

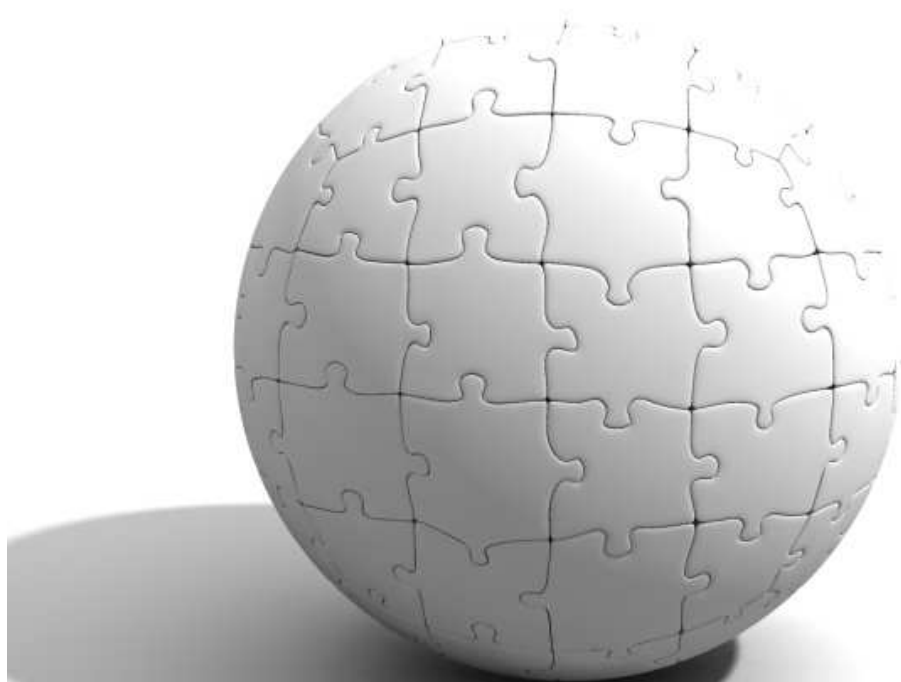
## LÓGICA DE PROGRAMAÇÃO COM "C"

Professora Lucélia Oliveira

Professora Poliana Ribeiro Tolentino

Agência Brasileira do ISBN

ISBN 978-85-67405-01-8



Lógica de programação é a  
técnica de encadear pensamentos  
para atingir determinado  
objetivo



## Sumário

<b>I - CONCEITOS INICIAIS.....</b>	<b>5</b>
Lógica de programação .....	5
Algoritmo .....	5
Instrução .....	7
EXERCÍCIOS - Algoritmos.....	7
Programa de Computador.....	8
Diagrama de Blocos .....	9
EXERCÍCIOS - Diagrama de blocos .....	10
<b>II - VARIÁVEIS.....</b>	<b>11</b>
Tipos de Variáveis .....	12
Numéricas .....	12
Lógico .....	12
Cadeia ou String ou Alfanuméricas.....	13
Caractere .....	13
Variáveis em C.....	14
Tipos Básicos em C:.....	14
Constantes em "C" .....	15
<b>III - EXPRESSÕES .....</b>	<b>15</b>
Operadores Aritméticos .....	16
Operadores Relacionais .....	17
Operadores Lógicos .....	18
EXERCÍCIOS - Identificadores e tipos de variáveis.....	19
<b>IV – COMANDOS BÁSICOS.....</b>	<b>20</b>
Comando de Atribuição .....	20
Comandos de Entrada e Saída .....	21
Comandos de Entrada e Saída em "C" .....	22
Exemplo - uso dos comando gets() e puts(): .....	23
Função printf() em "C" .....	24
Cadeia de caracteres em "C" .....	25
Primeiro Exemplo – Português Estruturado e "C".....	27
EXERCÍCIOS - Estrutura Sequencial.....	29
EXERCÍCIOS - Estrutura Sequencial.....	29
<b>V – ESTRUTURAS DE SELEÇÃO .....</b>	<b>31</b>



Seleção Simples .....	32
Seleção Composta .....	32
Seleção Simples .....	32
Seleção Composta .....	32
EXERCÍCIOS – Estrutura de Seleção .....	34
Alternativa de Múltiplas Escolhas - switch/case .....	37
EXERCÍCIOS – Alternativa de Múltiplas Escolhas .....	40
VI – ESTRUTURAS DE REPETIÇÃO .....	41
Comando Enquanto (While) .....	41
Comando Repita (do- while) .....	44
em "C" .....	44
EXERCÍCIOS – Comando Enquanto (WHILE) .....	45
Comando Para (For) .....	48
em "C" .....	49
EXERCÍCIOS – Comando Para (For) .....	50
REVISÃO - Lógica de Programação .....	53
VII - ESTRUTURA HOMOGÊNEA: VETORES .....	54
Trecho de Dimensionamento .....	54
Trecho de Entrada de Dados .....	55
Trecho de Saída de Dados .....	56
em "C" .....	57
EXERCÍCIOS .....	60
VIII - MATRIZES .....	61
EXERCÍCIOS - Matriz .....	63
IX – SUB-ROTINAS - FUNÇÕES .....	64
Função(function) .....	64
EXERCÍCIOS - Funções .....	67
ANEXO I – Questões de Raciocínio Lógico .....	68
ANEXO II - Transferência de Comandos: Português Estruturado para C .....	69
ANEXO III - Como compilar um programa na linguagem C .....	70



ANEXO IV – Exercícios Resolvidos.....	70
ANEXO V – Tabela ASCII .....	70



## I - CONCEITOS INICIAIS

### *Lógica de programação*

É a técnica de encadear pensamentos para atingir determinado objetivo. O aprendizado desta técnica é necessário, para quem quer trabalhar com desenvolvimento de sistemas e programas.

### *Algoritmo*

É uma sequência de passos finitos com o objetivo de solucionar um problema.

O estudo da lógica é o estudo dos métodos e princípios usados para distinguir o raciocínio correto do incorreto. Naturalmente, essa definição não pretende afirmar que só é possível argumentar corretamente com uma pessoa que já tenha estudado lógica. Afirmá-lo seria tão errôneo quanto pretender que só é possível correr bem, se estudou física e fisiologia, necessárias para a descrição dessa atividade. Alguns excelentes atletas ignoram completamente os processos complexos que se desenrolam dentro deles próprios quando praticam o esporte.

Assim também acontece no nosso dia-a-dia. Quantas vezes já vimos um algoritmo e não sabíamos que aquela sequência de passos chamava-se algoritmo.

Um exemplo bem frequente é quando queremos falar em algum telefone público.

---

Exemplo de algoritmo para falar em um telefone público

- 1 – Retirar o telefone do gancho;
- 2 – Esperar o sinal;
- 3 – Colocar o cartão;



- 4 – Discar o número;
- 5 – Falar no telefone;
- 6 – Colocar o telefone no gancho.

O algoritmo é exatamente esse conjunto de passos que resolveu o problema de uma pessoa falar no telefone. É como se fôssemos ensinar uma máquina a fazer alguma tarefa específica.

Outro exemplo clássico é um algoritmo para resolver o problema de fritar um ovo que poderia estar escrito em forma de uma receita. A receita é um algoritmo, pois é formada de ações que devem ser tomadas para fritar um ovo.

---

#### Exemplo de algoritmo para fritar um ovo

- 1 – pegar frigideira, ovo, óleo e sal;
- 2 – colocar óleo na frigideira;
- 3 – acender o fogo;
- 4 – colocar a frigideira no fogo;
- 5 – esperar o óleo esquentar;
- 6 – colocar o ovo;
- 7 – colocar o sal;
- 8 – retirar quando estiver pronto.

Cada linha do algoritmo pode-se chamar de uma instrução, logo, podemos dizer que um algoritmo é um conjunto de instruções.

Assim como fritar um ovo, nos algoritmos computacionais não podemos trocar ou omitir certas instruções, caso contrário não obteremos o resultado esperado. Por exemplo, se omitirmos a instrução acender o fogo, não teremos ao final do algoritmo um ovo frito.



### ***Instrução***

Indica a um computador uma ação elementar a ser executada.

Até as coisas mais simples podem ser descritas por um algoritmo. Por exemplo:

---

Algoritmo para o fim de semana

- 1 – vejo a previsão do tempo;
- 2 – se fizer sol  
    vou à praia;
- senão  
        vou estudar;
- 3 – almoçar;
- 4 – ver televisão;
- 5 – dormir.

### **EXERCÍCIOS - Algoritmos**

- 1 – Fazer um algoritmo para tomar banho:

---

---

---

---

---

---



2 – Crie um algoritmo para fazer uma prova:

---

---

---

---

---

---

---

3 – Faça um algoritmo para somar dois números:

---

---

---

---

---

### ***Programa de Computador***

Nada mais é do que um algoritmo escrito numa linguagem de computador (C, Pascal, Fortran, Delphi, Cobol, Java e outras). É a tradução para o inglês do algoritmo feito em português. O mais importante de um programa é a sua lógica, o raciocínio utilizado para resolver o problema, que é exatamente o algoritmo.

A forma de escrever um algoritmo em pseudocódigo (algoritmo que não usa nenhuma linguagem de programação) vai variar de autor para autor, pois, um traduz ao pé da letra a linguagem C, outro, o Pascal, outro, mistura as duas linguagens e assim por diante.



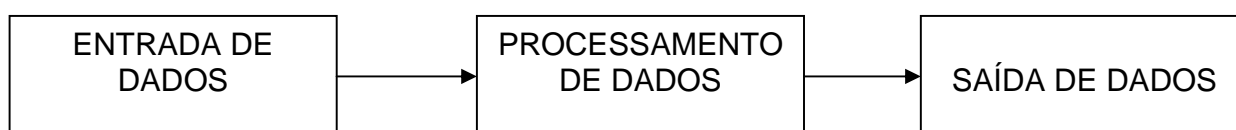


É importante lembrar que estas variações vão sempre ocorrer, podemos dizer que é uma variação de autores adotados.

## Fases

A principal finalidade de um computador é realizar a tarefa de processamento de dados, isto é, receber dados através de dispositivos de entrada que podem ser, por exemplo, teclado, mouse, scanner, entre outros; realizar operações com estes dados e gerar uma resposta que será expressa em um dispositivo de saída que pode ser, por exemplo, uma impressora, um monitor de vídeo, entre outros.

Entretanto ao montar um algoritmo, precisamos primeiro dividir o problema apresentado em três fases fundamentais:



**ENTRADA:** São os dados de entrada do algoritmo.

**PROCESSAMENTO:** São os procedimentos utilizados para chegar ao resultado final.

**SAÍDA:** São os dados já processados, os resultados, mostrados na tela do computador (monitor de vídeo) ou impressora.

## Diagrama de Blocos

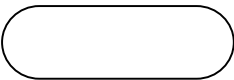
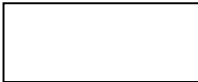


É uma forma padronizada para representar os passos lógicos de um determinado processamento.

Com o diagrama, também conhecido como fluxograma, podemos definir uma sequência de símbolos, com significado bem definido, portanto, sua principal função é a de facilitar a visualização dos passos de um processamento.



### Simbologia do Diagrama de Bloco

Existem diversos símbolos em um diagrama de bloco. Veja no quadro abaixo alguns dos símbolos que iremos utilizar:

Símbolo	Função
TERMINAL 	Indica o <b>início</b> ou <b>fim</b> de um processamento Exemplo: Início do algoritmo
 PROCESSAMENTO	Processamento em geral Exemplo: $x \leftarrow 2+3$
 ENTRADA MANUAL DE DADO	Indica entrada de dados pelo usuário via teclado Exemplo: Digite a nota da prova 1
 EXIBIR	Mostra informações ou resultados Exemplo: Mostre o resultado do cálculo

### EXERCÍCIOS - Diagrama de blocos

- Construir um diagrama de blocos que:
  - leia quatro números
  - calcule a média
  - mostre o resultado.
- Construa fluxograma que:
  - leia o salário de um empregado
  - calcule o novo salário sabendo que lê teve um aumento de 15%
  - mostre o resultado na tela
- Agora faça um algoritmo para o exercício 2.



## II - VARIÁVEIS

É um local na memória principal, isto é, um endereço que armazena um conteúdo.

O conteúdo de uma variável pode ser de vários tipos: inteiro, real, caractere (literal), string (cadeia de caracteres), lógico, entre outros.

Uma vez definidos o nome e o tipo de uma variável, não podemos alterá-los no decorrer de um algoritmo. Por outro lado, o conteúdo da variável pode ser modificado no decorrer do programa, de acordo com a necessidade.

Em algoritmos, as variáveis são definidas no início por meio do comando definido:

Nome da variável: tipo da variável;

### Exemplos:

A: inteiro;

X: real;

Nome: string (ou cadeia);

Regras para formação do nome de uma variável:

- 1 – O primeiro caractere é uma **letra**;
- 2 – Se houver mais de um caractere, poderemos usar letras ou dígitos;
- 3 – Nomes de variáveis escritas com letras maiúsculas serão diferentes de letras minúsculas em muitas linguagens de programação. Lembre-se: peso é **diferente** de PESO;
- 4 – Nenhuma palavra reservada poderá ser nome de uma variável.

Nomes Válidos	Nomes Não-Válidos
Nome, profissao,n, n1,PESO, A	2X -> Começa por algarismo peso do aluno -> espaço não é permitido



## ***Tipos de Variáveis***

### **Numéricas**

São aquelas que armazenam dados numéricos, podendo ser divididos em duas classes:

**Inteiro:** são aqueles que **não** possuem componentes decimais ou fracionários, podendo ser positivos ou negativos.

---

*Normalmente uma variável do tipo inteira poderá ocupar **1, 2 ou 4 bytes** na **MP**.*

---

**Exemplos:**

- 10      número inteiro positivo
- 10    número inteiro negativo

**Real:** são aqueles que podem possuir componentes decimais ou fracionários, podendo também ser positivos ou negativos.

---

*Normalmente uma variável do tipo real poderá ocupar **4 ou 8 bytes** na **MP**.*

---

**Exemplos:**

- 25.03      número real positivo com duas casas decimais
- 235.        número real positivo com zero casas decimais
- 10.5      número real negativo com uma casa decimal

### **Lógico**

Também conhecido como booleano. É representado no algoritmo pelos dois únicos valores lógicos possíveis: verdadeiro ou falso. Porém, é comum encontrar em outras referências, outros pares de valores lógicos como: sim/não, 1/0, true/false.

Em "C", um dado do tipo *boolean* armazena 0 (se for falso) ou 1 (se for verdadeiro)



## Cadeia ou String ou Alfanuméricas

São aquelas que possuem letras e/ou números. Pode, em determinados momentos, conter somente dados numéricos ou somente letras. Se usado somente para armazenamento de números, não poderá ser utilizado para operações matemáticas.

---

*O número de bytes possíveis para armazenamento de uma variável desse tipo dependerá da quantidade de caracteres.*

---

### Exemplos:

"Maria"	String de comprimento 5
"123"	String de comprimento 3
"A"	String de comprimento 1

## Caractere

É aquele que armazena apenas uma letra ou dígito, é uma string de comprimento 1. Se usado para armazenar número não poderá ser utilizado para operações matemáticas.

### Exemplos:

"A"	caractere que armazena uma letra
"5"	caractere que armazena um número



Em "C"

- Os caracteres são representados em C usando um byte.



- Dentro da classe dos caracteres temos as letras, maiúsculas e minúsculas, os dígitos, os símbolos de pontuação e os diferentes separadores.
- São considerados também caracteres a mudança de linha (\n), de tabulação(\t), etc.
- Os caracteres escrevem-se entre apóstrofes (exemplo: 'a', ',').
- Uma maneira de representar um caractere é por meio de um sistema de codificação onde um número decimal represente um caractere. Por exemplo, em ASCII (*American Standard for Information Exchange*) cada caractere é codificado com uma combinação de bits, dando um total de 128 combinações, suficiente para todas as letras minúscula e maiúsculas, os dígitos e símbolos comuns.



### Variáveis em C

- Os identificadores, na Linguagem C, devem seguir as seguintes regras de construção:
  - Os identificadores devem começar por uma letra (a - z , A - Z) ou um underscore (\_)
  - Os caracteres subsequentes devem ser APENAS letras, números ou sublinhados. (Não pode conter outros caracteres).
  - Em C, os identificadores podem ter até 32 caracteres, no máximo.
  - A linguagem C é *case sensitive*. Assim, identificadores como Soma, SOMA, soMA são considerados diferentes pelo compilador.

### ATENÇÃO!!!

Existem certos nomes que não podem ser usados como identificadores. São chamadas as **palavras reservadas** e são de uso restrito da linguagem C (comandos, estruturas, declarações, etc.).

### Tipos Básicos em C:

Os tipos char e int são inteiros e os tipos float e double são de ponto flutuante.



Tipo	Tamanho	Intervalo	Uso
<b>char</b> %c	1 byte	-128 a 127	Usado para armazenar um único caractere, é utilizado para guardar valores definidos na tabela ASCII
<b>int</b> %d	2 bytes	-32768 a 32767	Número inteiro, contador, controle de laço
<b>float</b> %f	4 bytes	3.4e-38 a 3.4e38	Real (precisão de 7 dígitos)



### Constantes em "C"

- Constantes são valores que são mantidos fixos pelo compilador durante o programa.
- Se pretendemos que o valor não seja alterado durante toda a execução de um programa, devemos declará-la como uma constante.
- Para isso, antes da declaração, usamos a palavra **const**.  
Ex.: `const int base=10;`  
`const float pi=3.14;`

## III - EXPRESSÕES

O conceito de expressão em termos computacionais está intimamente ligado ao conceito de expressão (ou fórmula) matemática, onde um conjunto de variáveis e constantes numéricas relaciona-se por meio de operadores compondo uma fórmula que, uma vez avaliada, resulta num valor.



### **Operadores Aritméticos**

Os operadores matemáticos são:

Operador	Função
+	Somar
-	Subtrair
*	Multiplicar
/	Dividir
%	Resto da divisão
++	Operador de soma unário
--	Operador de subtração unário

Exemplos de expressões aritméticas:

$$3 * 3 = 9$$

$$9 / 2 = 4.5$$

$$9 \% 2 = 1$$

$$2 / 9 = 0.22$$

$$2 + 4.6 = 6.6$$

#### **ATENÇÃO!!!**

Para que o resultado de uma divisão apresente a parte fracionada (Ex. 2.5), é necessário que esse resultado seja do tipo "float" ou "double". Caso contrário, desprezará a parte fracionada e apresentará somente a parte inteira (resultado = 2 e não 2.5, no caso de uma divisão de 5/2, por exemplo).





## ***Operadores Relacionais***

Uma expressão relacional é uma comparação realizada entre dois valores de mesmo tipo, tendo como resposta sempre um valor booleano (verdadeiro ou falso - "1" ou "0", no caso da Linguagem C). Esses valores são representados na relação por meio de constantes, variáveis ou expressões aritméticas.

Os operadores relacionais são:

Símbolo	Descrição
==	Igual
!=	Diferente
<=	Menor ou igual
>=	Maior ou igual
>	Maior que
<	Menor que

Exemplo de relações:

$X == 1 \quad y == 2 \quad z == 5$

$X * X + Y > Z$

Substituindo, temos:  $1 * 1 + 2 > 5$

$1 + 2 > 5$

$3 > 5$

Resultado desta expressão: FALSO



## ***Operadores Lógicos***

Uma expressão lógica serve para combinar resultados de expressões aritméticas e/ou relacionais, variáveis e/ou constantes, retornando verdadeiro ou falso.

Exemplo de operadores lógicos, matematicamente conhecidos:

E	&&
Ou	
Não	!

**E / &&** Uma expressão && (E) é verdadeira se todas as condições forem verdadeiras.

**OU/||** Uma expressão || (OU) é verdadeira se pelo menos uma condição for verdadeira.

**NÃO/!** Uma expressão NOT (NÃO) inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

<b>Tabela E (&amp;&amp;)</b>	<b>Tabela OU (  )</b>	<b>Tabela NÃO (!)</b>
V e V = V	V ou V = V	Não V = F
V e F = F	V ou F = V	Não F = V
F e V = F	F ou V = V	
F e F = F	F ou F = F	

As prioridades entre os operadores são:

1º - ( )

2º - funções

3º - Não

4º - \*, /, %, &&

5º - +, -, ||

6º - =, !=, <, <=, >, >=.

**Observação:** em caso de empate entre as prioridades, resolver da esquerda para a direita.



## EXERCÍCIOS - Identificadores e tipos de variáveis

1. Identifique o tipo dos dados:

- (a) inteiro/int;
- (b) real/float;
- (c) lógico/boolean;
- (d) cadeia ou literal/char[ ]

- |                |           |           |               |
|----------------|-----------|-----------|---------------|
| ( ) verdadeiro | ( ) 'c*d' | ( ) falso | ( ) '1 2 3 4' |
| ( ) 'aula'     | ( ) 897   | ( ) '345' | ( ) -18.589   |
| ( ) -0.342     | ( ) 35.23 | ( ) -23   | ( ) 'Maria'   |

2. Indique os identificadores como válidos ou inválidos:

- (a) identificador válido
- (b) identificador inválido

- |                   |            |                    |
|-------------------|------------|--------------------|
| ( ) ano           | ( ) ai!    | ( ) 3/1            |
| ( ) media_salario | ( ) A15B34 | ( ) nome-aluno     |
| ( ) média         | ( ) 'aula' | ( ) 5 <sup>a</sup> |

3. Faça a declaração de 2 variáveis do tipo inteiro, 2 do tipo real, 2 de um único caractere e 2 do tipo cadeia de caracteres.

4. Indique qual o resultado das expressões aritméticas abaixo:

Sendo:  $x = 6.0$      $y = 2$      $z = 4.0$      $a = 8$      $b = 7.5$      $c = 7.7$   
            $d = 12$      $p = 4$      $q = 3$      $r = 10$      $s = 2.5$

=====

- |                      |                              |
|----------------------|------------------------------|
| a) $x + y - z * a =$ | f) $((z / a) + b * a) - d =$ |
|----------------------|------------------------------|



- |               |                                 |
|---------------|---------------------------------|
| b) $d / y =$  | g) $100 * (q / p) + r =$        |
| c) $d \% y =$ | h) $p * (r \% q) - q/2$         |
| d) $y / d =$  | i) $\text{raiz}(r - (q * q)) =$ |
| e) $y \% d =$ | j) $(a + r) * r =$              |

5. Dadas as informações abaixo, informe qual o valor das relações (V ou F):

- a)  $a = 2.0, \quad b = 9.0, \quad \text{nome} = \text{'ana'}, \quad \text{profissao} = \text{'advogado'}$   
 $a + 1 \geq b * 0.5 \quad ( )$   
 $\text{nome} != \text{'ana'} \quad ( )$   
 $\text{profissao} = \text{'médico'} \quad ( )$
- b)  $a = 6.0, \quad b = 121.0, \quad \text{nome} = \text{'pedro'}, \quad \text{profissao} = \text{'médico'}$   
 $a + 1 \geq \text{raiz}(b) \quad ( )$   
 $\text{nome} != \text{'ana'} \quad ( )$   
 $\text{profissao} = \text{'médico'} \quad ( )$
- c)  $x = 3, \quad y = 4, \quad z = 16, \quad \text{nome} = \text{'maria'}, \quad \text{resultado} = \text{verdadeiro}$   
 $(x + y > z) \ \&\& \ (\text{nome} = \text{'maria'}) \quad ( )$   
 $(\text{resultado}) \ || \ (y \geq x) \quad ( )$   
 $(\text{not resultado}) \ \&\& \ (z \text{ div } y + 1 = x) \quad ( )$   
 $(\text{nome} = \text{'jósé'}) \ \&\& \ (x + y + z < (y * y)) \quad ( )$

## IV – COMANDOS BÁSICOS

### *Comando de Atribuição*

Este comando é utilizado para atribuir valores a variáveis e, em português (português estruturado), é representado por  $\leftarrow$  (seta da direita para esquerda).

Exemplo:  $X \leftarrow 10;$



Em "C", a atribuição é representada pelo sinal "=" (igual).

**ATENÇÃO!!!**

O sinal de igualdade em "C" é representado por "==" (dois sinais de igual)

Exemplos de comandos de atribuição em "C":

Cor = 'verde';

Teste = 0;

Media = (n1 + n2) / 2;

### ***Comandos de Entrada e Saída***

#### **Comando de Entrada**

O comando de entrada **LEIA** é utilizado para receber dados do teclado em Português estruturado.

É o comando que permite que o usuário digite dados, possibilitando um "diálogo com o computador". O dado digitado é armazenado na variável indicada no comando.

Lembre-se de que o nome de uma variável representa uma posição de memória.

Sintaxe:

Leia (nome de uma variável);

Exemplo:

Leia (n);

#### **Comando de Saída**

É o comando responsável por enviar um resultado, uma informação ao usuário. O valor de cada variável é buscado na memória e mostrado em um dispositivo de saída. Através desse comando o computador pode emitir os resultados e outras mensagens para o usuário através da tela do computador (monitor) ou uma impressora.



### Sintaxe:

Escreva (expressão ou variável ou constantes);

### Exemplos:

Escreva (' Aprender lógica com esta apostila ficou bem mais fácil!');

Escreva (' Digite o seu nome: ');

Escreva ( A + B );

Escreva ('A média das notas é = ', media);



### Comandos de Entrada e Saída em "C"

- Existem vários tipos de entrada e saída de dados pelo console. Por exemplo: gets(), puts(), scanf(), printf() e outros.
- Os mais comuns são as funções scanf() e printf(), pois podem ser usadas para qualquer tipo de dados existentes em C, além da facilidade para formatar esses dados.
- Se quisermos ler uma string, podemos também usar a função gets(). A função gets() coloca o terminador nulo na string, quando você aperta a tecla "Enter".
- Ela aceita escrever strings com espaço entre as palavras (nomes compostos).

Código especial	Descrição
\n	Nova linha
\t	Tab
\b	Retrocesso
\"	Aspas
\\	Barra
\f	Salta formulário
\0	Nulo

- A função puts() pode receber os códigos de barra invertida e é considerada muito mais rápida do que o printf().
- O único problema é que a puts() trabalha apenas com string de caracteres enquanto a printf() trabalha com todos os tipos de dados.



## Exemplo - uso dos comando gets() e puts():

```

1  /* Este programa lê uma string pelo comando gets
2     e mostra a string digitada pelo comando puts*/
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  int main() {
8      char nome[41];
9      puts("Digite uma string de no maximo 40 caracteres ");
10
11     gets(nome);
12     puts(nome);
13
14     system("pause");
15
16     return (0);
17 }
18
  
```



## Função scanf() em "C"

Por trabalhar com todos os tipos de dados, para a leitura de dados a partir do teclado, usaremos a função "scanf()". Para que ela funcione corretamente, deveremos observar alguns aspectos importantes:

1. Precisamos indicar, entre aspas, qual será o tipo de dados lido:

Símbolo	Tipo	Para que dados usar
%c	char	Para dados de um único caractere.
%d	int	Para números inteiros, ou seja, sem parte fracionada.
%f	float ou double	Para números "quebrados", ou seja, fracionados, como 3.5.
%s	char [ ]	Para cadeia de caracteres (textos)

2. Após o encerramento das aspas, deveremos indicar qual será o endereço da variável que deverá guardar o valor informado pelo usuário. Isso será feito usando o símbolo "&", seguido do nome da variável (Ex. &nota1)

Observando esses dois passos, a leitura de uma variável chamada "x", que armazenará um número inteiro, ficará assim: **scanf("%d", &x);**



## Função printf() em "C"

Para exibir uma mensagem na tela, usaremos a função "printf()", que deverá seguir os seguintes padrões:

1. Para exibir apenas um texto:

```
printf("Texto desejado");
```

2. Para exibir o resultado de uma soma, armazenada em uma variável do tipo inteiro, chamada "resultado":

```
printf("Soma = %d", resultado);
```

Observe que o comando "%d" será substituído, ao executar o programa, pelo valor armazenado na variável resultado.

### ATENÇÃO!!!

Para os outros tipos de dados, consulte a tabela de símbolos na seção anterior: "Comando de Entrada em 'C'".

**Exemplo - uso dos comando printf() e scanf():**





```
1 /* Este programa lê uma string pelo comando scanf()
2    e mostra a string digitada pelo comando printf() */
3
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 int main(){
8     char nome[40];
9     printf("Digite o seu nome ");
10
11     scanf("%s", &nome);
12     printf("%s, acabou de ganhar um super curso de Logica!!!\n", nome );
13
14     system("pause");
15
16     return(0);
17 }
```



## Cadeia de caracteres em "C"

Como você já deve ter percebido, na Linguagem "C", não existe um tipo "string" para cadeia de caracteres, usa-se um vetor de char.

Explicando melhor:

- Uma string é um conjunto ordenado de caracteres que pode ser armazenado sob forma de um vetor, um ponteiro.
- No C uma string é um vetor de caracteres terminado com um caractere nulo.
- O caractere nulo é um caractere com valor inteiro igual a zero (código ASCII igual a 0).
- O terminador nulo também pode ser escrito usando a convenção de barra invertida do C como sendo '\0'.

Para declarar uma string, podemos usar o seguinte formato geral:

```
char nome_da_string[tamanho];
```

```
Ex.: char nomeAluno[81];
```

Isto declara um vetor de caracteres (uma string) com número de posições igual a tamanho. Note que, como temos que reservar um caractere para ser o terminador nulo, temos que



declarar o comprimento da string como sendo, no mínimo, um caractere maior que a maior string que pretendemos armazenar.

Vamos supor que declaremos uma string de 7 posições e coloquemos a palavra João nela.

Nesse caso, teremos:

J	O	Ã	O	\0		
---	---	---	---	----	--	--

No caso acima, as duas células não usadas têm valores indeterminados.

Isso acontece porque o "C" não inicializa variáveis, cabendo ao programador essa tarefa.

Portanto as únicas células que são inicializadas são as que contêm os caracteres 'J', 'o', 'a', 'o' e '\0'.

**Espero que agora já esteja pronto(a) para iniciarmos nossos exercícios... Afinal é fazendo que se aprende, não é mesmo?**



## Primeiro Exemplo – Português Estruturado e "C"

Algoritmo que dá boas-vindas ao programador.

### Português Estruturado

Programa exemplo1;

Var

nome: literal;

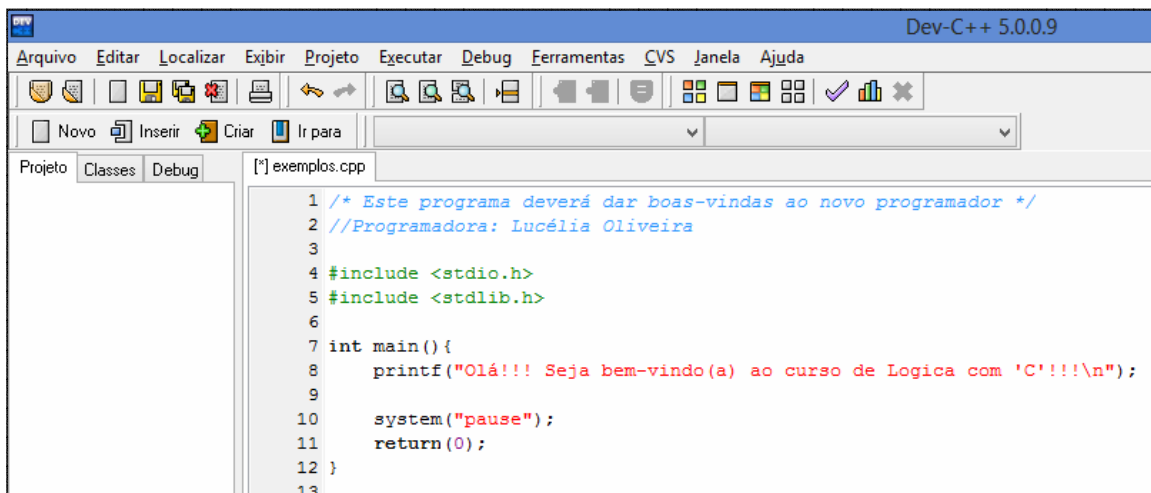
Início

    Escreva('Olá!!! Seja bem- vindo(a) ao curso de Logica com  
    'C'!!! ');

Fim.



em "C"



***Vamos compreender o programa que acabamos de construir:***

A linha `#include <stdio.h>` diz ao compilador que ele deve incluir o arquivo-cabeçalho `stdio.h`. Neste arquivo existem declarações de funções úteis para entrada e saída de dados



(std = standard, padrão em inglês; io = Input/Output, entrada e saída ==> stdio = Entrada e saída padronizadas). Toda vez que você quiser usar uma destas funções deve-se incluir este comando, como fizemos também na linha: `#include <stdlib.h>`, que nos permitiu usar o comando: `system("pause");`. O C possui diversos Arquivos-cabeçalho.

Quando construímos um programa, uma boa ideia é usar comentários que ajudem a elucidar o seu funcionamento. No caso acima, temos um comentário: `/* Um Primeiro Programa */`. O compilador C desconsidera qualquer coisa que esteja começando com `/*` e terminando com `*/`. Um comentário pode, inclusive, ter mais de uma linha. Quando o comentário tiver apenas uma linha, basta colocar `//` no início da linha que deseja comentar.

A linha `int main()` indica que estamos definindo uma função de nome `main`. Todos os programas em C têm que ter uma função `main`, pois é esta função que será chamada quando o programa for executado.

O conteúdo da função `main()` é delimitado por chaves `{ }`. O código que estiver dentro das chaves será executado sequencialmente quando a função for chamada. A palavra `int` indica que esta função retorna um inteiro.

A última linha do programa, `return(0);`, indica o número inteiro que está sendo retornado pela função, no caso o número 0.

A única coisa que o programa realmente faz é chamar a função `printf()`, passando a string (uma string é uma sequência de caracteres, como veremos brevemente) `"Olá!!! Seja bem-vindo(a) ao curso de Logica com 'C'!!!\n"` como argumento. É por causa do uso da função `printf()` pelo programa que devemos incluir o arquivo-cabeçalho `stdio.h`. A função `printf()`, neste caso, irá apenas colocar a string na tela do computador. O `\n` é uma constante chamada de constante barra invertida.

No caso, o `\n` é a constante barra invertida de "new line" e ele é interpretado como um comando de mudança de linha, isto é, após imprimir `"Olá!!! Seja bem-vindo(a) ao curso de Logica com 'C'!!!"` o cursor passará para a próxima linha. É importante observar também que os comandos do C terminam com `;"`.



## EXERCÍCIOS - Estrutura Sequencial

1. Faça um algoritmo que leia dois números, calcule e imprima a soma desses dois números.
2. Faça um programa que receba dois números reais, calcule e mostre a subtração do primeiro número pelo segundo.
3. Faça um programa que receba dois números inteiros, calcule e imprima a divisão do primeiro número pelo segundo.
4. Faça um programa que leia um número e informe a metade e o dobro desse número.
5. Escreva um programa que receba um número via teclado e informe em seguida a metade e o quadrado desse número.
6. Escrever um programa que permita receber o nome e a idade de uma pessoa e em seguida, informar o nome digitado e a idade da pessoa daqui a 30 anos.
7. Faça um programa que leia três notas de um aluno, calcule e imprima a média aritmética entre essas notas.
8. Faça um programa que receba dois números inteiros, calcule e imprima:
  - soma dos dois números;
  - subtração do primeiro pelo segundo;
  - subtração do segundo pelo primeiro;
  - produto dos dois números;
  - divisão do primeiro pelo segundo;
  - quociente inteiro da divisão do primeiro pelo segundo;



- resto da divisão do primeiro pelo segundo.

9. Faça um programa que receba quatro números inteiros, calcule e mostre a soma desses números.
10. Faça um programa que receba três notas e seus respectivos pesos. Calcule e mostre a média ponderada dessas notas.
11. Faça um programa que receba o valor do salário de um funcionário. Calcule e mostre o novo salário, sabendo-se que este sofreu aumento de 25%.
12. Faça um programa que receba o valor do salário de um funcionário e o percentual de aumento, calcule e mostre o valor do aumento e o novo salário.
13. Faça um programa que receba o valor do salário-base de um funcionário, calcule e mostre o salário a receber, sabendo-se que este funcionário tem gratificação de 5% sobre o salário-base e paga imposto de 7% sobre o salário-base.
14. Faça um programa que receba o valor do salário-base de um funcionário, calcule e mostre o salário a receber, sabendo-se que este funcionário tem gratificação de R\$ 1.000,00 e paga imposto de 10% sobre o salário-base.



## V – ESTRUTURAS DE SELEÇÃO

### *Conceitos*

Os algoritmos até agora seguiram um mesmo padrão: entrava-se com dados, esses eram processados e alguma informação era mostrada na tela, como resultado final.

Dessa forma, o computador mais parecia uma máquina de calcular. O aprendizado de novos conceitos, como a estrutura de seleção, nos dará uma visão maior da complexidade de tarefas que ele poderá executar.

Um exemplo do nosso dia-a-dia: imagine-se diante de um caixa eletrônico e suponha que sua senha seja 1234:

Na tela aparece a mensagem:

- Digite sua senha : ■

E o cursor ( ■ ou | ) fica piscando:

Você digita os algarismos da sua senha.

Neste momento, a Unidade Lógica e Aritmética (um dos componentes da CPU) verifica se os números que você digitou são iguais a **1234**. Caso tenha sido, a transação bancária continua; mas, se você digitou algo diferente, aparece na tela a mensagem: **SENHA INVÁLIDA**.

Podemos constatar que esta estrutura faz parte do nosso cotidiano:

- Se eu não tiver prova, vou ao clube; senão vou estudar.
- Se eu tiver aumento, troco de carro; senão espero o 13º salário.
- Se minha média for maior ou igual a sete, passo direto; senão faço exame final.

A única coisa diferente é a forma como iremos escrevê-la, vamos adaptar os algoritmos para uma linguagem específica de um modo formal, obedecendo as regras da linguagem.



## Sintaxes:

### Seleção Simples

**Se** condição **então**

**Início**

Comando (s);

**Fim;**



em "C"

### Seleção Composta

**Se** condição **então**

**Início**

Comando (s);

**Fim**

**Senão**

**Início**

Comando (s);

**Fim;**

### Seleção Simples

**if** (condição)

{

Comando (s);

}

### Seleção Composta

**If** (condição)

{

Comando (s);

}

**else**

{

Comando (s);

}

**Exemplo:** Programa que lê um número e mostra uma mensagem dizendo se o número digitado é ou não múltiplo de 5.





## Português Estruturado:

Programa ExemploSe;

Var

num, resto: inteiro;

Início

Escreva ('Digite um número: ');

Leia (num);

resto <- num mod 5;

Se resto = 0 então

Escreva (num, ' é múltiplo de 5')

Senão

Escreva (num, ' não é múltiplo de 5');

Fim.



em "C"

```
1 /* Programa que lê um número e mostra uma mensagem dizendo se
2 o número digitado é ou não múltiplo de 5 */
3 //Programadora: Lucélia Oliveira
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(){
9     //Declaração da variável
10    int num;
11    //Entrada de dados
12    printf("Informe o valor desejado: ");
13    scanf("%d", &num);
14    //Processamento
15    if(num % 5 == 0){
16        printf("%d é múltiplo de 5 \n", num);
17    }else
18        printf("%d não é múltiplo de 5 \n", num);
19
20    system("pause");
21    return(0);
```

### **ATENÇÃO!!!**

O início "{" e fim "}" são obrigatórios somente quando há mais de um comando dentro do *if/else*. Porém, poderão ser usados sempre que desejar.



## EXERCÍCIOS – Estrutura de Seleção

1. Faça um programa que leia um número e informe se o dobro dele é maior que 35.
2. Faça um programa que leia um número e informe se a metade dele é menor que 12.
3. Faça um programa que leia dois números e informe apenas se o primeiro é maior que o segundo.
4. Faça um programa que leia dois números e informe se o primeiro é igual ao segundo.
5. Faça um programa que leia dois números e informe se o primeiro é maior, menor ou igual ao segundo.
6. Faça um programa que leia três números e informe qual dos três é o maior.
7. Fazer um programa que receba um número e mostre se ele é positivo, negativo ou nulo.
8. Elabore um programa que leia um número e informe se ele ‘é par’ ou ‘é ímpar’.
9. Faça um programa que leia a altura e o sexo de uma pessoa, calcule e imprima seu peso ideal, utilizando as seguintes fórmulas:  
Para homens:  $(72.7 * h) - 58$   
Para mulheres:  $(62.1 * h) - 44.7$  (h = altura)
10. Faça um programa que receba quatro notas de um aluno, calcule e imprima a média aritmética das notas e a mensagem de aprovado para média superior ou igual a 7.0 ou a mensagem de reprovado para média inferior a 7.0.



11. Faça um programa que calcule e imprima o salário reajustado de um funcionário de acordo com a seguinte regra:
  - salários até R\$ 300,00, reajuste de 50%
  - salários maiores que R\$ 300,00, reajuste de 30%
12. A prefeitura do Rio de Janeiro abriu uma linha de crédito para os funcionários municipais. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Fazer um algoritmo que permita entrar com o salário bruto e o valor da prestação e informar se o empréstimo pode ou não ser concedido.
13. Ler um número qualquer e exibir na tela uma mensagem indicando se ele é positivo, negativo ou nulo (zero). Se ele for positivo, exibir também a raiz quadrada deste número. Se ele for negativo você deve escrever uma mensagem dizendo 'Não é possível calcular a raiz deste número'.
14. Ler um número inteiro e exibir na tela a mensagem 'Par' se ele for um número par, ou 'Ímpar' se ele for um número ímpar.
15. Faça um programa que receba três notas de um aluno, calcule e mostre a média aritmética e as mensagens de acordo com a tabela abaixo. Para os alunos de exame, calcule e mostre a nota que deverá ser tirada no exame para aprovação, considerando que a média no exame é de 6,0.

MÉDIA	MENSAGEM
[ 0,0 a ] 3,0	Reprovado
[ 3,0 a ] 7,0	Exame
[ 7,0 a 10,0 ]	Aprovado

16. Faça um programa que receba três números e os mostre em ordem crescente.
17. Dados três valores X, Y e Z, verificar se eles podem ser os lados de um triângulo e, se forem, verificar se é um triângulo equilátero, isósceles ou escaleno. Se eles não formarem um triângulo informar ao usuário tal situação. Considerar que:

Condição para ser triângulo: O comprimento de cada lado de um triângulo é menor que a soma dos outros dois lados.

Chama-se triângulo equilátero o triângulo que tem os três lados iguais;



Chama-se triângulo isósceles o que tem o comprimento de dois lados iguais;  
Chama-se triângulo escaleno o triângulo que têm os três lados diferentes.

18. Faça um programa que receba o código de origem de um produto e informe a sua procedência. A procedência obedece a seguinte tabela:

<b>Código de origem</b>	<b>Procedência</b>
1	Sul
2	Norte
3	Leste
4	Oeste
5 ou 6	Nordeste
7, 8 ou 9	Sudeste
10 até 20	Centro-oeste
21 até 30	Noroeste



## Alternativa de Múltiplas Escolhas - switch/case

O comando *switch* possui certa semelhança com a estrutura *if/else*. A principal diferença é que a estrutura *switch* não aceita expressões, aceita apenas variáveis.

O *switch* verifica a variável e executa o(s) comando(s) correspondente(s), se ele corresponder ao valor especificado no comando *case*.

O uso do *default* é opcional e será executado apenas se a variável, que está sendo testada, não for igual a nenhuma das variáveis testadas.

O comando *break* faz com que o *switch* seja interrompido assim que um dos *cases* for executado. Mas ele não é essencial ao comando *switch*. Porém, se após a execução da declaração não houver um *break*, o programa continuará executando os comandos seguintes até que o fim do comando *switch* ou outro *break* seja encontrado.

O *switch* é uma boa alternativa para quando o programa tiver muitos **SEs**, deixando-o com uma estrutura mais amigável.

### Sintaxe:

#### Português Estruturado

**Caso** <nome da variável> **seja**

Alvo 1: comando 1;

Alvo 2: comando 2;

Alvo n: comando n;

**Fim;**

Ou

**Caso** <nome da variável> **seja**

Alvo 1: comando 1;

Alvo 2: comando 2;

Alvo 3: comando 4;

**Senão**

comando 5;

**Fim;**



em "C"

```
switch(variável){  
    case 1: comando 1; break;  
    case n: comando n; break;  
    default: comando;  
}
```

### **Exemplo:**

Este programa lê um valor e mostra-o por extenso, se for 1, 2 ou 3. Se for um número maior que 3, informará que está fora do limite.

### **Português Estruturado**

```
Programa Exemplo;  
var  
    num: integer;
```

```
Início  
    Escreva('Informe o valor desejado: ');  
    Leia (num);
```

#### **Caso num seja**

```
    1: Escreva ('Um');  
    2: Escreva ('Dois');  
    3: Escreva ('Três');
```

Senão

```
    Escreva('Valor inválido!');
```

```
Fim;          //Fim do Caso
```

```
Fim.
```



em "C"

```
C:\Users\Poliana\Desktop\Poliana\ProgramasC\SwitchCaseFonte\pSwitchCase.cpp - Dev-C++ 5.5.3
Arquivo  Editar  Localizar  Exibir  Projeto  Executar  Ferramentas  CVS  Janela  Ajuda
(global)
pSwitchCase.cpp
1  /* Programa que lê um valor e mostra por extenso se corresponder a 1, 2 ou 3.
2  Se for diferente desses números, informa que o valor está fora do limite */
3  //Programadora: Lucélia Oliveira
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  int main()
9  {
10     //Declarando variável
11     int num;
12     //Entrada de dados
13     printf("Valor: ");
14     scanf("%d", &num);
15     //Processamento e saída
16     switch(num)
17     {
18         case 1: printf("Um \n"); break;
19         case 2: printf("Dois \n"); break;
20         case 3: printf("Três \n"); break;
21         default: printf("Valor fora do limite. \n");
22     }
23     system("pause");
24     return(0);
25 }
```

Compilador Recursos Registro do Compilador Depurador Resultados da Busca

Linha: 1    Col: 1    Sel: 0    Linhas: 25    Tamanho: 624    Inserir    Finalizada leitura



## EXERCÍCIOS – Alternativa de Múltiplas Escolhas

1. Faça um programa que receba o código de origem de um produto e informe a sua procedência. (**Obs.:** este é o exercício 14 da estrutura de seleção **se**, mas, agora ele será feito utilizando o escolha caso.) A procedência obedece a seguinte tabela:

Código de origem	Procedência
1	Sul
2	Norte
3	Leste
4	Oeste
5 ou 6	Nordeste

2. Escrever um algoritmo que leia um peso na Terra e o número de um planeta e imprima o valor do seu peso neste planeta. A relação de planetas é dada a seguir juntamente com o valor das gravidade relativas à Terra:

<i>Nº</i>	<i>gravidade relativa</i>	<i>Planeta</i>
1	0.37	Mercúrio
2	0.88	Vênus
3	0.38	Marte
4	2.64	Júpiter
5	1.15	Saturno
6	1.17	Urano

Para calcular o peso no planeta escolhido use a seguinte fórmula:

Peso no Planeta = Peso \* gravidade.





## VI – ESTRUTURAS DE REPETIÇÃO

Vamos supor que nossos programas precisarão ser executados mais de uma vez e, para que não tenhamos que reescrever trechos idênticos que aumentariam consideravelmente o tamanho do programa, utilizaremos as estruturas de repetição.

### ***Comando Enquanto (While)***

Esta estrutura é recomendada quando o número de repetições for desconhecido. Para que funcione corretamente é necessário observar:

- É necessário um teste (uma condição) para interromper a repetição, esta estrutura testa a condição e só executa o que está dentro dela, se for verdadeira.
- A variável que testa a condição deverá ter seu valor atribuído no início do programa através de um comando de leitura ou de atribuição, antes da estrutura do enquanto e dentro da estrutura, como último comando.
- O Enquanto começa com "{" (início) e termina com "}" (fim).
- Pode-se usar outras estruturas dentro do "Enquanto" (while), como: if, switch/case, etc.

### **Sintaxe:**

#### **Português Estruturado:**

```
ENQUANTO condição FAÇA  
INÍCIO  
    Comando 1;  
    Comando 2;  
    Comando n;  
FIM;
```

#### **C:**

```
WHILE (condição)  
{  
    Comando 1;  
    Comando 2;  
    Comando n;  
}
```



**Exemplo:** Faça um programa que leia vários números e imprima a metade de cada número, o programa termina quando o 0 (zero) for digitado:

### **PORTUGUÊS ESTRUTURADO:**

```
PROGRAMA Enquanto1;  
VAR  
num: INTEIRO;  
metade:REAL;  
INÍCIO  
ESCREVA ('Digite um número inteiro ');  
LEIA (num);  
ENQUANTO num <> 0 FAÇA  
INÍCIO  
metade <- num/2;  
ESCREVA ('A metade de ', num, ' é ', metade);  
ESCREVA ('Digite outro número ou zero para sair do programa: ');  
LEIA (num);  
FIM;  
FIM.
```



em "C"



Lógica de Programação com "C"  
Professoras: Lucélia Oliveira e Poliana Ribeiro Tolentino

```
1 /* Faça um programa que leia vários números e imprima a metade de cada número,
2    o programa termina quando o 0 (zero) for digitado: */
3 //Programadora: Lucélia Oliveira
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main() {
9     //Declaração da variável
10    int num;
11    float metade;
12    //Entrada de dados
13    printf("Digite um numero inteiro: ");
14    scanf("%d", &num);
15    //Processamento e saída
16    while (num != 0) {
17        metade = (float)num/2;
18        printf("A metade de %d e = %0.1f", num, metade);
19        printf("\nDigite outro numero ou zero para sair do programa: ");
20        scanf("%d", &num);
21    }
22
23    system("pause");
24    return(0);
25 }
```



### Comando Repita (do- while)

Estrutura recomendada quando o número de repetições for desconhecido, sendo necessária uma chave (um teste) para interromper a repetição.

Sua diferença em relação ao "while" é que ela testa a condição ao final, significando que ela executa o trecho pelo menos uma vez.

Essa estrutura existe para situações em o programador deseje que seja realizada pelo menos uma execução.

#### Sintaxe:

##### Português Estruturado:

faça  
    comandos;  
enquanto condição;

##### Pascal

do{  
    comandos;  
}while condição;



em "C"

```
Dev-C++ 5.0.0.9
Arquivo  Editar  Localizar  Exibir  Projeto  Executar  Debug  Ferramentas  CVS  Janela  Ajuda

[Ícone] Novo [Ícone] Inserir [Ícone] Criar [Ícone] Ir para

Projeto  Classes  Debug  [Ícone] exemplos.cpp

1  /* Faça um programa que leia vários números e imprima a metade de cada número,
2     o programa termina quando o 0 (zero) for digitado: */
3  //Programadora: Lucélia Oliveira
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  int main(){
9      //Declaração da variável
10     int num;
11     float metade;
12     //Entrada de dados
13     printf("Digite um numero inteiro: ");
14     scanf("%d", &num);
15     //Processamento e saída
16     do{
17         metade = (float)num/2;
18         printf("A metade de %d e = %.1f", num, metade);
19         printf("\nDigite outro numero ou zero para sair do programa: ");
20         scanf("%d", &num);
21     }while (num != 0);
22
23     system("pause");
24     return(0);
25 }
26
```



## EXERCÍCIOS – Comando Enquanto (WHILE)

1. Entrar com vários números e imprimir quantos números foram digitados.
2. O IBGE fez uma pesquisa, ele quer saber quantos dentistas foram entrevistados. Fazer um programa que pergunte aos entrevistados qual a sua profissão e ao final diga quantas pessoas foram entrevistadas e quantas são dentistas (considerar dentista, Dentista e DENTISTA).
3. Dado um país A, com 5.000.000 de habitantes e uma taxa de natalidade de 3% ao ano, e um país B com 7.000.000 de habitantes e uma taxa de natalidade de 2% ao ano, calcular e imprimir o tempo necessário para que a população do país A ultrapasse a população do país B.
4. Uma pousada estipulou o preço da diária em R\$ 40,00 e mais uma taxa de serviços diários de:
  - R\$ 15,00, se o número de dias for menor que 10;
  - R\$ 8,00, se o número de dias for maior ou igual a 10.Criar um algoritmo que imprima o nome, o valor da conta de cada cliente e ao final o total arrecadado pela pousada.
5. Criar um algoritmo que entre com vários números inteiros e positivos e informe a quantidade de números múltiplos de 3 (três). O programa será encerrado quando o usuário digitar 0 (zero) ou menos.
6. Criar um algoritmo que entre com vários números inteiros e positivos e informe a média dos números múltiplos de 3 (três). O programa será encerrado quando o usuário digitar 0 (zero) ou menos.



7. Uma fábrica produz e vende vários produtos e para cada um deles tem-se o nome, quantidade produzida e quantidade vendida. Criar um algoritmo que mostre:
  - Para cada produto, nome, quantidade no estoque e uma mensagem se o produto tiver menos de 50 itens no estoque.
8. Faça um programa que leia a idade e a altura de várias pessoas. Calcule e informe a média das alturas das pessoas com mais de 50 anos. Para encerrar o programa digite zero para idade.
9. Faça um programa que leia vários números, quando o zero for digitado o programa será finalizado. Mostre no final desse programa a soma dos números positivos, a soma dos negativos e a soma total dos positivos e negativos juntos.
10. O GDF realizou uma pesquisa entre vários habitantes do DF, coletando dados sobre o rendimento familiar e o número de filhos de cada família. O GDF quer saber:
  - A média dos rendimentos da população;
  - Média do número de filhos.
11. Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas no seu quadro de funcionários. Suponha que você seja o programador dessa empresa, criar um programa que leia para cada candidato a idade, o sexo e se tem experiência no serviço (S ou N). Para encerrar o programa, digite zero para idade. Calcule e escreva:
  - O número de candidatos do sexo feminino;
  - O número de candidatos do sexo masculino;
  - A idade média dos homens que já tem experiência no serviço.
12. Faça um programa que receba vários números positivos ou negativos, terminada por zero. O programa deve fornecer como saída, a soma dos números positivos e a soma dos números negativos.



13. Uma empresa classifica seus funcionários em três níveis de acordo com um índice de produtividade. São eles (1) Excelente, (2) Bom e (3) Regular. Cada nível acrescenta um abono ao salário base do funcionário, de acordo com a seguinte tabela:
- Excelente      80% do salário base;
  - Bom             50% do salário base;
  - Regular        30% do salário base.
- O programa deve ler a matrícula do funcionário, seu salário base e seu nível de abono. Calcular e imprimir o salário a ser pago (salário a ser pago é = salário base + abono). O programa será encerrado quando for digitado 0 para matrícula.
14. Faça um programa que leia os dados de vários alunos, contendo o número da matrícula, as três notas e a frequência. Calcule e mostre: para cada aluno o número de matrícula, a nota final e a mensagem (aprovado ou reprovado); a maior e a menor nota da turma; o total de alunos reprovados;
15. Faça um programa que receba a idade, a altura e o peso de várias pessoas. Calcule e mostre:
- A quantidade de pessoas com idade superior a 50 anos;
  - A média das alturas das pessoas com idade entre 10 e 20 anos;
  - A porcentagem de pessoas com peso inferior a 40 quilos entre todas as pessoas analisadas.
16. Uma empresa deseja aumentar seus preços em 20%. Faça um programa que leia o código, o preço de custo de vários produtos e que calcule o novo preço de cada um deles. Calcule também a média de preços com e sem aumento. Mostre o código e o novo preço de cada produto e, no final, as médias. A entrada de dados deve terminar quando for recebido um código de produto menor ou igual a zero.
17. Faça um programa que apresente um menu de opções para cálculo das seguintes operações entre dois números: adição, subtração, multiplicação e divisão. O programa deve permitir a escolha da operação desejada, a entrada dos números, a exibição do



resultado e a volta ao menu de opções. O programa só termina quando for escolhida a opção de saída.

### **Comando Para (For)**

Essa estrutura de repetição é utilizada quando se sabe o número de vezes que um trecho do programa deve ser repetido.

Sintaxe:

#### **Português Estruturado**

```
PARA a<-valor inicial ATÉ valor  
final FAÇA  
    INÍCIO  
        Comando 1;  
        Comando 2;  
    FIM;
```

#### **C**

```
FOR (a = valor inicial; a <=  
valor final; a++)  
{  
    Comando 1;  
    Comando 2;  
}
```

#### **Observações:**

1. O identificador (a variável "**a**" no exemplo anterior) tem que ser declarada do tipo inteiro (int) ou char.
2. A variável que controla a repetição poderá ser impressa se precisarmos dela para numerar uma lista, posicionar, etc.
3. A variável que controla a repetição **jamaís** deverá aparecer num comando de **leitura** dentro do bloco de repetição.





### Exemplo:

Programa para escrever os números de 1 até 10:

### Português Estruturado

```
Programa para1;  
Var  
    I : Inteiro;  
Início  
    Para i <- 1 até 10 faça  
        Início  
            Escreva (' - ', i);  
        fim;  
Fim.
```



em "C"

```
1 /* Programa para escrever os números de 1 até 10: */  
2 //Programadora: Lucélia Oliveira  
3  
4 #include <stdio.h>  
5 #include <stdlib.h>  
6  
7 int main(){  
8     //Processamento e saída  
9     for (int i = 1; i <= 10; i++){  
10         printf("%d - ", i);  
11     }  
12     printf("\n");  
13     system("pause");  
14     return(0);  
15 }  
16
```

## EXERCÍCIOS – Comando Para (For)

1. Imprimir todos os números de 100 até 1.
2. Criar um algoritmo que imprima todos os números pares no intervalo de 1 a 100.
3. Criar um algoritmo que entre com cinco números e imprima o quadrado de cada número.
4. Entrar com 10 números e imprimir a metade de cada número.
5. Criar um algoritmo que imprima todos os números de 1 até 100 e a soma deles.
6. Escreva um programa que receba a idade de 10 pessoas, calcule e imprima a quantidade de pessoas maiores de idade (idade  $\geq$  18 anos).
7. Entrar com nome, idade e sexo de 10 pessoas. Imprimir o nome se a pessoa for do sexo masculino e tiver mais de 21 anos.
8. Em uma eleição presidencial, existem três candidatos. Os votos são informados através de código. Os códigos utilizados são:  
1, 2, 3                       $\longrightarrow$  votos para os respectivos candidatos;  
0                                $\longrightarrow$  votos em branco  
outros códigos            $\longrightarrow$  votos nulos  
Escreva um programa que calcule e imprima:  
- total de votos para cada candidato;  
- total de votos nulos;  
- total de votos em branco.

9. Criar um algoritmo que entre com quatro notas de cada aluno de uma turma de 20 alunos e mostre:
  - A média de cada aluno;
  - A média da turma;
  - Quantos alunos foram aprovados (média 7.0)
10. Imprimir as tabuadas de multiplicar de 1 até 10.
11. Criar um algoritmo que deixe escolher qual a tabuada de multiplicar que se deseja imprimir
12. Um empresa está fazendo a estatística de seus funcionários, ela precisa saber quantas funcionárias têm com mais de 40 anos para encaminhá-las para exames de mamografia. Fazer um programa que leia o nome, a idade e o sexo de seus 10 funcionários e imprima o nome se for do sexo feminino e tiver mais de 40 anos.
13. Faça um programa que receba a idade de 10 pessoas. O programa deve calcular e mostrar a quantidade de pessoas com idade maior que 18 anos.
14. Faça um programa que mostre a tabuada de multiplicação (de 1 a 10) para os 6 primeiros números primos. Ao mudar de uma base para outra o programa deve mostrar uma mensagem ao usuário e aguardar que alguma tecla seja pressionada para então montar a tabuada para a próxima base.
15. Faça um programa que leia uma série de números positivos inteiros. Calcule e mostre o maior número digitado. A entrada de dados deve terminar quando um número negativo foi digitado.
16. Faça um programa que receba um número, calcule e mostre o fatorial desse número. Sabe-se que:  $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$ ;  
 $0! = 1$ , por definição.

17. Faça um programa que receba a idade de 15 pessoas, calcule e mostre a quantidade de pessoas em cada faixa etária.

Faixa etária	Idade
1	Até 15 anos
2	De 16 a 30 anos
3	De 31 a 45 anos
4	De 46 a 60 anos
5	Acima de 61 anos

## REVISÃO - Lógica de Programação

Veja o quanto você já evoluiu! Lembra-se do começo, quando tudo parecia muito mais difícil? Ainda há muitos desafios, mas já superamos muitos outros!

1. Faça um programa que permita entrar com a quantidade de pães e leite. Informe o valor da conta do cliente, sabendo que cada litro de leite custa R\$2,50 e cada pão custa R\$0,25.
2. Entrar com nome, sexo e idade de uma pessoa. Se a pessoa for do sexo masculino e tiver mais de 21 anos, exibir o nome e a mensagem: 'ACEITO'. Caso contrário, exibir o nome e a mensagem: 'NÃO ACEITO'.
3. Faça um programa que leia 50 números e informe, no final, a média aritmética dos números informados.
4. Faça um programa que leia o código de um produto do supermercado e informe que categoria o produto pertence de acordo com a relação a seguir:
  - 1 – limpeza;
  - 2 – alimentação;
  - 3 – bebidas;
  - 4 – têxtil;
  - 5 – açougue.
5. Criar um algoritmo que receba a idade e o estado civil (C – casado, S – solteiro, V viúvo, D – divorciado ou separado) de várias pessoas. Calcule e informe:
  - A quantidade de pessoas casadas;
  - A quantidade de pessoas solteiras;
  - A quantidade de viúvas com menos de 40 anos.
  - A média de idades.

## VII - ESTRUTURA HOMOGÊNEA: VETORES

Um vetor é um arranjo de elementos armazenados na Memória Principal, um após o outro, todos com o mesmo nome. A ideia é a mesma de uma matriz linha da matemática, isto é, várias colunas e uma linha.

2	4	5	8	12	3	56	34
1	2	3	4	5	6	7	8

A [2   4   5   8   12   3   56   34]

Esse é um vetor de 8 elementos, isto é, tem 8 variáveis, todas com o mesmo nome e diferentes por sua posição dentro do arranjo que é indicada por um **índice**.

Quando se tem somente uma linha, podemos omiti-la e colocar somente a coluna.

$A_1 = 2$ ,    $A_2 = 4$ ,    $A_3 = 5$ ,    $A_4 = 8$ ,    $A_5 = 12$ ,    $A_6 = 3$ ,    $A_7 = 56$ ,    $A_8 = 34$

Em português estruturado, representamos da seguinte forma:

$A[1] = 2$     $A[2] = 4$     $A[3] = 5$     $A[4] = 8$     $A[5] = 12$     $A[6] = 3$     $A[7] = 56$     $A[8] = 34$

Um algoritmo com vetor implica vários trechos para que possa funcionar corretamente. Esses trechos são **independentes**.

### Trecho de Dimensionamento

Para dimensionar um vetor, usamos o seguinte comando na declaração de variáveis:

## Português Estruturado

Var

nome: arranjo [dimensão] de tipo;

## C

tipo\_da\_variável nome [tamanho];

Onde dimensão, na prática, é o número de elementos:

No exemplo acima seria:

**Em Português Estruturado:** A: arranjo [1..8] de Inteiro;

**Em C:** int A[8];

## **ATENÇÃO!!!**

Na linguagem C a numeração começa sempre em zero. Isto significa que, no exemplo acima, os dados serão indexados de 0 a 7.

Para acessá-los vamos escrever:

A[0]            A[1]            ... A[7]

Mas ninguém o impede de escrever:

exemplo[30]            exemplo[103]

Se o programador observar os limites de validade para os índices, ele corre o risco de ter variáveis sobrescritas ou de ver o computador travar. Podem ocorrer *Bugs*.

## Trecho de Entrada de Dados

- normalmente, utiliza uma **estrutura de repetição**.
- se for a estrutura **para (for)**, deverá ter o **valor final** igual à **última posição** do vetor.
- se for a estrutura **enquanto (while)**, deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

## Português Estruturado

```
PARA L <- 1 ATÉ tamanho do vetor  
FAÇA  
INÍCIO  
    ESCREVA ('....');  
    LEIA (nome do vetor [ L ] );  
FIM;
```

## C

```
for (L = 1; L <= tam_vetor; L++)  
{  
    printf ('....');  
    scanf ("%d",&nome_vetor [L]);  
}
```

- Este trecho poderá ser incrementado com outros comandos como: estrutura de seleção (if), outro para (for), etc.

### Trecho de Saída de Dados

- Normalmente, utiliza uma **estrutura de repetição**.
- Se for a estrutura **para (for)**, deverá ter o **valor final** igual à **última posição** do vetor.
- Se for a estrutura **enquanto (while)**, deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

#### Português Estruturado

```
PARA L <- 1 até tamanho do vetor
FAÇA
INÍCIO
    ESCREVA (nome do vetor [ L ] );
FIM;
```

#### C

```
for (L = 1; L <= tam_vetor; L++)
{
    printf("%d", nome_vetor [L]);
}
```

- O trecho anterior poderá ser incrementado com outros comandos;
- **Jamais** deverá aparecer um comando **leia (scanf)** no trecho de saída;
- Este trecho poderá ser incrementado com outros comandos como: estrutura de seleção (if), outro para (for), etc.

### Exemplos:

**1 - Criar um programa que entre com dez números e imprima uma listagem contendo todos os números informados:**

Deverá aparecer na tela o seguinte resultado:

```
Digite número 1: 10
Digite número 2: 20
Digite número 3: 30
Digite número 4: 40
Digite número 5: 50
```

```
Digite número 6: 15
Digite número 7: 25
Digite número 8: 35
Digite número 9: 45
Digite número 10: 55
```



**OBS:** após digitar esses dados e pressionar a tecla Enter deverá aparecer:

10  
20  
30  
40  
50  
15  
25  
35  
45  
55

### **Português Estruturado**

```
Programa lerimp;  
var  
L: inteiro;  
nomes: arranjo [1..10] de literal;  
INÍCIO  
    Para L <- 1 até 10 faça  
    Início  
        Escreva ('Digite nome ', L, ' : ');  
        Leia (nomes [L]);  
    Fim;  
    Para L <- 1 até 10 faça  
    Início  
        Escreva (nomes [ L]);  
    Fim;  
  
FIM.
```



em "C"

```
Dev-C++ 5.0.0.9  
Arquivo  Editar  Localizar  Exibir  Projeto  Executar  Debug  Ferramentas  Janela  Ajuda  
Novo  Inserir  Criar  Ir para  
Projeto  Classes  Debug  [*] exemplos.cpp  
1  /* Criar um programa que entre com dez números  
2  e imprima uma listagem contendo todos os nomes: */  
3  //Programadora: Lucélia Oliveira  
4  
5  #include <stdio.h>  
6  #include <stdlib.h>  
7  
8  int main() {  
9      //Trecho de dimensionamento  
10     int num[10];  
11     //Trecho de entrada  
12     for (int l = 0; l < 10; l++) {  
13         printf("Digite numero %d: ", l+1);  
14         scanf("%d", &num[l]);  
15     }  
16     //Trecho de saída  
17     for (int l = 0; l < 10; l++) {  
18         printf("%d \n", num[l]);  
19     }  
20     system("pause");  
21     return(0);  
22 }  
23
```

**Exemplo 2 – Criar um programa que armazene nome e duas notas de 5 alunos e imprima uma listagem contendo nome, as duas notas e a média de cada aluno.**

### Português Estruturado

```

Programa imprimemedias;
Var
i: Inteiro;
nomes: arranjo [1..4] de literal;
pr1,pr2, media: arranjo [1..5] de real;
INÍCIO
    Para L <- 1 até 4 faça
    Início
        Escreva ('Nome do aluno ',L, ' : ');
        Escreva('Nome do(a) aluno(a) ',i,',': ');
        Leia (nomes[L]);
        Escreva ('Digite a 1ª nota: ');
        Leia (pr1[L]);
        Escreva ('Digite a 2ª nota: ');
        Leia (pr2[L]);
        media [L] <- (pr1 [L] + pr2[L]) / 2;
    Fim;
    Escreva (' Relação Final');
    Para L <- 1 até 4 faça
    Início
        Escreva (L, ' - ', nomes [L], ' 1ª nota: ', pr1[L], ' 2ª nota:
        ', pr2[L], ' média: ',media[L]);
    Fim;
FIM.

```



em "C"

```

Dev-C++ 5.0.0.9
Exibir Projeto Executar Debug Ferramentas CVS Janela Ajuda
[?] exemplos.cpp
1  /* Criar um programa que armazene nome e duas notas de 5 alunos e imprima uma
2  listagem contendo nome, as duas notas e a média de cada aluno. */
3  //Programadora: Lucélia Oliveira
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  int main(){
9      //Trecho de dimensionamento
10     char nomes[40][4];
11     float pr1[4], pr2[4], media[4];
12     //Trecho de entrada
13     for (int l = 0; l < 4; l++){
14         printf("Nome do aluno %d: ",l+1);
15         scanf("%s",&nomes[l]);
16         printf("Digite a 1a nota: ");
17         scanf ("%f",&pr1[l]);
18         printf ("Digite a 2a nota: ");
19         scanf("%f",&pr2[l]);
20         //processamento
21         media[l] = (pr1[l] + pr2[l])/2;
22     }
23     //Trecho de saída
24     printf ("===== Relação Final=====\n");
25     for (int l = 0; l < 4; l++){
26         printf("%d - %s - 1a nota: %0.1f - 2a nota: %0.1f ----> media: %0.1f\n",l+1, nomes[l], pr1[l],pr2[l],media[l]);
27     }
28     system("pause");
29     return(0);
30 }
31

```

### **Veja o resultado deste programa:**

Nome do aluno 1: MARCOS

Digite a 1ª nota: 10

Digite a 2ª nota: 5

Nome do aluno 2: PEDRO

Digite a 1ª nota: 8

Digite a 2ª nota: 6

Nome do aluno 3: MARIANA

Digite a 1ª nota: 5

Digite a 2ª nota: 7

Nome do aluno 4: ANA

Digite a 1ª nota: 10

Digite a 2ª nota: 9

### **Relação Final**

1 – MARCOS	1ª nota: 10	2ª nota: 5	média: 7.5
2 – PEDRO	1ª nota: 8	2ª nota: 6	média: 7
3 - MARIANA	1ª nota: 5	2ª nota: 7	média: 6
4 – ANA	1ª nota: 10	2ª nota: 9	média: 9.5

## EXERCÍCIOS

1. Armazenar 10 nomes em um vetor NOME e imprimir uma listagem numerada.
2. Armazenar 15 números inteiros em um vetor NUM e imprimir uma listagem contendo o número e uma das mensagens: par ou ímpar.
3. Armazenar 8 números em um vetor e imprimir todos os números. Ao final, teremos o total de números múltiplos de 3.
4. Armazenar nomes e notas da prova 1 e prova 2 de 15 alunos. Calcular e armazenar a média. Armazenar também a situação do aluno: AP ou RP. Imprimir uma listagem contendo nome, média e situação de cada aluno.
5. Armazenar nome e salário de 20 pessoas. Calcular e armazenar o novo salário sabendo-se que o reajuste foi de 8%. Imprimir uma listagem numerada com nome e novo salário.
6. Ler um vetor de 10 elementos e obter outro vetor, cujos componentes são o triplo dos respectivos componentes do primeiro vetor.
7. Entrar com números inteiros em um vetor A[50] e um vetor B[50]. Gerar e imprimir o veto C[50] que será a soma dos vetores A e B.

## VIII - MATRIZES

A estrutura da matriz é semelhante à do vetor, sendo que, pode possuir n dimensões. Assim, para fazer referência aos elementos de uma matriz, precisaremos de tantos índices quantas forem suas dimensões.

A declaração de uma matriz no "C" é obrigatória. Por exemplo, para se declarar uma matriz MAT de uma única dimensão (vetor) composta de 50 números inteiros, seria feito da seguinte forma:

```
int MAT[50];
```

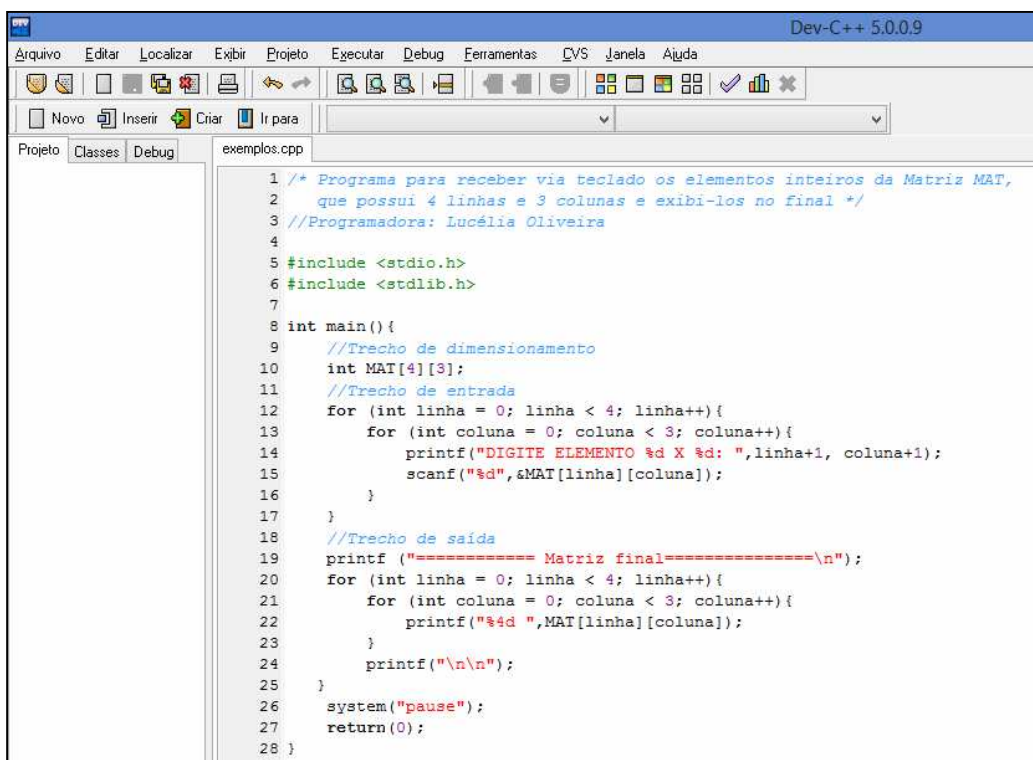
No caso de uma matriz bidimensional de 50 linhas e 100 colunas, composta de inteiros, seria assim:

```
int MAT[50][100];
```

**ATENÇÃO!!!**

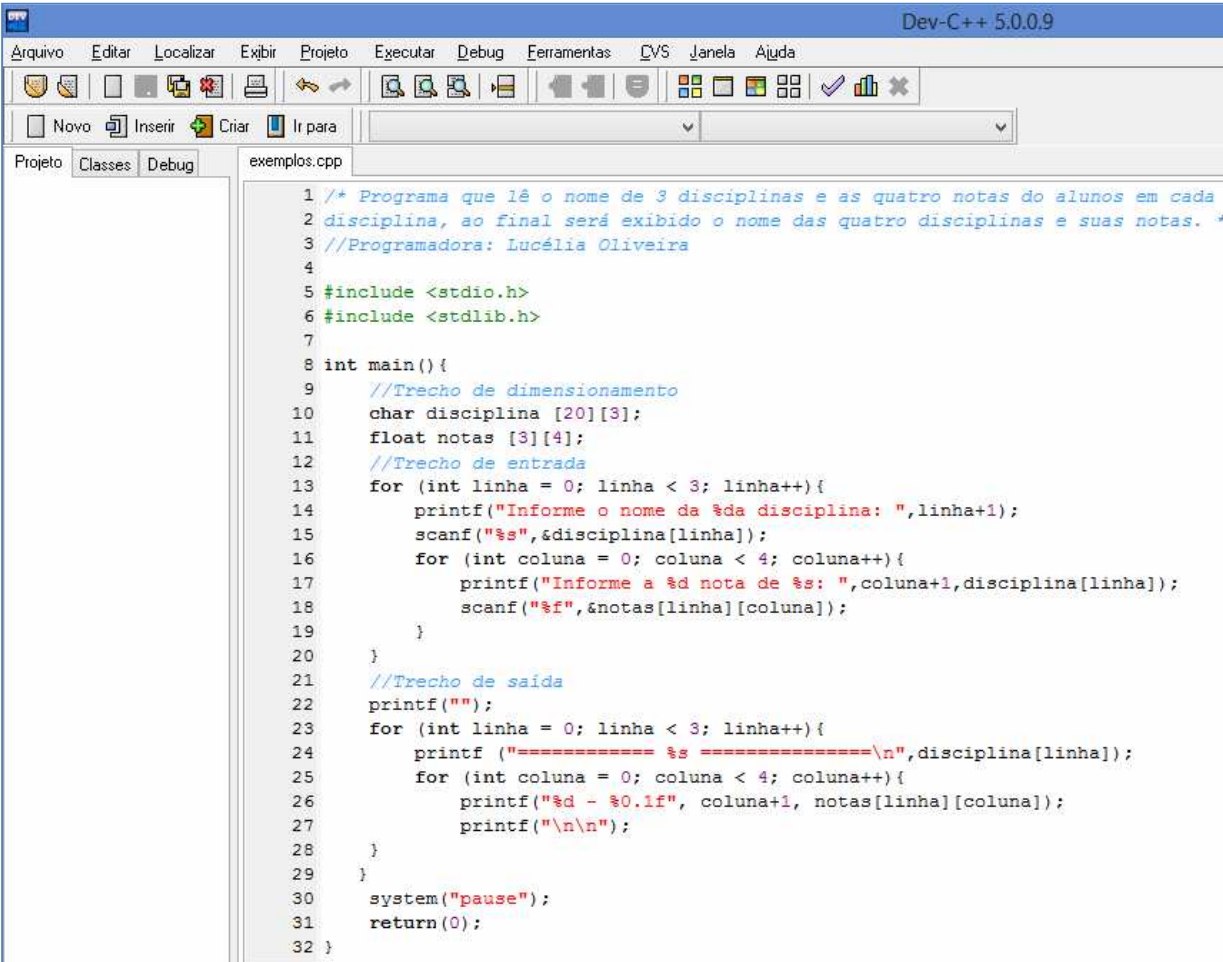
Como você já deve estar familiarizado com a Linguagem "C", a partir deste ponto, os exemplos serão apenas nessa linguagem.

**Exemplo:** Programa para receber via teclado os elementos inteiros da Matriz MAT, que possui 4 linhas e 3 colunas e exibi-los no final:



```
1 /* Programa para receber via teclado os elementos inteiros da Matriz MAT,
2    que possui 4 linhas e 3 colunas e exibi-los no final */
3 //Programadora: Lucélia Oliveira
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(){
9     //Trecho de dimensionamento
10    int MAT[4][3];
11    //Trecho de entrada
12    for (int linha = 0; linha < 4; linha++){
13        for (int coluna = 0; coluna < 3; coluna++){
14            printf("DIGITE ELEMENTO %d X %d: ",linha+1, coluna+1);
15            scanf("%d",&MAT[linha][coluna]);
16        }
17    }
18    //Trecho de saída
19    printf ("===== Matriz final =====\n");
20    for (int linha = 0; linha < 4; linha++){
21        for (int coluna = 0; coluna < 3; coluna++){
22            printf("%4d ",MAT[linha][coluna]);
23        }
24        printf("\n\n");
25    }
26    system("pause");
27    return (0);
28 }
```

**Exemplo 2:** Programa que lê o nome de 3 disciplinas e as quatro notas do alunos em cada disciplina, ao final será exibido o nome das quatro disciplinas e suas notas.



```
1  /* Programa que lê o nome de 3 disciplinas e as quatro notas do alunos em cada
2  disciplina, ao final será exibido o nome das quatro disciplinas e suas notas. */
3  //Programadora: Lucélia Oliveira
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  int main(){
9      //Trecho de dimensionamento
10     char disciplina [20][3];
11     float notas [3][4];
12     //Trecho de entrada
13     for (int linha = 0; linha < 3; linha++){
14         printf("Informe o nome da %da disciplina: ",linha+1);
15         scanf("%s",&disciplina[linha]);
16         for (int coluna = 0; coluna < 4; coluna++){
17             printf("Informe a %d nota de %s: ",coluna+1,disciplina[linha]);
18             scanf("%f",&notas[linha][coluna]);
19         }
20     }
21     //Trecho de saída
22     printf("\n");
23     for (int linha = 0; linha < 3; linha++){
24         printf ("===== %s =====\n",disciplina[linha]);
25         for (int coluna = 0; coluna < 4; coluna++){
26             printf("%d - %0.1f", coluna+1, notas[linha][coluna]);
27             printf("\n\n");
28         }
29     }
30     system("pause");
31     return(0);
32 }
```

## EXERCÍCIOS - Matriz

1. Entrar com valores em reais. Gerar e mostrar a matriz metade
2. Entrar com valores inteiros para duas matrizes 4 X 4. Gerar e mostrar a matriz SOMA.
3. Criar um programa que leia os elementos de uma matriz 5 X 5 e mostre os elementos da diagonal principal. Dica: Elementos da diagonal principal linha = coluna.
4. Criar um programa que leia os elementos de uma matriz 5 X 5 e mostre os elementos abaixo da diagonal principal. Dica: linha > coluna.
5. Criar um algoritmo que leia os elementos de uma matriz 5 X 5 e mostre a soma dos elementos da diagonal principal.
6. Faça um programa que possa armazenar o nome de 5 atletas de 3 países que participarão dos jogos de verão. Informar os nomes dos países e seus respectivos jogadores.
7. Faça um Microssistema para realizar as seguintes operações:
  - Soma de duas matrizes;
  - Subtração de duas matrizes;
  - Multiplicação de duas Matrizes;
  - Mostrar os valores da diagonal principal

As regras são:

- o Microssistema deve conter um menu de opções em que o usuário deve escolher a opção desejada;
- As matrizes devem conter no máximo três linhas por três colunas; o usuário é quem determina as dimensões das matrizes;
- Permitir realizar várias operações, isto é, após uma operação, voltar ao menu de opções.

## IX – SUB-ROTINAS - FUNÇÕES

Existem dois tipos de sub-rotinas: Funções e Procedimentos. Entre estes dois tipos de sub-rotinas existem algumas diferenças, mas, o conceito é o mesmo para ambas. O importante no uso prático destes dois tipos de sub-rotinas é distinguir as diferenças entre elas e como utiliza-las no momento mais adequado.

### Procedimento (Procedure)

Um procedimento é um bloco de programa contendo início e fim e é identificado por um nome, por meio do qual será referenciado em qualquer parte do programa principal. Quando uma sub-rotina é chamada dentro do programa principal, ela é executada e, ao seu término, o controle de processamento retorna para a próxima linha de instrução após a linha que efetuou a chamada da sub-rotina.

Com relação à criação da rotina, será idêntica a tudo o que foi estudado sobre programação. A Linguagem C, porém, não comporta as *procedures*, mas apenas as *functions*, que serão estudadas a seguir.

### Função(function)

Uma *function* possui a mesma funcionalidade de uma *procedure*, que é desviar a execução do programa principal para realizar uma tarefa específica, com uma única diferença: uma *function* sempre retorna um valor. Por isso, ao declarar uma *function*, declaramos também que tipo de dados esta função irá retornar.

#### Sintaxe:

```
tipo_de_retorno_da_função nome_da_function(parâmetros se houver){  
    Declaração das variáveis locais(se tiver);  
    {  
        Comandos;  
    }  
}
```

#### Exemplo:

```
int soma (int x,int y){  
    {  
        return x + y;  
    }  
}
```



O tipo de retorno é o tipo de variável que a função retornará após sua execução. O *default* é o tipo `int`, ou seja, se uma função não tiver um tipo explícito de retorno, ela retornará um dado do tipo inteiro (`int`).

A declaração de parâmetros é uma lista contendo a seguinte sintaxe:

`tipo nome1, tipo nome2, ... , tipo nome_n`

### ATENÇÃO!!!

O tipo de cada variável de entrada deve ser especificado separadamente, para cada uma das variáveis. É na declaração de parâmetros que informamos ao compilador quais serão as entradas da função (assim como, na saída, informamos tipo-de-retorno).

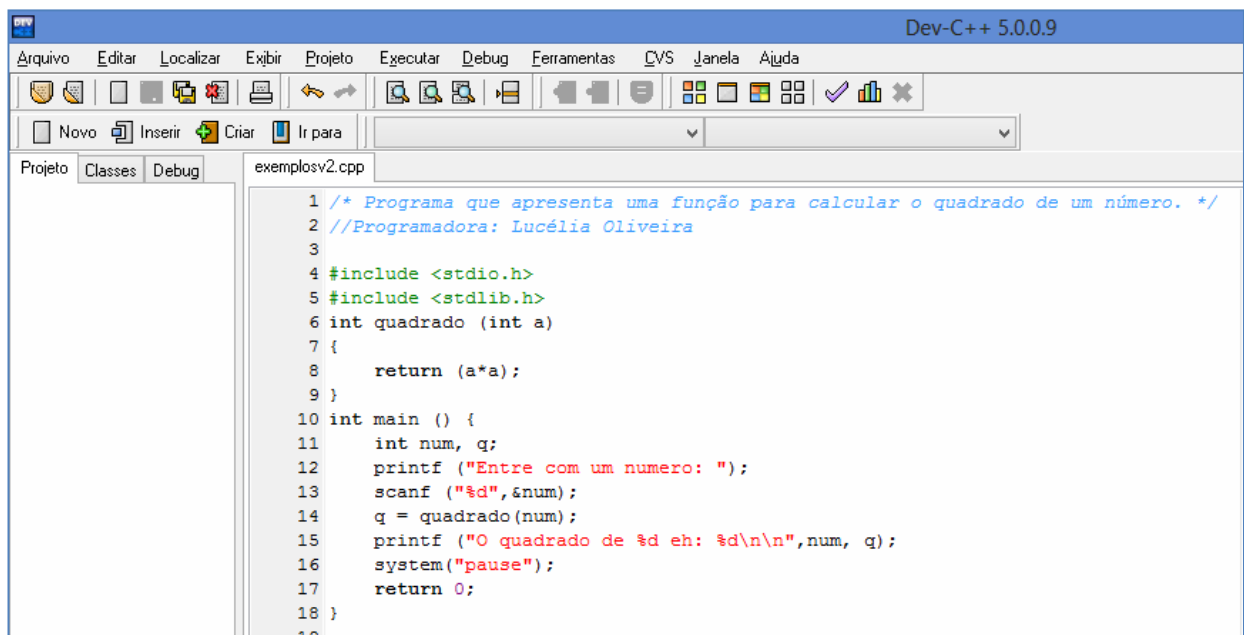
O corpo da função é como uma função principal - `main()`. Nele as entradas são processadas, saídas são geradas ou outras coisas são realizadas.

O comando *return* tem a seguinte forma geral:

`return valor_de_retorno; ou return;`

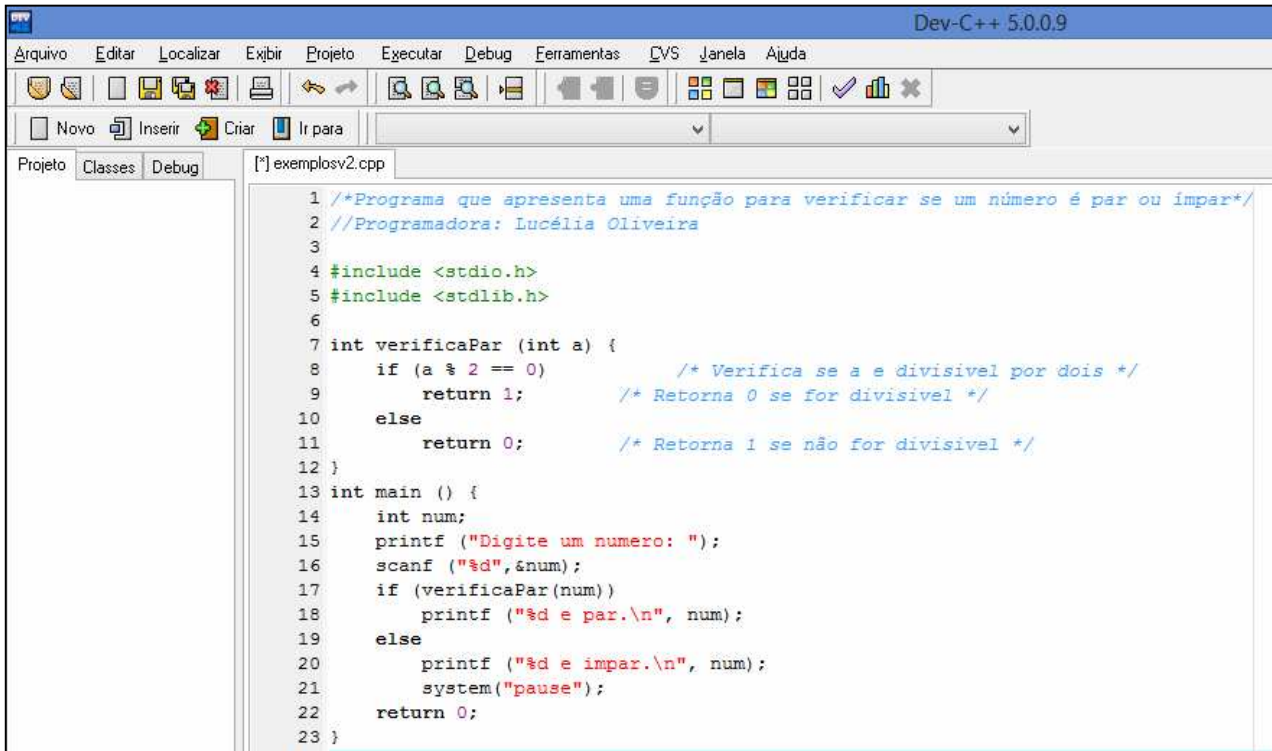
Quando uma função é executada e se chega a um comando *return*, a função é encerrada imediatamente e, se o valor de retorno é informado, a função retorna esse valor. Observe que o valor de retorno fornecido tem que ser compatível com o tipo de retorno declarado para a função.

**Exemplo 1:** Programa que apresenta uma função para calcular o quadrado de um número.



```
1  /* Programa que apresenta uma função para calcular o quadrado de um número. */
2  //Programadora: Lucélia Oliveira
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  int quadrado (int a)
7  {
8      return (a*a);
9  }
10 int main () {
11     int num, q;
12     printf ("Entre com um numero: ");
13     scanf ("%d", &num);
14     q = quadrado(num);
15     printf ("O quadrado de %d eh: %d\n\n", num, q);
16     system("pause");
17     return 0;
18 }
19
```

**Exemplo 2:** Programa que apresenta uma função para verificar se um número é par ou impar:



```
1 /*Programa que apresenta uma função para verificar se um número é par ou impar*/
2 //Programadora: Lucélia Oliveira
3
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 int verificaPar (int a) {
8     if (a % 2 == 0)          /* Verifica se a é divisível por dois */
9         return 1;          /* Retorna 0 se for divisível */
10    else
11        return 0;          /* Retorna 1 se não for divisível */
12 }
13
14 int main () {
15     int num;
16     printf ("Digite um numero: ");
17     scanf ("%d",&num);
18     if (verificaPar(num))
19         printf ("%d é par.\n", num);
20     else
21         printf ("%d é impar.\n", num);
22     system("pause");
23     return 0;
24 }
```

Observe que, como as funções retornam valores, podemos aproveitá-los para fazer atribuições, ou mesmo para que esses valores participem de expressões.

No segundo exemplo, tivemos mais de um *return* na função.

Se uma função retorna um valor, você não é obrigado a aproveitá-lo. Se você não fizer nada com o valor de retorno de uma função, ele será descartado. Por exemplo, a função `printf()` retorna um inteiro que nós nunca usamos para nada. Ele é descartado.

## EXERCÍCIOS - Funções

1. Criar um programa que efetue o cálculo de uma prestação em atraso através de uma função. Para tanto, utilize a fórmula  $PREST = VALOR + (VALOR * (TAXA/100) * TEMPO$ .
2. Elaborar um programa que possua uma função que efetue e permita apresentar o somatório dos N primeiros números inteiros, definidos pelo usuário.  $(1 + 2 + 3 + 4 + \dots + N)$ .
3. Elaborar um programa que contenha uma função que leia e retorne um valor. No programa principal (*function* `main()`), chame essa função de leitura três vezes, para a leitura de três valores, implementar uma *function* que retorne a soma dos quadrados de três valores. Na *function* `main()` chamar essas funções e mostrar o resultado final.
4. Faça um programa com uma *function* que calcula o fatorial de um número.
5. Faça um programa calculadora que apresente um menu de opções no programa principal. Este menu deverá dar ao usuário a possibilidade de escolher uma entre quatro operações aritméticas. Escolhida a opção desejada, deverá ser solicitada a entrada de dois números, e processada a operação deverá ser exibido o resultado.

## ANEXO I – Questões de Raciocínio Lógico

1. Preencha o quadro a seguir de tal maneira que a soma dos números que ficam sobre uma linha, ou sobre uma coluna, ou sobre uma diagonal, dê sempre 15 e todos os números têm de ser diferentes:


2. Entram num restaurante para jantar três mulheres, cada uma com duas filhas. Só existiam 7 lugares. Nenhuma ficou de pé. Como isso é possível?
3. Tenho 3 camisas: A, B e C. Uma é VERDE, uma é BRANCA e outra é AZUL, não necessariamente nesta ordem. Sabe-se somanete que a a camisa A é VERDE, B não É VERDE e C Não é AZUL. Quais as cores de A, B, e C, nessa ordem?
4. Dentro de uma caixa fechada, há uma bola branca e uma bola preta. Numa segunda caixa fechada, há duas bolas brancas e, numa terceira caixa fechada, há duas bolas pretas. Cada caixa possui uma etiqueta indicando o conteúdo das caixas, mas alguém misturou as três etiquetas de modo que **todas** as etiquetas estão erradas. Você seria capaz de escolher apenas uma das seis bolas de modo tal que, olhando a sua cor, você possa dizer o conteúdo de cada uma das caixas?

## ANEXO II - Transferência de Comandos: Português Estruturado para C

Veja como seria a transferência de alguns comandos de Português Estruturado para C:

**OBS:** Os dados da tabela estão em ordem alfabética.

Português Estruturado	C
← (atribuição)	=
= comparação	==
Algoritmo ou Programa	int main()
Cadeia (de caracteres) ou Literal Ex.: Nome: literal;	char[ ] char nome[40];
Caracter	char
E	&&
Enquanto n <> 0 faça	while (n != 0)
Escreva (".....")	printf(".....");
Início	{
Fim	}
Inteiro	int
Leia (nome da variável);	Para ler um inteiro, por exemplo: scanf ("%d",&nome da variável);
Nome:arranjo [1..10] de cadeia (ou string);	Nome: array [1..10] of string;
Ou	
Para i←1 até 10 faça	For (i= 1; i <= 10; i++)
Se x > 10 então	If (x > 10)
Senão	else

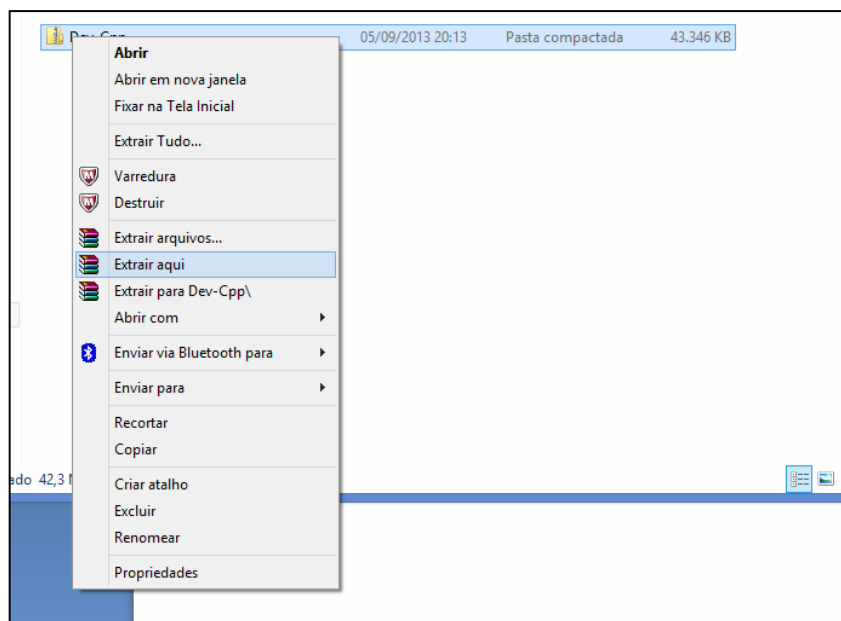
## ANEXO III - Como compilar um programa na linguagem C

Existem vários compiladores para a Linguagem de Programação em C. Entre eles, há os compiladores NetBeans, Turbo C, Dev-C++.

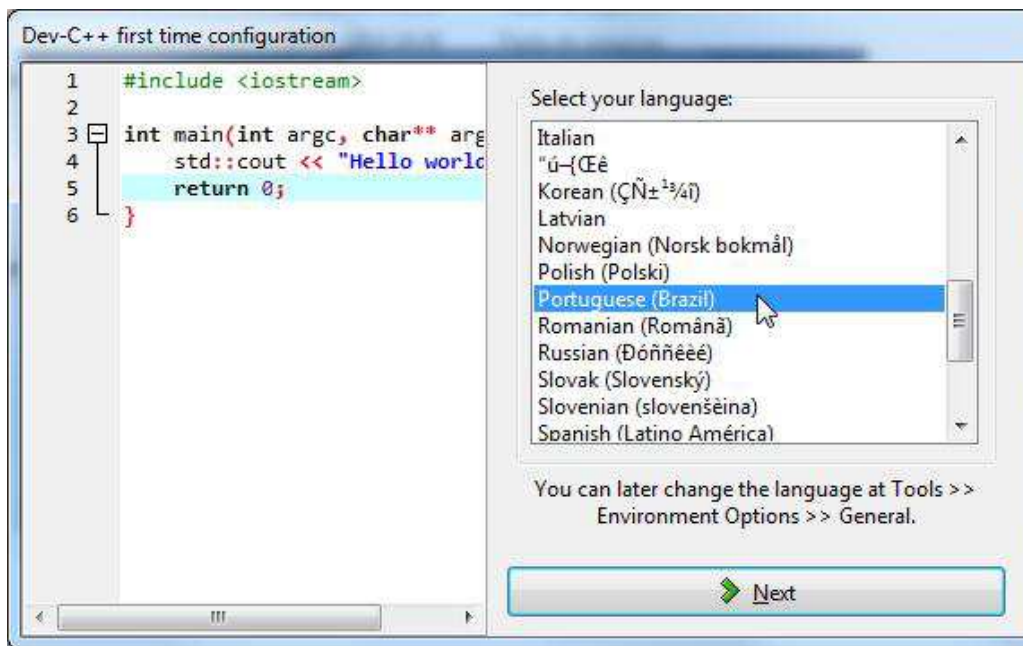
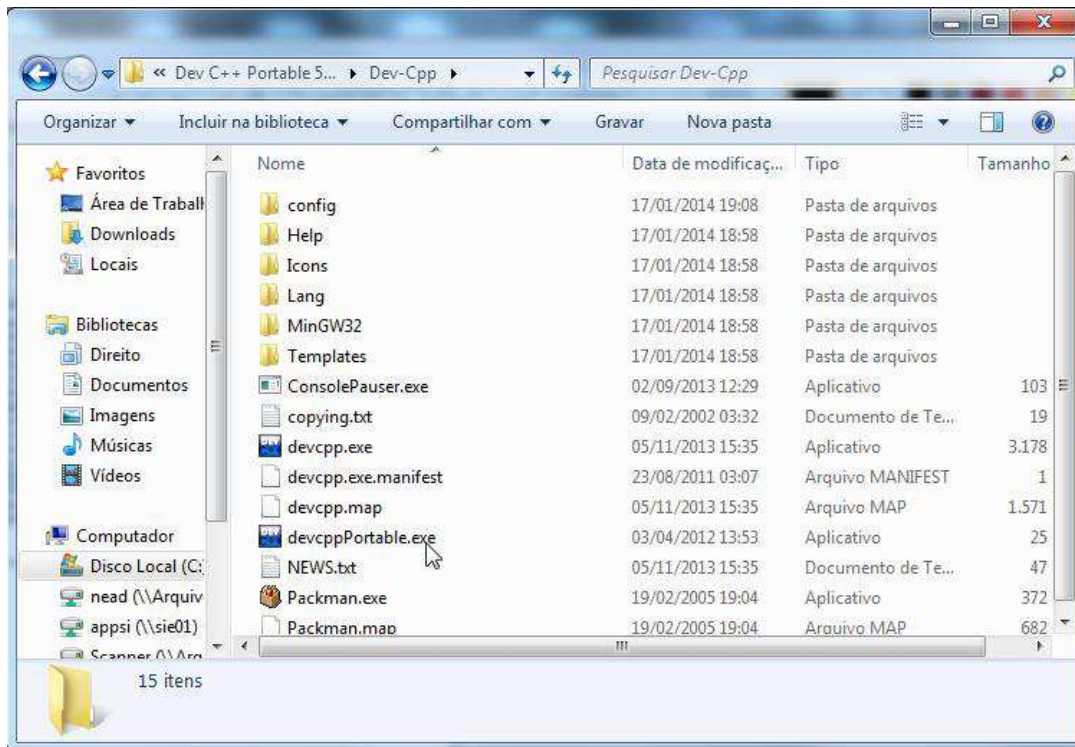
Para facilitar o trabalho de compilação por parte dos iniciantes, sugere-se o compilador Dev-C++ portátil, que não requer nenhum tipo de instalação prévia, além de se evitar problemas de incompatibilidades do compilador com o sistema operacional, como é o caso do Dev-C++ instalável com o Windows 8.

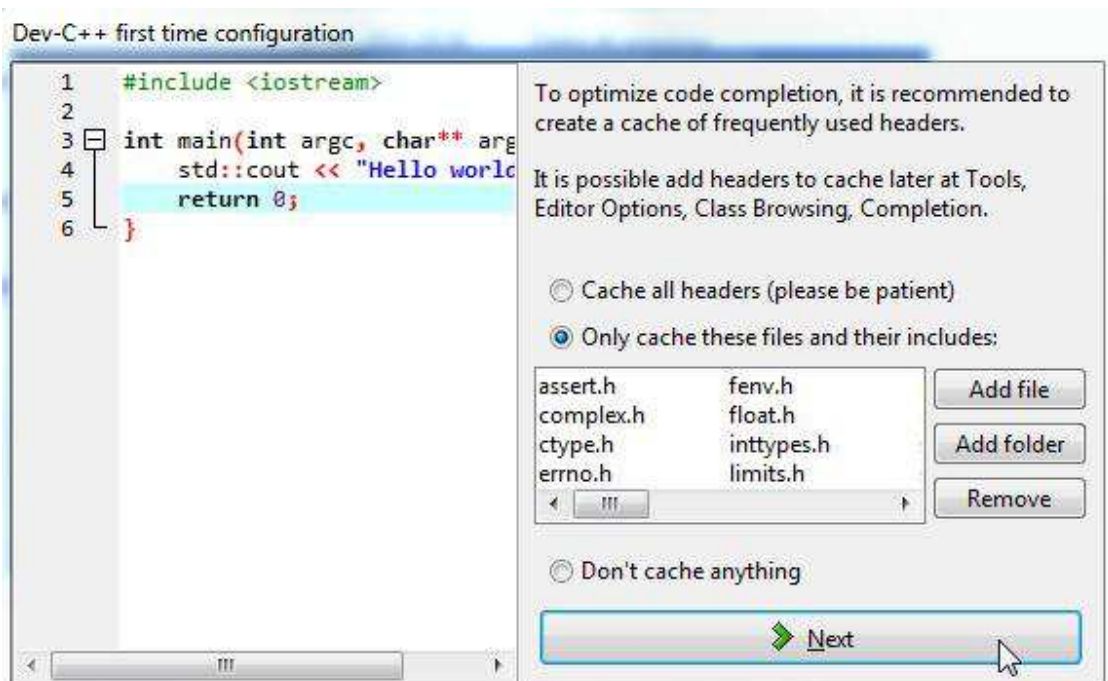
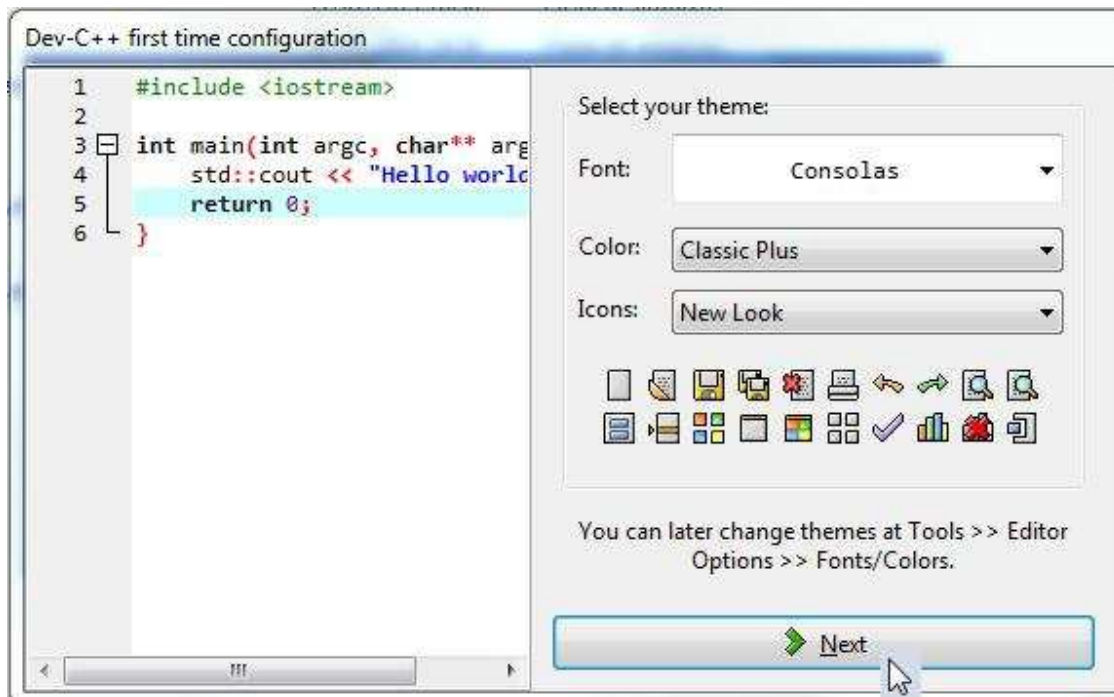
Esse compilador pode ser baixado gratuitamente. Após isso, basta descompactar a pasta e executar o arquivo executável "devcppPortable", conforme será mostrado no passo a passo a seguir:

**Passo 1** - Após salvar o arquivo "Dev-Cpp", contendo o Dev-C++ *portable* (portátil) na pasta desejada, clique com o botão direito sobre o arquivo e escolha a opção "Extrair aqui".

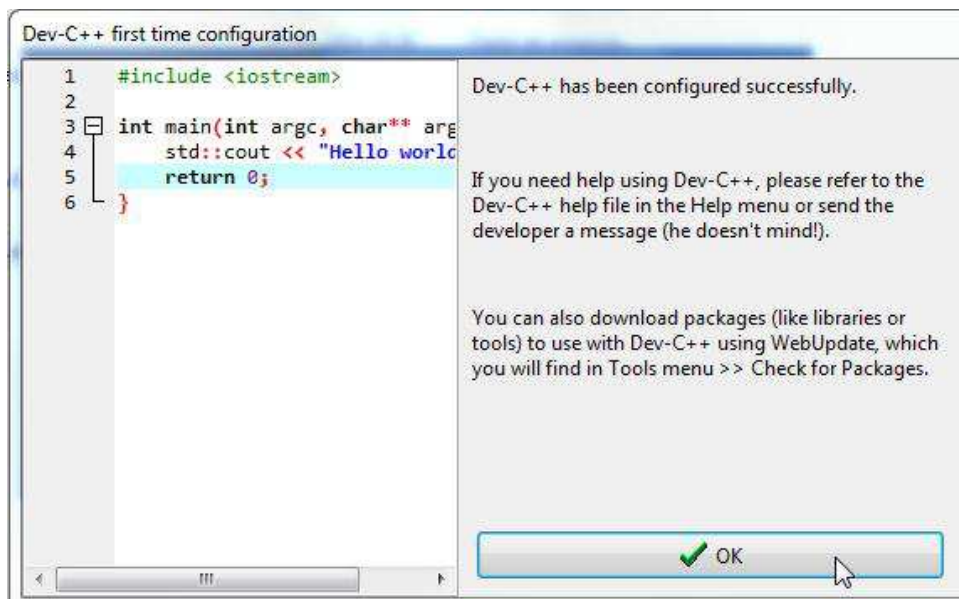


**Passo 2** - Abra a pasta Dev-Cpp e clique sobre o arquivo " devcppPortable". Execute a instalação conforme as figuras seguintes:

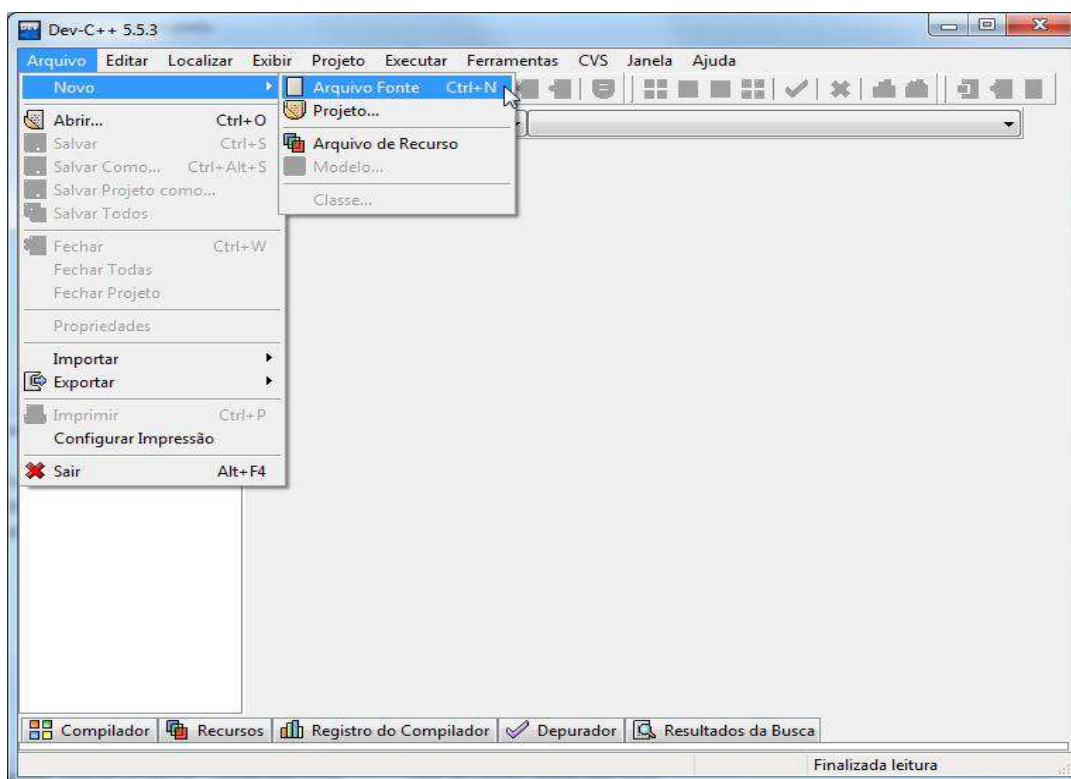






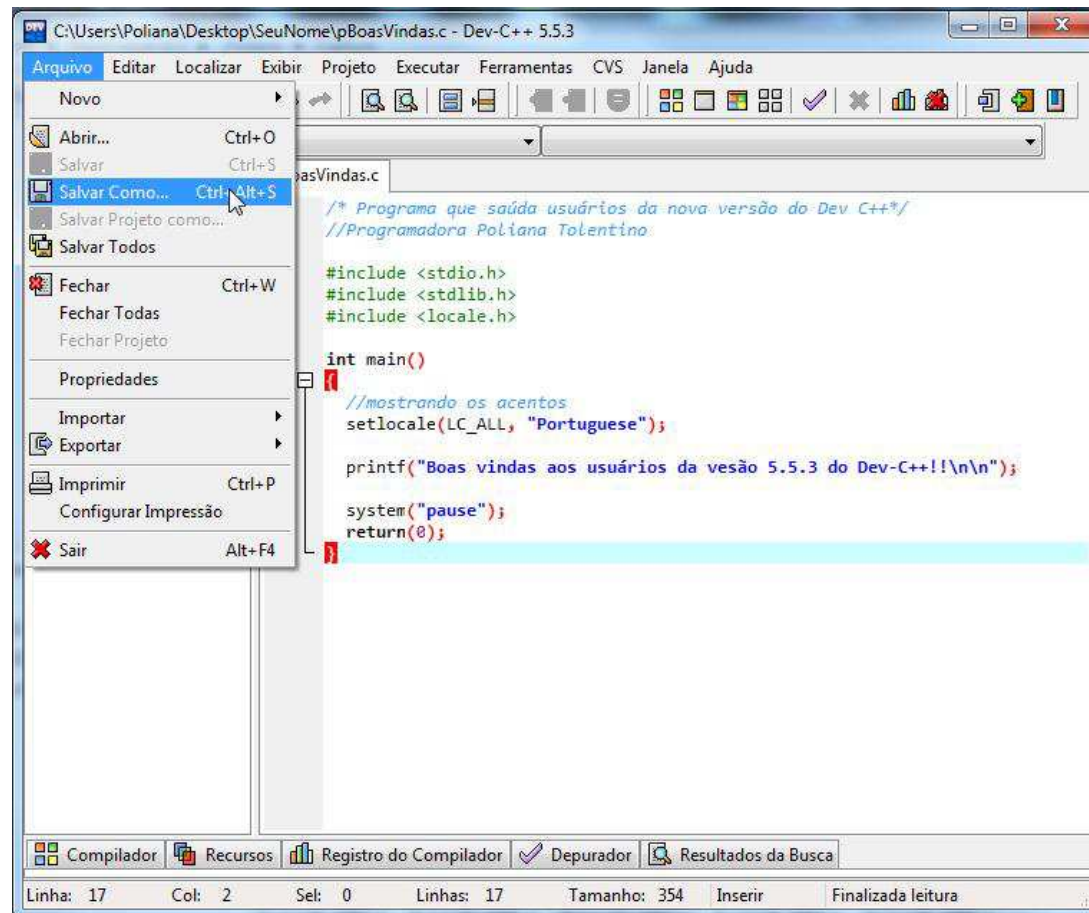


**Passo 3** - Clique nas opções Arquivo --> Novo --> Arquivo Fonte ou pressione as teclas (Ctrl + N)

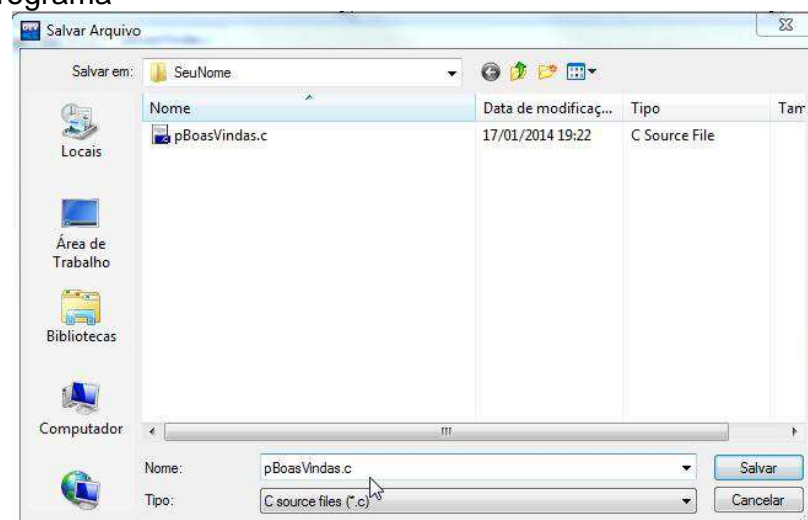


**Passo 4 -** Digite o seu programa (código fonte)

**Passo 5 -** Salve o código fonte gerado no local desejado: Arquivo --> Salvar Como

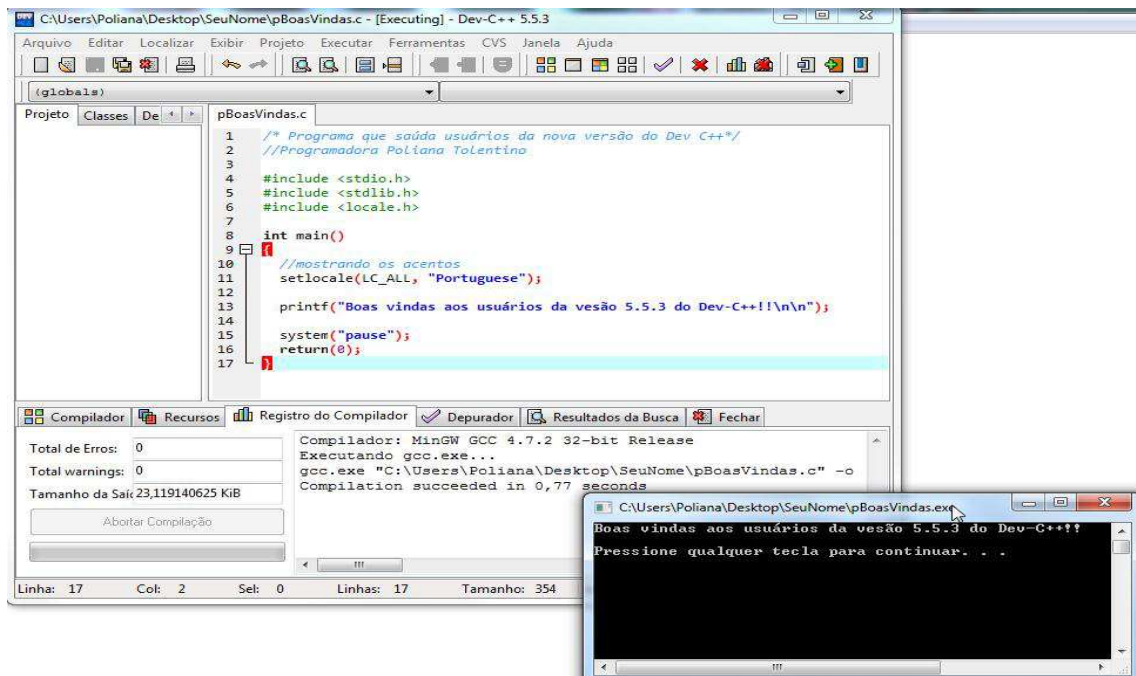
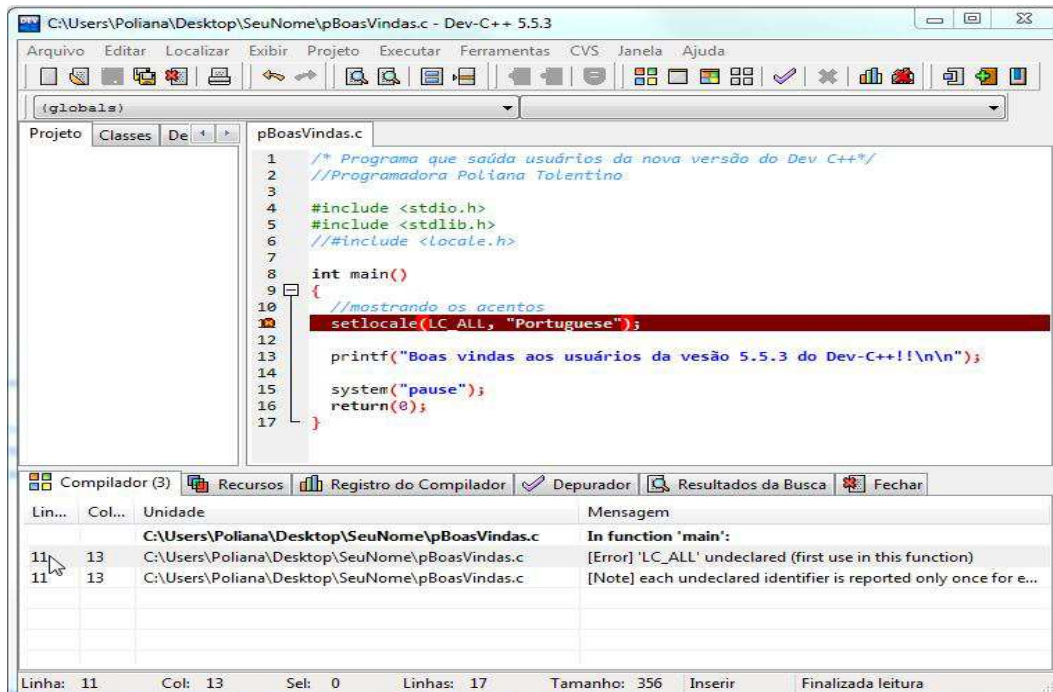


**Passo 6 -** Escolha o local onde deseja salvar seu programa e depois dê um nome ao seu programa

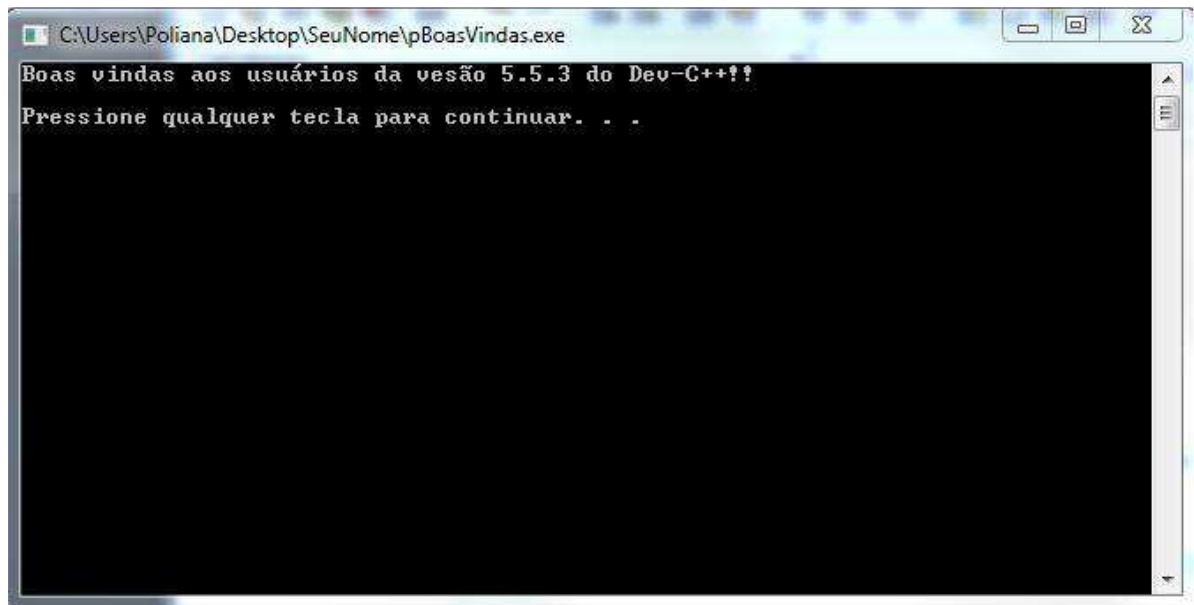


**Passo 7 -** Agora é só compilar o seu programa. Para isso, pressione a tecla "F11".

Caso ocorra algum erro, eles aparecerão abaixo do seu código fonte. Nesse caso, você terá que corrigir os erros e pressionar a tecla "F11" novamente. A tela abaixo só aparecerá quando todos os erros já tiverem sido corrigidos. Veja abaixo como serão apresentados os erros:



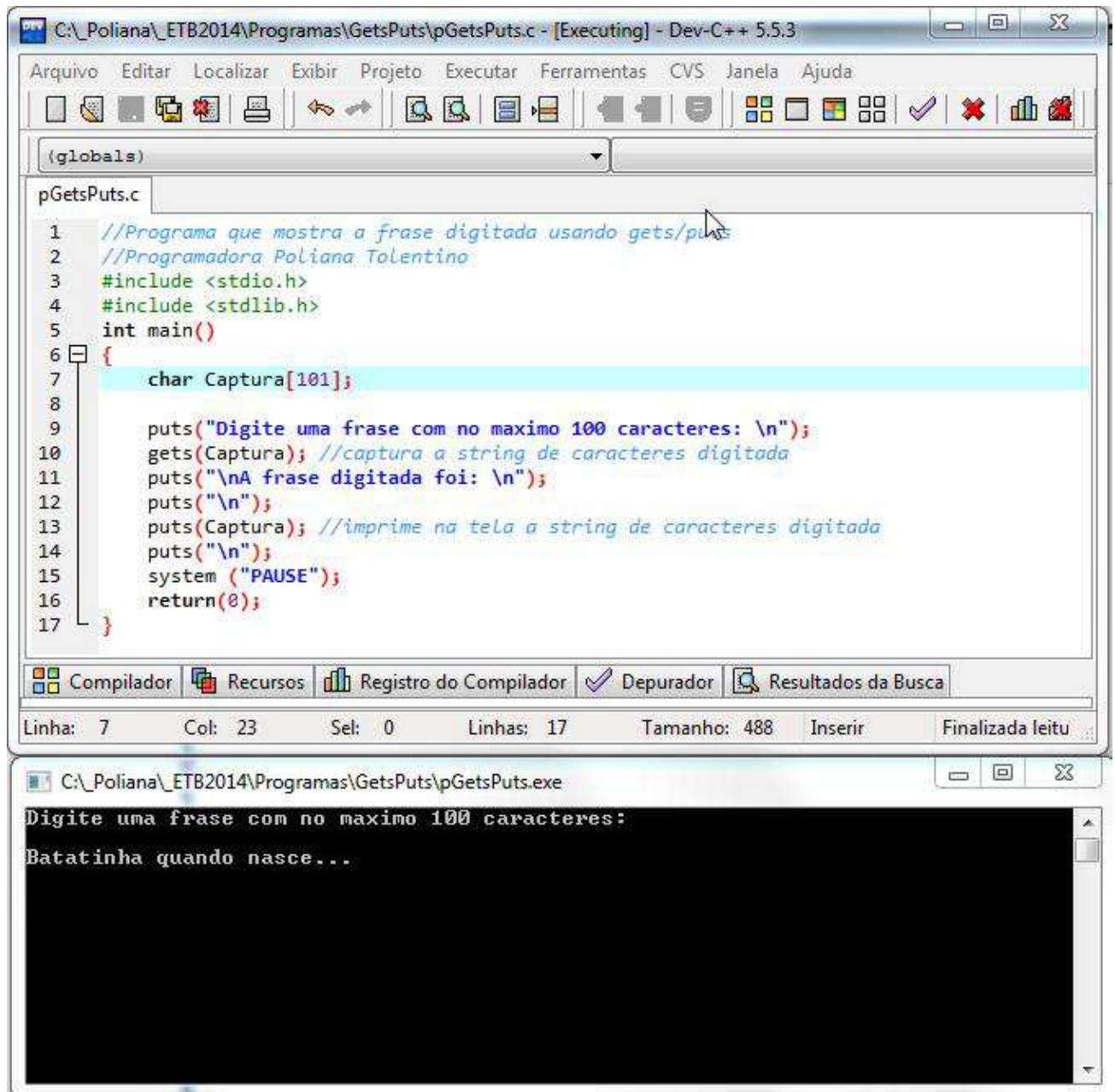
Se não ocorrer nenhum erro, aparecerá a tela do prompt de comando, com o resultado do programa que você construiu.



```
C:\Users\Poliana\Desktop\SeuNome\pBoasVindas.exe
Boas vindas aos usuários da versão 5.5.3 do Dev-C++!!
Pressione qualquer tecla para continuar. . .
```

## ANEXO IV – Exercícios resolvidos

Faça um programa que mostre uma frase com até 100 caracteres usando as funções “Gets – Put”.



The image shows a screenshot of the Dev-C++ 5.5.3 IDE. The top window displays the source code for a C program named `pGetsPuts.c`. The code is as follows:

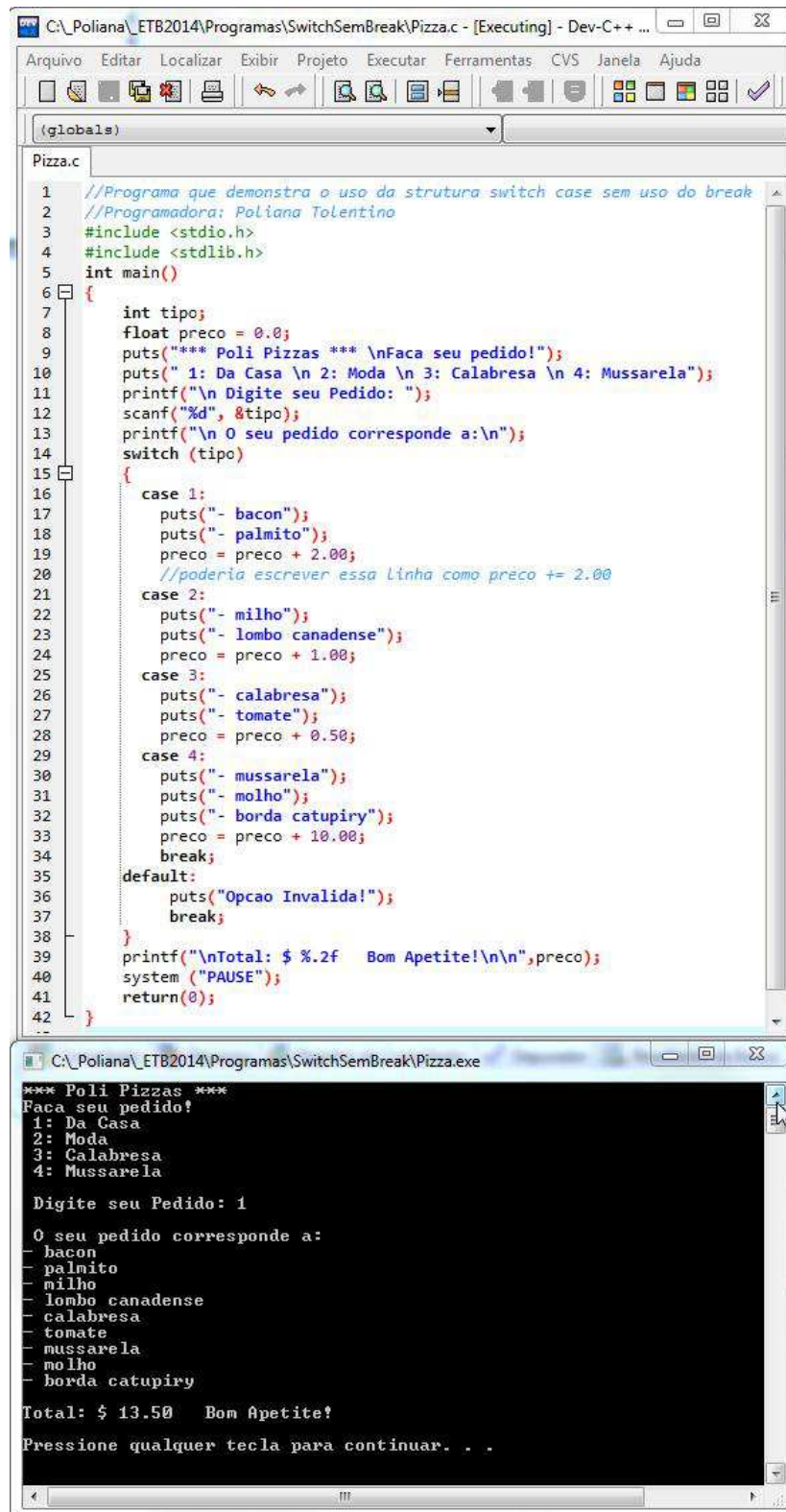
```
1 //Programa que mostra a frase digitada usando gets/puts
2 //Programadora Poliana Tolentino
3 #include <stdio.h>
4 #include <stdlib.h>
5 int main()
6 {
7     char Captura[101];
8
9     puts("Digite uma frase com no maximo 100 caracteres: \n");
10    gets(Captura); //captura a string de caracteres digitada
11    puts("\nA frase digitada foi: \n");
12    puts("\n");
13    puts(Captura); //imprime na tela a string de caracteres digitada
14    puts("\n");
15    system("PAUSE");
16    return(0);
17 }
```

The bottom window shows the execution of the program, `pGetsPuts.exe`. The output is:

```
Digite uma frase com no maximo 100 caracteres:
Batatinha quando nasce...
```



Faça um programa que monte um menu de uma pizzaria com incremento de ingredientes, usando a estrutura "Switch – Case - Default"



The image shows a screenshot of a Dev-C++ IDE with two windows. The top window, titled "C:\Poliana\ETB2014\Programas\SwitchSemBreak\Pizza.c - [Executing] - Dev-C++ ...", displays the source code for a C program named "Pizza.c". The code uses a switch-case structure to calculate the price of a pizza based on the selected toppings. The bottom window, titled "C:\Poliana\ETB2014\Programas\SwitchSemBreak\Pizza.exe", shows the program's execution output.

```
1 //Programa que demonstra o uso da estrutura switch case sem uso do break
2 //Programadora: Poliana Tolentino
3 #include <stdio.h>
4 #include <stdlib.h>
5 int main()
6 {
7     int tipo;
8     float preco = 0.0;
9     puts("*** Poli Pizzas *** \nFaca seu pedido!");
10    puts(" 1: Da Casa \n 2: Moda \n 3: Calabresa \n 4: Mussarela");
11    printf("\n Digite seu Pedido: ");
12    scanf("%d", &tipo);
13    printf("\n 0 seu pedido corresponde a:\n");
14    switch (tipo)
15    {
16        case 1:
17            puts("- bacon");
18            puts("- palmito");
19            preco = preco + 2.00;
20            //poderia escrever essa linha como preco += 2.00
21        case 2:
22            puts("- milho");
23            puts("- lombo canadense");
24            preco = preco + 1.00;
25        case 3:
26            puts("- calabresa");
27            puts("- tomate");
28            preco = preco + 0.50;
29        case 4:
30            puts("- mussarela");
31            puts("- molho");
32            puts("- borda catupiry");
33            preco = preco + 10.00;
34            break;
35        default:
36            puts("Opcao Invalida!");
37            break;
38    }
39    printf("\nTotal: $ %.2f   Bom Appetite!\n\n",preco);
40    system ("PAUSE");
41    return(0);
42 }
```

The execution output in the bottom window shows the program running and displaying the menu options. The user has entered '1' for the first option, and the program has calculated the total price as \$ 13.50. The output is as follows:

```
*** Poli Pizzas ***
Faca seu pedido!
1: Da Casa
2: Moda
3: Calabresa
4: Mussarela

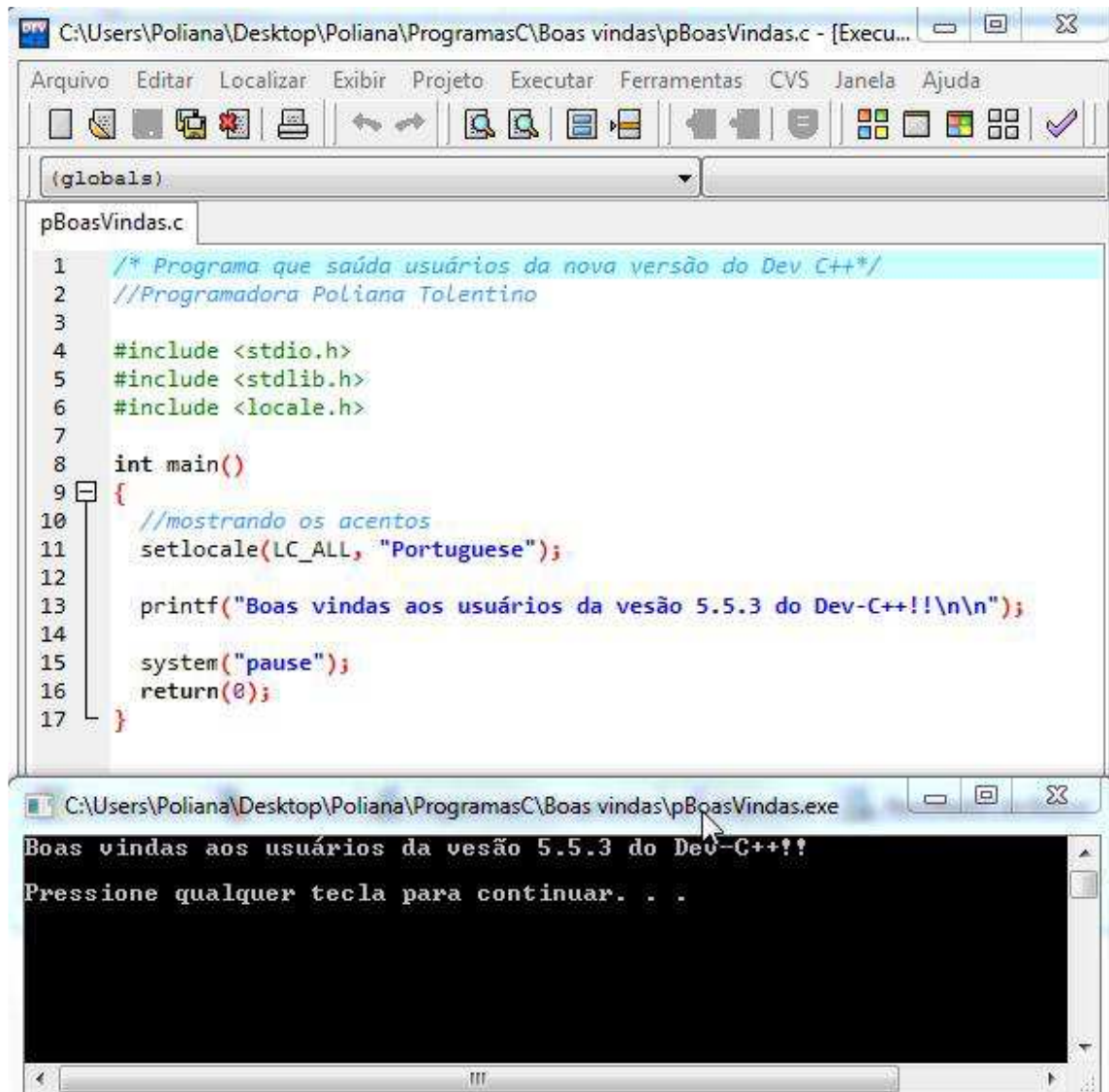
Digite seu Pedido: 1

0 seu pedido corresponde a:
- bacon
- palmito
- milho
- lombo canadense
- calabresa
- tomate
- mussarela
- molho
- borda catupiry

Total: $ 13.50   Bom Appetite!

Pressione qualquer tecla para continuar. . .
```

Faça um programa que mostre uma mensagem na tela sem desconfigurar os acentos.



The image shows a screenshot of the Dev-C++ IDE. The top window displays the source code for a C program named `pBoasVindas.c`. The code includes headers for `stdio.h`, `stdlib.h`, and `locale.h`. The `main` function sets the locale to Portuguese using `setlocale(LC_ALL, "Portuguese");`, prints a welcome message with accents, and then pauses the execution with `system("pause");`.

```
1  /* Programa que saúda usuários da nova versão do Dev C++ */
2  // Programadora Poliana Tolentino
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <locale.h>
7
8  int main()
9  {
10     // mostrando os acentos
11     setlocale(LC_ALL, "Portuguese");
12
13     printf("Boas vindas aos usuários da versão 5.5.3 do Dev-C++!!\n\n");
14
15     system("pause");
16     return(0);
17 }
```

The bottom window shows the execution output of the program. It displays the message "Boas vindas aos usuários da versão 5.5.3 do Dev-C++!!" followed by "Pressione qualquer tecla para continuar. . .".

## ANEXO V – Tabela ASCII

	00	16	32	48	64	80	96	112
0	NUL DLE			0	@	P	`	p
1	SOH DC1	!		1	A	Q	a	q
2	STX DC2	"		2	B	R	b	r
3	ETX DC3	#		3	C	S	c	s
4	EOT DC4	\$		4	D	T	d	t
5	ENQ NAK	%		5	E	U	e	u
6	ACK SYN	&		6	F	V	f	v
7	BEL ETB	'		7	G	W	g	w
8	BS CAN	(		8	H	X	h	x
9	HT EM	)		9	I	Y	i	y
10	LF SUB	*	:		J	Z	j	z
11	VT ESC	+	;		K	[	k	{
12	FF FS	,	<		L	\	l	
13	CR GS	-	=		M	]	m	}
14	SO RS	.	>		N	^	n	~
15	SI US	/	?		O	_	o	DEL