



INSTITUTO FEDERAL
Sul de Minas Gerais

2017

MEDIOTEC - REDE - E TEC

APOSTILA DE **Informática** Programação estruturada

Profa. Ms. Giselle Cristina Cardoso



Programação estruturada

Profa. Ms. Giselle Cristina Cardoso

2017

Sumário

Apresentação da disciplina	5
Introdução	5
Definição	5
Linguagem C	6
A estrutura Básica do Programa	7
Primeiro Programa	7
Declarando Variáveis	11
Função scanf()	13
Exercícios Resolvidos	14
Exercícios Propostos	15
Operadores	15
Operadores Aritméticos	15
Operadores de Incremento (++) e Decremento (--)	16
Exercícios Resolvidos	17
Exercícios Propostos	18
Operadores Relacionais	18
Operadores Lógicos	19
Exercícios Propostos	19
Comandos Condicionais	20
Comando IF-ELSE	20
Exercícios Resolvidos	23
Exercícios Propostos	25
Comandos de Repetição	27
Comando FOR	27
Exercícios resolvidos	28
Exercícios Propostos	30
Comando WHILE	30
Exercícios resolvidos	31
Exercícios Propostos	32
Vetores	33
Exercícios resolvidos	34
Exercícios Propostos	35
Referências	35
Anexo 1	36
Introdução ao Code::Blocks	36
Download e Instalação	36
Utilizando o Code::Blocks	41
Anexo 2	45
Resolução dos Exercícios Propostos	45
A estrutura Básica do Programa	45
Exercícios Propostos	45
Operadores	46
Exercícios Propostos	46
Operadores Relacionais	48
Exercícios Propostos	48
Comandos Condicionais	49

Exercícios Propostos.....	49
Comandos de Repetição.....	52
Exercícios Propostos.....	52
Comando WHILE	55
Exercícios Propostos.....	55
Vetores	57
Exercícios Propostos.....	57

Apresentação da disciplina

A disciplina Programação Estruturada tem como objetivo principal ensinar você a modelar, desenvolver programas e implementá-los em linguagem C. Para que o objetivo seja alcançado a disciplina será dividida em quatro partes:

1. **Introdução, Tipos de dados e Operadores:** será apresentado como criar um programa utilizando a Linguagem C, suas principais características, quais os tipos de dados e como declarar variáveis. Finalizando esta parte, serão mostrados os operadores aritméticos que serão utilizados nos programas.
2. **Estruturas condicionais:** será abordado o comando condicional “*IF-ELSE*”, quando será ensinado como utilizar condições no programa implementado em Linguagem C. Para introduzir esta parte, veremos os operadores relacionais e lógicos, que são muito utilizados nos comandos condicionais e comandos de repetição.
3. **Estruturas de repetição:** nesta parte será apresentado como repetir alguns trechos do programa. Serão abordados os comandos de repetição “*FOR*” e “*WHILE*”.
4. **Vetores:** para finalizar a disciplina, será mostrado como armazenar e manipular dados armazenados em um vetor.

Para um bom aproveitamento desta disciplina, sugiro que todos os exemplos, exercícios resolvidos e propostos sejam implementados e executados em um dos compiladores sugeridos e que seja realizado o teste de mesa.

Introdução

Nesta disciplina, utilizando o paradigma de Programação Estruturada e a Linguagem de Programação C, eu e você ensinaremos o computador a realizar algumas tarefas simples, como por exemplo:

- Calcular a área e o perímetro de uma figura geométrica;
- Mostrar ao usuário seu novo salário, sabendo que seu aumento dependerá de sua função na empresa;
- Calcular a média das notas dos alunos;
- Entre muitas outras tarefas.

Definição

Programação Estruturada é uma das formas existentes para fazer programas para computadores, em que se utilizam, basicamente, três estruturas:

- Sequencial;
- Condicional;
- Repetição.

Para se utilizar a Programação Estruturada no computador, é necessário utilizar uma Linguagem de Programação que é responsável por traduzir as instruções de determinada tarefa para o computador realizar.

Entre as Linguagens de Programação Estruturada mais utilizadas, podemos citar:

- C;
- C++;
- Basic;
- Pascal;
- Cobol.

Neste material serão abordadas as três estruturas básicas da Programação Estruturada, utilizando a Linguagem de Programação C.

Linguagem C

A Linguagem C foi criada por Dennis M. Ritchie e Ken Thompson, no Laboratório Bell, nos EUA, em 1972 (MIZRAHI, 2008), e é utilizada até hoje, tanto como linguagem para ensinar a programar como na área de desenvolvimento de software.

Você Sabia?

Que C é uma linguagem utilizada para programação de vários sistemas? Como exemplo podemos citar os Sistemas Operacionais, as Planilhas, os Processadores de Texto, etc.

Você Sabia?

Segundo Mizhari (2008), a maneira de se comunicar com o computador chama-se programa e a única linguagem que o computador entende chama-se linguagem de máquina. Portanto, todos os programas que se comunicam com a máquina devem estar em linguagem de máquina.

A forma de tradução que a Linguagem C utiliza é o compilador. Um compilador é responsável pela leitura das instruções do programa, verificando se está correto. Se não houver nenhum erro, ele é convertido para a linguagem de máquina, gerando um arquivo com a extensão .EXE (este não pode ser alterado), que será nosso programa. Se houver erro no código, este deve ser corrigido.

Existem vários compiladores para a Linguagem C, entre eles podemos citar o “Code::Blocks”(Anexo 1) e o OnlineGDB.com (https://www.onlinegdb.com/online_c_compiler).

OnlineGDB.com é um compilador online da Linguagem C, que pode ser utilizado em qualquer navegador de internet, não importando se esse navegador está instalado no desktop, notebook, dispositivos móveis, etc. Para acessar essa ferramenta, acesse o site: https://www.onlinegdb.com/online_c_compiler.

Neste material, utilizaremos o compilador OnlineGDB.com para ilustrar as implementações e execuções dos códigos em Linguagem C.

A estrutura Básica do Programa

O código de um programa na Linguagem C é composto de uma ou várias “funções”. A função principal é chamada “main”, ela deve existir em algum lugar do programa, pois marca o seu início. Nesta disciplina vamos trabalhar somente com esta função.

Abaixo podemos ver a estrutura da função “main”:

```
main() // primeira função a ser executada
{      // início da função main()
}      // término da função main()
```

Primeiro Programa

Nosso primeiro programa terá como objetivo colocar a mensagem “**Primeiro Programa!**” na tela. Para isso, vamos utilizar a função “printf()”, responsável por colocar informações na tela do computador. Ela é uma função de E/S (Entrada e Saída), em que no interior dos seus parênteses estão as informações passadas para a função printf(), isto é, “Primeiro Programa”, que chamamos de argumentos.

Você Sabia?

Na Linguagem C existem bibliotecas compostas por um conjunto de funções já prontas que podem ser incluídas no programa para a utilização. A biblioteca “stdio.h” é composta pelas funções de E/S (Entrada e Saída) de dados do sistema.

Para que a função “*printf()*” funcione, precisamos incluir a biblioteca “*stdio.h*”. Para se incluir uma biblioteca no programa, utilizamos a diretiva “*#include*”, responsável por importar a biblioteca para o seu programa.

Na Linguagem C há alguns pontos muito importantes que não podemos esquecer:

- Cada comando termina com um “;” (ponto e vírgula);
- Ela é uma linguagem “*case sensitive*”, ou seja, ela diferencia letras maiúsculas e minúsculas;
- Todos os seus comandos são escritos com letras minúsculas.

Abaixo está nosso primeiro programa:

```
#include <stdio.h>

main () {

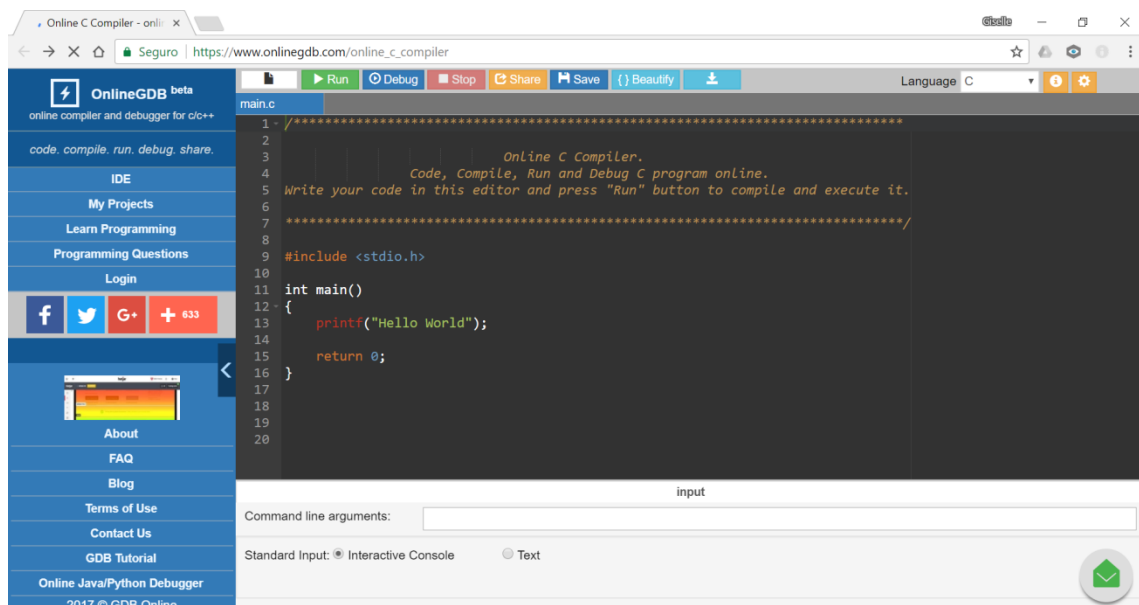
    printf("Primeiro Programa");

}
```

Neste caso, quando o programa encontra a linha “*printf*(“Primeiro Programa”);”, será impresso tela do computador a mensagem “*Primeiro Programa*”.

O próximo passo é implementarmos nosso programa no compilador e executá-lo.

Acesse o compilador OnlineGDB.com https://www.onlinegdb.com/online_c_compiler, irá aparecer a página ilustrada na figura a seguir:

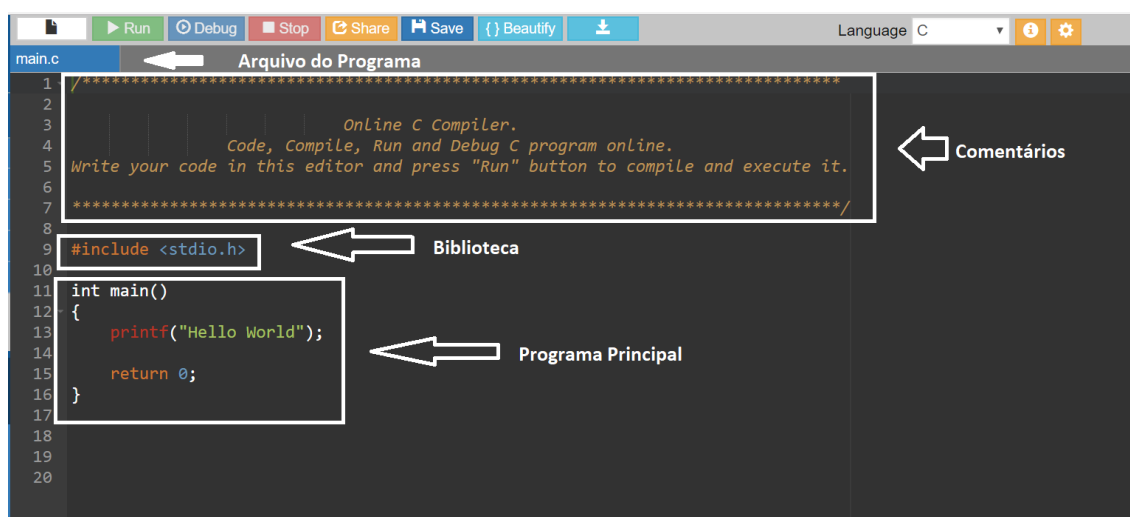


Nesta figura temos os seguintes componentes:

1. Barra de Ferramentas:

Componente	Função
	Criar novo arquivo.
	Compilar e executar o programa.
	Monitorar a execução do programa.
	Parar a execução do programa.
	Compartilhar o código do programa.
	Salvar em um repositório online o código do programa.
	Definir a estrutura do programa (identação).
	Fazer download do programa.
	Definição da Linguagem de Programação.
	Teclas de Atalho.
	Configuração da tela do compilador.

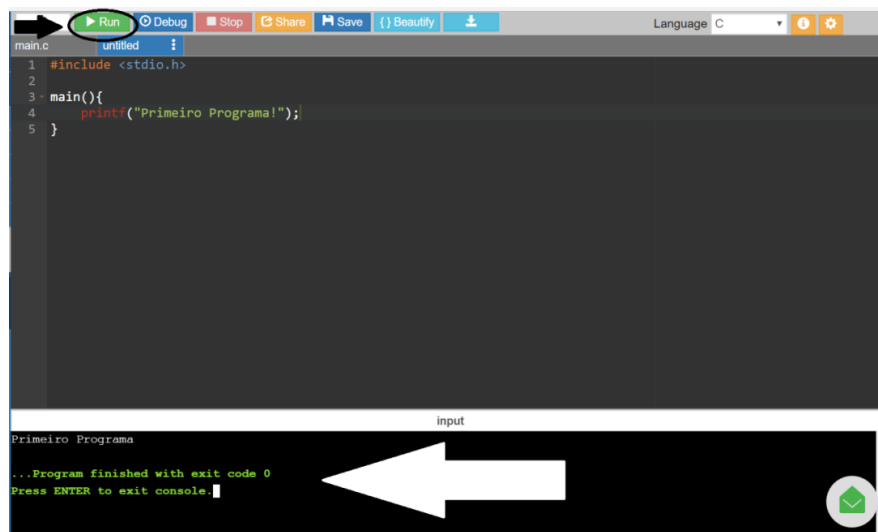
2. Tela do compilador:



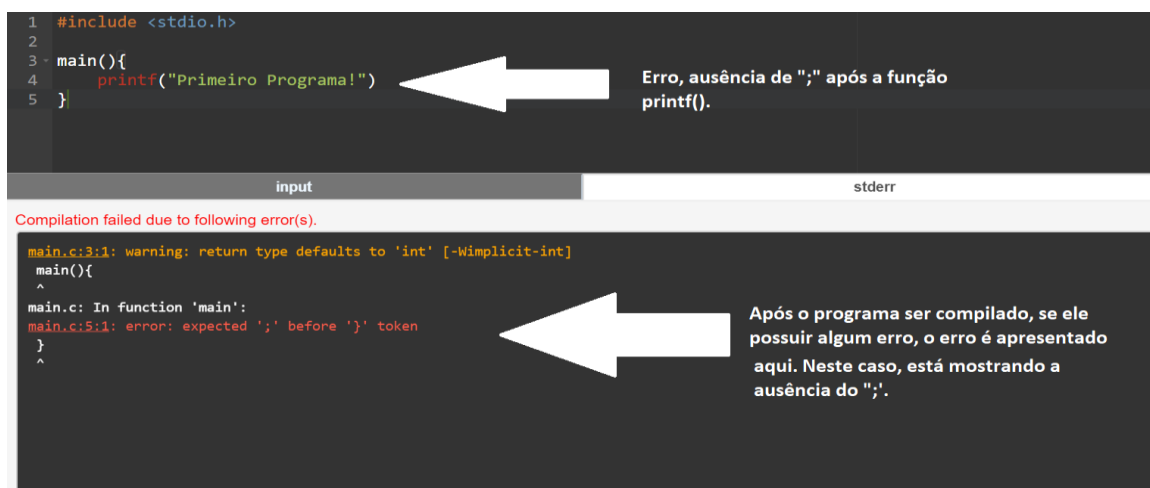
Para iniciarmos a implementação do nosso programa, digite o código do primeiro programa no compilador, conforme apresenta a figura abaixo:

```
main.c  untitled  ⋮
1  #include <stdio.h>
2
3  main(){
4      printf("Primeiro Programa!");
5  }
```

O próximo passo é compilar o programa, para isso clique no botão “RUN” na barra de ferramentas, se o programa estiver correto, ele será executado, conforme figura abaixo:



Se o programa apresentar algum erro, ele não vai rodar enquanto o erro não for corrigido, a figura abaixo ilustra um erro no programa após este ter sido compilado.



Para um bom aproveitamento do conteúdo da disciplina, a partir deste ponto, todos os programas devem ser compilados em um compilador da Linguagem C.

Declarando Variáveis

Variáveis podem ser definidas como espaços de memória do computador reservados para armazenar um valor, cada variável deve possuir um nome para que seu conteúdo (valor) possa ser acessado a qualquer instante, podendo ela, a cada instante, possuir valores diferentes.

Em um programa temos que declarar as variáveis que serão utilizadas durante sua execução. Portanto, chamamos de declaração de variáveis uma instrução para reservar uma quantidade de memória apropriada para armazenar o tipo especificado e indicar que o seu conteúdo será referenciado pelo nome dado a essa variável, pois todas as variáveis devem ser declaradas antes de ser utilizadas.

Os tipos de variáveis informam o tipo de dado (valor) que podem ser armazenados no espaço de memória reservado. Nesta disciplina vamos utilizar os tipos abaixo:

TIPO	DESCRIÇÃO
int	Espaço reservado para armazenar um número inteiro.
float	Espaço reservado para armazenar um número real.
char	Espaço reservado para armazenar um caracter (letra).

Exemplos:

- Reservar um espaço de memória para armazenar uma variável do tipo inteiro, como, por exemplo, a idade de uma pessoa:
 - `int idade;`
- Reservar um espaço de memória para armazenar uma variável do tipo real, como, por exemplo, a sua nota nesta disciplina:
 - `float nota;`
- Reservar um espaço de memória para armazenar uma variável do tipo caracter, como, por exemplo, uma letra:
 - `char letra;`

Você Sabia?

Que o nome de uma variável deve ter, obrigatoriamente, algum caracter.

Para atribuir um valor a uma variável, utilizamos o sinal de igual (=), que chamamos de operador de atribuição, como exemplificado a seguir:

- Atribuir o valor 15 à variável inteira idade:

- idade = 15;
- Atribuir o valor 8.5 para a variável real nota:
 - nota = 8.5; //utiliza-se “.” e não “,” para números reais.
- Atribuir a vogal “a” para a variável caracter letra:
 - letra = ‘a’; (os caracteres devem estar sempre entre aspas simples quando forem utilizados).

Para que uma variável possa ser utilizada como parâmetro da função “*printf()*”, é necessário especificar seu formato, ou seja, que tipo de informação será colocada na tela. Os formatos que utilizaremos serão:

Tipo	Formato
Inteiro (int)	%d
Real (float)	%f
Caracter (char)	%c

Não podemos esquecer de incluir a variável como parâmetro da função. Abaixo serão mostrados alguns exemplos:

	Função	O que sairá na tela?
Mostrar na tela o conteúdo da variável inteira idade	printf (“Idade = %d”, idade);	Idade = 15
Mostrar na tela o conteúdo da variável real nota	printf (“Nota = %f”, nota);	Nota = 8.5
Mostrar na tela o conteúdo da variável caracter letra	printf (“Letra = %c”, letra);	Letra = a

Portanto, a função “*printf()*” é formada basicamente por dois componentes, sendo o primeiro composto pelo que vai ser escrito na tela e a formatação da(s) variável(is) e o segundo é composto das variáveis. A função “*printf()*” com duas variáveis ficaria da seguinte forma:

printf (“Letra %c – Nota = %f”, letra, nota);

onde %c é o formato da variável letra e %f é o formato da variável nota.

Na tela apareceria o seguinte: “Letra a – Nota = 8.5”.

Exemplo:

```
#include <stdio.h>

main ( ){

    int num;

    num = 2;

    printf("Este e o numero dois: %d",num);

}
```

Alguns códigos especiais utilizados na função “*printf()*”:

\n	Mudar de linha (utilizado na função “ <i>printf()</i> ”)
// ou /*	Comentário (pode ser colocado em qualquer local do programa, não é compilado)

Função scanf()

Para fazer a leitura de uma informação (valor) do teclado, utiliza-se a função de E/S da biblioteca “stdio.h” chamada “*scanf()*”. Esta função possui o seguinte formato:

scanf (“expressão de controle”, argumento);

onde a “*expressão de controle*” deve conter o formato do valor a ser lido do teclado e o “argumento” deve ser o endereço(&) da variável em que o valor vai ser armazenado e a variável.

Exemplo:

```
#include <stdio.h>

main() {

    int num;

    printf ("Digite um valor:");

    scanf ("%d", &num);    //leitura do teclado de um valor inteiro, que será
                           //armazenado na variável num


    printf("Valor = %d", num);

}
```

Quando o usuário digita no teclado o valor solicitado, ele deve pressionar a tecla <ENTER> para que o valor seja enviado à função “*scanf()*”, conforme ilustrado abaixo:

```
main.c
1 #include <stdio.h>
2 main() {
3     int num;
4     printf("Digite um valor:");
5     scanf("%d", &num); //Leitura do teclado de um valor inteiro, que será
                        //armazenado na variável num
6     printf("Valor = %d", num);
7 }
8
9
10
11
```

input

Digite um valor:  Digite aqui um valor
Tecla <ENTER> 

Exercícios Resolvidos

1. Faça um programa que contenha somente um comando para imprimir na tela as informações abaixo:

Curso: Técnico em Informática
Disciplina: Programação Estruturada
Ano: 2017

Solução:

```
#include <stdio.h>

main(){
    printf(" Curso: Técnico em Informática \n Disciplina: Programação Estruturada\n Ano: 2017");
}
```

2. Faça um programa em linguagem C que contenha uma variável inteira e outra do tipo real. O programa deve solicitar ao usuário que digite um valor para cada uma das variáveis e apresentar os valores na tela.

Solução:

```
#include <stdio.h>

main(){
    int x;
    float z;
```

```
printf("Digite um valor inteiro:");
scanf("%d", &x);
printf("Digite um valor real:");
scanf("%f", &z);
printf("X = %d - Y = %f", x, z);
}
```

Exercícios Propostos

1. Faça um programa que mostre na tela os seguintes dados seus:
 - ✓ Nome
 - ✓ Curso
2. Faça um programa capaz de ler um caracter e um número inteiro do teclado. Após a leitura apresente-os na tela do computador.

Operadores

Os operadores que serão utilizados nesta disciplina podem ser divididos em:

- Operadores Aritméticos: são os operadores matemáticos que irão auxiliar na resolução de problemas que envolvem operações matemáticas.
- Operadores de Decremento e Incremento: são considerados operadores matemáticos especiais.
- Operadores Relacionais: são os operadores utilizados para fazerem comparações entre as variáveis.
- Operadores Lógicos: utilizados para realizar operações lógicas com duas ou mais expressões matemáticas ou lógicas.

Nas próximas seções serão apresentados todos esses operadores e suas utilizações.

Operadores Aritméticos

Os operadores aritméticos binários, conhecidos como operadores matemáticos (ilustrados abaixo), são aqueles que fazem operações sobre dois números e que utilizamos no nosso dia a dia. Os operadores mais comuns são:

=	Atribuição
+	Soma
-	Subtração
/	Divisão
*	Multiplicação

Para entendermos o que o computador está fazendo, realizamos o “teste de mesa”, que consiste em executar passo-a-passo o programa em um papel, com a finalidade de verificar os passos do programa e corrigir os erros encontrados. A partir de agora, é importante que a cada programa implementado seja realizado o teste de mesa, mesmo que não for solicitado.

Exemplo:

//transformar a temperatura de graus Fahrenheit para graus Celsius

```
#include <stdio.h>
main(){
float ftemp, ctemp;
printf ("Digite a temperatura em graus Fahrenheit:");
scanf ("%f",&ftemp); //ê ftemp do teclado
ctemp = (ftemp - 32) *5/9;
printf("Temperatura em graus Celsius: %f",ctemp);
}
```

Teste de Mesa		
ftemp	ctemp	Tela
32		
	0	
		Temperatura em graus Celsius: 0

Operadores de Incremento (++) e Decremento (--)

São operadores muito utilizados na Linguagem C, sendo que a principal função é abreviar alguns comandos. Na tabela abaixo são apresentados esses operadores, quando utilizados pós-fixados, e seu funcionamento:

++	Incrementa 1 na variável, depois da variável a ser utilizada.
--	Decrementa 1 na variável, depois da variável a ser utilizada.

Exemplo:

```
#include <stdio.h>

main(){
    int x, n = 5;

    x = n++;                      //equivale aos comandos "x = n;" e "x = x + 1;"
    printf ("x = %d n = %d",x,n);
}
```

Saída: x = 5 n = 6

Exercícios Resolvidos

1. Faça um programa para o usuário calcular a área de um quadrado. Lembre-se: $\text{Área} = \text{lado} * \text{lado}$.

Solução:

```
#include <stdio.h>

main(){
    float area, lado;
    printf ("Digite o lado do quadrado:");
    scanf ("%f", &lado);
    area = lado*lado;
    printf ("Area do quadrado = %f", area);
}
```

2. Faça um programa para o usuário calcular um aumento de 20% no seu salário.

Solução:

```
#include <stdio.h>

main(){
    float salario, novosalario;
    printf ("Digite o seu salario:");
    scanf ("%f", &salario);
    novosalario = salario + salario *20/100;
    printf ("Novo salario = %f", novosalario);
}
```

Exercícios Propostos

1. Faça um programa que solicite ao usuário dois números inteiros. Ele deve calcular e mostrar a soma, subtração, multiplicação e divisão destes dois números.
2. Escreva um programa que solicite 3 números reais e imprima na tela a média aritmética.
3. Faça um programa que calcule e mostre a área de um círculo. Sabe-se que: $\text{Area} = \pi \text{Raio}^2$ e $\pi = 3,1415$.
4. Escreva um programa no qual você digita o ano de nascimento, o ano atual, e ele calcula e informa quantos anos você tem ou vai fazer neste ano.
5. Faça um programa que receba o preço de uma compra, calcule e mostre o preço final da compra, sabendo-se que esta sofreu um desconto de 15%.

Operadores Relacionais

Estes operadores são utilizados para fazerem comparações entre dois valores, na Linguagem C temos os seguintes operadores:

Operador	Descrição
==	Igual
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
!=	Diferente

O resultado de uma operação que utiliza operadores relacionais assume 1 (verdadeiro) se for verdadeiro ou 0 (falso) se for falso.

Exemplo:

```
#include <stdio.h>

main() {
    int verdade, falso;
```

```

verdade = (15<20);
falso = (15==20);
printf("Verdadeiro = %d, Falso = %d", verdade, falso);
}

```

Saída: Verdadeiro = 1, Falso = 0

Operadores Lógicos

Os operadores lógicos são operadores que fazem operações somente com valores 1 (verdadeiro) e 0 (falso) e são muito utilizados nos comandos condicionais e comandos de repetição em qualquer linguagem de programação. Existem 3 operadores lógicos, sendo eles o E (AND), OU (OR) e o NÃO (NOT).

Na Linguagem C, o operador E é representado por "&&", o operador OU é representado por "||" e o operador NÃO é representado por "!". A tabela a seguir ilustra o resultado das operações com cada um desses operadores:

a	b	!a	!b	a && b	a b
1	1	0	0	1	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0

Exemplo: Suponha que temos três variáveis: A = 5, B = 8 e C = 1. Os resultados das expressões seriam:

Expressões	Resultado	Comentários
(A == B) && (B > C)	0	(A == B) é falso e (B > C) é verdadeiro
(A != B) (B < C)	1	(A != B) é verdadeiro e (B < C) é falso
! (A > B)	1	(A > B) é falso
(A < B) && (B > C)	1	(A < B) é verdadeiro e (B > C) é verdadeiro
(A >= B) (B == C)	0	(A >= B) é falso e (B == C) é falso
! (A <= B)	0	(A <= B) é verdadeiro

Você Sabia?

Que operadores aritméticos têm precedência sobre os operadores relacionais e lógicos? Ou seja, as operações matemáticas devem ser executadas primeiro.

Exercícios Propostos

1. Sabendo que $A = 5$, $B = 9$ e $C = 1$, informe se as expressões abaixo são verdadeiras ou falsas.
 - a) $(A+B) > C$
 - b) $B \geq (A - 3)$
 - c) $C == (B - A)$
 - d) $(B + A) \leq C$
 - e) $(C+A) > B$
2. Sabendo que $A = 6$, $B = 5$, $C = 4$ e $D = 7$, informe se as expressões abaixo são verdadeiras ou falsas.
 - a) $(A > C) \ \&\& \ (C \leq D)$
 - b) $(A \geq C) \ \&\& \ (D \geq C)$
 - c) $(A + B) > (C + D) \ || \ (D \geq C)$
 - d) $!((A+C) < D \ || \ (A==B))$

Comandos Condicionais

Os comandos condicionais são responsáveis por determinar se um trecho de programa deve ou não ser executado a partir do resultado de uma expressão lógica. Existem dois tipos de comandos condicionais, são eles o "IF-ELSE" e o "SWITCH-CASE". Nesta disciplina vamos estudar somente o comando "IF-ELSE", por se tratar do comando condicional mais utilizado.

Comando IF-ELSE

Existem algumas formas diferentes de se utilizar o comando "IF-ELSE", como apresentado abaixo:

- A primeira maneira de se utilizar o comando condicional "IF" é quando temos somente um **comando** dentro dele.

Sintaxe:

```
if (condição)
    comando;
```

Desta maneira, quando se encontra o comando condicional "IF", a **condição** é avaliada, se ela for verdadeira, o **comando** é executado e segue executando os demais comandos que compõem o programa. Se a **condição** for falsa, o **comando não** é executado e segue executando os demais comandos que compõem o programa.

Exemplo 1:

```
#include <stdio.h>
main(){
int x = 3, y = 4;
if (x < y)    //avaliar a condição x<y (3 < 4), ou seja, é uma condição verdadeira
    printf("x é menor que y"); // executa o comando, pois a condição é verdadeira
printf("\nFim do programa."); //executa independentemente da condição,
}
```

Exemplo 2:

```
#include <stdio.h>
main(){
int x = 3, y = 4;
if (x > y)    //avaliar a condição x>y (3 > 4), ou seja, é uma condição falsa
    printf("x é maior que y"); // não executa o comando, pois a condição é falsa
printf("\nFim do programa."); //executa independentemente da condição,
}
```

- A segunda maneira de se utilizar o comando condicional "IF" é quando temos vários **comandos** dentro dele.

Sintaxe:

```
if (condição)
{
    //marca o início do comando condicional "IF"
    comando1;
    comando2;
} //marca o fim do comando condicional "IF"
```

Desta maneira, quando se encontra o comando condicional "IF", a **condição** é avaliada, se ela for verdadeira, os **comandos** são executados e segue executando os demais comandos que compõem o programa. Se a **condição** for falsa, os **comandos não** são executados e segue executando os demais comandos que compõem o programa.

Exemplo 1:

```
#include <stdio.h>
main(){
int x = 3, y = 4;
if (x < y){    //avaliar a condição x<y (3 < 4), ou seja, é uma condição verdadeira
    printf("x é menor que y"); //executa o comando, pois a condição é verdadeira
    printf("\nx é menor que y");//executa o comando, pois a condição é verdadeira
}
```

```

    }
    printf("\nFim do programa."); //executa independentemente da condição,
}

```

Exemplo 2:

```

#include <stdio.h>

main(){
    int x = 3, y = 4;
    if (x > y){ //avaliar a condição x>y (3 > 4), ou seja, é uma condição falsa
        printf("x é maior que y"); // não executa o comando, pois a condição é falsa
        printf("\nx é maior que y"); // não executa o comando, pois a condição é falsa
    }
    printf("\nFim do programa."); //executa independentemente da condição,
}

```

- A terceira maneira de se utilizar o comando condicional "IF" é utilizar o comando complementar "ELSE". Neste caso, tanto o comando "IF" quanto o comando "ELSE" podem ter somente um ou vários **comandos** dentro deles. Importante não esquecer que o "ELSE" não possui condição.

Sintaxe:

```

if (condição)
{
    //marca o início do comando condicional "IF"
    comando1;
    comando2;
} //marca o fim do comando condicional "IF"

else
{
    //marca o início do comando complementar "ELSE"
    Comando3;
    Comando4;
} //marca o fim do comando complementar "ELSE"

```

Desta maneira, quando se encontra o comando condicional "IF", a **condição** é avaliada, se ela for verdadeira, os **comandos** que compõem o "IF" são executados. Se a **condição** for falsa, os **comandos** que serão executados são os **comandos** que compõem o comando complementar "ELSE". Após a execução do "IF" ou "ELSE", segue executando os demais comandos que compõem o programa.

Exemplo 1:

```
#include <stdio.h>

main(){
    int x = 3, y = 4;
    if (x < y)      //avaliar a condição x<y (3 < 4), ou seja, é uma condição verdadeira
        printf("x é menor que y"); //executa o comando, pois a condição é verdadeira
    else
        printf("x é maior que y"); //não executa o comando, pois o "IF" foi executado
    printf("\nFim do programa."); //executa independentemente da condição,
}
```

Exemplo 2:

```
#include <stdio.h>

main(){
    int x = 3, y = 4;
    if (x > y)      //avaliar a condição x>y (3 > 4), ou seja, é uma condição falsa
        printf("x é maior que y"); // não executa o comando, pois a condição é falsa
    else
        printf("x é maior que y"); //executa o comando, pois o "IF" não foi executado
    printf("\nFim do programa."); //executa independentemente da condição,
}
```

Você Sabia?

Um comando condicional "IF-ELSE" pode ser utilizado dentro de outro comando condicional "IF-ELSE".

Exercícios Resolvidos

1. Faça um programa para mostrar se o número inteiro digitado pelo usuário é positivo.

Solução:

```
#include <stdio.h>

main(){
    int n;
    printf ("Digite um número inteiro:");
    scanf ("%d", &n);
    if (n>=0)
```

```

    printf("Número Positivo");
}

```

2. Faça um programa para mostrar se o número inteiro digitado pelo usuário é positivo ou negativo.

Solução:

```

#include <stdio.h>

main(){
    int n;
    printf("Digite um número inteiro:");
    scanf ("%d", &n);
    if (n>=0)
        printf("Número Positivo");
    else
        printf("Número Negativo");
}

```

3. Faça um programa em que o usuário digite o valor de um celular e calcule o desconto de acordo com as regras abaixo:

10% para celulares com valor até R\$500,00 (inclusive);

20% para celulares com valor acima de R\$500,00 e inferior R\$1.500,00 (inclusive);

30% para celulares com valor acima de R\$1.500,00 e inferior a R\$3.000,00 (inclusive);

40% para celulares com valor acima de R\$3.000,00.

Após o cálculo, o programa deve mostrar o preço com desconto ao usuário.

Existem 2 formas de implementar uma solução para este problema:

Solução1 - utilizando somente "IF":

```

#include <stdio.h>

main(){
    float valor, valorcomdesconto;
    printf("Digite o valor do celular:");
    scanf ("%f", &valor);
    if (valor<=500)
        valorcomdesconto = valor - valor*0.1;
    if ((valor > 500) && (valor <=1500))
        valorcomdesconto = valor - valor*0.2;
    if ((valor >1500) && (valor <=3000))
        valorcomdesconto = valor - valor*0.3;
}

```



```

if (valor > 3000)
    valorcomdesconto = valor - valor*0.4;
printf("Valor final com desconto = %f", valorcomdesconto);
}

```

Solução2 - utilizando "IF-ELSE":

```

#include <stdio.h>
main(){
    float valor, valorcomdesconto;
    printf("Digite o valor do celular:");
    scanf("%f", &valor);
    if (valor<=500)
        valorcomdesconto = valor - valor*0.1;
    else
        if (valor <=1500)
            valorcomdesconto = valor - valor*0.2;
        else
            if (valor <=3000)
                valorcomdesconto = valor - valor*0.3;
            else
                valorcomdesconto = valor - valor*0.4;
    printf("Valor final com desconto = %f", valorcomdesconto);
}

```

Exercícios Propostos

1. Faça um programa em linguagem C em que o usuário digita 3 números inteiros e ele mostra na tela qual é o menor número.
2. Sabendo que uma escola realiza 4 provas bimestrais durante seu ano letivo, faça um programa que leia as 4 notas (N1,N2,N3,N4), calcule a média das notas e indique a situação do aluno, conforme abaixo:

Nota	Situação
Média >= 6.0	Aprovado
Média < 6.0 e Média >=4.0	Recuperação
Média < 4.0	Reprovado

3. Desenvolva um programa para calcular a conta de seu celular, você deve digitar quantos minutos foram utilizados e calcular conforme as regras a seguir:
- Até 100 minutos, incluindo a internet, total da conta R\$39,99;
 - Até 200 minutos, incluindo a internet, total da conta R\$59,99;
 - Até 500 minutos, incluindo a internet, total da conta R\$99,99;
 - Acima de 500 minutos, incluindo a internet, paga-se R\$99,99 mais R\$1,00 cada minuto excedente.

Não se esqueça de apresentar na tela o total da conta.

4. Após a execução do algoritmo a seguir, qual será a saída?

```
#include <stdio.h>

main(){
    int x,y,z,aux;
    x = 8;
    y = 6;
    z = 1;
    aux = 0;
    if((x>y)||((x>z))){
        if (z>y){
            aux = x;
            x = y;
            y = aux;
        }
        else{
            aux = x;
            x = z;
            z = aux;
        }
    }
    if (y>z){
        aux = y;
        y = z;
        z = aux;
    }
    printf("x = %d y = %d z = %d",x,y,z);
}
```

Comandos de Repetição

Em alguns momentos, alguns trechos do programa devem ser repetidos várias vezes. Para que esses trechos de programa se repitam utilizamos os comandos de repetição. Existem 3 tipos de comandos de repetição: “FOR”, “WHILE” e “DO-WHILE”. Nesta disciplina vamos focar nos dois primeiros comandos.

Comando FOR

O comando “FOR” é o comando de repetição utilizado para se repetir um determinado trecho do programa um número definido de vezes, ou seja, quando utilizamos esse comando, sabemos quantas vezes o trecho de código vai se repetir.

O comando de repetição “FOR” é composto por comandos, como segue abaixo:

Sintaxe:

```
for (inicialização; condição; incremento){  
    comandos;  
}
```

Onde:

- **inicialização:** instrução de atribuição, onde atribuímos um valor inicial à variável responsável por contar quantas vezes o trecho de programa vai se repetir. A inicialização é executada uma única vez, quando o comando “for” vai começar;
- **condição:** instrução de teste que controla o comando de repetição, ela é avaliada como verdadeira ou falsa toda vez para verificar se o trecho de programa deve ser repetido ou não. Se verdadeira, o trecho é repetido, e se for falsa, o programa passa para a instrução seguinte;
- **incremento:** define como a variável que controla o número de repetição do comando de repetição será alterada antes de avaliar se o trecho será repetido. Essa instrução é executada, toda vez, imediatamente após a execução do trecho de programa que está sendo repetido. Pode ser feito tanto incremento quanto decremento da variável de controle.

Exemplo 1:

```
/*Imprime os números de 0 a 3*/  
#include <stdio.h>  
main(){  
    int contador;  
    for (contador=0;contador<=3;contador++){ // repetir 10 vezes  
        printf("Contador = %d \n",contador);  
    }  
}
```

Teste de Mesa:

	Variável: Contador	Tela	Observação
Antes do comando “FOR”			
Ao iniciar o comando “FOR”	0		Incrementa a variável contador
			Contador é menor que 3
		Contador = 0	
Primeira vez	1		
			Contador é menor que 3
		Contador = 1	
Segunda vez	2		Incrementa a variável contador
			Contador é menor que 3
		Contador = 2	
Terceira vez	3		Incrementa a variável contador
			Contador é menor que 3
		Contador = 3	
Quarta vez	4		Incrementa a variável contador
			Contador é menor que 3? Não, portanto, sai do laço.
Após o comando for			

Você Sabia?

- Um comando de repetição pode ser utilizado dentro de outro comando de repetição;
- Um comando condicional pode ser utilizado dentro de um comando de repetição;
- Um comando de repetição pode ser utilizado dentro de um comando condicional.

Exercícios resolvidos

1. Modifique o exemplo para as seguintes situações:

- a. Mostrar na tela os números pares entre 0 e 10.

Solução:

```
#include <stdio.h>

main(){
    int contador;
    for (contador=0;contador<=10;contador = contador + 2){
        printf("Contador = %d \n",contador);
    }
}
```

b. Mostrar na tela os números de 9 a 0 (decrecente).

Solução:

```
#include <stdio.h>

main(){
    int contador;
    for (contador=9;contador>=0;contador--){
        printf("Contador = %d \n",contador);
    }
}
```

2. Faça um programa para calcular a média aritmética de 6 números reais.

Solução:

```
#include <stdio.h>

main(){
    float numero, soma=0, media;           //variável soma será utilizada para armazenar a soma //de
                                           todos os números

    int contador;
    for (contador=0;contador<6;contador++){
        printf("Digite um número:");
        scanf("%f", &numero);
        soma = soma+numero;                //a cada execução deste comando adiciona à variável o
                                           //número digitado
    }

    media = soma/6;
    printf ("Media =%f", media);
}
```

Exercícios Propostos

1. Faça um programa para calcular a soma de 10 números inteiros digitados pelo usuário e apresentá-los na tela.
2. Faça um programa em C em que o usuário digita a quantidade de alunos. Depois ele deve ler a nota de cada um dos alunos, calcular e mostrar a média aritmética das notas de todos os alunos. Se algum aluno estiver com a nota abaixo de 6.0, mostrar a mensagem na tela: "Nota baixa".
3. Faça um programa que leia 10 números inteiros e que determine e mostre qual o maior número lido.
4. Faça um programa em que o usuário digite um número e ele apresente na tela a tabuada do 0 ao 10 do número digitado, conforme abaixo:

Número digitado: 5

0 x 5 = 0

1 x 5 = 5

2 x 5 = 10

3 x 5 = 15

4 x 5 = 20

5 x 5 = 25

6 x 5 = 30

7 x 5 = 35

8 x 5 = 40

9 x 5 = 45

10 x 5 = 50

5. Durante uma corrida de bicicleta com 10 voltas de duração foram anotados para um ciclista, na ordem, os tempos registrados em cada volta. Fazer um programa em C para ler os tempos das 10 voltas, calcular e imprimir:
 - ✓ melhor tempo e a volta em que ocorreu;
 - ✓ o tempo total de todas as voltas.
 - ✓ tempo médio das N voltas.

Comando WHILE

O comando de repetição "WHILE" repete um determinado trecho que programa enquanto sua condição for verdadeira. Se a condição for verdadeira, o trecho de programa do comando "WHILE" é executado uma vez e a condição é avaliada novamente. Esse ciclo é repetido até que a condição se torne falsa, então o comando de repetição termina e o programa passa para a próxima linha após o comando "WHILE".

Sintaxe:

```
while (condição){
```

```
comandos;  
}
```

Você Sabia?

Que p comando de repetição “WHILE” é mais utilizado em situações em que o laço pode ser terminado inesperadamente por condições desenvolvidas dentro dele mesmo?

Exemplo:

```
// mostrar na tela os números de 0 a 10  
#include <stdio.h>  
main(){  
int contador = 0;  
while (contador <= 10){  
    printf("Contador = %d\n",contador);  
    contador++;  
}  
}
```

Exercícios resolvidos

1. Modifique o exemplo para as seguintes situações:
 - a. Mostrar na tela os números pares entre 0 e 10.

Solução:

```
#include <stdio.h>  
main() {  
    int contador =0;  
    while (contador <=10){  
        printf("Contador = %d \n",contador) ;  
        contador = contador + 2;  
    }  
}
```

- b. Mostrar na tela os números de 9 a 0 (decrescente).

Solução:

```
#include <stdio.h>  
main() {
```

```

    int contador = 9;
    while (contador >=0){
        printf("Contador = %d \n",contador);
        contador --;
    }
}

```

2. Faça um programa para calcular a soma de vários números reais. Ele deve apresentar a soma na tela quando o usuário digitar 0.

Solução:

```

#include <stdio.h>

main(){
    float numero=1, soma=0;           //número=1, para que ele seja diferente de 0
    while (numero !=0){               // condição: número diferente de 0
        printf("Digite um número: ");
        scanf("%f", &numero);
        soma = soma+numero;           //a cada execução deste comando adiciona à variável o
                                      //número digitado
    }
    printf ("Soma =%f", soma);
}

```

Exercícios Propostos

1. Escreva um programa que leia várias notas e calcule a média, sabendo que ele deve mostrar a média quando a nota lida for menor que 0.
2. Desenvolva um programa que faça a captura de N números reais. O programa deverá calcular o somatório e a média desses números. Pergunte o valor de N ao usuário.
3. Desenvolva um programa que faça a captura de 10 números reais e calcule a média. Na sequência, o programa deverá solicitar a inserção de mais 10 números reais. Para cada um desses números digitados, o programa deverá informar se está abaixo ou acima da média dos 10 primeiros números digitados.
4. Faça um programa que receba a idade de várias pessoas e que calcule e mostre a quantidade de pessoas com idade maior ou igual a 30 anos. O programa deve parar de ler as idades quando ler uma idade igual a 0.

Vetores

Muitas vezes, em um programa precisamos de dezenas ou centenas de variáveis, neste caso fica complicado declarar tantas variáveis. Para resolver esse problema, podemos definir a variável como um vetor, ou seja, em uma única variável podemos armazenar vários valores de um mesmo tipo.

A declaração de um vetor na Linguagem C é realizada da seguinte forma “tipo_da_variável nome[tamanho]”, onde tamanho define quantos valores serão armazenados na variável. Para que fique mais fácil imaginar um vetor, podemos representá-lo da seguinte forma:

conteúdo										
posição	0	1	2	3	4	5	6	7	8	9

Onde temos um vetor com 10 posições, ou seja, nesse vetor podemos armazenar 10 valores.

Na Linguagem C, a primeira posição do vetor é a posição 0 (zero), nesse caso em um vetor de 10 posições a última posição será a posição 9.

Exemplo:

- Declarar uma variável para armazenar 10 valores inteiros:
`int vetor[10];`
- Declarar uma variável para armazenar 5 notas de um aluno:
`float notas[5];`

Para armazenar um conteúdo (valor) em uma determinada posição de um vetor é necessário somente indicar a posição do vetor em que o valor deve ser colocado, como apresentado abaixo:

- Armazenar o valor 7 na terceira posição do “vetor”:
`vetor[2] = 7;`

		7							
0	1	2	3	4	5	6	7	8	9

A terceira posição do vetor é a posição com índice 2, pois começamos a contar da posição 0.

- Armazenar a nota 6.5 na primeira posição do vetor “notas”:
`vetor[0] = 6.5;`

Exemplo:

```
//programa para ler 10 valores inteiros para um vetor e mostrar na tela
#include <stdio.h>
main(){
    int vetor[10], i;
    //leitura de 10 valores inteiros
    for (i=0; i<10; i++){
```

```

        printf("Digite um número:");
        scanf("%d", &vetor[i]);
    }
    //mostrar os valores armazenados em um vetor
    for (i=0; i<10; i++)
        printf(" %d ", vetor[i]);
}

```

Exercícios resolvidos

1. Faça um programa para armazenar 10 números reais em um vetor. Para finalizar, ele deve apresentar os valores armazenados e a soma desses números.

Solução:

```

#include <stdio.h>

main(){
    float vetor[10], soma=0;
    int i;
    //leitura de 10 valores reais e soma
    for (i=0; i<10; i++){
        printf("Digite um número:");
        scanf("%f", &vetor[i]);
        soma = soma + vetor[i];
    }
    //mostrar os valores armazenados em um vetor
    for (i=0; i<10; i++)
        printf(" %f ", vetor[i]);
    //mostrar a soma dos números armazenados no vetor
    printf("\nSoma = %f",soma);
}

```

2. Faça um programa para armazenar 10 números em um vetor e calcular a média. Este programa deve mostrar, ao finalizar, quais números estão acima da média.

Solução:

```

#include <stdio.h>

main(){
    float vetor[10], soma=0, media;
    int i;
    //leitura de 10 valores reais e soma
    for (i=0; i<10; i++){

```

```

        printf("Digite um número:");
        scanf("%f", &vetor[i]);
        soma = soma + vetor[i];
    }
    //cálculo da média
    media = soma/10;
    //mostrar os valores armazenados que estão acima da média
    for (i=0; i<10; i++)
        if (vetor[i]>media)
            printf(" %f ", vetor[i]);
    }

```

Exercícios Propostos

1. Faça um programa que, tendo como dados de entrada do teclado 10 valores reais que deverão ser armazenados em um vetor, calcule a média e mostre quantos elementos são superiores à média calculada.
2. Faça um programa que, tendo como dados de entrada do teclado 10 valores inteiros, mostre a posição em que foi armazenado o maior valor.
3. Faça um programa que armazene 10 números inteiros digitados pelo usuário e como saída ele deve apresentar o vetor ao contrário.
4. Uma marca possui 10 produtos, cada um com seu código (0 a 9) e o seu custo. Ao longo da semana, o sistema acumula o valor gasto na produção de cada produto. Utilize um vetor para armazenar essas informações. O programa, ao finalizar, apresenta o valor gasto por cada produto. Para que o programa seja finalizado, entra-se no sistema com um código do produto igual a -1. Você está convidado a desenvolver um programa para resolver este problema.

Referências

CODE::BLOCKS. The IDE with all the features you need, having a consistent look, feel and operation across platforms. <<http://www.codeblocks.org/>> Acesso em: 18 out. 2017.

GDB ONLINE. OnlineGDB^{beta}: online compiler and debugger for c/c++. Disponível em <https://www.onlinegdb.com/online_c_compiler> Acesso em: 18 out. 2017.

MIZRAHI, V. V. **Treinamento em Linguagem C**. 2. ed. Prentice Hall Brasil, 2008.

Anexo 1

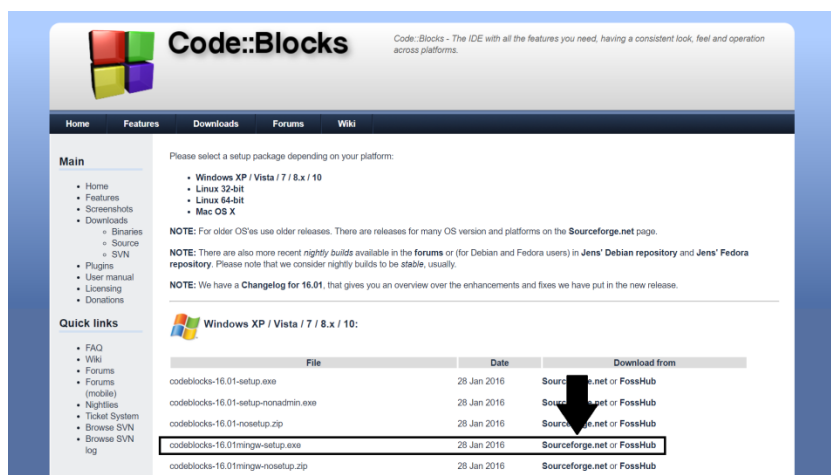
Introdução ao Code::Blocks

O Code::Blocks é um ambiente de desenvolvimento integrado de código aberto e multiplataforma, utilizado também como compilador da Linguagem de Programação C. Sua última versão foi lançada em 2016. Possui versões de instalação para os Sistemas Operacionais Windows, Linux e MAC. Mais informações sobre essa ferramenta podem ser encontradas no seu site oficial: <http://www.codeblocks.org/>

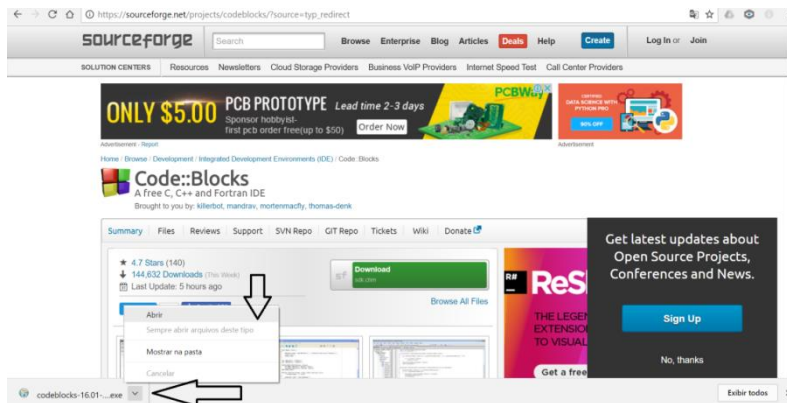
Este material tem como objetivo mostrar passo-a-passo a instalação do Code::Blocks no Sistema Operacional Windows XP/Vista/8.x/10.

Download e Instalação

1. Acesse o site: <http://www.codeblocks.org/downloads/26>
2. A primeira atividade é iniciar o download do arquivo da ferramenta (**codeblocks-16.01mingw-setup.exe**) no site, clique então no repositório “Sourceforge.net”, conforme ilustrado na figura abaixo:

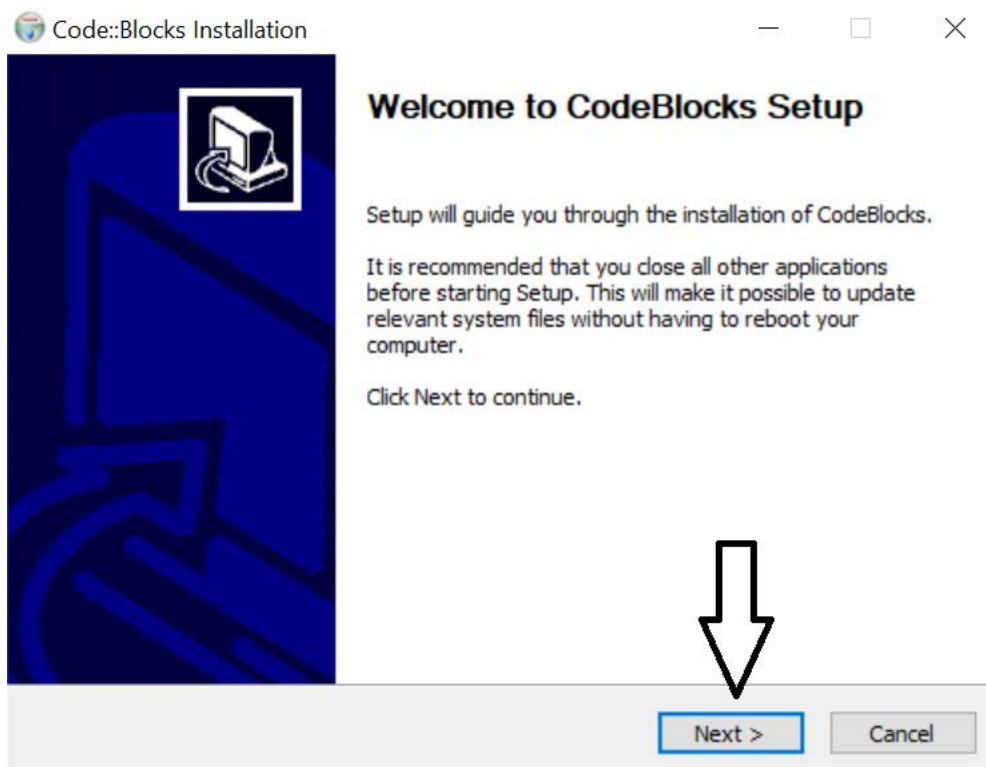


Irá iniciar o download da ferramenta. Aguarde até ser concluído. Após a conclusão, clique na seta que está à direita do arquivo de download e em seguida na opção “Abrir”, conforme figura abaixo:

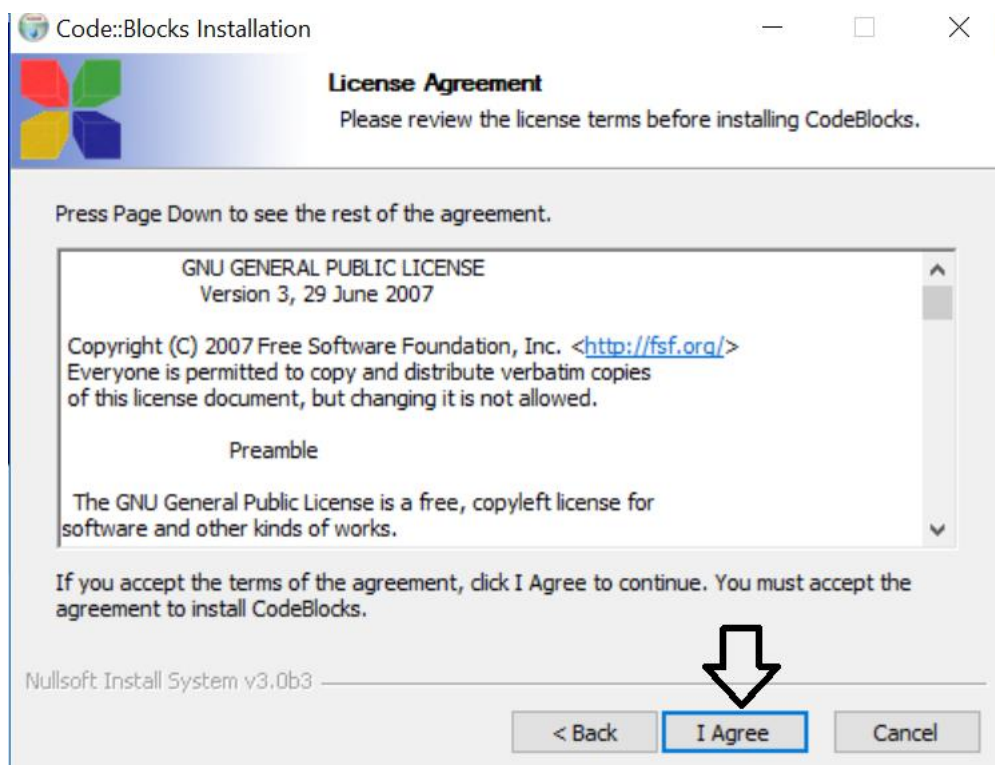


Aparecerá uma mensagem do Sistema Operacional pedindo a autorização para fazer alterações, clique em **“Sim”** para iniciar a instalação.

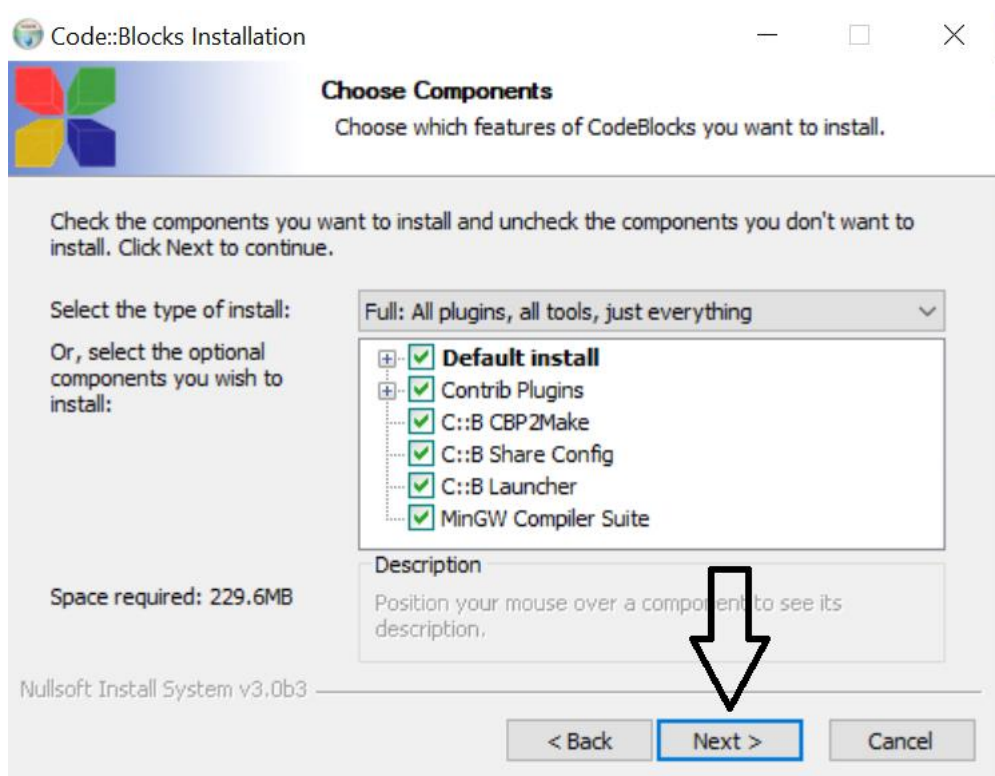
3. Ao iniciar a instalação, aparecerá a tela abaixo, clique na opção **“Next”**.



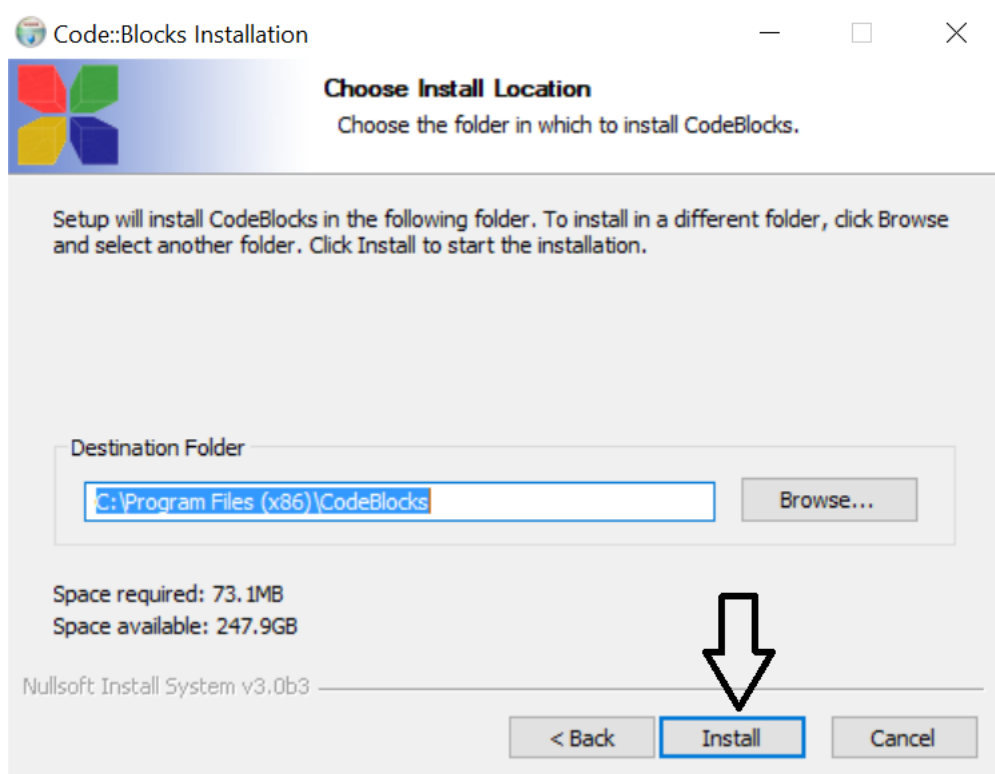
Em seguida, aparecerá outra tela, solicitando que você aceite o termos de utilização, faça a leitura, se concordar com os termos, clique em **“I Agree”**, conforme figura abaixo:



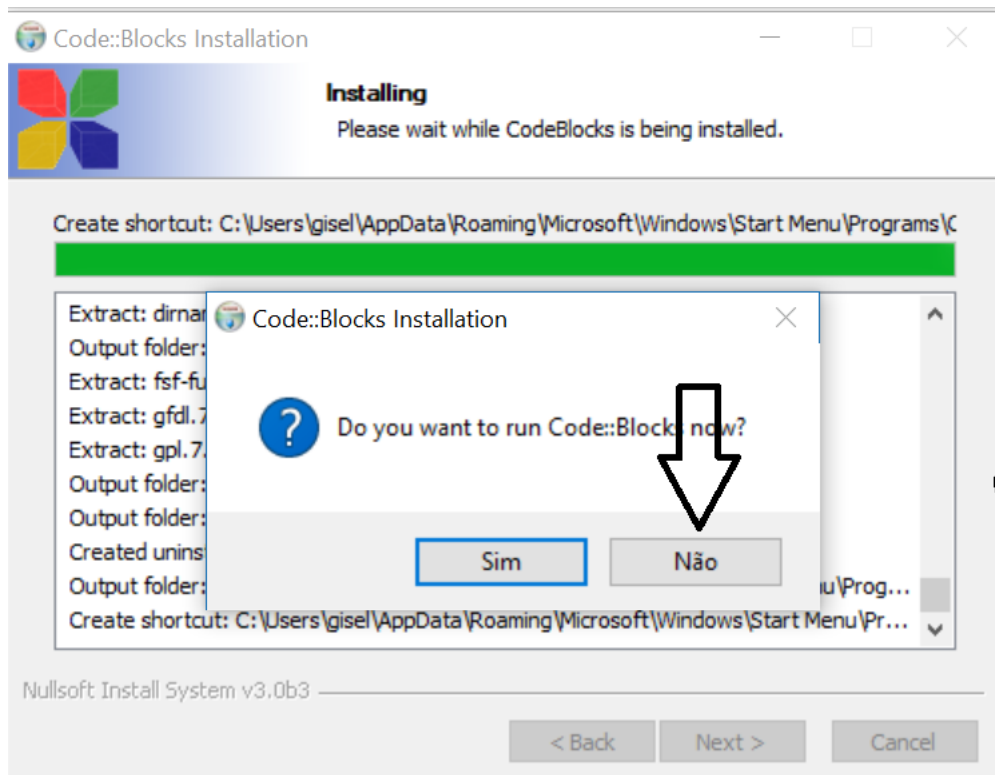
Em seguida serão apresentados quais componentes devem ser instalados, não modifique nenhuma opção e clique em “Next”, conforme ilustrado abaixo:



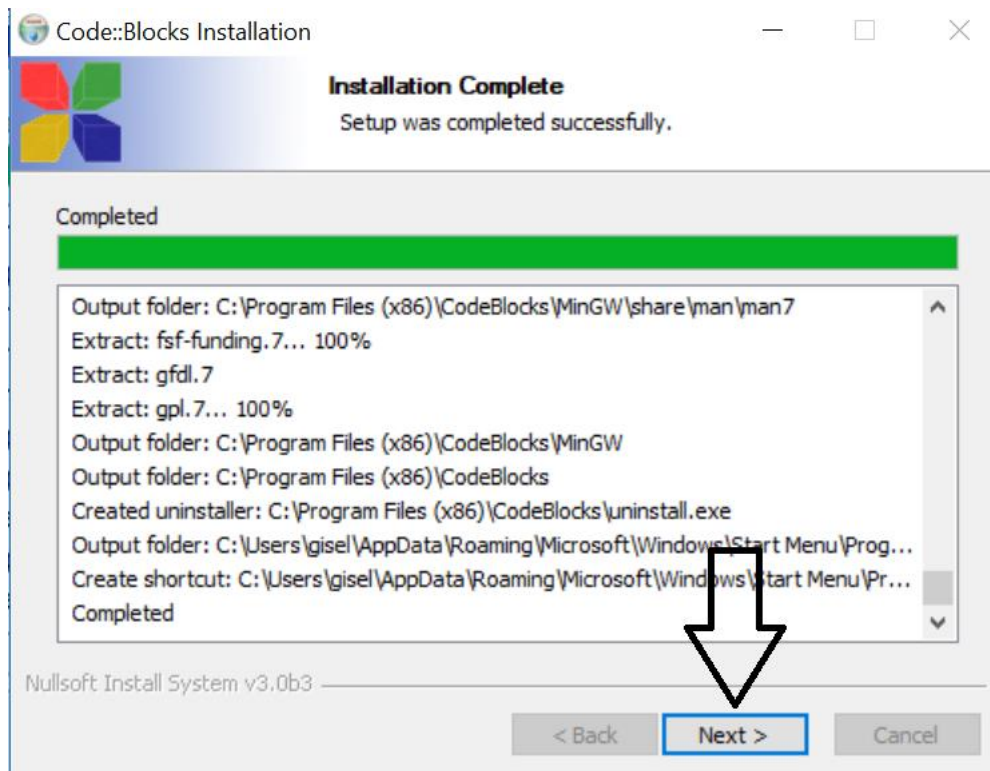
O próximo passo é iniciar a instalação, será solicitado o local onde deve ser instalado o Code::Blocks, não é necessário fazer nenhuma modificação, clique na opção “Install”, conforme abaixo:



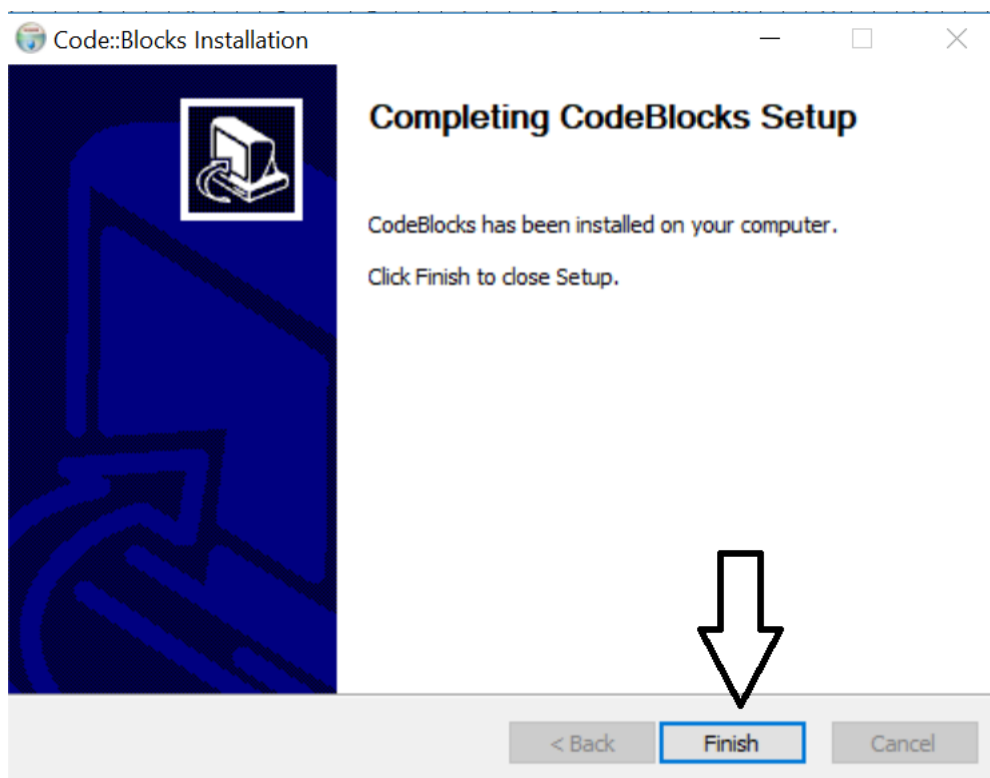
Após a instalação, irá aparecer uma mensagem se gostaria de iniciar o Code::Blocks, selecione “Não”, como aparece abaixo:



Na tela seguinte, clique em “Next”:



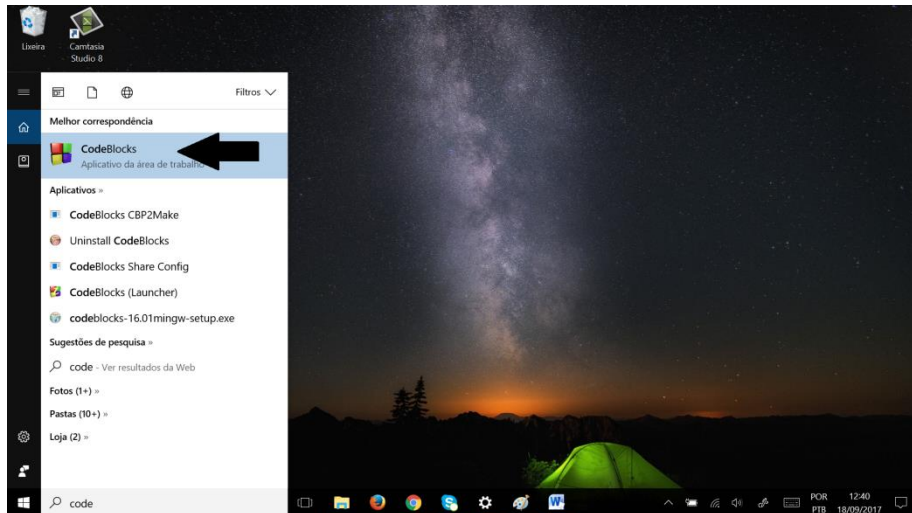
Na última tela clique em "Finish":



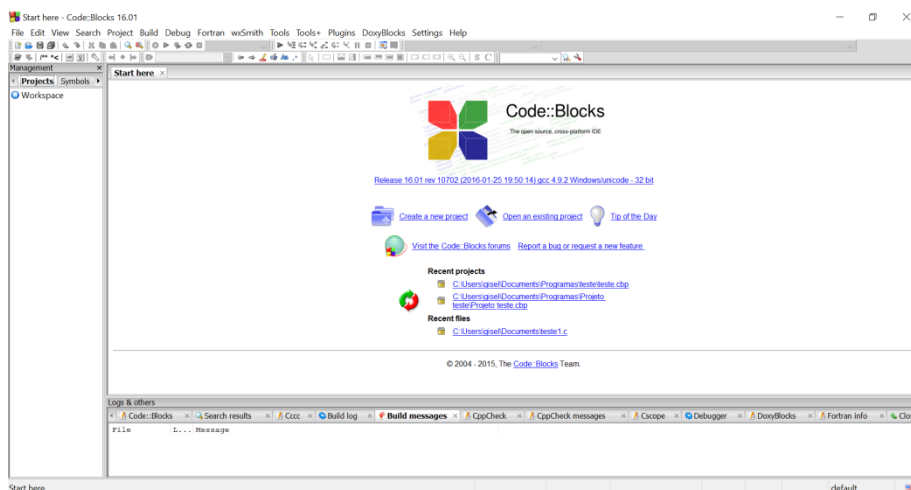
A instalação foi concluída com sucesso.

Utilizando o Code::Blocks

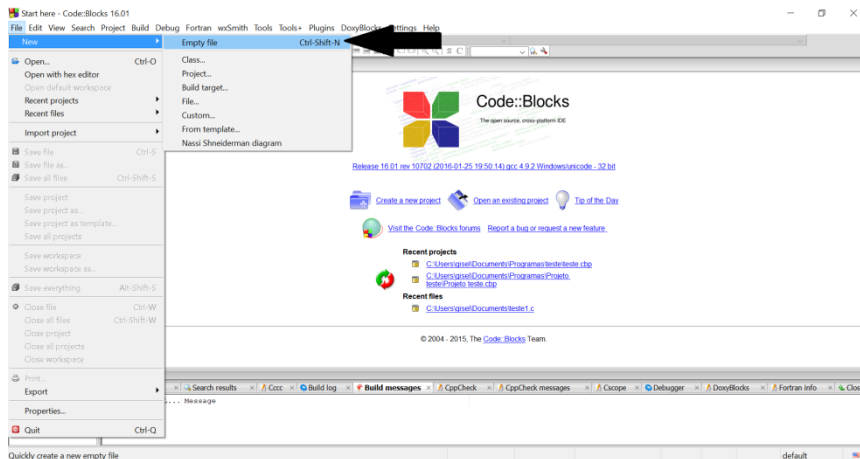
1. O primeiro passo é acessar o Code::Blocks, para isso vá à opção “Iniciar” do Windows e selecione o Aplicativo **Code::Blocks**, conforme ilustrado abaixo:



2. Será a aberto o Code::Blocks, conforme ilustrado na figura abaixo.



3. Em seguida, clique em File —> New —> Empty file, conforme ilustrado abaixo, ou CTRL+SHIFT+N.

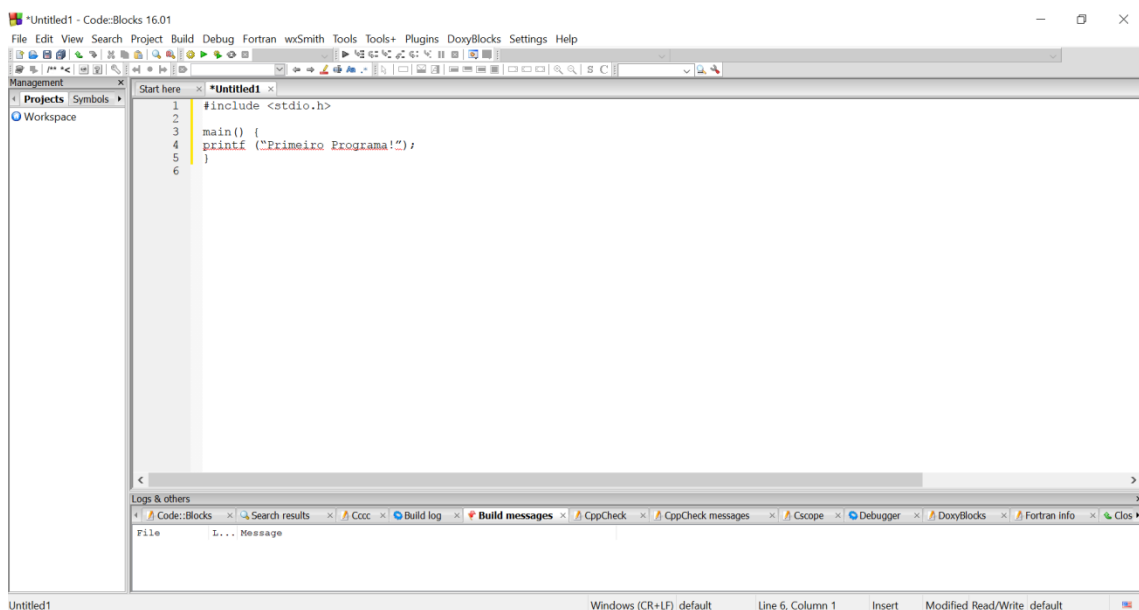


4. Na janela que foi criada você vai digitar, compilar e executar seu primeiro programa em C, seguindo os passos abaixo.

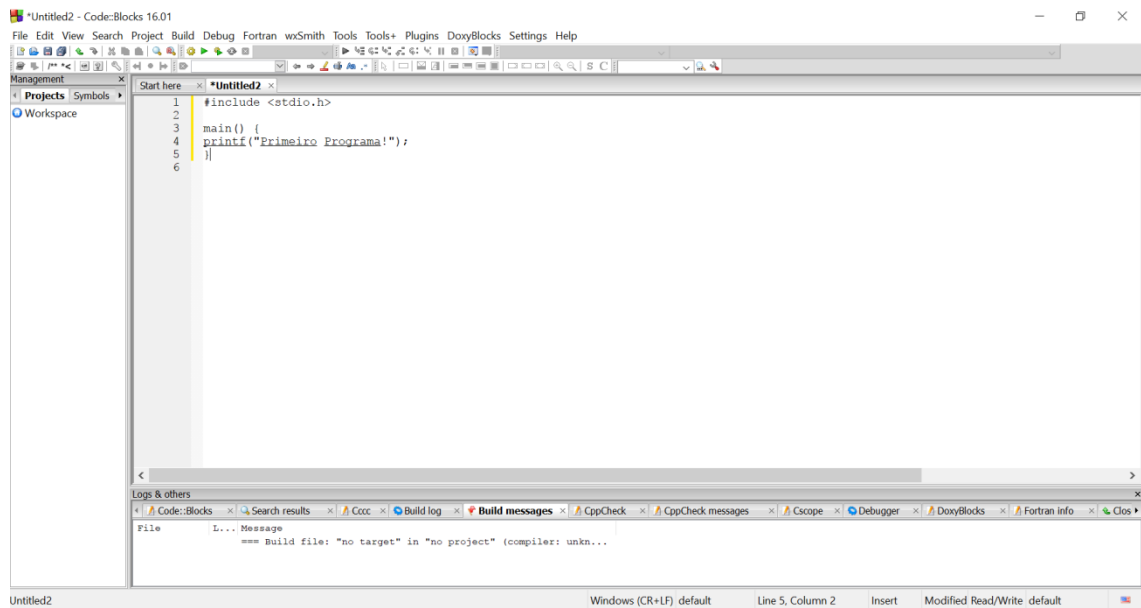
- Digite o programa a seguir na janela criada:

```
#include <stdio.h>

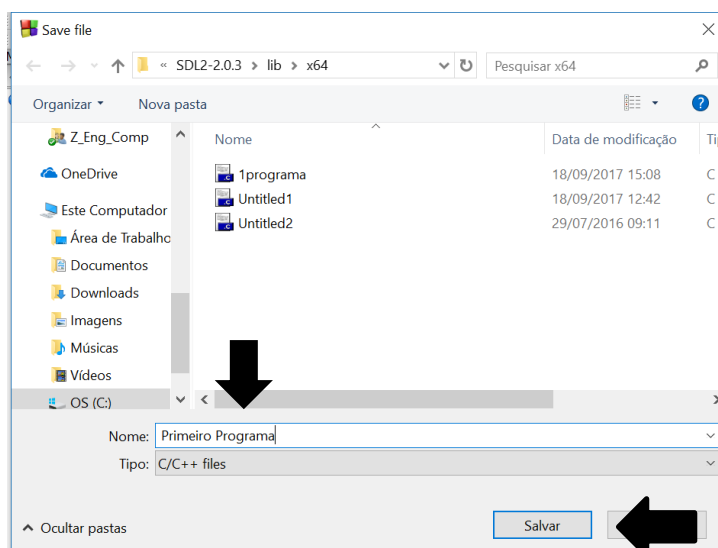
main() {
printf("Primeiro Programa!");
}
```



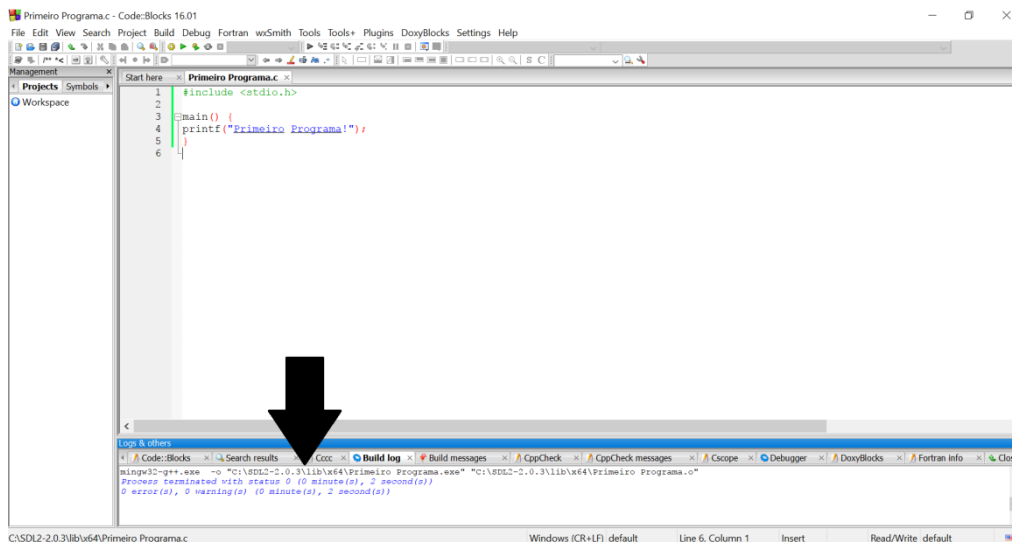
- Agora compile, clique em Build ———> Compile current file, ou CTRL+SHIFT+F9.



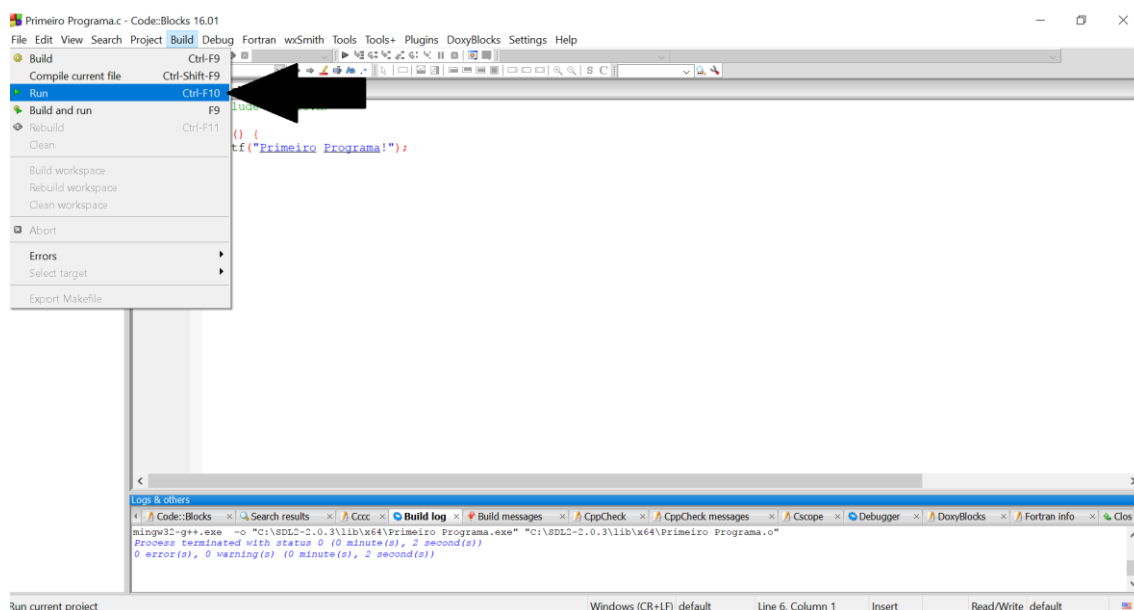
Irá aparecer a opção para Salvar o arquivo, coloque o nome do arquivo e salve-o, conforme abaixo.



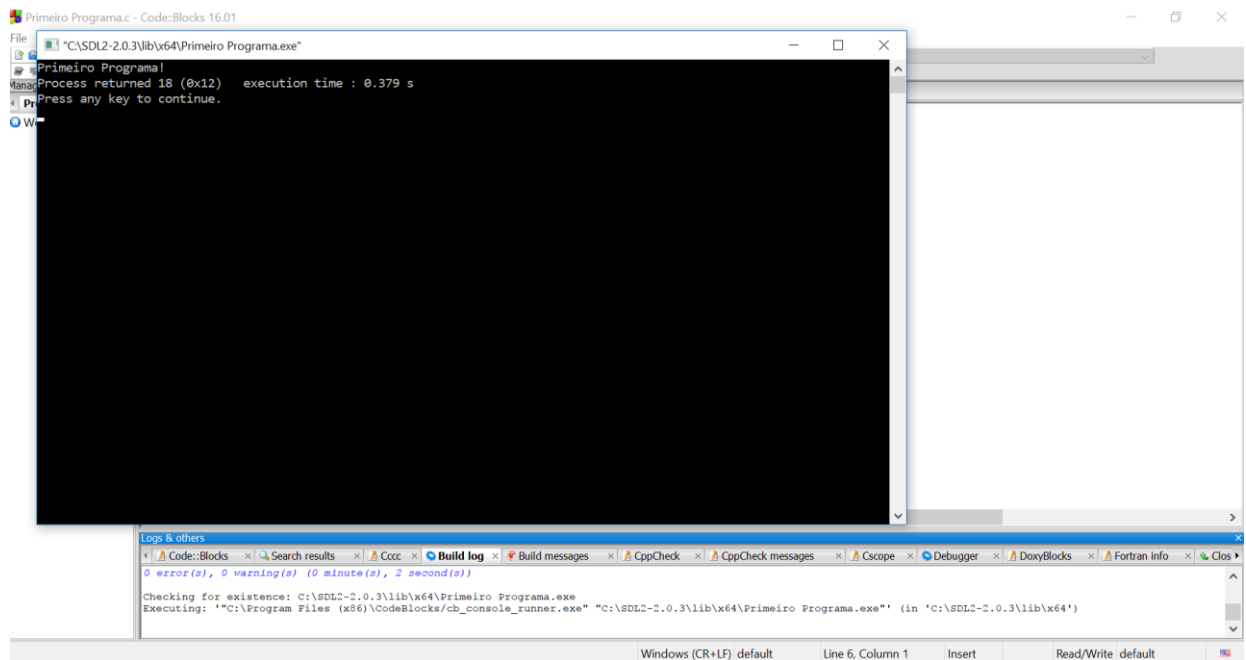
Se tudo der certo, na janela de **log**, abaixo do programa, haverá a indicação de que o programa foi compilado com sucesso: 0 errors, 0 warnings.



Com o programa compilado sem erros, a pasta onde foi salvo o arquivo contém agora o programa executável ".exe." Para executar o programa, clique no botão Build —> Run da janela principal do Code::Blocks.



Deverá surgir uma nova janela, de DOS, mostrando a execução de seu programa, como pode ser visto na figura abaixo.



Note que na janela do DOS aparece a mensagem *“Pressione qualquer tecla para continuar . . .”*. Depois de conferir o resultado, pressione qualquer tecla para que a janela do DOS desapareça. Se a janela do DOS não desaparecer, para fechá-la clique no X, no canto superior direito da mesma.

Anexo 2

Resolução dos Exercícios Propostos

A estrutura Básica do Programa

Exercícios Propostos

1. Faça um programa que mostre na tela os seguintes dados seus:

✓ Nome

✓ Curso

```
#include <stdio.h>
```

```
main(){
```

```
printf(" Nome: Giselle Cristina Cardoso \n Curso: Técnico em Informática");
```

```
}
```

2. Faça um programa capaz de ler um caracter e um número inteiro do teclado. Após a leitura, apresente-os na tela do computador.

```
#include <stdio.h>

main(){
char c;
int x;
printf("Digite um caracter:");
scanf("%c", &c);
printf("Digite um número inteiro:");
scanf ("%d", &x);
printf (" Caracter = %c \n Inteiro = %d", c, x);
}
```

Operadores

Exercícios Propostos

1. Faça um programa que solicite ao usuário dois números inteiros, ele deve calcular e mostrar a soma, subtração, multiplicação e divisão desses dois números.

```
#include <stdio.h>

main(){
int n1, n2, resultado;
printf("Digite um número inteiro:");
scanf("%d", &n1);
printf("Digite outro número inteiro:");
scanf ("%d", &n2);

//calcula da soma
resultado = n1+n2;
printf (" Soma = %d \n", resultado);

//calcula da subtração
resultado = n1-n2;
printf (" Subtração = %d \n", resultado);

//calcula da multiplicação
resultado = n1*n2;
printf (" Multiplicação = %d \n", resultado);

//calcula da divisão
```

```

resultado = n1/n2;
printf (" Divisão = %d \n", resultado);
}

```

2. Escreva um programa que solicite 3 números reais e imprima na tela a média aritmética.

```

#include <stdio.h>
main(){
float n1, n2, n3, media;
printf("Digite um número:");
scanf("%f", &n1);
printf("Digite um número:");
scanf ("%f", &n2);
printf("Digite um número:");
scanf ("%f", &n3);
//calculo da média
media = (n1+n2+n3)/3; //deve-se colocar os parênteses para que a soma seja realizada
//primeiro
printf (" Média = %f", media);
}

```

3. Faça um programa que calcule e mostre a área de um círculo. Sabe-se que: $\text{Area} = \pi \cdot (\text{Raio})^2$ e $\pi = 3,1415$.

```

#include <stdio.h>
main(){
float area, raio;
printf("Digite o raio do círculo:");
scanf("%f", &raio);
area = 3.1415 * raio*raio;    //Raio²= Raio*raio
printf (" Área = %f", area);
}

```

4. Escreva um programa no qual você digita o ano de nascimento e o ano atual e ele calcula e informa quantos anos você tem ou vai fazer neste ano.

```

#include <stdio.h>

```

```

main(){
int anonascimento, anoatual, idade;
printf("Digite o ano atual:");
scanf("%d", &anoatual);
printf("Digite o ano de nascimento:");
scanf("%d", &anonascimento);
idade = anoatual - anonascimento;
printf (" Idade = %d", idade);
}

```

5. Faça um programa que receba o preço de uma compra, calcule e mostre o preço final da compra, sabendo-se que esta sofreu um desconto de 15%.

```

#include <stdio.h>

main(){
float preco_compra, preco_final;
printf("Digite o valor da compra:");
scanf("%f", &preco_compra);
preco_final = preco_compra - preco_compra * 15/100;
printf (" Preço Final = %f", preco_final);
}

```

Operadores Relacionais

Exercícios Propostos

1. Sabendo que A = 5, B = 9 e C = 1, informe se as expressões abaixo são verdadeiras ou falsas.

	Solução
f) (A+B) > C	Verdadeiro (1), pois (5 + 9) > 1
g) B >= (A - 3)	Verdadeiro (1), pois 9 > (5 - 3)
h) C == (B - A)	Falso (0), pois 1 não é igual a (9 - 5)
i) (B + A) <= C	Falso (0), pois (9 + 5) não é menor que 1
j) (C+A) > B	Falso (0), pois (1 + 5) não é maior que 9

2. Sabendo que A = 6, B = 5 e C = 4 e D = 7, informe se as expressões abaixo são verdadeiras ou falsas.

	Solução
e) (A > C) && (C <= D)	Verdadeiro (1), pois (6 > 5) E (4 <=7)

f) $(A \geq C) \ \&\& \ (D \geq C)$	Verdadeiro (1), pois $(6 \geq 4)$ E $(7 \geq 4)$
g) $(A + B) > (C + D) \ \ (D \geq C)$	Verdadeiro (1), pois $(6 + 5)$ não é maior que $(4 + 7)$ OU $(7 \geq 4)$
h) $!((A+C) < D \ \ (A==B))$	Verdadeiro (1), pois $(6 + 4) < 7$ OU $(6 == 5)$ é falso, a negação (!) da falsidade é uma verdade.

Comandos Condicionais

Exercícios Propostos

1. Faça um programa em linguagem C em que o usuário digita 3 números inteiros e ele mostra na tela qual é o menor número.

```
#include <stdio.h>

main(){
    int n1, n2, n3;
    printf("Digite um número:");
    scanf ("%d",&n1);
    printf("Digite um número:");
    scanf ("%d",&n2);
    printf("Digite um número:");
    scanf ("%d",&n3);
    if ((n1<n2) && (n1<n3))
        printf("%d é o menor", n1);//n1 é o menor
    if ((n2<n1) && (n2<n3))
        printf("%d é o menor", n2);//n2 é o menor
    if ((n3<n1) && (n3<n2))
        printf("%d é o menor", n3);//n3 é o menor
}
```

1. Sabendo que uma escola realiza 4 provas bimestrais durante seu ano letivo, faça um programa que leia as 4 notas (N1,N2,N3, N4), calcule a média das notas e indique a situação do aluno, conforme abaixo:

Nota	Situação
Média ≥ 6.0	Aprovado
Média < 6.0 e Média ≥ 4.0	Recuperação
Média < 4.0	Reprovado

```

#include <stdio.h>

main(){
float n1, n2, n3, n4, media;
printf("Digite a nota 1:");
scanf ("%f",&n1);
printf("Digite a nota 2:");
scanf ("%f",&n2);
printf("Digite a nota 3:");
scanf ("%f",&n3);
printf("Digite a nota 4:");
scanf ("%f",&n4);
//cálculo da média
media = (n1+n2+n3+n4)/4;
if (media>=6)
    printf("Aluno Aprovado");
if ((media<6) && (media >=4))
    printf("Aluno está de recuperação");
if (media < 4)
    printf("Aluno Reprovado");
}

```

2. Desenvolva um programa para calcular a conta de seu celular. Você deve digitar quantos minutos foram utilizados e calcular conforme as regras a seguir:
- Até 100 minutos, incluindo a internet, total da conta R\$39,99;
 - Até 200 minutos, incluindo a internet, total da conta R\$59,99;
 - Até 500 minutos, incluindo a internet, total da conta R\$99,99;
 - Acima de 500 minutos, incluindo a internet, paga-se R\$99,99 mais R\$1,00 cada minuto excedente.

Não se esqueça de apresentar na tela o total da conta.

Solução 1:

```

#include <stdio.h>

main(){
float minutos, total_da_conta;
printf("Digite os minutos utilizados:");
scanf ("%f",&minutos);
if (minutos <=100)

```

```

    total_da_conta = 39.99;
if ((minutos > 100) && (minutos <= 200))
    total_da_conta = 59.99;
if ((minutos > 200) && (minutos <= 500))
    total_da_conta = 99.99;
if (minutos > 500)
    total_da_conta = 99.99 + (minutos-500);
printf("Total da conta = R$ %f",total_da_conta);
}

```

Solução 2:

```

#include <stdio.h>

main(){
    float minutos, total_da_conta;
    printf("Digite os minutos utilizados:");
    scanf("%f",&minutos);
    if (minutos <=100)
        total_da_conta = 39.99;
    else // se minutos for maior que 100
        if (minutos <= 200)
            total_da_conta = 59.99;
        else // se minutos for maior que 200
            if (minutos <= 500)
                total_da_conta = 99.99;
            else // se minutos for maior que 500
                total_da_conta = 99.99 + (minutos-500);
    printf("Total da conta = R$ %f",total_da_conta);
}

```

3. Após a execução deste algoritmo qual será a saída?

```
#include <stdio.h>
```

main(){	Teste de Mesa				
int x,y,z,aux;	x	y	z	aux	Observação
x = 8;	8				
y = 6;		6			
z = 1;			1		
aux = 0;				0	
if((x>y) (x>z)){					(x>y) (x>z) é verdade, executa
if (z>y){					(z>y)é falsidade, executa o "else"
aux = x;					

x = y;					
y = aux;					
}					
else{					Executa os comandos
aux = x;				8	
x = z;	1				
z = aux;			8		
}					
}					
if (y>z){					(y>z) é falso, não executa
aux = y;					
y = z;					
z = aux;					
}					
printf("x = %d y = %d z = %d",x,y,z);					
}					
Saída na tela: x = 1 y = 6 z = 8					

Observação: Refaça este exercício com outros valores.

Comandos de Repetição

Exercícios Propostos

1. Faça um programa para calcular a soma de 10 números inteiros digitados pelo usuário e apresentá-los na tela.

```
#include <stdio.h>

main(){
    int num, contador, soma = 0;
    for (contador=0; contador <10; contador++){
        printf ("Digite um número inteiro: ");
        scanf ("%d", &num);
        soma = soma + num;
    }
    printf ("Soma = %d", soma);
}
```

2. Faça um programa em C em que o usuário digita a quantidade de alunos. Depois ele deve ler a nota de cada um dos alunos, calcular e mostrar a média aritmética das notas de todos os alunos. Se algum aluno estiver com a nota abaixo de 6.0, mostrar a mensagem na tela: "Nota baixa".

```
#include <stdio.h>
```

```

main(){
float nota, n, contador, soma = 0, media;
printf("Digite a quantidade de alunos:");
scanf ("%f",&n);
for (contador=0; contador <n; contador ++){
    printf ("Digite a nota: ");
    scanf ("%f", &nota);
    soma = soma + nota;
    if (nota < 6.0)
        printf ("Nota Baixa\n");
    }
media = soma/n;
printf ("Média = %f", media);
}

```

3. Faça um programa que leia 10 números inteiros e que determine e mostre qual o maior número lido.

```

#include <stdio.h>
main(){
int n, contador, maior;
for (contador=0; contador <10; contador ++){
    printf ("Digite um número: ");
    scanf ("%d", &n);
    if (contador == 0) //atribuir o primeiro valor a variável maior, determinando que o
                        //primeiro valor é o maior

        maior = n;
    else // demais valores devem ser comparados com a variável maior, para se achar o
        //maior valor digitado
        if (n > maior)
            maior = n;
    }
printf ("Maior = %d", maior);
}

```

4. Faça um programa em que o usuário digite um número e ele apresente na tela a tabuada do 0 ao 10 do número digitado, conforme abaixo:

Número digitado: 5

0 x 5 = 0

1 x 5 = 5

2 x 5 = 10

3 x 5 = 15

4 x 5 = 20

5 x 5 = 25

6 x 5 = 30

7 x 5 = 35

8 x 5 = 40

9 x 5 = 45

10 x 5 = 50

```
#include <stdio.h>

main(){
    int n,i, resultado;
    printf("Digite o número:");
    scanf("%d",&n);
    for (i=0; i<=10; i++){
        resultado = i*n;
        printf("\n%d x %d = %d",i, n, resultado);
    }
}
```

5. Durante uma corrida de bicicleta com 10 voltas de duração foram anotados para um ciclista, na ordem, os tempos registrados em cada volta. Fazer um programa em C para ler os tempos das 10 voltas, calcular e imprimir:

- ✓ melhor tempo e a volta em que ocorreu;
- ✓ o tempo total de todas as voltas.
- ✓ tempo médio das N voltas.

```
#include <stdio.h>

main(){
    float i,volta, soma=0,tempo, melhortempo;
    for (i=1;i<=10;i++){
        printf("Digite o tempo da volta:");
        scanf("%f", &tempo);
        if (i==1){ //inicializa a variável melhor tempo na primeira volta
            melhortempo=tempo;
            volta=i;
        }
    }
}
```

```

else
    if(tempo<melhortempo){
        melhortempo=tempo;
        volta=i;
    }
soma = soma + tempo;
}
printf("Melhor tempo: %f Volta: %f\n", melhortempo,volta);
printf("Tempo total: %f\n", soma);
printf("Tempo médio: %f", soma/10);
}

```

Comando WHILE

Exercícios Propostos

1. Escreva um programa que leia várias notas e calcule a média, sabendo que ele deve mostrar a média quando a nota lida for menor que 0.

```

#include <stdio.h>
main(){
    float nota, contador=0, soma=0,media;
    while(nota>=0){
        printf("Digite a nota ou um número negativo para parar:");
        scanf("%f",&nota);
        if (nota>=0){
            soma = soma + nota;
            contador++;
        }
    }
    media = soma/contador;
    printf("Média = %f", media);
}

```

2. Desenvolva um programa que faça a captura de N números reais. O programa deverá calcular o somatório e a média desses números. Pergunte o valor de N ao usuário.

```

#include <stdio.h>
main(){
    float n, contador=1,numero, soma=0,media;

```

```

printf("Digite quantos números serão lidos:");
scanf ("%f",&n);
while(contador <=n){
    printf("Digite um número:");
    scanf("%f",&numero);
    soma = soma + numero;
    contador++;
}
media = soma/n;
printf("Soma = %f\n", soma);
printf("Média = %f", media);
}

```

3. Desenvolva um programa que faça a captura de 10 números reais e calcule a média. Na sequência, o programa deverá solicitar a inserção de mais 10 números reais. Para cada um desses números digitados, o programa deverá informar se está abaixo ou acima da média dos 10 primeiros números digitados.

```

#include <stdio.h>
main(){
    float numero, contador=0, soma=0,media;
    while(contador<10){
        printf("Digite um número:");
        scanf("%f",&numero);
        soma = soma + numero;
        contador++;
    }
    media = soma/10;
    printf("Média = %f\n", media);
    contador = 0;
    while(contador<10){
        printf("Digite um número:");
        scanf("%f",&numero);
        if (numero < media)
            printf ("Numero abaixo da média!\n\n");
        else
            printf ("Número acima da média!\n\n");
        contador++;
    }
}

```



```
}  
}
```

4. Faça um programa que receba a idade de várias pessoas e que calcule e mostre a quantidade de pessoas com idade maior ou igual a 30 anos. O programa deve parar de ler as idades quando ler uma idade igual a 0.

```
#include <stdio.h>  
main(){  
    int idade = 1, contador=0;  
    while(idade!=0){  
        printf("Digite a idade:");  
        scanf("%d",&idade);  
        if (idade >= 30)  
            contador++;  
    }  
    printf("Número de idades superiores ou igual a 30 anos: %d\n", contador);  
}
```

Vetores

Exercícios Propostos

1. Faça um programa que, tendo como dados de entrada do teclado 10 valores reais que deverão ser armazenados em um vetor, calcule a média e mostre quantos elementos são superiores à média calculada.

```
#include <stdio.h>  
main(){  
    float vetor[10],soma=0,media, contador=0;  
    int i;  
    for (i=0;i<10;i++){  
        printf("Digite um valor:");  
        scanf("%f", &vetor[i]);  
        soma = soma + vetor[i];  
    }  
    media = soma/10;  
    printf("Média = %f\n", media);  
}
```

```

    for (i=0;i<10;i++){
        if(vetor[i] > media)
            contador++;
    }
    printf("Números superiores a média: %f", contador);
}

```

2. Faça um programa que, tendo como dados de entrada do teclado 10 valores inteiros, mostre a posição em que foi armazenado o maior valor.

```

#include <stdio.h>

main(){
    int vetor[10], i, maior, posicaomaior;
    for (i=0;i<10;i++){
        printf("Digite um valor:");
        scanf("%d", &vetor[i]);
    }
    maior=vetor[0];
    posicaomaior=0;
    for (i=1;i<10;i++){
        if(vetor[i] > maior){
            maior = vetor[i];
            posicaomaior=i;
        }
    }
    printf("Maior número: %d – Posição: %d", maior, posicaomaior);
}

```

3. Faça um programa que armazene 10 números inteiros digitados pelo usuário e como saída ele deve apresentar o vetor ao contrário.

```

#include <stdio.h>

main(){
    int vetor[10], i;
    for (i=0;i<10;i++){
        printf("Digite um valor:");
        scanf("%d", &vetor[i]);
    }

    for (i=9;i>=0;i--)

```

```
    printf(" %d", vetor[i]);
}
```

4. Uma marca possui 10 produtos, cada um com seu código (0 a 9) e o seu custo. Ao longo da semana, o sistema acumula o valor gasto na produção de cada produto. Utilize um vetor, para armazenar essas informações. O programa, ao finalizar, apresenta o valor gasto por cada produto. Para que o programa seja finalizado, entra-se no sistema com um código do produto igual a -1. Você está convidado a desenvolver um programa para resolver esse problema.

```
#include <stdio.h>

main(){
    float valor, vetor[10]={0,0,0,0,0,0,0,0,0,0};
    //inicializando o vetor com todas as posições igual a 0
    int i, codigo=0;
    while(codigo!=-1){
        printf("Digite o codigo do produto (0-9) ou -1 para parar:");
        scanf("%d", &codigo);
        if(codigo!=-1){
            printf ("Digite o valor gasto:");
            scanf("%f",&valor);
            vetor[codigo]=vetor[codigo]+valor;
        }
    }
    for (i=0;i<10;i++)
        printf("Produto  %d = R$%f\n", i,vetor[i]);
}
```

