

CAPÍTULO 1

INTRODUÇÃO

1.1 História

A origem do nome da linguagem C é muito simples. É a linguagem que sucede a linguagem B. Por sua vez, a linguagem B teve seu nome retirado da inicial do local onde ela foi desenvolvida: Laboratórios Bell. A primeira versão da linguagem C foi escrita e implementada por D.M. Ritchie. Foi inicialmente publicada no livro "The C Programming Language", por B.W. Kernighan & D.M. Ritchie em 1978.

Diversas versões de C, incompatíveis, foram criadas. Estas versões funcionavam somente com um determinado compilador, rodando apenas em uma única plataforma, o que tornava os códigos computacionais muito restritos a determinadas condições. Em 1983, a ANSI (American National Standards Institute) fundou uma comissão para definir uma versão padronizada para a linguagem C. Esta versão chamou-se ANSI C. Desta forma, simplesmente compilando o código fonte em qualquer sistema, um programa escrito em ANSI C funciona em praticamente qualquer computador.

1.2 Estruturação de um programa em C

A seguir é fornecido um programa exemplo, contendo algumas das principais funcionalidades da linguagem C. Cada item do programa será detalhado mais adiante no curso.

```
main()
{
    /* Programa exemplo */
    float a, b, c;

    printf("Digite dois numeros:\n");
    scanf("%f", &a);
    scanf("%f", &b);
    if(a > b)
        c = a * a;
    else
        c = b * b;
    printf("Quadrado do maior numero digitado: %f\n", c);
}
```

A primeira linha do programa, `main()`, indica que é a primeira função a ser executada, ou seja, é por onde o programa começa a execução.

O abre-chaves, `{`, na segunda linha começa o corpo da função.

A terceira linha, `/* Programa exemplo */`, é um comentário e é ignorada pelo compilador.

Na quarta linha são declaradas as três variáveis que serão utilizadas pelo programa.

A quinta linha é uma linha vazia. É ignorada pelo compilador e pode ser utilizada em qualquer lugar dentro de um código em C. Normalmente, utiliza-se para separar partes lógicas de um código.

As linhas 7 e 8 recebem dois valores do teclado, através da função `scanf()`.

As próximas 4 linhas calculam o valor da variável `c`, dependendo da comparação entre `a` e `b`, feita pelo bloco **if-else**.

A linha 13 imprime o resultado na tela, através da função `printf()`.

O fecho-chaves, `}`, na última linha encerra a função.

1.3 Tipos

A linguagem C possui 5 tipos básicos. São os tipos **char**, **int**, **float**, **double** e **void**. A tabela abaixo apresenta algumas propriedades de cada tipo.

Tipo	Descrição	Tamanho	Intervalo
char	caractere	1 bytes	-128 a 127 ou 0 a 255
int	inteiro	2 ou 4 bytes	-32768 a 32767 ou -214783648 a 214783647
float	ponto flutuante	4 bytes	-1.7E38 a 1.7E38 (precisão de 6 dígitos)
double	ponto flutuante de dupla precisão	8 bytes	-1.7E38 a 1.7E38 (precisão de 16 dígitos)
void	vazio	0 bytes	-

O tamanho do tipo inteiro varia com o compilador utilizado. Este tipo possui ainda três variações, a seguir:

Tipo	Descrição	Tamanho	Intervalo
<code>long int</code>	inteiro longo	4 bytes	-214783648 a 214783647
<code>short int</code>	inteiro curto	2 bytes	-32768 a 32767
<code>unsigned int</code>	inteiro sem sinal	2 ou 4 bytes	0 a 65535 ou 0 a 4294967295

1.4 Variáveis

Em C, todas as variáveis precisam ser declaradas. A declaração tem a forma

tipo nome-da-variável

ou

tipo nome-da-variável1, da-variável2, ...

onde *tipo* é o tipo da variável e *nomes-das-variáveis* são separadas por vírgulas.

Os nomes das variáveis devem começar com uma letra ou o sublinhado ('A' a 'Z', 'a' a 'z' e '_'). O restante do nome pode ser composto por letras, sublinhado ou números. Também não são permitidos como nomes de variáveis palavras reservadas pela linguagem C. A tabela abaixo fornece alguns nomes de variáveis válidos e não-válidos.

Nomes válidos	Nomes não válidos
ABC	/ABC
tomate	int
_g	g*
agua_do_mar	agua-do-mar
carro5000	5000carro

1.5 Constantes

Existem diversos tipos de constantes: inteira, ponto flutuante, caractere e cadeia de caracteres.

Constante numérica	Significado
10	constante inteira
017	constante octal
0xFF, 0XF0	constante hexadecimal
64L	constante longa
78678537	constante longa (implícito)
74.1, 1., .5	constante de ponto flutuante

Constantes de caractere são representadas entre apóstrofos (') e equivalem ao número pelo qual o caractere é representado na máquina. A maioria das máquinas utiliza a representação ASCII (American Standard Code for Information Interchange). Para permitir portabilidade, constantes de caractere devem ser utilizadas no lugar de seus equivalentes inteiros.

Constante de caractere	Valor ASCII
'A'	65
'Z'	90
'='	61

Caracteres especiais são representados com a barra invertida (\) seguida de um determinado caractere.

Constante de caractere	Caractere representado
'\n'	caractere de mudança de linha (LF)
'\r'	caractere de retorno de carro (CR)
'\t'	caractere de tabulação (TAB)
'\\'	caractere de barra invertida
'\0'	caractere nulo
'\"'	caractere apóstrofo
'\"'	caractere aspas

O caractere nulo ('\0') é colocado à direita da cadeia, indicando o seu final.

Cadeia	Significado	Tamanho
"ABCDE"	Cadeia de caracteres armazenando os caracteres 'A', 'B', 'C', 'D', 'E' e '\0'	6 bytes
""	Cadeia de caracteres armazenando o caractere '\0'	1 byte

1.6 Entrada e Saída Básicas

Nesta seção são apresentadas duas funções que serão utilizadas ao longo dos exercícios propostos no curso. São as funções printf() e scanf(). O conceito de funções será detalhado no Capítulo 4.

1.6.1 A Função printf()

A função printf() é uma das funções de E/S (entrada e saída) que podem ser usadas em C. Ela não faz parte da definição da linguagem C, sendo incluída em uma biblioteca (*stdio.h*) fornecida juntamente com os compiladores. Esta função serve para apresentar na tela uma expressão definida pelo usuário, e segue a sintaxe

```
printf("expr. de controle", argumento1, argumento2, ...),
```

onde *expr. de controle* é uma expressão definida, que pode conter alguns códigos, apresentados na tabela a seguir. Quando a função `printf()` encontra um destes códigos, ela o substitui pelo argumento fornecido. Os argumentos podem ser nenhum ou quantos argumentos se fizerem necessários.

Código <code>printf()</code>	Formato
<code>%c</code>	caractere simples
<code>%d</code>	decimal
<code>%e</code>	notação científica
<code>%f</code>	ponto flutuante
<code>%g</code>	<code>%e</code> ou <code>%f</code> (o mais curto)
<code>&o</code>	octal
<code>%s</code>	cadeia de caracteres
<code>%u</code>	decimal sem sinal
<code>%x</code>	hexadecimal
<code>%ld</code>	decimal longo
<code>%lf</code>	ponto flutuante longo (double)

Exemplos:

```
printf("Teste geral");
Saída: Teste geral
```

```
printf("Esta casa tem %d quartos\n", 2);
Saída: Esta casa tem 2 quartos
```

```
printf("Nome: %s\nSexo: %c\nIdade: %d\n", "Pedro", 'M', 18);
Saída: Nome: Pedro
Sexo: M
Idade: 18
```

1.6.2 A Função `scanf()`

A função `scanf()` é outra das funções de E/S (entrada e saída) que podem ser usadas em C. Lê do teclado dados e coloca os valores fornecidos pelo usuário nas variáveis utilizadas como parâmetro da função. Sua sintaxe é

```
scanf("expr. de controle", &argumento1, &argumento2, ...),
```

onde a expressão de controle utiliza os mesmos códigos da função `printf()`.

Exemplo:

```
int i, j;
float f;
char c;

scanf("%d%d", &i, &j);
scanf("%f", &f);
scanf("%c", &c);
```

O operador de endereço (&), que precede os argumentos da função, retorna o primeiro byte ocupado pela variável na memória do computador, e será detalhado no capítulo de ponteiros.