Técnicas computacionais

Cap. 2: Linguagem C

Grazione de Souza

PPGMC/IPRJ/UERJ

2022/1

Conteúdo

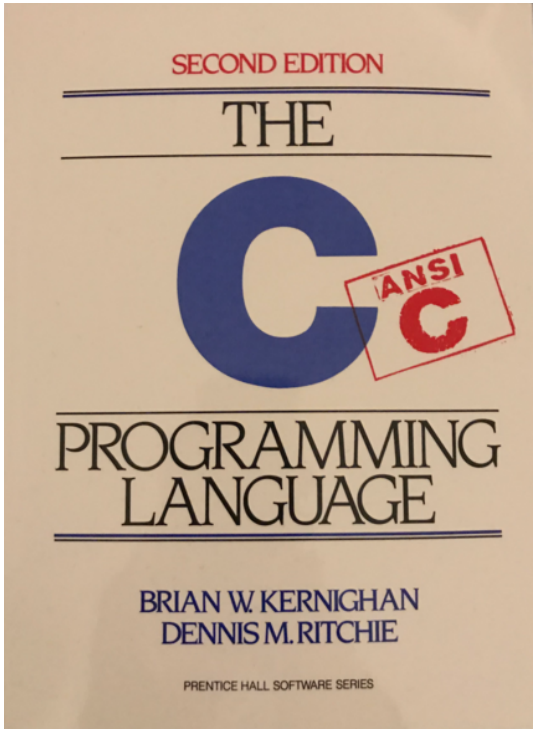

Figura 1: Livro de Kernighan e Ritchie.

Introdução à Linguagem C

Origem da linguagem C: Dennis Ritchie - AT & T Bell Laboratories - 1972

Algumas características da linguagem C:

- largamente utilizada;
- se estende a arquiteturas mais recentes;
- eficiência/desempenho;
- acesso de baixo nível;
- poucas palavras-chaves e
- biblioteca padrão externa.

O seguinte breve histórico pode ser considerado para a linguagem C:

- 1972: origem da linguagem C;
- 1978: *The C Programming Language* foi publicado e há a primeira especificação da linguagem;
- 1989: C89 standard (conhecido como ANSI C ou Standard C);
- 1990: ANSI C é adotado pela ISO, conhecido como C90;
- 1999: C99 standard (maior parte da base anterior é compatível, não completamente implementado em muitos compiladores);
- 2007: trabalho no novo padrão C, C1X, é anunciado;
- 2011: novo padrão, conhecido como C11 ou C1X;
- 2017: última versão, C17;
- 2020: versão C2X em testes.

A linguagem é utilizada na programação de sistemas, como por exemplo em:

- sistemas operacionais, como o Linux;
- micro-controladores em automóveis e aviões;
- sistemas embarcados, como em telefones e eletrônica portátil e
- sistemas de áudio e TV.

Algumas derivações do C são: C++, Objective C e C#.

Algumas linguagens influenciadas pelo C são: Java, Perl, Python.

Linguagem C é de baixo nível, levando a executável mais rápido (geralmente) quando com comparado a outras linguagens com o mesmo ou maior distanciamento do usuário.

Escrevendo códigos em C

A extensão dos códigos fonte é .c, arquivo a ser editado diretamente.

O gcc é o compilador básico (incluído na maioria das distribuições Linux)

O executável tem extensão .o no Linux e .exe no Windows.

Algumas IDEs (*Integrated development environment*), os ambientes de desenvolvimento integrado são: Eclipse, Microsoft Visual C++, KDevelop, Xcode, Dev C++, etc. Tratam-se de ferramentas que permitem edição, compilação e execução e são muito convenientes para códigos maiores.

O Eclipse pode ser encontrado em http://www.eclipse.org/cdt/, sendo que para um projeto novo é possível se utilizar New, C Project e escolher "Hello World ANSI C Project"para um projeto simples que pode ser incrementado.

Primeiro código em C

Um primeiro código em C pode ser criado no estilo "Hello, world!", para mostrar a estrutura básica do arquivo .c, a sintaxe, a função main(), as declarações básicas e como usar comentários.

```
/* Comentários sobre o conteúdo do código */

Insira #include e definições
Protótipos de funções e declarações de variáveis
função main() é definida
{
        Corpo da função main
}
Definição de outra função
{
        Corpo da outra função
}
```

Os comentários usando /* e */ podem ocupar várias linhas, como em

```
/* hello.c – our first C program
Criado por J. J. Silva, 01/11/2010 */
```

Os comentários são completamente ignorados pelo compilador e podem aparecer em qualquer local. Para comentários em uma única linha é possível utilizar:

```
// comentario em 1 linha
```

Arquivos de cabeçalho contém constantes, funções e outras declarações. Por exemplo, tem-se
#include <stdio.h>

que serve para se ler os conteúdos da biblioteca stdio.h, que é a biblioteca padrão para entrada e saída usando o console e arquivos.

Para o exemplo abordado, tem-se

```
/* hello.c – our first C program
Criado por J. J. Silva, 01/11/2010 */
#include <stdio.h> /* funcoes basicas de I/O */
```

Do ponto de vista de funções, é possível utilizar

```
/* hello.c – our first C program
Criado por J. J. Silva, 01/11/2010 */

#include <stdio.h> /* funcoes basicas de I/O */
void hello();
void main()
{
        hello();
}
void hello()
{
        printf("Hello!")
}
```

A biblioteca stdio.h é parte do padrão do C.

Outros importantes arquivos de cabeçalho são: ctype.h, math.h, stdlib.h, string.h, time.h

Para detalhes (registro requerido):

www.unix.org/single_unix_specification/

Variáveis devem ser declaradas antes de serem usadas.
Exemplos de declaração:

int n;
float phi.

int: tipo inteiro.

float: tipo ponto flutuante.

Variáveis não inicializadas assumem um valor padrão.

Exemplos de inicialização (tendo-se escrito acima int n):

- n = 3;
- float phi = 1.6180339887;
- int a, b, c = 0, d = 4;

Supondo x e y serem variáveis, tem-se as seguintes operações de aritmética binária: x+y, x-y, x*y, x/y, x%y

Exemplo de cálculo:
y = x+3*x/(y-4);

Números são válidos nas expressões.
Ponto e vírgula encerra a expressão.
Aritmética e atribuição:

- x += y;
- x -= y;
- x*= y;
- x/=y;
- x %= y;

Ordem dos operadores:

- +,- (sinal) direita para a esquerda;
- *,/,% esquerda para a direita;
- +,- esquerda para a direita;
- =,+=,-=,*=,/=,%= direita para a esquerda.

Parêntesis são usados para sobrepor a ordem de avaliação.

Considera-se x = 2.0 e y = 6.0. Avalia-se

float z = x+3*x/(y-4);

1. Avalia-se a expressão dentro dos parêntesis

float z = x+3*x/(y-4);
float z = x+3*x/2.0;

2. Avalia-se multiplicações e divisões, da esquerda para a direita float

z = x+3*x/2.0;
float z = x+6.0/2.0;
float z = x+3.0;

3. Avalia-se a adição

float z = x+3.0;
float z = 5.0;

4. Atribuição
z = 5.0.

Com parênteses é possível se obter z = 4.0:

float z = (x+3*x)/(y-4);

Funções também devem ser declaradas antes do uso, sendo a declaração chamada de protótipo de função, como em

```
int fatorial (int);
```

ou

```
int fatorial (int n);
```

Protótipos para muitas funções comuns estão na biblioteca padrão do C.

Forma geral:

tipo de retorno nome da função (arg1,arg2,...);

Os argumentos são variáveis locais, passadas na chamada da função.

O valor de retorno é o resultado quando se sai da função.

O uso de void indica ausência de argumento e/ou valor de retorno:

```
int rand (void);
void teste (void);
```

A função main() é o "ponto de entrada"para o programa em C. Na versão mais simples, não há entrada e a saída é 0 quando bem sucedida e não nula como sinal de algum erro:

```
int main(void);
```

Exemplo de função main com dois argumentos:

```
int main(int argc, char **argv);
```

Na declaração da função tem-se

```
declaração da função
{
        declaração de variáveis
        declarações do programa;
}
```

A declaração deve coincidir com o protótipo (os nomes dos argumentos na chamada, não). Não há uso de ponto e vírgula no final da declaração.

As chaves definem um bloco ou região de código. As variáveis declaradas em um bloco existem somente nele. As declarações de variáveis estão antes das outras declarações.

```
/* função main ( ) */
int main (void) /* ponto de entrada */
{
        /* escrever mensagem para o console */
        puts("hello students");
        return 0; /* exit (0 => success ) */
}
```

A função puts() leva a saída de texto no console (stdout). O conjunto de caracteres é escrito entre aspas duplas. Para terminar a função, tem-se return 0.

Alternativamente, o conjunto de caracteres, string, pode ser armazenado primeiramente

```
i n t main ( void ) / * ponto de entrada * / {
        const char msg [ ] = "hello students";
        /* mensagem para o console */
        puts(msg);
```

A palavra chave const qualifica uma variável como constante. O tipo char representa um único caracter, escrito entre aspas simples, como em 'a', '3', 'n'. Ao usar

```
const char msg[]
```

tem-se um vetor constante de caracteres.

As strings são armazenadas como vetores de caracteres, sendo o último nulo.

Os caracteres especiais são especificados usando contrabarra antes, tais como apóstrofe, aspas e a própria contrabarra.

Espaço, tabulação e mudança de linha são feitos usando \b, \t, e \n.

\ooo e \xhh são usados para códigos octal e hexadecimal ASCII, e.g., \x41 - 'A', \060 - '0'

stdout e stdin representam a saída e a entrada usando o console.

Usando puts(string) e putchar(char), respectivamente, string e caracter são postos no console. Usando

char = getchar()

retorna um caracter a partir de stdin. Também há, similarmente string = gets(string)