

# ***LÓGICA PARA REDES DE COMPUTADORES***

## **CAPÍTULO 3 - COMO CRIAR ALGORITMOS UTILIZANDO A LINGUAGEM DE PROGRAMAÇÃO C?**

Ana Paula da Costa Cardoso

# Introdução

Para quem é iniciante em programação, há perguntas que ficam o tempo todo rondando a mente. De que forma fazer meu algoritmo rodar? Como transformar estes comandos em um programa que o usuário possa usar? Às vezes, o anseio de construir seu primeiro programa atrapalha um pouco no aprendizado da lógica. Mas, pode se tranquilizar que estamos prontos para dar o primeiro passo: escolher a linguagem de programação. Mas qual linguagem utilizar?

Existem infinitas linguagem disponíveis, tais como: C, C++, C#, Pascal, Fortran e Java. E na medida que você for se especializando, o tipo de aplicação também definirá o tipo de linguagem a ser utilizada. Mas não se preocupe, porque é fácil migrar de uma linguagem para outra, se você compreender bem a lógica de programação.

Agora, vamos focar em aprender a linguagem de programação C! Primeiro, você deve entender a estrutura da linguagem, os tipos básicos de dados e alguns comandos de entrada e saída de dados. Neste momento, você já conseguirá rodar seus programas interagindo com o usuário.

O próximo passo é conhecer as formas do comando condicional em linguagem C e desenvolver programas utilizando estas formas. E para finalizar este capítulo, iremos compreender os três tipos de comandos de repetição, o que permitirá a construção de programas mais sofisticados.

Vamos conhecer a linguagem de programa C.

Acompanhe com atenção e bons estudos!

## 3.1 Introdução à linguagem de programação C

A linguagem de programação C é de alto nível e compilada. De alto nível significa que o programador contará com comandos parecidos com o idioma da língua inglesa e compilada, implica que, depois de escrito o programa em um editor de texto, será necessário um programa tradutor, chamado de compilador, que converte esta linguagem de alto nível em uma linguagem de baixo nível, linguagem de máquina (DEITEL; DEITEL, 2011).

Então, quando estamos falando de uma linguagem, inicialmente, precisamos conhecer seus comandos, sua estrutura e qual ferramenta precisamos para desenvolver o programa.

Neste tópico, você vai conhecer um pouco da história da Linguagem C, sua estrutura, os tipos de dados e alguns comandos de entrada e saída de dados.

### 3.1.1 História da linguagem de programação C

O cientista da computação Denis Ritchie (1941-2011) se inspirou em duas outras linguagens, B e BCPL, para desenvolver a linguagem C, em 1972, nas dependências da

*Bell Telephone Laboratories.* Denis tinha o objetivo de escrever o sistema operacional Unix.



Figura 1 - Transformando os algoritmos em programas.

Fonte: Modella, Shutterstock, 2019.

A linguagem C passou a ser conhecida por todos os programadores, devido à disseminação do uso do Unix, fazendo com que diversas organizações desenvolvessem várias versões para a linguagem C. Porém, todas essas versões implicam problemas de portabilidade.

Assim, para resolver isso, a ANSI (*American National Standards Institute*) criou um comitê em 1983, para estabelecer um padrão para a linguagem C, utilizado até a atualidade (DAMAS, 2007).

---

## VOCÊ QUER VER?

O filme *Transcendence: a revolução* (PFISTER; PAGLEN, 2014) é interessante, para abrir os horizontes sobre como uma rede de computadores pode interferir no mundo. Na trama, um cientista especializado em inteligência artificial realiza experiências com intenção de criar robôs com emoções humanas. Este tipo de experimento causa bastante controvérsias, o que põe sua vida em risco. Ao sofrer um atentado, ele convence a esposa e seu melhor amigo a usá-lo como cobaia em seu próprio experimento.

---

Agora, vamos entender mais sobre como funciona a estrutura de programas no C.

### 3.1.2 Estrutura de Programa

Um programa em linguagem C pode conter uma ou mais funções, sendo obrigatória a presença da função *main*, cuja forma é:

```
tipo nome_funcao (parametros)
{
    //declaracao de variaveis
    //linhas de codigo
}
```

Este formato não se refere somente à função *main*, mas é um formato geral de funções. A figura a seguir ilustra a estrutura de um programa em C.

Observe a figura e perceba que temos um espaço reservado para incluir bibliotecas. Não se preocupe, daqui a pouco você entenderá para que servem estas bibliotecas. Logo em seguida, vemos a função *main* seguida de duas chaves ({}), uma de abertura, linha 2, e a outra de fechamento, linha 5. Estas chaves indicam onde começam e onde terminam as linhas de código que pertencem à função principal. Neste exemplo, o tipo foi omitido e esta omissão significa que é do tipo *int*.

Este é um programa escrito em linguagem de programação C, e o que ele faz? Nada,

absolutamente nada.

```
1 /* incluir bibliotecas*/
2 main()
3 {
4     /* declarar variaveis */
5
6
7     //linhas de comando
8     //linhas de comando
9 }
```

The diagram illustrates the structure of a C program. On the left, line numbers 1 through 9 are listed vertically. Lines 1, 2, 4, 7, and 8 contain text in blue, while lines 3, 5, 6, and 9 are empty. To the right of the lines, a tree diagram shows a root node at level 3, which branches down to levels 4 and 5. Level 4 further branches into levels 7 and 8. Level 5 is a single node. Level 6 is also a single node.

Figura 2 - Estrutura de um programa em C.

Fonte: Elaborada pela autora, 2019.

Você deve estar com muita ansiedade para escrever seu primeiro programa, né? Como foi dito, anteriormente, você precisará de um ambiente para escrever seus programas. Este ambiente pode ser um editor de texto ou uma IDE (*Integrated Development Environment*), ou seja, um ambiente de desenvolvimento integrado.

Existem várias IDEs robustas, feitas para programação em grande escala. Mas, para quem está iniciando o melhor é uma ferramenta mais simples. Eu sugiro o Code::Blocks ou o DevC++. São duas ferramentas muito populares e utilizadas para quem está iniciando na arte de programar. É abundante a quantidade de material e sites de discussão sobre o assunto.

Agora, você irá preparar a sua máquina para desenvolver este capítulo. Faça uma busca na internet, baixe a sua ferramenta, instale de forma correta e volte para continuarmos o nosso estudo.

---

## VOCÊ QUER LER?

Manzano (2015) traz uma boa explicação sobre os tipos de compiladores C e algumas IDEs disponíveis no mercado no livro “Estudo dirigido de linguagem C acompanhada de uma xícara de café”. Ele mostra de forma clara e objetiva a instalação do Code::Block, Visual Studio, Xcode e Orwell DevC++. Além de deixar detalhado o processo do compilador C. O livro é um bom auxílio para você criar seu ambiente de trabalho e está disponível na biblioteca virtual: <<https://Animabrasil.blackboard.com/webapps/ga-bibliotecaSSO-BBLEARN/homeMinhaBiblioteca>> (<https://Animabrasil.blackboard.com/webapps/ga-bibliotecaSSO-BBLEARN/homeMinhaBiblioteca>)>.

---

Não se preocupe se você não possui uma máquina em casa para treinar. Escreva seus programas em uma folha e quando for estudar, digite seus programas em um editor de texto, ou abra a ferramenta disponível na máquina para rodar programas em C. Se a máquina que você está usando para estudar, não possuir uma ferramenta específica, procure na internet *sites* para compilar programas em C *online*. Existem várias opções, basta escolher uma.

### **3.1.3 Tipos de dados, constantes e variáveis**

Os dados da linguagem C são tipados, ou seja, toda vez que você for manipular dados será necessário especificar o tipo de dado. Existem cinco tipos pré-definidos: *int*, *char*, *float*, *double* e *void*. Além dos tipos pré-definidos, a linguagem C possui modificadores, tais como: *short*, *long*, *signed* e *unsigned*.

O quadro a seguir mostra os principais tipos de dados e suas características. Observe que cada tipo possui um tamanho em *bytes*, isto significa o limite da faixa de valores a ser utilizado.

Tipo	Descrição	Tamanho em Bytes	Faixa de valores
char	Caracter	1	Um caractere será representado entre aspas simples. Por exemplo: 'a'
int	inteiro	2 ou 4	-32.768 a 32.767
short int	Inteiro curto	2	-32.768 a 32.767
long int	Inteiro longo	4	-2.147.483.648 a 2.147.483.647
unsigned int	Inteiro sem sinal	2 ou 4	0 a 65.535
signed int	Inteiro com sinal	2	-32.768 a 32.767
signed short int	Inteiro curto com sinal	2	-32.768 a 32.767
unsigned short int	Inteiro curto sem sinal	2	0 a 65.535
signed long int	Inteiro longo com sinal	4	-2.147.483.648 a 2.147.483.647
unsigned long int	Inteiro longo sem sinal	4	0 a 4.294.967.295
float	Ponto flutuante com precisão simples	4	3.4 E-38 a 3.4E+38
double	Ponto flutuante com precisão simples	8	1.7 E-308 a 1.7E+308
long double	Ponto flutuante com precisão dupla longo	16	3.4E-4932 a 1.1E+4932
void			vazio

Quadro 1 - Tipos de dados em linguagem C para compiladores de 32 bits.

Fonte: Elaborado pela autora, 2019.

Por exemplo, um tipo *unsigned int* que possui 2 bytes (16 bits). Um campo com 16 bits pode ter  $2^{16} = 65.536$  possibilidades de representação, para números inteiros sequenciais podemos representar os números de 0 a 65.535. Isso significa que se declararmos uma variável como *unsigned int* e tentarmos guardar um valor de 66.000, o programa retornaria com um erro. Observe a figura a seguir.

```
1 #include<iostream>
2 using namespace std;
3 main()
4 {
5     unsigned int z;
6     z=4294967296;
7     cout<< "Tamanho do tipo:"<<sizeof(unsigned int) ;
8     cout<<"\n"<<z;
9 }
```

```
Tamanho do tipo:4
0
-----
Process exited after 6.186 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 3 - Prog01 - Exemplo de tamanho em bytes do tipo *unsigned int*.

Fonte: Elaborada pela autora, 2019.

Uma forma de você saber quantos *bytes* cada tipo possui na máquina que está trabalhando é utilizar o *sizeof*. Observe que nesta máquina de 64 *bits*, o *unsigned int* tem o tamanho de 4 *bytes*, o que significa que pode armazenar de 0 a 4.294.967.295. No programa acima, foi utilizado o número 4.294.967.296. Veja o que ele imprimiu na tela para o usuário, o valor de zero.

---

## VOCÊ SABIA?

Você sabe quem inventou a internet? Foi o Departamento de Defesa dos EUA, no final da década de 1960. A ideia era construir uma rede que fosse capaz de sobreviver a uma destruição parcial, já que na época, a Guerra Fria entre os EUA e a antiga União Soviética estava em seu auge e a ameaça de uma guerra nuclear era bastante presente (FESCINA, 2016). Leia este artigo sobre a história de criação da internet acessando o site <<https://super.abril.com.br/mundo-estranho/quem-inventou-a-internet/>>.

---

Como já vimos, os principais tipos de dados no C, agora podemos estudar como são declaradas as variáveis.

Para declarar variáveis basta colocar o tipo e depois as variáveis separadas por vírgula. Para finalizar, coloque um ponto e vírgula.

**tipo nome\_variavel, nome\_variavel2;**

Agora, vamos passar algumas observações importantes. Clique na interação a seguir.

A linguagem de programação C é *case sensitive*, o que significa que ele faz diferenciações entre maiúsculas e minúsculas. Por exemplo, a variável *nome* é diferente das variáveis *Nome* e *NOME*.

Declare as variáveis logo após a chave de abertura da função *main*.

É possível atribuir um valor a uma variável na declaração. Por exemplo:  
***int x = 0, y = 10.***

Nome das variáveis:

- não utilize acentuação;
- não podem começar por dígitos;
- podem conter letras do alfabeto, dígitos e o caractere *underscore* (\_);
- não pode utilizar as palavras-chave da linguagem, por exemplo: *int*, *char*, *for* etc.

Além das variáveis, o C permite a criação de constantes. Para criar uma constante, use a diretiva `#define`:

**#define nome\_constante valor**

Por exemplo, `#define PI 3.1415`

As constantes são declaradas fora e antes da função *main*. Também existe um padrão que requer que o nome das constantes sejam escritos com letras maiúsculas.

### **3.1.4 Operadores lógicos e matemáticos**

O comando de atribuição em C é o sinal de `=`, no qual a variável fica no lado esquerdo e a expressão fica no lado direito.

O quadro a seguir mostra os principais operadores matemáticos e lógicos, utilizados em C.

Operador	Descrição
+	Soma
*	Produto
/	Divisão
%	Resto da divisão de inteiros
&&	Operador lógico E
	Operador lógico OU
!	Operador lógico NÃO
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
!=	Diferente
== (dois sinais de igual)	Igual
//	Comentário de linha
/* */	Comentário de parágrafo
++	É um operador de incremento. Por exemplo, $i = i + 1$ , pode ser escrito como $i++$
--	É um operador de decremento. Por exemplo, $i = i - 1$ , pode ser escrito como $i--$

Quadro 2 - Principais operadores matemáticos e lógicos utilizados em C.

Fonte: Elaborado pela autora, 2019.

O quadro a seguir mostra os operadores de atribuição que podem ser utilizados de forma simplificada.

Operador	Significado	Exemplo		
$+=$	soma e atribui	$x += y$	igual	$x = x + y$
$-=$	subtrai e atribui	$x -= y$	igual	$x = x - y$
$*=$	multiplica e atribui	$x *= y$	igual	$x = x * y$
$/=$	divide e atribui quociente	$x /= y$	igual	$x = x / y$
$\%=$	divide e atribui resto	$x \%= y$	igual	$x = x \% y$
$\&=$	E bit a bit e atribui	$x \&= y$	igual	$x = x \& y$
$ =$	OU bit a bit e atribui	$x  = y$	igual	$x = x   y$
$^=$	OU exclusivo e atribui	$x ^= y$	igual	$x = x ^ y$
$<<=$	desloca à esquerda e atribui	$x <<= y$	igual	$x = x << y$
$>>=$	desloca à direita e atribui	$x >>= y$	igual	$x = x >> y$

Quadro 3 - Operadores de atribuição simplificada.

Fonte: BACKES, 2013, p. 62.

**Importante!**

- Em qualquer operação que utilize inteiros, o retorno da expressão será um número inteiro (DAMAS, 2007). Por exemplo:  
 $5/4 = 1$ .
- Para números reais, *float* ou *double*, deve-se utilizar o ponto em pelo menos um dos operandos (DAMAS, 2007). Por exemplo:  
 $5/4.0 = 1.25$ .

---

## VOCÊ QUER LER?

Para trabalhar com cadeia de caracteres no C, será necessário utilizarmos funções específicas para a sua manipulação. O artigo “Cadeia de Caracteres - Funções de Entrada e Saída”, de Elaine Gatto, mostra como utilizar vários comandos de entrada e saída de dados responsáveis pela leitura e manipulação das cadeias de caracteres. Algumas funções que ela detalha: *gets*, *fgets*, *sscanf*, *putchar*, entre outras (GATTO, 2017). Acesse o artigo em: <<https://www.embarcados.com.br/cadeia-de-caracteres-funcoes-de-entrada-e-saida/>> (<https://www.embarcados.com.br/cadeia-de-caracteres-funcoes-de-entrada-e-saida/>)>.

---

Agora que já vimos alguns conceitos básicos sobre o C, vamos conhecer alguns comandos de entrada e saída de dados.

### 3.2 Comandos de entrada e saída de dados

A linguagem C possui vários comandos de entrada e saída de dados, tais como *printf*, *putc* *scanf*, *getc* entre outros, além do *cin* e *cout* do C++. Cada comando tem a sua utilidade e aplicação específicas. Os comandos mais utilizados para saída de dados são o *printf* da linguagem C e *cout* do C++, e os mais utilizados para entrada de dados são: *scanf* do C e o *cin* do C++.

Lembra das bibliotecas? Pois bem, em C, como em outras linguagens de programação, podemos inserir arquivos com funções pré-definidas. Estes arquivos recebem o nome de bibliotecas. Em C para incluir uma biblioteca utilizamos a diretiva `#include`.

Vamos conhecer a estrutura dos comandos de saída *printf* e *cout* e dos comandos de entrada, *scanf* e *cin*, e suas bibliotecas a seguir.

#### 3.2.1 Comandos de saída de dados

O primeiro comando de saída de dados que iremos aprender é o *printf*, que significa saída formatada. Ele imprime valores no vídeo para o usuário.

O *printf* possui dois parâmetros:

```
printf("tipos_de_saida", lista_de_variaveis);
```

Cada um dos termos é apresentado na interação a seguir. Clique para ver.

<b>Tipos de saída</b>	São caracteres que especificam o tipo de dados que serão mostrados nas variáveis.
<b>Lista de variáveis</b>	São as variáveis cujos valores serão impressos no vídeo para o usuário. Elas devem ser separadas por vírgula.

O quadro a seguir mostra alguns caracteres e seus respectivos tipos de dados a serem utilizados no *printf*.

Alguns tipos de saída	
%c	Escrita de um caractere ( <b>char</b> )
%d ou %i	Escrita de números inteiros ( <b>int</b> ou <b>char</b> )
%u	Escrita de números inteiros sem sinal ( <b>unsigned</b> )
%f	Escrita de números reais ( <b>float</b> ou <b>double</b> )
%s	Escrita de vários caracteres
%p	Escrita de um endereço de memória
%e ou %E	Escrita em notação científica

Quadro 4 - Alguns caracteres para os tipos de saída.

Fonte: BACKES, 2013, p. 36.

Para o *printf* utilizaremos a biblioteca padrão do C: stdio.h, onde seu nome vem de standard input-output header que significa cabeçalho padrão de entrada e saída.

Observe o exemplo a seguir, na figura abaixo. Na linha 6, temos um *printf* simplificado, ou seja, sem nenhuma variável, com apenas a mensagem. Já na linha 7, temos o *printf* com dois tipos de dados, o tipo inteiro e o real. Para o tipo inteiro utiliza-se o %d e para o real, o %f. O escape \n foi utilizado para pular linhas. Preste atenção na tela de saída.

```
1 #include<stdio.h>
2 main()
3 {
4     int x = 2;
5     float y = 5.2;
6     printf("Ola\n");
7     printf("Valor de x = %d \n Valor de y %f", x, y);
8 }
```

```
Ola
Valor de x = 2
Valor de y 5.200000
-----
Process exited after 5.72 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 4 - Prog02: Exemplo do printf.  
Fonte: Elaborada pela autora, 2019.

A linguagem C possui constantes pré-definidas que permitem o envio de caracteres de controle para o vídeo. O quadro a seguir ilustra alguns destas constantes, chamadas de escape.

Código	Comando
\a	Som de alerta (bip)
\b	Retrocesso (backspace)
\n	Nova linha (new line)
\r	Retorno de carro (carriage return)
\v	Tabulação vertical
\t	Tabulação horizontal
\'	Apóstrofe
\"	Aspa
\\\	Barra invertida (backslash)
\f	Alimentação de folha (form feed)
\?	Símbolo de interrogação
\0	Caractere nulo (cancela a escrita do restante)

Quadro 5 - Alguns caracteres de escape ou barra invertida.

Fonte: BACKES, 2013, p. 47.

O próximo comando é o *cout*, do C++. Ele utiliza a biblioteca *iostream* e o operador de inserção `<<`. Para utilizar a biblioteca *iostream* é necessário incluir o comando *using namespace std*, logo abaixo das bibliotecas.

Analise o exemplo a seguir, na próxima figura. São utilizados os dois comandos, *printf* e *cout*. O *cout* não usa parâmetros, apenas o operador `<<` para separar as constantes das variáveis. Na linha 11, o caractere para informar o tipo de dado *float*, está como `%.3f`. Este `.3` entre o `%` e o `f`, significa o número de casas após a vírgula.

```

1 #include<iostream>
2 #include<stdio.h>
3 using namespace std;
4 main()
{
5     int x = 2;
6     float y = 5.2;
7     cout<<"Ola\n";
8     cout<<"Valor de x = " << x <<"\n Valor de y = "<< y << "\n";
9     printf("Ola\n");
10    printf("Valor de x = %d \n Valor de y %.3f", x, y);
11 }
12 
```

```

Ola
Valor de x = 2
Valor de y = 5.2
Ola
Valor de x = 2
Valor de y 5.200
-----
Process exited after 8.149 seconds with return value 0
Pressione qualquer tecla para continuar. . .

```

Figura 5 - Prog03: Exemplos de cout e printf.

Fonte: Elaborada pela autora, 2019.

Vamos conhecer os comandos de entrada de dados *scanf* e *cin*.

### 3.2.2 Comandos de entrada de dados

O comando de entrada de dados *scanf*, entrada formatada, recebe um conjunto de valores lidos do teclado. A biblioteca para utilizá-lo é a mesma do *printf*, stdio.h. Ele possui dois campos, uma para identificar os tipos de dados que serão lidos e o outro é para identificar as variáveis (BACKES, 2013). A sua sintaxe é:

**scanf(“tipos\_entrada”, variaveis);**

Vamos entender cada um dos termos na interação a seguir. Clique para ver.

<b>Tipos_e ntrada</b>	São caracteres que especificam o tipo de dados que serão lidos.
<b>Variávei s</b>	São as variáveis onde serão armazenados os dados lidos, separadas por vírgula e precedidas do caractere &.

Veja o exemplo na figura a seguir, de um programa para somar dois números utilizando o *scanf*.

```
1 #include<stdio.h>
2 main()
3 {
4     int x, y, soma;
5     printf("Digite dois numeros\n");
6     scanf("%d %d", &x, &y);
7     soma = x + y;
8     printf("A soma de %d com %d e %d", x, y, soma);
9 }
```

```
Digite dois numeros
5
10
A soma de 5 com 10 e 15
-----
Process exited after 23.22 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 6 - Prog04: Exemplo utilizando o *scanf*.

Fonte: Elaborada pela autora, 2019.

O próximo comando de entrada de dados é o *cin*. Ele utiliza o operador *>>* e a sua biblioteca é a mesma do *cout*, iostream.

Vamos fazer o exemplo da soma de dois números com este comando. Veja a figura a seguir. Observe que para o usuário não faz a menor diferença, aliás, ele nem sabe o que está acontecendo, qual o comando que você está utilizando.

```
1 #include<stdio.h>
2 #include<iostream>
3 using namespace std;
4 main()
5 {
6     int x, y, soma;
7     printf("Digite dois numeros\n");
8     scanf("%d %d", &x, &y);
9     soma = x + y;
10    printf("A soma de %d com %d e %d\n", x, y, soma);
11
12    cout<<"Digite dois numeros\n";
13    cin >> x >> y;
14    soma = x + y;
15    cout<<"A soma de "<< x << " com " << y << " e " << soma;
16 }
```

```
Digite dois numeros
12
-30
A soma de 12 com -30 e -18
Digite dois numeros
12
-30
A soma de 12 com -30 e -18
-----
Process exited after 112.9 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 7 - Prog05 - A soma de dois números usando o printf/scanf e cout/cin.

Fonte: Elaborada pela autora, 2019.

Vamos a mais um exemplo! Para calcular a área de um círculo utilizamos a fórmula  $A = \pi r^2$ , onde  $r$  é o raio da circunferência. O programa que calcula a área de um círculo é ilustrado pela figura a seguir.

```
1 #include<stdio.h>
2 #include<math.h>
3 #define PI 3.1416
4 main()
5 {
6     float raio, area;
7     printf("Digite o valor do raio\n");
8     scanf("%f", &raio);
9     area = PI * pow(raio, 2);
10    printf("A area e %f", area);
11 }
```

Digite o valor do raio  
1.5  
A area e 7.068600  
-----  
Process exited after 9.727 seconds with return value 0  
Pressione qualquer tecla para continuar. . .

Figura 8 - Programa para calcular a área de um círculo.

Fonte: Elaborada pela autora, 2019.

Clique na interação a seguir para ver algumas observações a respeito deste programa.

Foi utilizado duas bibliotecas, a stdio por causa dos comandos *printf*/*scanf* e a biblioteca math.h por causa da função pow, linhas 1 e 2.

A função pow calcula a potência de um número, ou seja,  $\text{pow}(a, b) = a^b$ , linha 10.

A constante PI foi definida com o valor de 3.1416, linha 4. Este valor não poderá ser alterado neste programa.

Para números reais (*float*) deve-se utilizar o ponto e não a vírgula.

Início da função *main*, linha 5 e término na linha 12.

Muito parecido com o algoritmo, né? Por isso a lógica é tão importante. Na linguagem, a IDE que você estiver utilizando corrige erros de sintaxe, mas não erros de lógica. Experimente apagar o ponto e vírgula, por exemplo, da linha 7. O programa não irá compilar, irá aparecer um erro informando que falta o finalizador de comando.

Vamos brincar com a brincadeira dos sete erros? Analise o programa para calcular a média ponderada das notas de um concurso. Serão quatro provas, sendo a primeira com peso 1, a segunda com peso 2, a terceira com peso 3 e a quarta com peso 4. Então, para calcular a média final basta usar a seguinte fórmula:

$$\text{media} = \frac{(1 \times N1 + 2 \times N2 + 3 \times N3 + 4 \times N4)}{(1+2+3+4)} = \frac{(1 \times N1 + 2 \times N2 + 3 \times N3 + 4 \times N4)}{(10)}$$

O programa apresenta sete erros de sintaxe, ou seja, apenas de erros no uso dos comandos. Não olhe as respostas, tente localizar cada um dos erros.

```

1 #include<iostream>
2 main()
3 {
4     float n1, n2, n3, n4, media
5     printf("Digite as quatro notas\n");
6     scanf("%f %d %f %f", &n1, &n2, &n3, &n4);
7     media == (n1 + 2*n2 + 3*n3 + 4*xn4)/10;
8     printf("A media e %f", Media);
9 }
```

Figura 9 - Prog07: Programa para calcular a média ponderada de 4 notas, mas com sete erros.

Fonte: Elaborada pela autora, 2019.

Resposta:

- Linha 1: a biblioteca que o *printf*/*scanf* utilizam é a *stdio.h* e não a *iostream*;

- Linha 4: falta um ponto e vírgula no final da linha;
- Linha 5: falta fechar o parêntese do *printf*;
- Linha 6: todas as quatros notas são do tipo *float*, então o tipo de entrada correto é *%f*.
- Linha 7: o comando de atribuição é *=* e não *==*;
- Linha 7: o operador matemático de multiplicação é o *\** e não o *x*. No caso, ele iria acusar que a variável *x* não foi definida.
- Linha 8: como o C é *case sensitive*, a variável *media* é diferente da variável *Media*.

Estes erros são os mais comuns para os iniciantes e confesso, que as vezes, para os mais experientes também.

Veja como é o programa corrigido, próxima figura. Faça uma análise linha a linha, comparando os dois programas, o correto, e o com erros.

The screenshot shows a terminal window with the following content:

```
1 #include<stdio.h>
2 main()
3 {
4     float n1, n2, n3, n4, media;
5     printf("Digite as quatro notas\n");
6     scanf("%f %f %f %f", &n1, &n2, &n3, &n4);
7     media = (n1 + 2*n2 + 3*n3 + 4*n4)/10;
8     printf("A media e %f", media);
9 }
```

D:\ArquivosPessoais\DtCom\CAP03\prog08.exe

```
Digite as quatro notas
5
6
8
10
A media e 8.100000
-----
Process exited after 17.25 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 10 - Prog08: Programa para calcular a média ponderada de 4 notas.

Fonte: Elaborada pela autora, 2019.

Antes de prosseguirmos para o próximo tópico, passe todos os algoritmos que você fez para a linguagem C. É a melhor forma de treinar e compreender a sua lógica. Então, agora vamos ver como o C trabalha com o comando condicional.

### 3.3 Estrutura de Seleção com C

A linguagem C possui várias estruturas de seleção ou o famoso comando condicional SE. Em C o SE tornar-se *if* e pode ser utilizado de diversas formas: simples, composto, homogêneo e heterogêneo. Simples será na utilização do comando apenas com o bloco do ENTÃO, ou seja, somente quando a condição retornar verdadeira. Cabe lembrar, que no C, o comando condicional *if* não possui a palavra-chave equivalente ao ENTÃO. Composto ou completo, é quando se tem os blocos, tanto, do ENTÃO, quanto, do SENÃO. Depois vêm os homogêneos e heterogêneas, que nada mais é do que *ifs* aninhados, ou seja, um dentro do outro. Um outro comando de seleção é o *switch/case* que é o nosso CASO em algoritmo.

Vamos começar pelo comando condicional *if* e suas variações.

### 3.3.1 Estrutura de Seleção: simples, composta e homogênea e heterogênea

O comando condicional SE, em linguagem C, é o comando *if* e sua sintaxe, na forma simples, é:

```
if (condicao)
{
    Linhas de codigo;
}
```

Onde:

- o *if* é uma palavra-chave;
- a palavra chave *then*, ENTÃO do SE, não existe na linguagem C;
- condição é uma expressão lógica que retornará falso ou verdadeiro;
- chaves: indicam onde começam e onde terminam as linhas de código, que serão executadas caso a condição retorne verdadeira;
- as chaves são opcionais, caso se tenha APENAS uma linha de código; esta observação serve para todos os comandos do C.

---

## VOCÊ SABIA?

A linguagem C tem uma forma simplificada de permitir o uso do *if/else* por meio do operador? (ponto de interrogação), sendo ele um operador ternário. Sua forma geral é ***condição?verdadeiro : falso;*** este operador é limitado, mas em certas situações, ele se torna muito prático. Por exemplo, media  $\geq 7$ ? “aprovado”: “reprovado”, se a media for maior ou igual a 7, o programa irá imprimir aprovado, senão irá imprimir reprovado (CARVALHO FILHO, 2016). Entenda mais sobre este operador, acessando o artigo “Operador Ternário em C Curso C”, disponível em: <<http://excript.com/linguagem-c/operador-ternario-c.html>>.

---

Observe o programa a seguir, próxima figura, onde temos um exemplo de omissão das chaves. Como é apenas uma linha de código, no caso o *printf*, a saída será a mesma.

```
1 #include<stdio.h>
2
3 main()
4 {
5     int num;
6
7     printf("Digite um numero \n");
8     scanf("%d", &num);
9     if(num>3)
10    {
11         printf("\n %d e maior que 3", num);
12     }
13     if(num>3)
14         printf("\n %d e maior que 3", num);
15
16 }
```

D:\ArquivosPessoais\DtCom\CAP03\prog09.exe

Digite um numero  
5

5 e maior que 3  
5 e maior que 3

-----

Process exited after 12.37 seconds with return value 0

Pressione qualquer tecla para continuar. . .

Figura 11 - Prog09: Exemplo de *if* com e sem as chaves.

Fonte: Elaborada pela autora, 2019.

Mas, agora, observe o próximo programa, na figura a seguir, consegue perceber o erro? Olhe a tela de saída, o usuário digitou o número 5, então deveria aparecer para ele apenas as mensagens das linhas 11 e 12, "Voce digitou o numero 5" e "Ele e maior que 3". Mas, apareceu a mensagem da linha 16, "Ele e menor que 3". Isso aconteceu porque no *if* da linha 14, tem-se duas linhas de código, linhas 15 e 16, mas não foi usada as chaves, onde o programa entende que se tem apenas uma única linha de código, no exemplo, a linha 15.

```
1 #include<stdio.h>
2
3 main()
4 {
5     int num;
6
7     printf("Digite um numero \n");
8     scanf("%d", &num);
9     if(num>3)
10    {
11        printf("\n Voce digitou o numero %d", num);
12        printf("\n Ele e maior que 3");
13    }
14    if(num<3)
15        printf("\n Voce digitou o numero %d", num);
16        printf("\n Ele e menor que 3");
17 }
```

D:\ArquivosPessoais\DtCom\CAP03\prog10.exe

```
Digite um numero
5

Voce digitou o numero 5
Ele e maior que 3
Ele e menor que 3
-----
Process exited after 24.1 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 12 - Prog10: programa com erro por causa da falta das chaves.

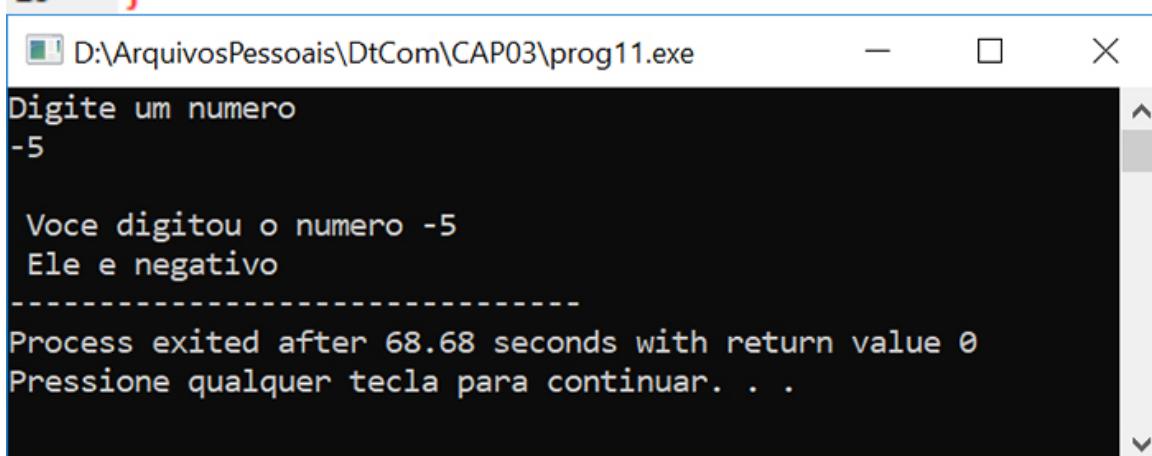
Fonte: Elaborada pela autora, 2019.

O comando condicional *if* composto é aquele onde temos o ELSE, o mesmo que SENÃO do SE. Assim, se a condição retornar verdadeira, irá executar o bloco do *então*, pois no C, é ausente a palavra *then*, se retornar falso irá executar o bloco do *else*. Sua sintaxe é:

```
if (condicao)
{
    Linhas de codigo;
}
else
{
    Linhas de codigo;
}
```

A próxima figura mostra um programa para verificar se um número é positivo ou negativo utilizando o *if* composto.

```
1 #include<stdio.h>
2
3 main()
4 {
5     int num;
6
7     printf("Digite um numero \n");
8     scanf("%d", &num);
9     if(num>=0)
10    {
11        printf("\n Voce digitou o numero %d", num);
12        printf("\n Ele e positivo");
13    }
14    else
15    {
16        printf("\n Voce digitou o numero %d", num);
17        printf("\n Ele e negativo");
18    }
19 }
```



```
D:\ArquivosPessoais\DtCom\CAP03\prog11.exe
Digite um numero
-5

Voce digitou o numero -5
Ele e negativo
-----
Process exited after 68.68 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 13 - Prog11: Programa para verificar se o número é positivo ou negativo.

Fonte: Elaborada pela autora, 2019.

Depois da seleção composta, temos a seleção encadeada ou aninhada. Isso acontece quando precisamos colocar um *if* dentro do outro. O programa a seguir, figura abaixo, mostra a utilização de um *if* aninhado. Ele ordenada de forma crescente três números digitados pelo usuário. Por exemplo, se o usuário digitar os números 5, 4 e 3, o programa imprimirá na tela a sequência de 3, 4 e 5.

```
1 #include<stdio.h>
2 main()
{
3     int a,b,c;
4     printf("Digite 3 numeros \n");
5     scanf("%d %d %d", &a, &b, &c);
6     if((a>b)&&(a>c))
7     {
8         if(b>c)
9         {
10            printf("Ordem crescente: %d, %d e %d", c, b, a);
11        }
12        else
13        {
14            printf("Ordem crescente: %d, %d e %d", b, c, a);
15        }
16    }
17    else
18    {
19        if(b>c)
20        {
21            if(a>c)
22            {
23                printf("Ordem crescente: %d, %d e %d", c, a, b);
24            }
25            else
26            {
27                printf("Ordem crescente: %d, %d e %d", a, c, b);
28            }
29        }
30        else
31        {
32            if(b>a)
33            {
34                printf("Ordem crescente: %d, %d e %d", a, b, c);
35            }
36            else
37            {
38                printf("Ordem crescente: %d, %d e %d", b, a, c);
39            }
40        }
41    }
42 }
43 }
```

Figura 14 - Prog12: Programa que ordena de forma crescente 3 números digitados pelo usuário.

Fonte: Elaborada pela autora, 2019.

Para facilitar o entendimento deste programa, a figura a seguir mostra o fluxograma para ordenar de forma crescente três números, ou seja, equivalente ao programa da figura anterior.

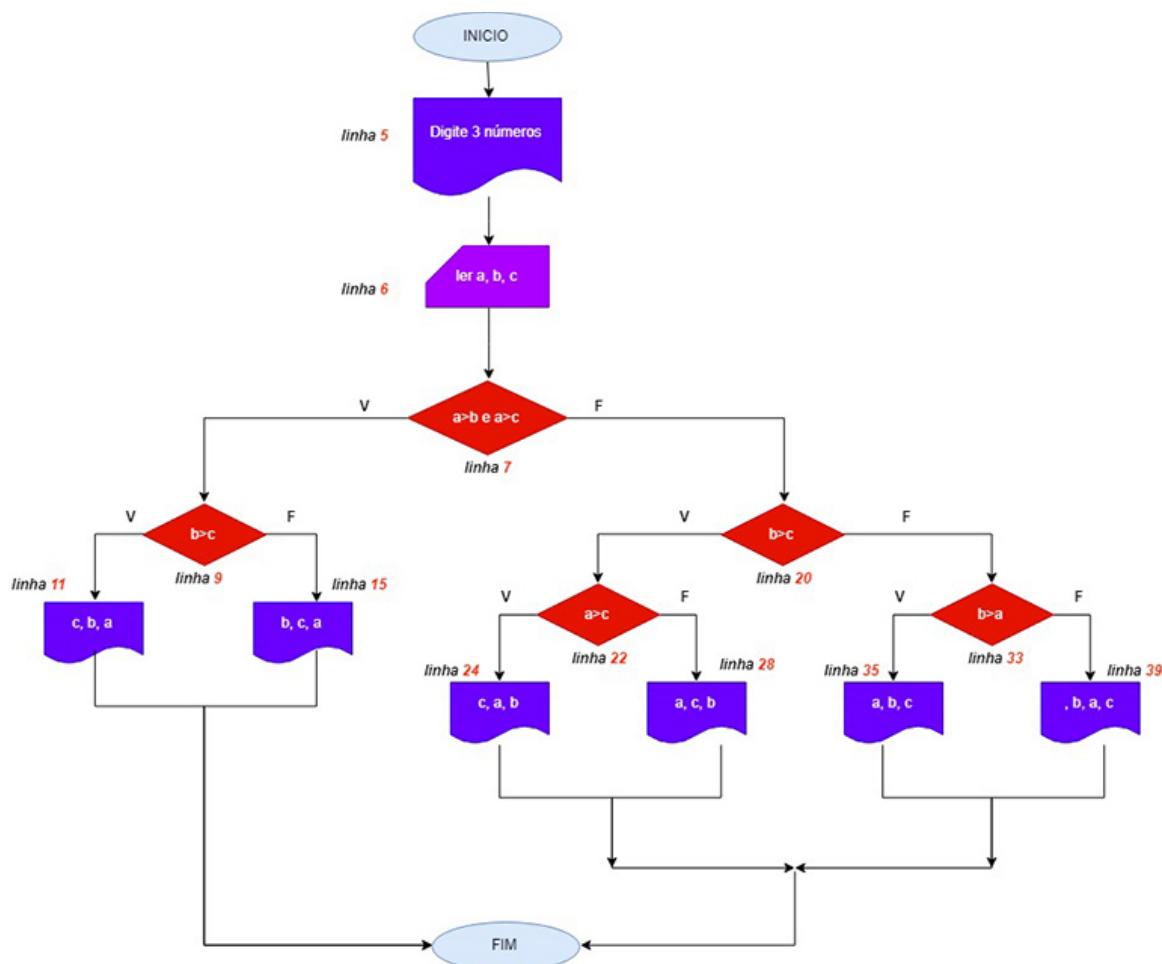


Figura 15 - Fluxograma para ordenar de forma crescente 3 números.

Fonte: Elaborada pela autora, 2019.

Vamos compreender este programa. Acompanhe com o programa e o fluxograma. No fluxograma foi colocado o número das linhas correspondentes ao programa.

- Linhas 5 e 6: usuário entra com três números.
- Linha 7: o teste verifica se *a* é o maior número entre os três.
  - o Se for verdadeiro, segue para a linha 9.
  - o Se for falso, segue para a linha 20.
- Linha 9: aqui *a* é o maior número entre os três, então entre *b* e *c*, qual será o maior?
  - o Se retornar verdadeiro: sequência *c, b, a* (linha 11) e finaliza programa.
  - o Se retornar falso: sequência *b, c, a* (linha 15) e finaliza programa.
- Linha 20: neste ponto sabemos que *a* não é o maior número, ele pode ser o menor de todos ou o número do meio. Então, o maior número está entre *b* e *c*.
  - o Se *b* é o maior, vai para a linha 22.
  - o Se *c* é o maior, vai para a linha 33.
- Linha 22: quem será o número do meio *a* ou *c* (*a>c*)?
  - o Se for *a*: sequência *c, a, b* (linha 24) e finaliza programa.
  - o Se for *c*: sequência *a, c, b* (linha 28) e finaliza programa.
- Linha 33: neste ponto *c* é o maior número. Quem será o segundo maior, *a* ou *b* (*b>a*)?
  - o Se for *b*: sequência *a, b, c* (linha 35) e finaliza programa.
  - o Se for *a*: sequência *b, a, c* (linha 39) e finaliza programa.

Para construir este programa, você pode digitá-lo ou criar o seu, o importante é entender o que foi feito e como foi feito.

A figura a seguir, mostra o mesmo programa, apenas com as chaves retiradas. Analise os dois programas e responda qual é o melhor. O melhor, com certeza, é aquele que você entendeu. Existe uma tendência de que com o aumento da experiência a gente quer economizar nos códigos. Neste momento, seu foco é na compreensão e desenvolvimento da lógica de programação.

```
1 #include<stdio.h>
2 main()
3 {
4     int a,b,c;
5     printf("Digite 3 numeros \n");
6     scanf("%d %d %d", &a, &b, &c);
7     if((a>b)&&(a>c))
8         if(b>c)
9             printf("Ordem crescente: %d, %d e %d", c, b, a);
10    else
11        printf("Ordem crescente: %d, %d e %d", b, c, a);
12    else
13        if(b>c)
14            if(a>c)
15                printf("Ordem crescente: %d, %d e %d", c, a, b);
16            else
17                printf("Ordem crescente: %d, %d e %d", a, c, b);
18        else
19            if(b>a)
20                printf("Ordem crescente: %d, %d e %d", a, b, c);
21            else
22                printf("Ordem crescente: %d, %d e %d", b, a, c);
23 }
```

Figura 16 - Prog12: Programa que ordena de forma crescente 3 números digitados pelo usuário com a omissão das chaves.

Fonte: Elaborada pela autora, 2019.

Agora, podemos seguir para o próximo comando, a estrutura de seleção de múltiplas escolhas.

### 3.3.2 Estrutura de seleção: múltiplas escolha

Uma outra estrutura de seleção que o C suporta é o *switch*. Ele verifica se uma determinada variável possui um valor específico. Sua sintaxe é:

```
switch (variavel)
{
    case valor1:
        linhas de codigo;
        break;
    case valor2:
        linhas de codigo;
        break;
    default:
        linhas de codigo;
}
```

Ele funciona assim, se o valor da variável for *valor1*, o programa irá executar as linhas de código do primeiro *case*, sairá do *case* quando executar o *break*. A opção *default* é opcional e será executado quando o valor da variável não for igual a nenhum dos valores descritos nos *cases*.

A *variável* só poderá ser do tipo *int* ou *char*, com as suas variações. Vamos ver o exemplo a seguir, próxima figura, que ilustra o programa para mostrar uma tela principal.

The figure shows the source code of `prog13.c` and three terminal windows. The source code uses a `switch` statement to handle user input for two options:Cadastro de alunos and Cadastro de notas.

```
1 #include<stdio.h>
2 main()
3 {
4     int op, qde;
5     printf("TELA PRINCIPAL \n");
6     printf("1 - Cadastro de alunos \n");
7     printf("2 - Cadastro de notas \n");
8     scanf("%d", &op);
9     switch (op)
10    {
11        case 1:
12            printf("\nDigite a quantidade de alunos");
13            scanf("%d", &qde);
14            printf("\nSao %d alunos matriculados", qde);
15            break;
16        case 2:
17            printf("\nCadastro de Notas");
18            break;
19        default:
20            printf("\nVoce escolheu uma opcao diferente de 1 e 2");
21            break;
22    }
23 }
24 }
```

The first terminal window shows the program's output for option 1 (Cadastro de alunos). It asks for the number of students (45) and then displays the message "Sao 45 alunos matriculados".

```
D:\ArquivosPessoais\DtCom\CAP03\prog13.exe
TELA PRINCIPAL
1 - Cadastro de alunos
2 - Cadastro de notas
1
Cadastro de Alunos
Digite a quantidade de alunos45

Sao 45 alunos matriculados
-----
Process exited after 10.77 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

The second terminal window shows the program's output for option 2 (Cadastro de notas). It simply displays "Cadastro de Notas".

```
D:\ArquivosPessoais\DtCom\CAP03\prog13.exe
TELA PRINCIPAL
1 - Cadastro de alunos
2 - Cadastro de notas
2

Cadastro de Notas
-----
Process exited after 10.15 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

The third terminal window shows the program's output for an invalid option (6). It displays the message "Voce escolheu uma opcao diferente de 1 e 2".

```
D:\ArquivosPessoais\DtCom\CAP03\prog13.exe
TELA PRINCIPAL
1 - Cadastro de alunos
2 - Cadastro de notas
6

Voce escolheu uma opcao diferente de 1 e 2
-----
Process exited after 10.43 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 17 - Prog13: Programa Tela principal usando o `switch`.

Fonte: Elaborada pela autora, 2019.

Nosso próximo tópico é sobre comandos de repetição. Vamos estudar o `for`, `while` e

*do/while* que equivalem ao PARA, ENQUANTO E REPETIR ATE de algoritmos.

### 3.4 Estrutura de repetição com C

Vamos abordar neste tópico os três comandos de repetição PARA, ENQUANTO e REPETIR ATE em linguagem C. O comando equivalente ao PARA é o comando *FOR*, para o comando ENQUANTO temos o comando *WHILE* e para o comando REPETIR ATE, seu equivalente será o DO/*WHILE*. Os comandos *for* e *while* são estruturas de repetição com teste no início e o comando *do/while* possui o teste no final.

### VOCÊ O CONHECE?

Você sabe quem é Ada Lovelace, ou Ada Augusta Byron King, condessa de Lovelace? Pertence a ela, o título da primeira pessoa a criar um programa. Foi ela quem criou um programa para a Máquina de Charles Babbage. Esta máquina deu início aos primeiros computadores eletrônicos. Ada escreveu um algoritmo para calcular a sequência de Bernoulli entre 1842 e 1843. (SANTOS, 2011). Conheça o restante da história da programação em: <<http://www.techtudo.com.br/platb/desenvolvimento/2011/06/20/historia-da-programacao-como-tudo-comecou/>> (<http://www.techtudo.com.br/platb/desenvolvimento/2011/06/20/historia-da-programacao-como-tudo-comecou/>)>.

Nas três estruturas, o programa irá repetir enquanto a condição retornar verdadeira. O comando *for* possui três campos de controle, e o bloco de linhas de código dentro do comando só será executado a primeira vez, se a condição retornar verdadeira. O mesmo ocorre com o comando *while*, mas este só possui um campo, o campo da condição. No comando *do/while* o programa entra no laço sem fazer testes, o que faz com que o bloco de comandos seja executado pelo menos uma vez, sem fazer o teste.

Vamos começar pelo comando *FOR*.

#### 3.4.1 Introdução à estrutura de repetição com teste no início

A estrutura de repetição com teste no início *FOR*, possui três campos separados por ponto e vírgula:

```
for (inicializar_variaveis; condicao; incremento)
{
    //linhas de codigo
    //linhas de codigo
}
```

O quadro a seguir mostra a explicação de cada campo.

Campo	Descrição
inicializar_variaveis	Será o primeiro a ser executado. Aceita mais de uma variável separadas por vírgula.
condicao	Expressão lógica. Se retornar verdadeira: o bloco de comandos será executado. Se retornar falsa: seguirá para a primeira linha após o fim do comando.
incremento	Espaço para incrementar uma ou mais variáveis, desde que separadas por vírgula.

Quadro 6 - Campos do comando de repetição for.

Fonte: Elaborado pela autora, 2019.

Vamos ver o próximo exemplo, figura a seguir. Um programa para ler dois números, um maior que o outro, então, temos uma faixa de números. O programa irá imprimir os números sequentes e subsequentes do menor número, e antecedentes do maior número até que eles se encontrem.

```
1 #include<stdio.h>
2 main()
3 {
4     int i, j, n1, n2;
5     printf("Digite 2 numeros\n");
6     scanf("%d %d", &n1,&n2);
7     for( i=n1, j=n2 ; i <= j; i++, j--)
8     {
9         printf("\n%d ", i);
10        printf("%d ", j);
11    }
12 }
13 }
```

D:\ArquivosPessoais\DtCom\CAP03\prog14.exe

```
Digite 2 numeros
5
25

5 25
6 24
7 23
8 22
9 21
10 20
11 19
12 18
13 17
14 16
15 15
-----
Process exited after 9.625 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 18 - Prog14: Exemplo do comando for.

Fonte: Elaborada pela autora, 2019.

É um programa que merece algumas observações:

- no comando *for*, linha 7, no campo de inicialização, as variáveis *i* e *j* são inicializadas com os valores de *n1* e *n2*;
- no campo de incremento, as duas variáveis são incrementadas de 1. Observe que estamos utilizando a versão simplificada do operador de incremento e decremento, ou seja,  $i = i + 1$  pode ser escrito como  $i++$  e  $j = j - 1$  pode ser escrito como  $j--$ .

Vamos imprimir o alfabeto na tela? Analise o próximo programa. Ele imprime as letras do alfabeto na tela para o usuário. Veja que o incremento está na variável do tipo *char*. Interessante, né?

```
1 #include<stdio.h>
2 main()
3 {
4     char letra;
5     printf("Alfabeto Maiusculo\n");
6     for( letra='A'; letra <= 'Z'; letra++)
7     {
8         printf("%c ", letra);
9     }
10    printf("\nAlfabeto Minusculo\n");
11    for( letra='a'; letra <= 'z'; letra++)
12    {
13        printf("%c ", letra);
14    }
15
16 }
```

```
D:\ArquivosPessoais\DtCom\CAP03\prog15.exe
Alfabeto Maiusculo
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Alfabeto Minusculo
a b c d e f g h i j k l m n o p q r s t u v w x y z
-----
Process exited after 6.233 seconds with return value 0
Pressione qualquer tecla para continuar. . . .
```

Figura 19 - Prog15: Programa para imprimir o alfabeto na tela.

Fonte: Elaborada pela autora, 2019.

O que acontece quando colocamos um comando de repetição dentro do outro? Veja o exemplo a seguir, próxima figura. São três comandos *for*, aninhados, um dentro do outro. O mais interno, com a variável *k*, vai de 0 a 2. O próximo é o com a variável *j*, que vai de 10 a 30, de 10 em 10. E o mais externo, com a variável *i*, recebe o valor de 100 e de 200. Observe que, para cada valor dos comandos externos, o *for* interno roda todos os seus valores. Como um relógio, o *for* mais interno será para os segundos, depois o outro será para os minutos e o mais externo, será para as horas.

```
1 #include<stdio.h>
2 main()
3 {
4     int i, j, k;
5     for(i=100; i<=200; i = i + 100)
6     {
7         for(j=10; j<=30 ; j = j + 10)
8         {
9             for(k=0; k<=2; k++)
10            {
11                printf("%d : %d %d\n", i, j, k);
12            }
13            printf("\n");
14        }
15        printf("\n");
16    }
17 }
```

```
D:\ArquivosPessoais\DtCom\CAP03\prog16.exe
100 : 10 0
100 : 10 1
100 : 10 2

100 : 20 0
100 : 20 1
100 : 20 2

100 : 30 0
100 : 30 1
100 : 30 2

200 : 10 0
200 : 10 1
200 : 10 2

200 : 20 0
200 : 20 1
200 : 20 2

200 : 30 0
200 : 30 1
200 : 30 2

Process exited after 5.824 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 20 - Prog16: Programa com 3 comandos de repetição for aninhados.  
Fonte: Elaborada pela autora, 2019.

O próximo comando é o *while*, equivalente ao ENQUANTO.

### 3.4.2 Estrutura de repetição com teste no início usando flag de parada

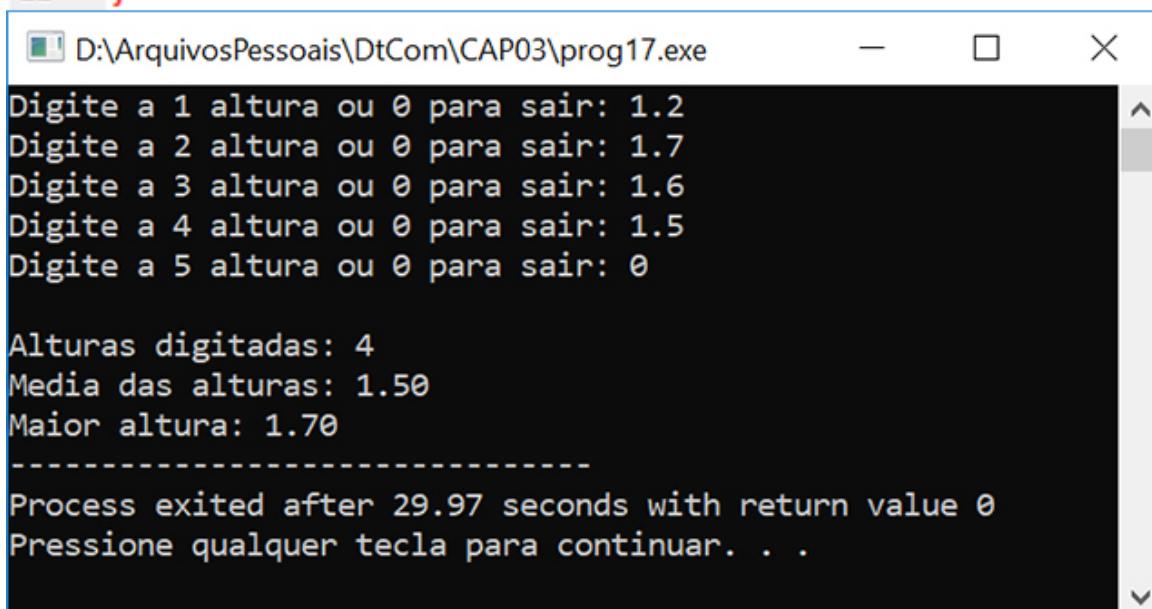
O comando `while` é uma estrutura de repetição com teste no início e com apenas um campo. Sua forma é:

```
while (condicao)
{
    //linhas de codigo
    //linhas de codigo
}
```

A condição é uma equação lógica que controla do comando. Caso retorne verdadeira, as linhas de código que estão dentro do comando, entre as chaves, serão executadas até o momento em que esta condição se tornar falsa.

Analise o exemplo a seguir, próxima figura. É um programa para ler uma sequência de alturas e informar: a maior altura, a média de alturas e quantas alturas foram lidas.

```
1 #include<stdio.h>
2 main()
3 {
4     float h=1, soma=0, cont=0, maiorh=0;
5     while( h != 0 )
6     {
7         printf("Digite a %.0f altura ou 0 para sair: ", cont+1);
8         scanf("%f", &h);
9         if(h!=0)
10        {
11            soma = soma + h;
12            cont++; //o mesmo que cont=cont+1
13            if( h > maiorh )
14            {
15                maiorh = h;
16            }
17        }
18    }
19    printf("\nAlturas digitadas: %.0f", cont);
20    printf("\nMedia das alturas: %.2f", soma/cont);
21    printf("\nMaior altura: %.2f", maiorh);
22 }
```



```
D:\ArquivosPessoais\DtCom\CAP03\prog17.exe
Digite a 1 altura ou 0 para sair: 1.2
Digite a 2 altura ou 0 para sair: 1.7
Digite a 3 altura ou 0 para sair: 1.6
Digite a 4 altura ou 0 para sair: 1.5
Digite a 5 altura ou 0 para sair: 0

Alturas digitadas: 4
Media das alturas: 1.50
Maior altura: 1.70
-----
Process exited after 29.97 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 21 - Prog17: Programa para calcular a maior e a média das alturas e quantas foram lidas.

Fonte: Elaborada pela autora, 2019.

Vamos analisar este programa da figura anterior.

- Linha 5: comando `while`, testa se `h` é diferente de 0.

- Se for verdadeiro pula para a linha 7.
- Se for falso sai do comando de repetição, ou seja, pula para a linha 19.
- Linha 7: o  $cont+1$  não altera o valor da variável  $cont$ , ele apenas imprime para o usuário o valor de  $cont+1$ , então, se  $cont$  tiver o valor de 0, irá aparecer para o usuário o valor de 1.
- Na linha 8: irá ler o número digitado pelo usuário.
- Linha 9: um teste condicional para separar as alturas do número zero.
  - Caso o usuário digite uma altura, o programa irá executar as linhas de 10 a 17.
  - Se o usuário digitou o zero, significa que deseja sair, então vai para a linha 17 (fechamento das chaves) e depois para a linha 5, com  $h$  valendo 0. Ao testar a condição do `while`,  $h!=0$ , retornará falso, seguindo para a linha 19.
- Linha 11: é um acumulador que armazena todas as alturas. Segue para a linha 12.
- Linha 12: é um contador utilizado na forma abreviada. Segue para a linha 13.
- Linhas 13 a 16: trecho para achar a maior altura. Quando o usuário digitar uma altura maior do que a que está armazenada, ele atualiza a altura. O detalhe aqui é a inicialização da variável que irá guardar a maior altura. Siga a dica:
  - Para calcular o maior valor, inicialize a variável com o menor valor que não será utilizado. Por exemplo, no nosso caso, menor valor que não será utilizado é altura zero.
  - Para calcular o menor valor, inicialize a variável com o maior valor que não será utilizado. Por exemplo, se fôssemos calcular a menor altura, o maior valor que não será utilizado seria uma altura de 5m ou mais.

Agora que tal um problema aplicado?

## CASO

A gerente de uma empresa de instalação e projetos de redes de computadores ficou curiosa para saber dados a respeito da idade de seus funcionários, como a idade do funcionário mais velho e a quanto tempo ele trabalha na empresa e a idade média de seus funcionários. Para isso, ela te contratou para desenvolver um sistema que informe estes dados. O programa deverá ficar em um computador situado no refeitório da empresa, assim, os funcionários passarão pelo computador e digitarão os dados. A cada lida dos dados de um funcionário, imprimir todas as informações solicitadas, ou seja, quantos funcionários já participaram, qual a idade do funcionário mais velho e o tempo que ele trabalha na empresa, a média de idade dos funcionários e a média de tempo de trabalho dos funcionários na empresa. O usuário terá uma opção para digitar novos dados.

**Sugestões:** utilize um comando de repetição; ao calcular a maior idade aproveite para guardar o tempo de casa deste funcionário; tente resolver antes de olhar a resposta; bom trabalho!

A seguir, é apresentado um programa para atender às necessidades da empresa e uma tela de execução para ilustrar o funcionamento.

```
1 #include<stdio.h>
2 main()
3 {
4     float somaid=0, somatempo=0, tempo, tempomaior ;
5     int id, op=1, maiorid=0, contfunc=0;
6     while( op != 2 )
7     {
8         printf("Idade: ");
9         scanf("%d", &id);
10        printf("\nTempo de casa: ");
11        scanf("%f", &tempo);
12        somaid = somaid + id;
13        somatempo = somatempo + tempo;
14        contfunc++; //o mesmo que cont=cont+1
15        if( id > maiorid )
16        {
17            maiorid = id; //armazena a maior idade
18            tempomaior = tempo;//armazena o tempo do funcionário com a maior idade
19        }
20        printf("\nFuncionarios participantes: %d", contfunc);
21        printf("\nIdade do funcionario mais velho %d e o seu tempo de casa %.1f", maiorid, tempomaior);
22        printf("\nMedia de idade: %.1f", somaid/contfunc);
23        printf("\nMedia de tempo de casa: %.1f", somatempo/contfunc);
24        printf("\nNova consulta? 1 - Sim 2 - Nao");
25        scanf("%d", &op);
26    }
27 }
```

D:\ArquivosPessoais\DtCom\CAP03\caso.exe

```
Idade: 24
Tempo de casa: 5

Funcionarios participantes: 1
Idade do funcionario mais velho 24 e o seu tempo de casa 5.0
Media de idade: 24.0
Media de tempo de casa: 5.0
Nova consulta? 1 - Sim 2 - Nao1
Idade: 56

Tempo de casa: 10

Funcionarios participantes: 2
Idade do funcionario mais velho 56 e o seu tempo de casa 10.0
Media de idade: 40.0
Media de tempo de casa: 7.5
Nova consulta? 1 - Sim 2 - Nao2

-----
Process exited after 40.69 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

O último comando que estudaremos é o comando *do/while*. Ele é o equivalente ao REPETIR ATE. Vamos compreender o seu funcionamento no próximo item.

### 3.4.3 Estrutura de repetição com teste no final e variável de controle

Nossa última estrutura de repetição é o *do/while*, ele é o equivalente ao REPETIR ATE. É um comando que possui apenas um campo, o de condição, ao final do comando. Uma característica importante é que o bloco de comandos dentro da estrutura será executado, pelo menos, uma vez. Se a condição retornar verdadeira, o programa

permanece no laço, se retornar falso, saíra do laço. Sua sintaxe é:

```
do
{
    //linhas de codigo
    //linhas de codigo
} while(condicao);
```

Vamos ver um exemplo do uso deste comando. O programa, a seguir, cria uma tela principal com as opções de saldo, deposito, saque e sair. Isso mesmo, uma tela de um caixa eletrônico de um banco.

Duas novas funções foram utilizadas aqui, `system("cls")` e `system("pause")`, ambas pertencentes à biblioteca `stdlib.h`. O `cls` limpa a tela e o `pause` provoca uma pausa na execução, sendo necessária a intervenção do usuário para sair deste ponto. Esta configuração funciona apenas para o sistema operacional Windows.

Para cada opção, o programa executa a função desejada. Para o saldo, basta imprimir na tela o valor da variável `saldo`, cujo valor inicial é de zero.

Para a opção de `deposito`, o sistema pede o valor a ser depositado e atualiza o saldo do cliente. Para o `saque`, é verificado se o valor a ser sacado existe na conta. Esta verificação ocorre na linha 39. Caso o cliente tenha saldo suficiente, o programa efetua o saque atualizando o saldo e enviando uma mensagem ao usuário. Já na situação, onde o cliente deseja sacar um valor superior ao seu saldo, o programa não atualiza o saldo e envia uma mensagem ao cliente, seu saldo é insuficiente.

Analise o programa, segundo cada tela de execução.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 main()
5 {
6     int op;
7     float saldo, valor;
8     do
9     {
10         system("cls");
11         printf("Tela Principal\n");
12         printf("1- Saldo\n");
13         printf("2- Depósito\n");
14         printf("3- Saque\n");
15         printf("4- Sair\n");
16         scanf("%d", &op);
17         if(op==1)
18         {
19             system("cls");
20             printf("SALDO\n");
21             printf("Saldo atual: R$%.2f\n", saldo);
22             system("pause");
23         }
24         if(op==2)
25         {
26             system("cls");
27             printf("DEPÓSITO\n");
28             printf("Valor do depósito: ");
29             scanf("%f", &valor);
30             saldo += valor;
31             printf("Depósito efetuado com sucesso!\n");
32             system("pause");
33         }
34         if(op==3)
35         {
36             system("cls");
37             printf("SAQUE\n");
38             printf("Valor do saque: ");
39             scanf("%f", &valor);
40             if((saldo-valor)>=0)
41             {
42                 saldo -= valor;
43                 printf("Saque efetuado com sucesso!\n");
44             }
45             else
46                 printf("Saldo insuficiente\n");
47             system("pause");
48         }
49     }while(op!=4);
}
```

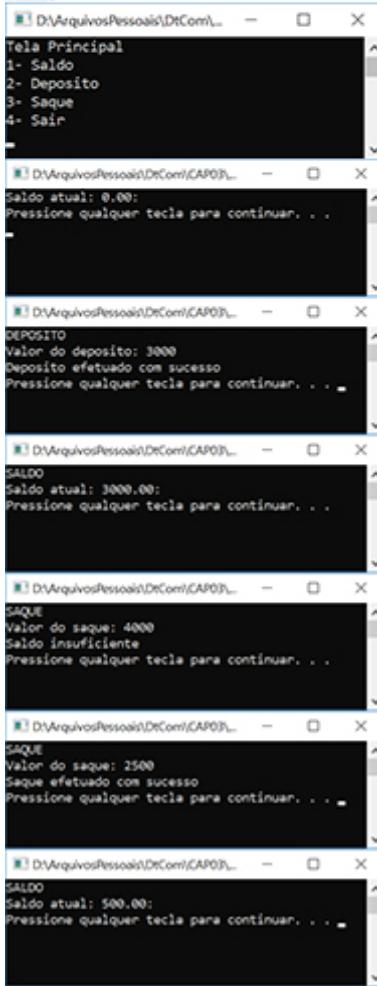


Figura 22 - Prog18: Programa para um caixa eletrônico.

Fonte: Elaborada pela autora, 2019.

Assim, conseguimos entender e aplicar os diferentes tipos de dados básicos do C,

observando como declarar as variáveis e suas regras de nomeação, quais são os principais operadores matemáticos, relacionais e lógicos.

Perceba que é fácil compreender a estrutura da linguagem C, que trabalha com funções, sendo a principal e necessária em todos os programas, a *main*.

A partir deste ponto, você já consegue criar seus programas, utilizando os comandos de entrada e saída de dados, o comando condicional *if*, simples, composto e aninhado, o *switch/case*, e os três comandos de repetição: *for*, *while* e *do/while*.

## Síntese

Concluímos o capítulo! Aprendemos aqui os princípios básicos das formas de construção de programas com a Linguagem de Programação C e como codificar seus algoritmos. Agora você já é capaz de desenvolver programa e colocá-los para rodar. Também estudamos formas de desenvolver programas sequenciais e estruturas de seleção simples, composta, homogênea e heterogênea. E vimos como funciona o laço de repetição com teste no início, com teste no início e *flag* de parada e com teste no final e variável de controle.

Neste capítulo, você teve a oportunidade de:

- conhecer os tipos básicos do C, *int*, *float*, *char* e *void*, e suas variações, como declarar as variáveis e as regras de nomeação, os principais operadores matemáticos, lógicos e relacionais da linguagem C;
- entender a estrutura de funcionamento da linguagem C;
- aplicar os comandos de entrada e saída, *scanf* e *printf*, em seus programas;
- empregar o comando condicional *if*, no modo simples, composto e na forma aninhada;
- aplicar o comando de repetição *for*, construir programas utilizando o comando *while* e *do/while*;
- comparar os três comandos de repetição *for*, *while* e *do/while* em aplicações específicas.



◀ Clique para baixar o conteúdo deste tema.

## Bibliografia

- BACKES, A. **Linguagem C completa e descomplicada.** Rio de Janeiro: Campus, 2013. 400p.
- DAMAS, L. **Linguagem C.** 10. ed. Rio de Janeiro: LTC, 2007.
- DEITEL, P.; DEITEL, H. **C Como Programar.** 6. ed. São Paulo: Pearson Prentice Hall, 2011.
- CARVALHO FILHO, C. R. **Operador Ternário em C Curso C.** Excript.com, 08 mai. 2016. Disponível em: <<http://excript.com/linguagem-c/operador-ternario-c.html>> (<http://excript.com/linguagem-c/operador-ternario-c.html>)> . Acesso em: 25/01/2019.
- FESCINA, D. **Quem inventou a internet?** Portal Super Interessante, publicado em 12 jan. 2016. Disponível em: <<https://super.abril.com.br/mundo-estranho/quem-inventou-a-internet/>> (<https://super.abril.com.br/mundo-estranho/quem-inventou-a-internet/>)> . Acesso em: 25/01/2019.
- GATTO, E. C. **Cadeia de caracteres:** funções de entrada e saída. Portal Embarcados, publicado em 7 dez. 2017. Disponível em: <<https://www.embarcados.com.br/cadeia-de-caracteres-funcoes-de-entrada-e-saida/>> (<https://www.embarcados.com.br/cadeia-de-caracteres-funcoes-de-entrada-e-saida/>)> . Acesso em: 25/01/2019.
- MANZANO, J. A. N. G. **Estudo dirigido de linguagem C Acompanhada de uma xícara de café.** São Paulo: Erica, 2015.
- PFISTER, W.; PAGLEN, J. **Transcendence:** A Revolução. Direção: Wally Pfister. Produção: Broderick Johnson, Andrew A. Kosove, Kate Cohen, Marisa Polvino, Annie Marter, David Valdes e Aaron Ryder. Produção executiva: Christopher Nolan e Emma Thomas. Cor. 119 min. Reino Unido/China/Estados Unidos. 2014.
- SANTOS, R. **História da Programação:** Como tudo começou! Portal Techtudo, publicado em 20/06/11. Disponível em: <<http://www.techtudo.com.br/platb/desenvolvimento/2011/06/20/historia-da-programacao-como-tudo-comecou/>> (<http://www.techtudo.com.br/platb/desenvolvimento/2011/06/20/historia-da-programacao-como-tudo-comecou/>)> . Acesso em: 25/01/2019.