ALEXANDRE PEREIRA

C e Algoritmos

2ª Edição
Revista e Atualizada

EDIÇÕES SÍLABO

C e Algoritmos

ALEXANDRE PEREIRA

2ª EDIÇÃO
Revista e Atualizada

EDIÇÕES SÍLABO

Visite a Sílabo na rede

www.silabo.pt

Editor: Manuel Robalo

FICHA TÉCNICA:

Título: C e Algoritmos
Autor: Alexandre Pereira
© Edições Sílabo, Lda.
Capa: Pedro Mota

1ª Edição – Lisboa, janeiro de 2013
2ª Edição – Lisboa, janeiro de 2017
Impressão e acabamentos: Cafilesa – Soluções Gráficas, Lda.
Depósito Legal: 419981/17
ISBN: 978-972-618-871-1

EDIÇÕES SÍLABO, LDA.

R. Cidade de Manchester, 2
1170-100 Lisboa
Tel.: 218130345
Fax: 218166719
e-mail: silabo@silabo.pt
www.silabo.pt

Índice

Capítulo 1

Introdução

1.1. Breve sinopse 11

1.2. Ambientes de desenvolvimento 12

1.2.1. Dev C++ 13

1.2.2. gcc 17

Capítulo 2

Linguagem C

2.1. Estruturas básicas 19

2.1.1. Expressão 19

2.1.2. Instrução 20

2.1.3. Função 21

2.1.4. Comentário 21

2.1.5. Programa 22

2.2. Tipos de dados, variáveis e constantes 24

2.2.1. Variáveis 24

2.2.2. Constantes 26

2.2.3. Tipos escalares 27

2.2.4. Conversão de tipos 40

2.2.5. `void` 42

2.3. Operadores 43

2.3.1. Precedência dos operadores 43

2.3.2. Operadores aritméticos 45

2.3.3. Operadores relacionais e de igualdade 46
2.3.4. Operadores lógicos 46
2.3.5. Operadores bit a bit 47
2.3.6. Operadores de atribuição 53
2.3.7. Operadores unários 54
2.3.8. Operadores primários 58
2.3.9. Outros operadores 59
2.4. Instruções de controlo de fluxo 60
2.4.1. `if` 61
2.4.2. `switch` 62
2.4.3. `while` 63
2.4.4. `do-while` 64
2.4.5. `for` 65
2.4.6. `break` e `continue` 66
2.4.7. `goto` 67
2.5. Apontadores e vetores 68
2.5.1. Apontadores 68
2.5.2. Aritmética de apontadores 72
2.5.3. Parâmetros de funções e apontadores 74
2.5.4. Vetores 76
2.5.5. Apontadores e vetores 79
2.5.6. Apontadores, vetores e funções 80
2.5.7. Cadeias de carateres 81
2.5.8. Funções de manipulação de cadeias 84
2.5.9. Vetores multidimensionais 86
2.5.10. Vetores de apontadores 88
2.5.11. Apontadores para apontadores 89
2.6. Acesso à memória 90
2.7. Visibilidade e longevidade 92
2.7.1. Variáveis externas 92
2.7.2. Variáveis internas 93
2.7.3. Visibilidade e longevidade de variáveis 93
2.7.4. Qualificador `extern` 94
2.7.5. Qualificador `static` 94
2.7.6. Qualificador `register` 96
2.7.7. Qualificadores `auto` e `volatile` 97

2.8. Estruturas, uniões e `typedef` 97
2.8.1. Estruturas 98
2.8.2. Inicialização de estruturas 100
2.8.3. Acesso a estruturas 100
2.8.4. Vetores de estruturas 102
2.8.5. Uniões 102
2.8.6. Nomes de tipos 104
2.9. Funções 105
2.9.1. Recursividade 106
2.9.2. Apontadores para funções 107
2.9.3. Argumentos para a função `main` 109
2.10. Entradas e saídas 110
2.10.1. Manipulação de carateres 110
2.10.2. Manipulação de linhas 113
2.10.3. `printf` 115
2.10.4. `scanf` 117
2.10.5. Ficheiros 120
2.10.6. Escrita em ficheiros 122
2.10.7. Leitura de ficheiros 124
2.10.8. Outras operações sobre ficheiros 127
2.11. Pré-processador 128
2.11.1. Processamento de macroinstruções 129
2.11.2. Inclusão de ficheiros 132
2.11.3. Compilação condicional 133
2.11.4. Controlo de erros 135
2.11.5. Controlo do compilador 136
2.12. Bibliotecas de funções 137
2.12.1. ctype.h 138
2.12.2. string.h 138
2.12.3. stdio.h 139
2.12.4. stdlib.h 141
2.12.5. math.h 141
2.13. Normas da linguagem C 142
2.13.1. C89 142
2.13.2. C99 143
2.13.3. C11 150

Capítulo 3

Estruturas de dados e algoritmos

3.1. Eficiência dos algoritmos 157
3.1.1. Notação O-grande 158
3.2. Vetores de tamanho fixo 159
3.2.1. Acesso linear 160
3.2.2. Pesquisa linear 161
3.2.3. Pesquisa binária 163
3.2.4. Inversão 166
3.2.5. Bubblesort 168
3.2.6. Ordenação por inserção 174
3.2.7. Shellsort 176
3.2.8. Quicksort 180
3.3. Listas 185
3.3.1. Elementos de uma lista 185
3.3.2. Gestão de uma lista 187
3.3.3. Inserção no início da lista 187
3.3.4. Impressão da lista 191
3.3.5. Inserção no fim da lista 193
3.3.6. Múltiplas listas 195
3.4. Listas genéricas 197
3.4.1. Pesquisa genérica 201
3.4.2 Remoção 203
3.4.3. Bubblesort 205
3.4.4. Quicksort 207
3.4.5. Escrita e leitura em ficheiros 211
3.5. Filas e pilhas 214
3.6. Vetores dinâmicos 215
3.6.1. Gestão de um vetor dinâmico 216
3.6.2. Inserção 217
3.6.3. Impressão 218
3.6.4. Pesquisa 219
3.6.5. Ordenação 221
3.6.6. Remoção 222

3.7. Árvores 224
3.7.1. Árvores AVL 225
3.7.2. Árvores B 232
3.8. Resumo da eficiência dos algoritmos 237

Bibliografia 239
Anexo 241
Lista de tabelas 243
Lista de quadros 245
Lista de figuras 249
Lista de códigos fonte 251
Índice remissivo 257

Capítulo 1

Introdução

1.1. Breve sinopse

O C é uma linguagem de programação criada por Dennis Ritchie no início da década de 70 do século XX. É uma linguagem de complexidade baixa, estruturada, imperativa e uma linguagem com caraterísticas maioritariamente do paradigma de programação procedimental. A linguagem teve uma primeira versão documentada por Dennis Ritchie e Brian Kernighan no livro *The C Programming Language*, em 1978, onde foram acrescentados alguns elementos novos à sintaxe e corrigidas algumas ambiguidades da linguagem original. Mais tarde, em 1989, o American National Standards Institute publicou um standard para a linguagem C, que introduziu melhorias nalguns elementos da sintaxe. O standard mais atual da linguagem foi aprovado em dezembro de 2011 e tomou a designação de C11. Os exemplos deste livro são escritos, maioritariamente, usando a norma C89. Na secção 2.13 são apresentadas as extensões introduzidas na linguagem com as normas C99 e C11.

Nas décadas de 50 e 60 do século passado, as linguagens de programação então existentes, nomeadamente o Fortran, o Cobol e o Basic, não continham instruções de controlo de fluxo fechadas, ou estruturadas. A programação de blocos de código não sequenciais era obtida recorrendo a saltos incondicionais: utilizando uma instrução GOTO, ou equivalente. A ideia de estrutura e estruturação de software surgiu apenas a partir da segunda metade da década de 60 e foi levada à prática em implementações como as linguagens de programação ALGOL, Pascal e Ada. O C

começou a ser desenvolvido no fim da década de 60 como uma linguagem estruturada e, apesar de manter uma instrução de salto incondicional – `goto` – tem também um conjunto de instruções de controlo de fluxo estruturadas, assim como funções estruturadas.

A linguagem C tem uma complexidade baixa, ou seja, é uma linguagem constituída por construtos simples que podem ser facilmente traduzidos em linguagens máquina e, por esse motivo, é uma linguagem adequada para a programação de software de sistema. Os núcleos dos sistemas Unix, Linux, Windows e Mac OS são programados maioritariamente em C. Muitas das funções das bibliotecas do C são simultaneamente funções dos sistemas operativos Unix e Linux.

Embora os conceitos de função e procedimento não tenham, em C, uma distinção explícita como têm em Pascal e, do ponto de vista puramente sintático, o C possua apenas funções e não procedimentos, a linguagem C tem, como foi dito atrás, caraterísticas, na sua maioria, do paradigma de programação procedimental. Quer isto dizer que o C privilegia uma abordagem *top-down* para a resolução de problemas: a criação de funções e módulos, com soluções parciais, que podem ser invocados por funções de mais alto nível e, em última análise, pela função `main`.

Muitas linguagens de programação, posteriores à linguagem C, utilizaram a mesma sintaxe de base do C. Entre elas contam-se o Java, o Javascript, o Actionscript, o PHP, o Perl, o C# e o C++.

1.2. Ambientes de desenvolvimento

Algumas linguagens de programação não utilizam compiladores: os programas escritos nessas linguagens são executados à medida que são lidos e validados por um interpretador. É o caso das linguagens Javascript e PHP mencionadas atrás.

Ao contrário das linguagens interpretadas, os programas em linguagem C precisam de ser previamente compilados antes de poderem ser executados. Existem compiladores de C gratuitos disponíveis para os sistemas operativos mais comuns.

Em Linux é comum usar o **gcc** como compilador de programas em C. O **gcc** costuma vir instalado, de base, em todas as distribuições de Linux.

Para Windows existem diversas ferramentas gratuitas disponíveis. O Dev C++ é um ambiente de desenvolvimento com editor, compilador e depurador, que exige poucos recursos do computador, sendo por isso uma boa opção. Pode ser encontrado em http://www.bloodshed.net/.

No Mac OS também se pode utilizar o **gcc**, tal como em Linux. Para instalar o **gcc** no Mac OS, descarrega-se o pacote **xcode** a partir do endereço http://connect.apple.com/. É necessário criar, previamente, uma conta de programador no sítio de Internet da Apple.

Nas secções seguintes, indica-se como compilar o programa abaixo com o Dev C++ e o **gcc**.

Código fonte 1.1. Programa simples em C

```
#include <stdio.h>

int main() {
          printf("Olá Mundo\n");
}
```

É necessário deixar uma linha em branco por baixo da última chaveta do código anterior.

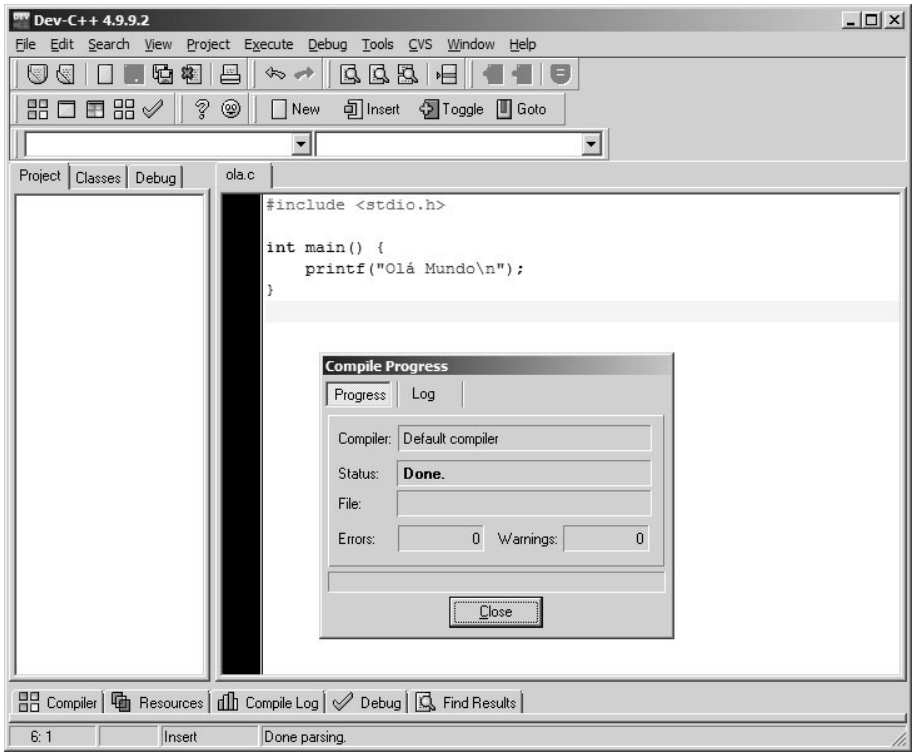
1.2.1. Dev C++

Para compilar o programa do Código fonte 1.1, abrir o Dev C++ e executar os passos abaixo:

- Nos menus, selecionar: *File* → *New* → *Source File*
- Digitar o programa apresentado atrás (ver Código fonte 1.1)
- Para gravar o ficheiro, selecionar nos menus: *File* → *Save As...*
- Surge a caixa de diálogo *Save File*
  - — Em *Save in*, selecionar a diretoria pretendida
  - — Na opção *Save as type*, selecionar *C source files (*.c)*
  - — Em *File name*, colocar o nome pretendido para o ficheiro
- Para compilar o programa, selecionar nos menus: *Execute* → *Compile*
- Após a compilação com sucesso, premir o botão *Close* da caixa *Compile Progress* (ver Figura 1.1)

- Caso a compilação tenha tido sucesso, o Dev C++ coloca o ficheiro executável, resultado da compilação, na mesma diretoria do código fonte.

Figura 1.1. Compilação com sucesso



Para executar o programa compilado, executar os passos seguintes:

- Abrir uma janela de consola. Para isso, no teclado premir as teclas ⊞+R.[1] Alternativamente pode premir o botão *Start*,[2] na barra de tarefas do Windows, e executar o comando **cmd**.[3]
- Na janela de consola, digitar "**cd**", ou seja, as teclas **cd** seguidas de um espaço

(1) Premir a tecla Janela (⊞) e, com esta premida, premir a tecla R.

(2) Botão *Iniciar*, no Windows em português.

(3) No Windows XP: *Start* → *Run* (Iniciar → Executar, no Wndows em português) e depois digitar **cmd** e premir *OK*. No Windows 7, premir *Start*, digitar **cmd** e premir a tecla ENTER.

- Depois, a partir do Explorador do Windows, arrastar o ícone da diretoria que contém o executável para cima da janela de consola (ver Figura 1.2)
- Largar o botão do rato sobre a janela de consola. O caminho da diretoria é escrito nesta janela.
- Na janela de consola, premir a tecla ENTER.
- Para verificar se o ficheiro executável se encontra nessa diretoria, digitar **dir** e premir ENTER. O resultado deverá ser semelhante ao da Figura 1.3
- Para executar o programa, digitar o nome do executável na consola e premir ENTER. O resultado deverá ser o da Figura 1.4. De notar que a letra acentuada pode não ser impressa corretamente na janela de consola. Neste caso, foi impresso "Olβ Mundo". Isto deve-se a um problema na codificação de carateres do Windows.

Figura 1.2. Arrastar diretoria para janela de consola

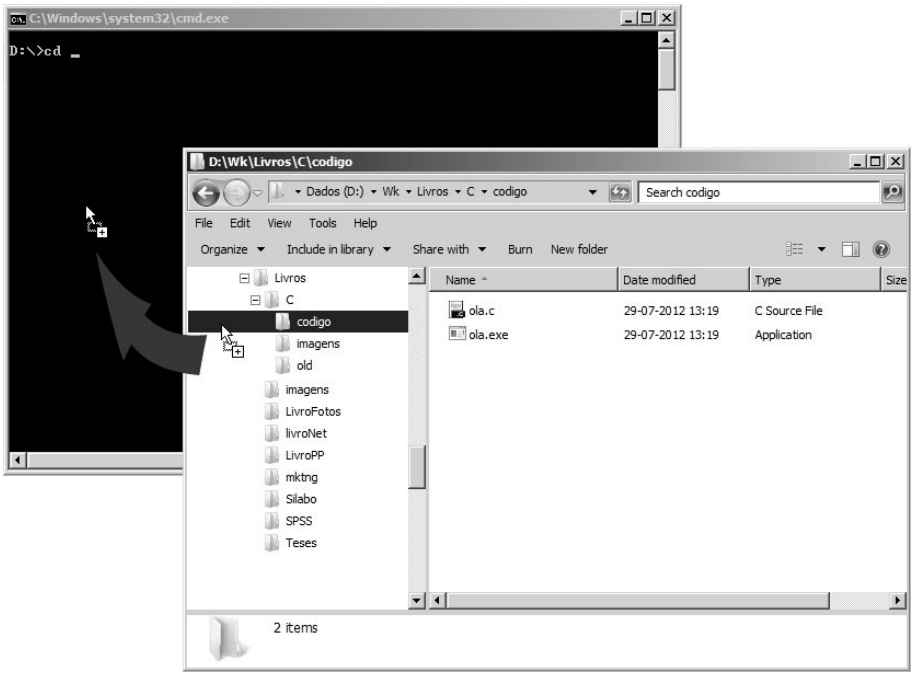


Figura 1.3. Listagem de diretoria

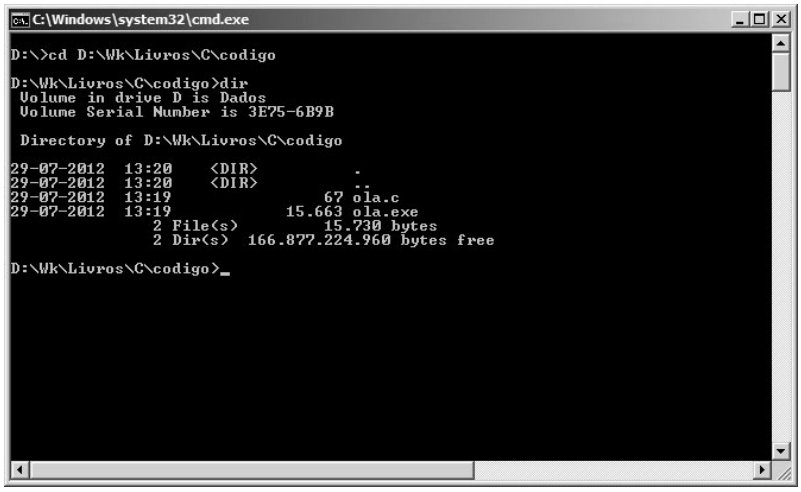
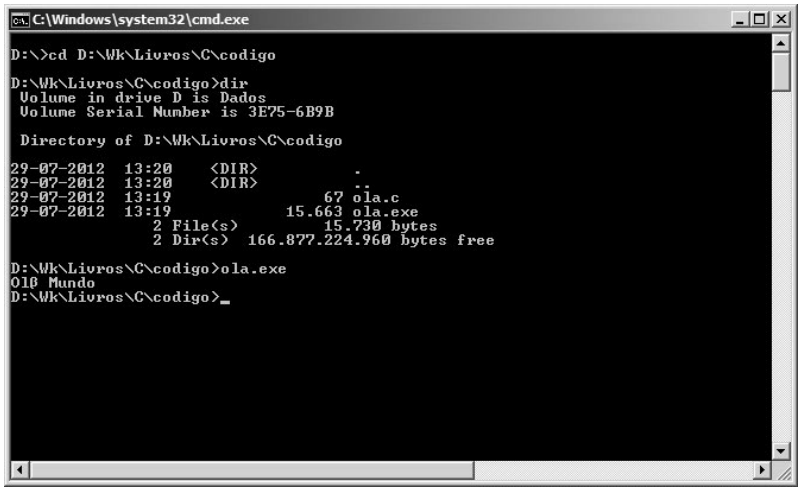


Figura 1.4. Execução do programa em Windows



1.2.2. gcc

Para compilar o programa anterior com o **gcc**, em Linux ou Mac OS, executar os passos seguintes:

- Abrir um editor de texto simples e digitar o texto do programa apresentado atrás (ver Código fonte 1.1)
- Gravar o ficheiro na diretoria pretendida
- Numa janela de consola, deslocar-se até à diretoria onde foi gravado o ficheiro com o código fonte e executar o seguinte comando:

  gcc -o ola ola.c

- Depois, para verificar se o ficheiro executável foi criado, digitar o comando:

  ls -alsp

- Deve obter-se um resultado semelhante ao da Figura 1.5
- Para executar o programa, digitar o comando seguinte:

  ./ola

- O resultado do programa executado é apresentado na Figura 1.6. De notar que, em Linux, a letra acentuada é impressa corretamente, contrariamente ao que acontece em Windows.

Figura 1.5. Compilação em Linux

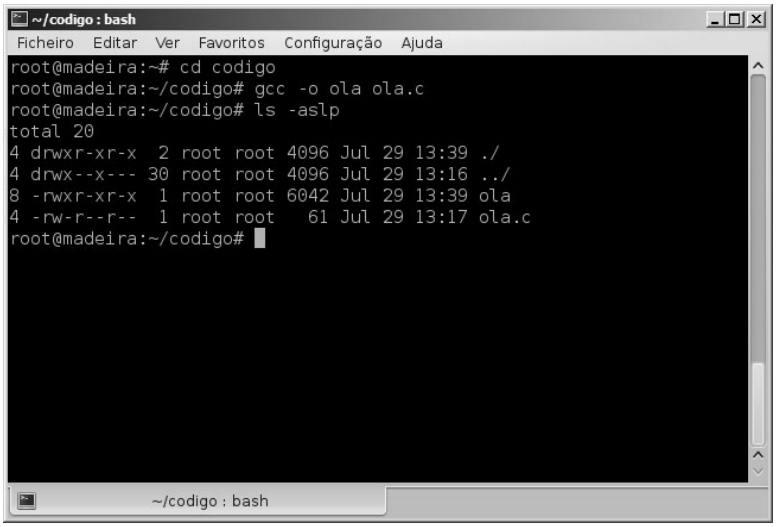
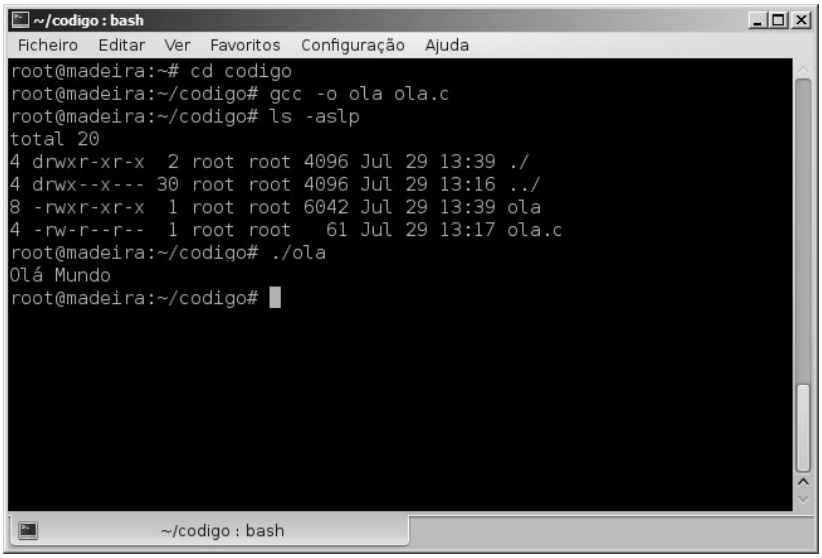


Figura 1.6. Execução do programa em Linux



Capítulo 2

Linguagem C

Neste capítulo, é apresentada a sintaxe da linguagem C, de acordo com a norma C89. São também referidos alguns elementos propostos pelas normas C99 e C11.

2.1. Estruturas básicas

Nesta secção apresentam-se as estruturas básicas da linguagem. Estas estruturas constituem os elementos com os quais se criam programas em C. Os termos descritos nesta secção serão utilizados ao longo do livro com o significado que aqui lhes é dado, sendo, por isso, importante compreendê-los e memorizá-los.

2.1.1. Expressão

Uma expressão é um conjunto de ações que, após calculado, pode produzir um resultado ou um valor. Pode ser constituída por uma sequência de valores numéricos e operadores, variáveis e operadores, uma invocação de uma função, ou uma combinação destes elementos.

Seguem-se alguns exemplos de expressões.

Código fonte 2.1. Expressão aritmética com literais e operadores

```
(5 + 3) * 7
```

Código fonte 2.2 Expressão aritmética com uma variável e invocação de função

```
1 + sqrt(x)
```

2.1.2. Instrução

Uma instrução é uma expressão terminada por um símbolo de ponto e vírgula “;”. A instrução é a unidade independente mais elementar de um programa em C.

Código fonte 2.3. Instrução de definição de variável

```
int numero;
```

Código fonte 2.4. Instrução de atribuição

```
numero = 3 + 2;
```

Código fonte 2.5. Instrução com invocação de função

```
printf(“Olá Mundo”);
```

2.1.3. Função

Uma função pode ser vista como um bloco de instruções ao qual é atribuído um nome, que pode ser invocado a partir desse nome, e que pode receber parâmetros e devolver um valor.

A definição de uma função em C inclui:

- O tipo de valor devolvido
- O nome da função
- A lista e tipo de argumentos
- O corpo da função, com declarações e instruções.

Segue-se um exemplo de definição de uma função que devolve o máximo de dois valores passados por parâmetro.

Código fonte 2.6. Exemplo de definição de uma função

```
int maximo (int a, int b) {
     int c;
     if (a > b)
          c = a;
     else
          c = b;
     return c;
}
```

2.1.4. Comentário

Um comentário, numa linguagem de programação, é um bloco de texto ignorado pelo compilador. Os comentários são utilizados para descrever o código em linguagem natural. Até à versão C89, o C permitia apenas um formato de comentários multilinha delimitado pelos carateres /* e */. A partir da versão C99, o C permite também comentários que se iniciam com a sequência // e terminam no fim da linha.

Código fonte 2.7. Exemplo de comentário multilinha

```
/* Este comentário ocupa
mais do que uma linha
*/
```

Código fonte 2.8. Exemplo de comentário de uma linha

```
// Este comentário termina no fim desta linha
```

Segue-se a aplicação de um comentário para descrever uma função.

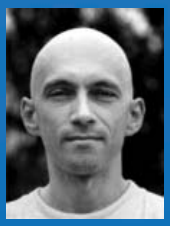
Código fonte 2.9. Comentário que descreve uma função

```
/*
* Função: mínimo
* Recebe: a e b, dois valores inteiros
* Devolve: o mínimo desses dois valores
*/
int minimo (int a, int b) {
     if (a < b)
          return a;
     else
          return b;
}
```

2.1.5. Programa

Qualquer programa em C tem que conter uma função denominada `main`, que é a função principal e aquela a partir da qual se inicia a execução do programa.

Em C todas as funções são definidas no âmbito global, separadas umas das outras. Não é possível definir funções dentro de outras funções.



ALEXANDRE PEREIRA

Mestre em Engenharia Eletrotécnica e de Computadores pelo Instituto Superior Técnico. Doutorando em Ciências da Comunicação na Universidade Lusófona de Humanidades e Tecnologias. Professor na Licenciatura em Aplicações Multimédia e Videojogos da Universidade Lusófona. Professor na Licenciatura em Engenharia Informática da Universidade Europeia, Laureate International Universities.

O C é uma linguagem de programação incontornável no estudo e aprendizagem das linguagens de programação. É um precursor das linguagens de programação estruturadas e a sua sintaxe foi reutilizada em muitas linguagens posteriores, mesmo de paradigmas diferentes, entre as quais se contam o Java, o Javascript, o Actionscript, o PHP, o Perl, o C# e o C++.

Este livro apresenta a sintaxe da linguagem C tal como especificada pelas normas C89, C99 e C11, da responsabilidade do grupo de trabalho ISO/IEC JTC1/SC22/WG14.

É, por isso, uma ferramenta indispensável para:

- Professores;
- Estudantes;
- Programadores;
- ***Web designers***;
- Autodidatas.

Para além da sintaxe da linguagem C, são também apresentados conceitos e aplicações práticas de estruturas de dados e algoritmos, nomeadamente:

- Vetores, listas e árvores;
- Inserção, remoção, pesquisa, impressão de estruturas;
- Diversos algoritmos de ordenação de estruturas;
- Eficiência temporal e espacial dos diversos algoritmos apresentados.

C e Algoritmos

449

ISBN 978-972-618-871-1

9 789726 188711