
Construindo um duplex .....	229
Dissecando um Destrutor .....	231
Por que você precisa do destrutor .....	231
Trabalhando com destrutores .....	232
<b>Capítulo 16: Fazendo Argumentos Construtivos .....</b>	<b>237</b>
Equipando os Construtores com Argumentos .....	238
Usando um construtor .....	238
Exigindo Muito do Ajudante: Sobrecarregando o Construtor .....	240
Estabelecendo Construtores Padrões.....	243
Construindo Membros de Classe .....	245
Construindo um membro de dados complexo.....	245
Construindo um membro de dados constante.....	251
Reconstruindo a Ordem de Construção .....	251
Objetos locais construídos em ordem.....	252
Objetos estáticos construídos apenas uma vez.....	252
Todos os objetos globais construídos antes de main().....	253
Os objetos globais não são construídos em uma ordem fixa .....	254
Membros construídos na ordem em que são declarados.....	255
Destrutores destroem na ordem inversa dos construtores.....	256
Construindo Arrays.....	256
Construtores como uma Forma de Conversão .....	257
<b>Capítulo 17: Os Construtores de Cópia e de Movimento.....</b>	<b>259</b>
Copiando um Objeto.....	259
Por que você precisa do construtor de cópia .....	260
Usando o construtor de cópia.....	260

O Construtor de Cópia Automático .....	262
Criando Cópias Superficiais versus Cópias Profundas .....	263
Um Longo Caminho até os Temporários .....	267
Evitando temporários, permanentemente .....	268
O construtor de movimento .....	269
<b>Capítulo 18: Membros Estáticos: Será que</b>	
<b>Amaciante de Roupa Pode Ajudar? .....</b>	<b>273</b>
Definindo Um Membro Estático .....	273
Por que você precisa de membros estáticos .....	274
Usando membros estáticos.....	274
Referindo-se aos membros de dados estáticos .....	275
Usos para membros de dados estáticos .....	276
Declarando Funções	
Membro Estáticas .....	277
Mas Sobre o Que É isso Afinal? .....	280
<b>Parte IV: Herança .....</b>	<b>283</b>
<b>Capítulo 19: Herdando uma Classe .....</b>	<b>285</b>
Eu Preciso da Minha Herança? .....	286
Como uma Classe Herda? .....	287
Usando uma subclasse .....	289
Construindo uma subclasse.....	290
Destruindo uma subclasse .....	291
Herdando construtores .....	291
Tendo um Relacionamento TEM_UM.....	292
<b>Capítulo 20: Examinando Funções de Membro</b>	
<b>Virtuais: Elas São Reais? .....</b>	<b>295</b>
Por Que Você Precisa de Polimorfismo .....	298
Como o Polimorfismo Funciona .....	299
Quando Não É uma Função Virtual? .....	300
Considerando Observações Virtuais .....	302
<b>Capítulo 21: Fatorando Classes .....</b>	<b>305</b>
Fatorando .....	305
Implementando Classes Abstratas.....	309
Descrevendo o conceito de classe abstrata .....	310
Criando uma classe honesta a partir de uma classe abstrata .....	312
Passando classes abstratas.....	312
<b>Parte V: Segurança .....</b>	<b>315</b>
<b>Capítulo 22: Um Novo Operador de Atribuição,</b>	
<b>Caso Você Decida Aceitá-lo .....</b>	<b>317</b>
Comparando Operadores com Funções .....	317
Inserindo um Novo Operador .....	318

Criar Cópias Superficiais É um Grande Problema .....	319
Sobrecarregando o Operador de Atribuição .....	320
Sobrecarregando o Operador de Índice .....	325
O Construtor e o Operador de Movimento .....	326
<b>Capítulo 23: Usando o Fluxo de I/O .....</b>	<b>329</b>
Como Funciona o Fluxo de I/O .....	329
Objetos de fluxo padrão.....	330
Fluxo de Entrada/Saída .....	331
Modos de abertura.....	333
Ei, arquivo, em qual posição você está? .....	334
Você pode me mostrar um exemplo? .....	334
Outros Métodos das Classes de Fluxo .....	337
Lendo e escrevendo fluxos diretamente.....	339
Controlando formatos .....	341
Qual o problema com endl? .....	343
Posicionando o ponteiro dentro de um arquivo.....	343
Usando as Subclasses stringstream.....	344
Manipulando os Manipuladores .....	347
<b>Capítulo 24: Tratamento de Erros — Exceções .....</b>	<b>351</b>
Justificando um Mecanismo de Erro Novo? .....	353
Examinando o Mecanismo de Exceção .....	354
Que Tipos de Coisas Eu Posso Lançar? .....	357
Apenas Passando.....	359
<b>Capítulo 25: Herdando Heranças Múltiplas .....</b>	<b>361</b>
Descrevendo o Mecanismo de Herança Múltipla .....	361
Acertando as Ambiguidades de Herança.....	363
Adicionando Herança Virtual.....	364
Construindo Objetos de Herança Múltipla .....	370
Expressando uma Opinião Contrária .....	371
<b>Capítulo 26: Templates Tentadores em C++.....</b>	<b>373</b>
Generalizando uma Função em um Gabarito(Template).....	374
Modelos de Classe .....	376
Dicas para Usar Gabaritos .....	379
Instanciações de Modelo Externo .....	380
Implementando uma Lista Inicializadora.....	380
<b>Capítulo 27: Padronizando na Biblioteca</b>	
<b>    Padrão de Gabaritos (STL).....</b>	<b>383</b>
O Contêiner string.....	384
Iterando as Listas .....	389
Abrindo caminho por uma lista .....	390
Operações em uma lista inteira .....	392
Você pode me mostrar um exemplo? .....	392

<b>Capítulo 28: Escrevendo Código a Prova de Hackers .....</b>	<b>395</b>
Entendendo os Motivos dos Hackers .....	395
Entendendo Injeção de Código .....	398
Examinando um exemplo de injeção de SQL.....	398
Evitando injeção de código.....	400
Estourando Buffers por Diversão e Lucro .....	401
Posso ver um exemplo?.....	401
Como uma chamada empilha? .....	404
Hackeando BufferOverflow .....	408
Evitando estouro de buffer — primeira tentativa.....	412
Evitando estouro de buffer — segunda tentativa.....	413
Outro argumento para a classe string.....	416
Por que não usar funções string sempre? .....	417
<b>Parte VI: A Parte dos Dez .....</b>	<b>421</b>
<b>Capítulo 29: 10 Maneiras de Evitar Adicionar</b>	
<b>Bugs ao Seu Programa.....</b>	<b>423</b>
Habilitando Todos os Avisos e Mensagens de Erro .....	423
Adote um Estilo de Código Limpo e Consistente .....	424
Limite a Visibilidade.....	425
Comente Seu Código Enquanto Você o Escreve.....	426
Verifique Cada Passo Pelo Menos Uma Vez.....	427
Evite Operadores Sobrecarregados .....	427
Controle o Heap Sistemáticamente.....	427
Use Exceções para Tratar Erros .....	428
Declare Destrutores Virtuais.....	428
Evite Herança Múltipla.....	430
<b>Capítulo 30: 10 Maneiras de Proteger Seus</b>	
<b>Programas dos Hackers.....</b>	<b>431</b>
Não Tire Conclusões sobre a Entrada de Usuário .....	432
Tratando Falhas com Elegância .....	433
Mantendo um Log do Programa .....	434
Siga um Bom Processo de Desenvolvimento .....	435
Implemente um Bom Controle de Versão.....	436
Autentique os Usuários com Segurança .....	437
Gerencie Sessões Remotas .....	440
Complique Seu Código .....	441
Assine Seu Código com um Certificado Digital .....	444
Use Criptografia Sempre Que Necessário .....	445
<b>Índice .....</b>	<b>447</b>

# Introdução

---

**B**em-vindos ao *C++ Para Leigos, Tradução da 7ª Edição*. Pense neste livro como “Edição Essencial do Leitor”, trazendo até você tudo que você precisa saber sobre começar a programar sem aquele papo chato.

## Sobre Este Livro

*C++ Para Leigos* é uma introdução a linguagem C++. Eu começo do início (por onde mais?) e apresento dos conceitos novos até técnicas mais sofisticadas. Não suponho que você possua nenhum conhecimento prévio (pelo menos, não em programação).

Este livro é cheio de exemplos. Cada conceito é documentado em inúmeros trechos e diversos programas completos.

Diferente dos outros livros, *C++ Para Leigos* considera o “porquê” tão importante quanto o “como”. Os recursos de C++ são como uma peça de quebra-cabeça. Em vez de apenas apresentar os recursos, eu acho importante que você entenda como eles se encaixam. Você também pode usar o livro como uma referência: se quiser entender mais sobre Templates (Gabaritos), pule para o Capítulo 26. Cada capítulo contém referências necessárias para os capítulos anteriores caso você não leia os capítulos em sequência.

*C++ Para Leigos* não considera um sistema operacional específico. É tão prático para programadores Macintosh ou Linux quanto para desenvolvedores Windows. O livro não aborda nem programação Windows nem .NET.

Você tem que dominar uma poderosa linguagem de programação, como C++, mesmo que seu primeiro plano seja se tornar um programador da aplicação Windows ou .NET. Quando você terminar *C++ Para Leigos*, estará apto a continuar na sua área de especialização, qualquer que seja.

Em uma época tão moderna cheia de hackers, aprender programação defensiva é importante, mesmo para iniciantes, então eu abordo conceitos importantes para prevenir que seu programa seja hackeado.

## O que é C++?

C++ é uma linguagem padrão de baixo nível e orientada a objeto. Sendo uma linguagem de baixo nível parecida e compatível com seu antecessor C, C++ pode gerar programas muito rápidos e eficientes. Geralmente é usada para criação de jogos, software de gráficos, controle de hardware e outras aplicações em que o desempenho realmente conta.

Como uma linguagem orientada a objeto, C++ tem o poder e a agilidade de escrever programas em grande escala. C++ é uma das linguagens de programação mais populares para todos os tipos de programas. A maioria dos programas que você usa em seu computador todos os dias são escritos em C++ (ou seu subconjunto, a linguagem C).

C++ foi certificada como padrão 99,9% puro, o que faz ela ser uma linguagem portátil. Um compilador padrão C++ existe na maioria dos sistemas operacionais. Algumas versões suportam extensões da linguagem básica — principalmente, Visual Studio e Visual Studio Express da Microsoft incluem um compilador C++ que implementa várias extensões que permite que seus programas interajam melhor com outras linguagens .NET. Apesar disso, seria melhor se os alunos aprendessem o padrão C++ primeiro. Aprender as extensões é fácil, uma vez que você domine o básico demonstrado aqui.

Quando eu descrevo uma mensagem que você vê na tela, ela aparece desta forma:

```
Hi mom!
```

Além disso, a lista de códigos aparece assim:

```
// algum programa
int main()
{
    ...
}
```

Se você está entrando nesses programas manualmente, você precisa inserir o texto exatamente como é mostrado, com uma exceção: a quantidade de *espaços em branco* (espaços, tabulações, e linhas novas) não é relevante. Você não pode colocar um espaço no meio de uma palavra-chave, mas você não precisa se preocupar em inserir espaços a mais ou a menos.



Porém, diferenciar maiúsculas e minúsculas É importantíssimo. Se ele diz `int`, ele não está dizendo `Int` ou `INT`!



As palavras reservadas do C++ normalmente são baseadas em palavras inglesas com sentidos parecidos. Isso pode dificultar a leitura de uma frase contendo inglês e C++, caso não haja um pouco de assistência. Para dar uma mãozinha, os nomes dos comandos e funções em C++ aparecem em fonte diferente, deste jeito. Além disso, os nomes das funções são sempre seguidos de parênteses aberto e fechado, como `myFavoriteFunction()`. Os argumentos para a função só são mencionados quando há uma necessidade específica de facilitar a leitura.

Às vezes, irei lhe pedir pra usar comandos de menu, como File ⇄ Open. Essa instrução quer dizer para usar o teclado ou mouse para abrir o menu File e escolher a opção Open.

Cada recurso novo é introduzido como resposta para estas três perguntas:

- ✓ *O que é este recurso novo?*
- ✓ *Por que ele foi introduzido na linguagem?*
- ✓ *Como ele funciona?*

Pedaços pequenos de código estão espalhados pelos capítulos. Cada um demonstra um recurso recém-introduzido ou destaca algum ponto importante. Esses trechos podem não estar completos e certamente não fazem nada de relevante. No entanto, cada conceito é demonstrado em, pelo menos, um programa funcional para que você pode executar e brincar no seu próprio computador.

Um programa do mundo real pode tomar muitas páginas. Porém, ver tal programa é uma ferramenta didaticamente importante para qualquer leitor. Eu incluí uma série de programas junto com uma explicação de como eles funcionam.

Eu uso um programa simples que chamo de BUDGET. O programa começa a vida como um simples e orientado funcionalmente BUDGET1. Esse programa mantém um conjunto de contas bancárias simples, corrente e poupança. O leitor é motivado a revisar esse programa no final da Parte II. A versão seguinte, BUDGET2, adiciona os conceitos de orientação a objeto apresentados na Parte III. Os exemplos vão evoluindo, usando mais e mais recursos da linguagem, até terminar em BUDGET5, que você deve revisar após dominar todos os capítulos no livro. Os programas BUDGET estão incluídos com o código-fonte do livro disponível para download em [www.altabooks.com.br](http://www.altabooks.com.br), procurando pelo título do livro.

## Ícones Usados Neste Livro



Área técnica que você pode pular em uma primeira leitura.



As dicas destacam um ponto que pode poupar muito tempo e esforço.



Lembre-se disso. É importante.



Lembre-se disso, também. Ele pode espionar quando você menos esperar e gerar um daqueles bugs difíceis de encontrar.



Esse ícone sinaliza algumas adições de 2011 na linguagem comparada ao padrão anterior (conhecido como C++ 2003). Se você já é familiarizado com C++ e algo parece completamente novo ou se algo não funciona com sua versão atual das ferramentas C++, pode ser por causa da adição de 2011.



Esse ícone sinaliza adições propostas ao padrão C++ 2014. Esses recursos não estão implementados no Code::Blocks/gcc que está disponível até o momento desta escrita. Talvez eles estejam disponíveis em [www.codeblocks.org](http://www.codeblocks.org) até você ler isto.

## Além do Livro

C++ Para Leigos inclui estes serviços online de fácil download:

- ✓ Você pode acessar a Folha de Cola Online, através do endereço: [www.altabooks.com.br](http://www.altabooks.com.br). Procure pelo título do livro/ISBN. Na página da obra, em nosso site, faça o download completo da Folha de Cola, bem como de erratas e possíveis arquivos de apoio. Os iniciantes vão querer imprimir e deixar de fácil acesso enquanto lê todos os capítulos. Como o socialismo assustador, eventualmente, a sintaxe C++ vai se tornar sua segunda língua e você não precisará mais da folha de cola.

- ✓ O código-fonte para todos os exemplos no livro podem ser baixados no site da editora [www.altabooks.com.br](http://www.altabooks.com.br), procurando pelo título do livro. Os programas estão organizados pelo número do capítulo. Eu incluí um arquivo de projeto para Code::Blocks (mais sobre ele na próxima parte, e explico os arquivos de projeto no Capítulo 1).
- ✓ Este livro usa o ambiente Code::Blocks de código-fonte e o compilador C++ GCC de graça. A versão do Code::Blocks usada na escrita deste livro (Versão 13.12) está disponível para download em [www.codeblocks.org/downloads/source/5](http://www.codeblocks.org/downloads/source/5). Eu incluí versões para o Windows (2000 e posteriores) e Macintosh (10.6 e posteriores). As versões para o Linux estão disponíveis online também. O Capítulo 1 contém instruções de como baixar e instalar o Code::Blocks. Você pode encontrar versões mais recentes de Code::Blocks e versões para diferentes versões do Linux em [www.codeblocks.org/downloads/binaries](http://www.codeblocks.org/downloads/binaries).
- ✓ Exceto pelo material disponibilizado em [www.altabooks.com.br](http://www.altabooks.com.br), os sites indicados estão em inglês e a editora não se responsabiliza pelo conteúdo, material ou permanência de sites de terceiros.



Se você realmente for em [www.codeblocks.org](http://www.codeblocks.org), certifique-se de baixar uma versão que possua o compilador gcc.

Se você já possui um compilador C++ instalado em seu computador que você prefere usar, fique à vontade contanto que seja compatível com o padrão C++ (a maioria é). Nem todos os compiladores implementaram o padrão 2011 ainda, então eu sinalizei essas extensões no livro. Além disso, se você usar um compilador diferente, sua tela não se parecerá muito com as figuras do livro.



Não recomendo usar os pacotes Visual Studio ou Visual Studio Express com este livro. Ele contém muitas extensões designadas para serem compatíveis com o .NET Framework. Uma vez que você aprenda C++ no Code::Blocks, você pode aprender programação .NET no Visual Studio.

## De Lá Para Cá, Daqui Para Lá

Descobrir uma nova linguagem de programação não é um esporte muito sensacional. Tentarei tornar o menos doloroso possível, mas você tem que ligar seu computador e mergulhar de cabeça na programação. Aqueça bem os dedos, abra bem a lombada do livro para que ele fique na horizontal perto do seu teclado (e para que você não tenha como devolver para a livraria), e mergulhe.

Se você encontrar um problema, primeiro verifique no questionário de perguntas frequentes (FAQ) em [www.stephendavis.com](http://www.stephendavis.com) — em inglês.

# Parte I

começando com  
**Programação**

em

**C++**

## *Nesta parte...*

- ✓ Explicando a construção dos blocos
- ✓ Declarando variáveis
- ✓ Definindo operadores matemáticos
- ✓ Usando operadores lógicos

## Capítulo 1

# Escrevendo Seu Primeiro Programa em C++

.....

### *Neste capítulo*

- ▶ Descobrindo C++
  - ▶ Instalando o Code::Blocks no Windows, Linux Ubuntu ou Macintosh OS X
  - ▶ Criando seu primeiro programa em C++
  - ▶ Executando seu programa
- .....

**E**ntão, cá estamos: ninguém mais além de você e eu. Nada para fazer, exceto começar. Mas vale esclarecer alguns conceitos fundamentais.

Um computador é uma máquina magnificamente rápida, mas incrivelmente estúpida. Um computador pode fazer qualquer coisa que você pedir (dentro do possível), mas ele faz *exatamente* o que foi pedido — nada a mais e nada a menos.

Talvez, infelizmente para nós, os computadores não entendam nenhuma linguagem humana adequada — eles também não falam inglês. Ah, já sei o que eu você vai dizer: “já vi computadores que entendiam inglês”. O que você realmente viu foi um computador executando um *programa* que entendia inglês moderadamente.

Os computadores entendem uma linguagem geralmente chamada de *linguagem de computação* ou *linguagem de máquina*. É possível, mas extremamente difícil para os seres humanos falarem linguagem de máquina. Entretanto, computadores e humanos deram um jeito de se entender usando linguagens intermediárias como C++. Os humanos podem falar C++ (mais ou menos), e C++ pode ser convertido em linguagem de máquina para que o computador entenda.

## Dominando Conceitos de C++

Um programa C++ é um arquivo de texto contendo uma sequência de comandos C++ unidos de acordo com as regras da gramática C++. Esse arquivo de texto é conhecido como *arquivo-fonte* (provavelmente porque é a fonte de todos os problemas). Um arquivo-fonte C++ geralmente recebe a extensão `.CPP` assim como um arquivo do Adobe Acrobat termina em `.PDF` ou um arquivo batch do MS-DOS (se lembra disso?) termina em `.BAT`.

O objetivo de programar em C++ é escrever uma sequência de comandos que podem ser convertidos em um programa de linguagem de máquina que realmente *faz* o que queremos. Essa conversão é chamada de *compilação* e é função do compilador. O código de máquina que você escreveu tem que ser combinado com configurações e instruções de desmontagem e algumas rotinas padrões de bibliotecas em um processo chamado *ligação*. Juntos, compilar e ligar são conhecidos como *construir*. Os arquivos de *máquina executáveis* gerados recebem a extensão `.EXE` no Windows. Eles não recebem nenhuma extensão em específico no Linux ou no Macintosh.

Isso parece bem fácil — então qual o problema? Prossiga.

Para escrever um programa, você necessita de dois programas específicos. Um (um editor) é o que você usa para escrever seu código enquanto você executa seu arquivo-fonte `.CPP`. O outro (um compilador) converte seu arquivo-fonte em um arquivo de máquina executável que carrega seus comandos reais (abrir uma planilha, emitir ruídos estranhos, rebater asteroides na órbita, o que quer que seja).

Hoje em dia, os desenvolvedores de ferramenta normalmente associam compilador e editor em um único pacote — um *ambiente de desenvolvimento*. Após terminar de inserir os comandos que criam seu programa, você necessita somente clicar em um botão para construir o arquivo executável.

Felizmente, existem ambientes de domínio público em C++. Eu uso um deles neste livro — o ambiente Code::Blocks. Esse editor funcionará com muitos compiladores diferentes. Mas a versão do Code::Blocks associada ao compilador GNU gcc usada para escrever este livro está disponível para download no Windows, no Macintosh e em várias versões do Linux, como descrito na parte de instalação deste capítulo.

Apesar do Code::Blocks ser de domínio público, há um incentivo para que você pague uma pequena taxa para apoiar seu desenvolvimento contínuo. Você não *tem* que pagar para usar Code::Blocks, mas você pode contribuir para a causa, se quiser. Veja o site do Code::Blocks para mais detalhes.

Eu testei os programas deste livro no Code::Blocks 13.12 que vem com a versão 4.7.1 do gcc incluído. Essa versão do gcc implementa a maioria dos padrões C++ 2011.



Você pode usar versões diferentes do gcc ou compiladores diferentes, se você preferir, mas talvez eles não implementem o padrão 2011 por completo. Por esse motivo, as extensões 2011 estão marcadas com o ícone '11 ao lado.



O compilador gcc não implementava nenhuma das extensões adicionadas ao padrão C++ 2014 enquanto eu escrevi este livro, mas eu o incluí, onde aplicável, porque algum dia ele será.

Tudo bem, eu admito: este livro é um pouco centrado no Windows. Eu testei todos os programas do livro no Windows 2000/XP/Vista/7/8, Linux Ubuntu e Macintosh OS X. Eu aponto as diferenças entre os sistemas operacionais no texto.

Além disso, eu incluo instruções de instalação para cada um dos sistemas operacionais acima neste capítulo. As versões do Code::Blocks e do gcc estão disponíveis para outras versões do Linux e do Macintosh OS. Os programas devem funcionar com essas também.



O pacote Code::Blocks/gcc gera programas de 32 bits, mas não suporta facilmente criar programas em janelas. Os programas deste livro são executados a partir de um prompt de linha de comando e escrevem para a linha de comando. Tão entediante quanto isso parece ser, eu recomendo que você trabalhe primeiro pelos exemplos deste livro para aprender C++ antes de lidar com desenvolvimento em janelas. Programação em C++ e no Windows são duas coisas distintas e (para o bem da sua sanidade) devem permanecer assim na sua mente.

Siga os passos na próxima seção para instalar o Code::Blocks e montar seu primeiro programa em C++. O objetivo desse programa é converter um valor de temperatura inserido pelo usuário de grau Celsius para Fahrenheit.

## Instalando o Code::Blocks

O site [www.codeblocks.org](http://www.codeblocks.org) possuía a versão mais recente do ambiente Code::Blocks na época desta escrita para Windows, Linux Ubuntu e Macintosh OS X 10.6 ou posterior. Siga as instruções de instalação que se aplicam ao seu sistema operacional abaixo.



## Windows

O ambiente Code::Blocks vem em um arquivo executável de fácil instalação que é compatível com todas as versões do Windows após o Windows 2000. Eis um resumo da instalação do ambiente:

1. **Faça o download do executável `codeblocks-13.12.mingw-setup.exe` em [www.codeblocks.org/downloads/source/5](http://www.codeblocks.org/downloads/source/5).**

Salve o executável em sua área de trabalho ou em algum lugar fácil de encontrar.



A versão 4.71 do compilador GCC está incluída. Essa não é a versão mais nova mas é a recomendada pelo Code::Blocks. Se você quiser a versão 4.81 mais atualizada e com pequenos bugs, você pode fazer o download e instalar `codeblocks-13.12.mingwsetup-TDM-GCC-481.exe`. Eu testei os programas deste livro com as duas versões mas usei a versão 4.71 para esta escrita.

2. **Clique duas vezes no programa após o download completo.**
3. **Dependendo da versão do Windows que você estiver usando, pode aparecer o tradicional aviso pop-up “Um programa não identificado quer acessar o seu computador”. Se for o caso, clique em Permitir para iniciar a instalação.**
4. **Clique em Next depois de fechar todas as aplicações abertas como você foi avisado na caixa de diálogo Welcome do Code::Blocks Setup Wizard.**
5. **Leia o End User License Agreement (mais conhecido como EULA) e então clique I agree se você puder viver sob essas condições.**

Na verdade, você não tem muitas opções — o pacote não será instalado se você não aceitar. Presumindo que você *realmente* clique em OK, o Code::Blocks abre uma caixa de diálogo mostrando as opções de instalação. As opções padrão estão corretas.

6. **Clique no botão Next.**

O programa de instalação permite que você instale somente um subconjunto dos recursos. Você precisa selecionar ao menos Default Install e o MinGW Compiler Suite. O padrão é instalar tudo — é a melhor opção.



Se o MinGW Compiler Suite não estiver nas opções, você deve ter instalado a versão do Code::Blocks que não inclui o gcc. Essa versão não funcionará corretamente.

## 7. Clique Install e aceite o Destination Folder padrão.

O Code::Blocks começa a copiar um grande número de arquivos para o seu disco rígido. Depois, ele pergunta “Do you want to run Code::Blocks now?”

## 8. Clique Yes para iniciar o Code::Blocks.

O Code::Blocks perguntará qual compilador você pretende usar. O padrão GNU GCC Compiler é a seleção adequada.

## 9. Dentro de Code::Blocks, escolha Settings ⇨ Compiler.

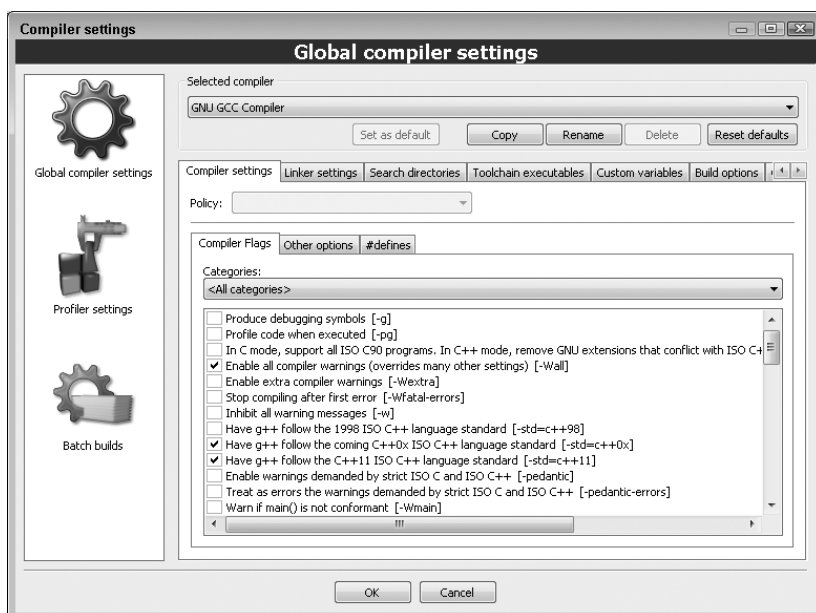
## 10. Selecione a aba Compiler Flags.

## 11. Certifique-se de selecionar as três opções, como mostra a Figura 1-1:

- Enable All Compiler Warnings
- Have g++ Follow the Coming C++0x ISO C++ Language Standard
- Have g++ Follow the C++11 ISO C++ Language Standard

O padrão C++ 2011 era para ser o padrão C++ 2008 ou 2009. Como não estava claro, o padrão ficou conhecido como o padrão 0x. Ele não foi completamente aceito até 2011. No gcc, C++ 0x e C++ 11 se referem ao mesmo padrão.

**Figura 1-1:**  
Verifique se as opções Enable All Compiler Warnings e C++ 2011 estão marcadas.



**12. Selecione a aba Toolchain Executables. Ela deve se parecer com a Figura 1-2.**

O local padrão para o compilador gcc é o subdiretório MinGW\bin dentro do diretório Code::Blocks



Se o local padrão estiver vazio, então Code::Blocks não sabe onde o compilador gcc está e não será possível montar seus programas. Certifique-se de fazer o download da versão do Code::Blocks que contém o gcc e de incluir o MinGW durante a instalação. Se você estiver usando um compilador gcc existente que você já tenha instalado, você deverá indicar ao Code::Blocks onde ele está localizado no seu disco rígido.

**13. Feche a caixa de diálogo Settings.**

**14. Clique em Next na caixa de diálogo Code::Blocks Setup e depois em Finish para completar o programa de configuração.**

O programa de configuração fecha.

## Linux Ubuntu

O Code::Blocks não contém o gcc no Linux, portanto a instalação é um processo de dois passos. Primeiro, você precisa instalar o gcc. Somente depois você poderá instalar o Code::Blocks.

**Figura 1-2:**  
Verifique se o diretório de instalação do compilador está correto.

