

# Algoritmos Computacionais e Estruturas de Dados

Prof. Me. Felipe Borges

# Prof. Felipe Borges

Doutorando em Sistemas de Potência – UFMA – Brasil

Mestre em Sistemas de Potência – UFMA – Brasil

MBA em Qualidade e Produtividade – FAENE – Brasil

Graduado em Engenharia Elétrica – IFMA – Brasil

Graduado em Engenharia Elétrica – Fontys – Holanda

Técnico em Eletrotécnica – IFMA – Brasil

Projetos e Instalações Elétricas – Engenharia – Banco do Brasil

Desenvolvimento e Gestão de Projetos – Frencken Engineering BV

# IDE - Ambiente de Desenvolvimento Integrado

IDE (do inglês *Integrated Development Environment*) ou **Ambiente de Desenvolvimento Integrado**, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

Exemplos de IDE que Geram código para C e C++:

- DEV-C++
- Code::Blocks
- Turbo C

# IDE - Ambiente de Desenvolvimento Integrado

As características e ferramentas mais comuns encontradas nos IDEs são:

- Editor: edita o código-fonte do programa escrito nas linguagens suportadas pela IDE;
- Compilador: compila o código-fonte do programa, editado em uma linguagem específica e a transforma em linguagem de máquina;
- Depurador (*debugger*): auxilia no processo de encontrar e corrigir defeitos no código-fonte do programa, na tentativa de aprimorar a qualidade de software;

# Linguagens de programação

Linguagens de programação são conhecimentos escritos e formais que seguem um conjunto de instruções e regras para o desenvolvimento de softwares.

# Linguagens de programação

Um programa de computador é um conjunto de instruções que representam um algoritmo para a resolução de algum problema. Estas instruções são escritas através de um conjunto de códigos (símbolos e palavras).

Este conjunto de códigos possui regras de estruturação lógica e sintática própria. Diz-se que este conjunto de símbolos e regras formam uma linguagem de programação.

# Tradução



- MONTADOR (*assembler*)
  - Tradutor para linguagens.
- COMPILADOR:
  - Traduz todo o programa de uma vez.
- INTERPRETADOR:
  - Traduz o programa instrução por instrução.

# Estrutura básica de um programa C

diretivas para o pré-processador

declaração de variáveis globais

main () {

    declaração de variáveis locais da função main

    comandos da função main

}



# Diretivas para o processador - Bibliotecas

- Diretiva `#include` permite incluir uma biblioteca
- Bibliotecas contêm funções pré-definidas, utilizadas nos programas
- Exemplos

→	<code>#include &lt;stdio.h&gt;</code>	Funções de entrada e saída
→	<code>#include &lt;stdlib.h&gt;</code>	Funções padrão
	<code>#include &lt;math.h&gt;</code>	Funções matemáticas
	<code>#include &lt;system.h&gt;</code>	Funções do sistema
	<code>#include &lt;string.h&gt;</code>	Funções de texto

# O ambiente Dev-C++

- O Dev-C++ é um ambiente de desenvolvimento de programas em C e C++ com editor, compilador, bibliotecas e debugger
- Pode ser baixado de <https://sourceforge.net/projects/orwellddevcpp/>
- Vamos criar apenas programas na linguagem **C**

# Usando o Dev-C++

- Inicie o Dev-C++ pelo ícone ou pelo menu
- Crie um novo arquivo, com o comando *File, New Source File*
- Edite o programa da página seguinte

# Usando o Dev-C++

```
#include <stdio.h>

main() {

    printf ("Alo mundo!");

    system("pause");

}
```

# Usando o Dev-C++

- Salve o programa com o nome **exemplo.c**. Para tanto, selecione o menu File, Save unit as
- Compile o programa com o comando **Executar, Compilar** ou com a tecla **F9**
- Se houver algum erro de sintaxe, aparece uma ou mais mensagens no rodapé da janela. Neste caso, corrija o programa e repita.
- Se não houver erros, execute o programa com o comando **Executar, Executar** ou com a tecla **F10**

# Dicas

- **Termine todas as linhas com ;**
- Sempre salve o programa antes de compilar
- **Sempre compile o programa antes de executar**
- Quando ocorrer um erro de compilação, dê um duplo clique sobre a mensagem de erro para destacar o comando errado no programa
- Verifique também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o ;
- **Use comentários, iniciados por //**

# Template

```
#include <stdio.h>
main()
{
    printf ("Alo mundo!");
    system("pause");
}
```

# Declarações

- Declaram as variáveis e seus tipos
- Os nomes das variáveis devem conter apenas letras, dígitos e o símbolo \_
- Até 32 caracteres
- Os principais tipos são: **int**, **float**, e **char**
- Exemplos

```
int n;  
int quantidade_valores;  
float x, y, somaValores;  
char sexo;  
char nome[40];
```

**Linguagem C diferencia letras maiúsculas de minúsculas!**

```
int n, N;
```

n é diferente de N!



# Exemplo

Real: n1, n2, n3, media

```
#include <stdio.h>
main()
{
    real n1, n2, n3, media;

    system("pause");
}
```

# Comando de atribuição

- Atribui o valor da direita à variável da esquerda
- O valor pode ser uma constante, uma variável ou uma expressão
- Exemplos

**x = 4;**    --> **lemos: x recebe 4**

y = x + 2;

y = y + 4;

valor = 2.5;

# Entrada e Saída

- **Função scanf**

```
scanf ("formatos", &var1, &var2, ...)
```

Exemplos:

```
int i, j;  
float x;  
char c;  
char nome[10];  
scanf("%d", &i);  
scanf("%d %f", &j, &x);  
scanf("%c", &c);  
scanf("%s", nome);
```

<b>%d</b>	<b>inteiro</b>
<b>%f</b>	<b>float</b>
<b>%lf</b>	<b>double</b>
<b>%c</b>	<b>char</b>
<b>%s</b>	<b>palavra</b>

# Exemplo

Real: n1, n2, n3, media

ler n1, n2, n3

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

```
main()
```

```
{
```

```
    float n1, n2, n3, media;
```

```
    scanf ("%f %f %f",&n1, &n2, &n3);
```

```
    system("pause");
```

```
}
```

# Operadores Matemáticos

Operador	Exemplo	Comentário
+	$x + y$	Soma x e y
-	$x - y$	Subtrai y de x
*	$x * y$	Multiplica x e y
/	$x / y$	Divide x por y
% Ou <b>mod</b> no português	$x \% y$	Resto da divisão de x por y
++	$x++$	Incrementa em 1 o valor de x
--	$x--$	Decrementa em 1 o valor de x

# Exemplo

Real: n1, n2, n3, media

ler n1, n2, n3

$media = (n1 + n2 + n3) / 3$

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

```
main()
```

```
{
```

```
    float n1, n2, n3, media;
```

```
    scanf ("%f %f %f",&n1, &n2, &n3);
```

```
    media=(n1+n2+n3)/3;
```

```
    system("pause");
```

```
}
```

# Entrada e Saída

- Função **printf**

```
printf ("formatos", var1, var2, ...)
```

Exemplos:

```
int i, j;  
float x;  
char c;  
char nome[10];  
printf("%d", i);  
printf("%d, %f", j, x);  
printf("%c", c);  
printf("%s", nome);
```

%d	inteiro
%f	float
%lf	double
%c	char
%s	palavra

# Exemplo

Real: n1, n2, n3, media

ler n1, n2, n3

$media = (n1 + n2 + n3) / 3$

exibir media

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

```
main()
```

```
{
```

```
    real n1, n2, n3, media;
```

```
    scanf ("%lf %lf %lf",&n1, &n2, &n3);
```

```
    media=(n1+n2+n3)/3;
```

```
    printf ("%f",media);
```

```
    system("pause");
```

```
}
```



# Exemplo

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

```
main()
```

```
{
```

```
    real n1, n2, n3,
```

```
    scanf ("%f %f %f",  
           &n1, &n2, &n3);
```

```
    media=(n1+n2+n3)/3;
```

```
    printf ("%f",media);
```

```
    system("PAUSE");
```

```
}
```

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

```
main()
```

```
{
```

```
    float n1, n2, n3, media;
```

```
    printf("Digite 3 notas: ");
```

```
    scanf ("%f %f %f",&n1, &n2, &n3);
```

```
    media=(n1+n2+n3)/3;
```

```
    printf ("A média é %0.2f",media);
```

```
}
```

# Exercício

1) Tendo como dados de entrada a altura de uma pessoa, construa um programa que calcule (e mostre) seu peso ideal, utilizando a seguinte fórmula:

- $\text{peso ideal} = (72.7 * \text{altura}) - 58$

# Funções Matemáticas

Função	Exemplo	Comentário
<code>ceil</code>	<code>ceil(x)</code>	Arredonda o número real para cima; <code>ceil(3.2)</code> é 4
<code>cos</code>	<code>cos(x)</code>	Cosseno de x (x em radianos)
<code>exp</code>	<code>exp(x)</code>	e elevado à potencia x
<code>fabs</code>	<code>fabs(x)</code>	Valor absoluto de x
<code>floor</code>	<code>floor(x)</code>	Arredonda o número para baixo; <code>floor(3.2)</code> é 3
<code>log</code>	<code>log(x)</code>	Logaritmo natural de x
<code>log10</code>	<code>log10(x)</code>	Logaritmo decimal de x
<code>pow</code>	<code>pow(x, y)</code>	Calcula x elevado à potência y
<code>sin</code>	<code>sin(x)</code>	Seno de x
<code>sqrt</code>	<code>sqrt(x)</code>	Raiz quadrada de x
<code>tan</code>	<code>tan(x)</code>	Tangente de x

```
#include <math.h>
```

# Operadores Relacionais

Operador	Exemplo	Comentário
==	$x == y$	O conteúdo de x é igual ao de y
!=	$x != y$	O conteúdo de x é diferente do de y
<=	$x <= y$	O conteúdo de x é menor ou igual ao de y
>=	$x >= y$	O conteúdo de x é maior ou igual ao de y
<	$x < y$	O conteúdo de x é menor que o de y
>	$x > y$	O conteúdo de x é maior que o de y

# Operadores Lógicos

- **&& (E lógico)**: retorna verdadeiro se ambos os operandos são verdadeiros e falso nos demais casos.  
Exemplo: `if( a>2 && b<3 )`.
- **|| (OU lógico)**: retorna verdadeiro se um ou ambos os operandos são verdadeiros e falso se ambos são falsos.  
Exemplo: `if( a>1 || b<2 )`.
- **! (NÃO lógico)**: usada com apenas um operando. Retorna verdadeiro se o operando é falso e vice-versa.  
Exemplo: `if( !var )`.

# Operadores Lógicos

Tabela E	Tabela OU	Tabela NÃO
$V \text{ e } V \rightarrow V$	$V \text{ ou } V \rightarrow V$	$\text{N\~ao } V \rightarrow F$
$V \text{ e } F \rightarrow F$	$V \text{ ou } F \rightarrow V$	$\text{N\~ao } F \rightarrow V$
$F \text{ e } V \rightarrow F$	$F \text{ ou } V \rightarrow V$	
$F \text{ e } F \rightarrow F$	$F \text{ ou } F \rightarrow F$	

# Estrutura condicional simples

- Comando **if**

```
if (condição)  
    comando;
```

```
if (condição) {  
    comando1;  
    comando2;  
    comando3;  
}
```

```
if (a<menor)  
    menor=a;
```

```
if (a<menor) {  
    menor=a;  
    printf ("%d", menor);  
}
```

em pseudo-código:

se (a<menor) então menor=a;

# Estrutura condicional composta

- Comando **if...else**

```
if (condição)  
    comando;  
else  
    comando;
```

```
if (condição) {  
    comando1;  
    comando2;  
} else {  
    comando3;  
    comando4;  
}
```

```
if (peso==peso_ideal){  
    printf ("Vc esta em forma!");  
}  
else{  
    printf ("Necessario fazer dieta!");  
}
```

em pseudo-código:

```
se (peso=peso_ideal) entao  
    exibir "Vc está em forma!"  
senao  
    exibir "Necessário fazer dieta!"  
fimse
```



# Exemplo

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

```
main()
```

```
{
```

```
    real n1, n2, n3,  
    scanf ("%f %f %f",  
            &n1, &n2, &n3);
```

```
    media=(n1+n2+n3)/3;  
    printf ("%f",media);
```

```
    system("PAUSE");
```

```
}
```

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

```
main()
```

```
{
```

```
    float n1, n2, n3, media;  
    printf("Digite 3 notas: ");  
    scanf ("%f %f %f",&n1, &n2, &n3);  
    media=(n1+n2+n3)/3;  
    printf ("A média é %0.2f",media);
```

```
    if(media>=7){  
        printf("Voce esta aprovado,  
        com a media %f",media);  
    }  
    else{  
        printf("Voce esta reprovado");  
    }  
}
```

# Exercício

- 2) Faça um programa que leia um número inteiro e mostre uma mensagem indicando se este número é positivo ou negativo.
  
- 3) Explique porque está errado fazer `if (num=10) ...` O que irá acontecer?

# Exercício

2)

```
1  #include<stdio.h>
2  main(){
3      int num;
4      printf("Digite um \t numero \n");
5      scanf("%d",&num);
6
7      if(num>=0){
8          printf("O numero eh positivo");
9      }
10     else{
11         printf("O numero eh negativo");
12     }
13 }
```

# Estrutura de repetição

- Comando **for**

```
for (var=valor inicial; condição; incremento)
    comando;
```

```
for (var=valor inicial; condição; incremento)
{
    comando1;
    comando2;
    comando3;
}
```

## Exemplo:

```
for (cont=3; cont<=11; cont++) {  
    printf ("%d",cont);  
}
```

## Pseudo-código:

**Para CONT = 3 até 11 repetir**  
**Mostrar CONT**

# Exercícios

4) Faça um programa que escreva de 50 a 100.

# Exercícios

4) Faça um programa que escreva de 50 a 100.

```
1 #include<stdio.h>
2 main(){
3     int cont;
4     for(cont=50; cont<=100; cont++){
5         printf("%d", cont);
6     }
7 }
```

# Estrutura de repetição

- Comando **while**

```
while (condição)
    comando;
```

```
while (condição) {
    comando1;
    comando2;
    comando3;
}
```

Exemplo:

```
while (N != 0) {
    scanf ("%d",&N);
}
```

Pseudo-código:

MAIOR = 0

N = 1

**Enquanto** (N <> 0) **repetir**

Ler N

# Exercício

5) Faça um programa que conte de 1 a 10 usando o comando **while**.



# Exercício

5) Faça um programa que conte de 1 a 10 usando o comando **while**.

# Exercício

5) Faça um programa que conte de 1 a 10 usando o comando **while**.

```
1  #include<stdio.h>
2  main(){
3      int cont;
4      cont=1;
5      while(cont<=10){
6          printf("%d, ", cont);
7          cont=cont+1;
8          //cont++;
9      }
10 }
```

# Estrutura de repetição

- Comando **do...while**

```
do {  
    comando  
} while (condição);
```

```
do {  
    comando1;  
    comando2  
    comando3;  
} while (condição);
```

Exemplo:

**cont=0;**

```
do {  
    cont = cont + 1;  
    printf("%d \n",cont);  
} while (cont < 10);
```

Em pseudo-código:

CONTADOR = 0

**Repetir**

CONTADOR = CONTADOR + 1

exibir CONTADOR

**enquanto** CONTADOR < 10

# Exercício

6) Faça um programa que conte de 1 a 10 usando o comando **do...while**.

# Vetores (array)

- Trata-se de automatizar a declaração de um grande número de dados de um mesmo tipo simples. As variáveis assim declaradas se acessam através de um índice de tipo int.
- Declaração:
  - `int v[100];`
  - primeira posição =0;
  - última posição=99;
- Atribuição:
  - `v [9] = 87;`
- Acessar um valor:
  - `a = v[9];`

# Vetores (array)

- `int v[10];`

0	
1	
2	
3	
4	
5	<b>10</b>
6	
7	
8	
9	

`V[5]=10;`

`printf ("%d",V[5]);`

# Strings

- Não existe um tipo String em C.
- Strings em C são uma array do tipo char que termina com '\0'.
- Para literais String, o próprio compilador coloca '\0'.

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

```
main(){
```

```
    char re[] = "lagarto";
```

```
    printf ("%s", re);
```

```
    system("pause");
```

```
}
```

# Para ler uma String

- Comando gets

```
#include <stdio.h>
main(){
    char re [80];
    printf ("Digite o seu nome: ");
    gets(re);
    printf ("Oi %s\n", re);
    system("pause");
}
```



# Funções

- São estruturas que permitem ao programador separar o código do seu programa em blocos.
- Uma função tem a seguinte forma :

```
tipo_de_retorno Nome_da_funcao (parâmetros) {  
    /*corpo da função */  
}
```

# Funções que não retornam valor :

```
#include <stdio.h>


void ehPar (int x){
    if (x % 2) {
        printf ("O numero nao eh par!\n");
    }
    else {
        printf ("O numero eh par!\n");
    }
}

int main(){
    ehPar (3);
    system("pause");
}
```

# Funções que retornam valor

```
#include <stdio.h>
#include <stdlib.h>
int ehPar (int x){
// o operador % retorna o resultado da divisão por 2
    if ((x % 2)==0) {
        return 0;
    }
    else{
        return 1;
    }
}
int main(){
    int i = ehPar (5);
    if (i!=0){
        printf ("O numero eh impar! \n");
    }
    else{
        printf ("O numero eh par! \n");
    }
    system("pause");
}
```

# Operadores de Atribuição



Operador	Exemplo	Comentário
=	$x = y$	Atribui o valor de $y$ a $x$
+=	$x += y$	Equivale a $x = x + y$
-=	$x -= y$	Equivale a $x = x - y$
*=	$x *= y$	Equivale a $x = x * y$
/=	$x /= y$	Equivale a $x = x / y$
%=	$x \% = y$	Equivale a $x = x \% y$

## Exercícios – Faça em Linguagem C

- 1) Transforme um valor em dólar, para reais. Sendo q um dólar vale R\$ 5,27.

## Exercícios – Faça em Linguagem C

2. O Shopping da Ilha agora usa uma nova tarifação para o estacionamento, mostrado na tabela abaixo:

TEMPO	VALOR
ATÉ 15 MINUTOS	GRÁTIS
ATÉ 3 HORAS	R\$ 8,00
A CADA HORA ADICIONAL (ACIMA DAS 3 PRIMEIRAS)	R\$ 2,00

O programa recebe o tempo em minutos e diz quanto o usuário precisa pagar.

## Exercícios – Faça em Linguagem C

3. O usuário digita três lados de um triângulo e o programa diz se os valores formam ou não um triângulo.

Obs: Pela regra, para se formar um triângulo, cada lado tem que ser menor que a soma dos outros dois lados. Ou seja  $(C < A+B)$ ,  $(B < A+C)$  e  $(A < B+C)$ .

## Exercícios – Faça em Linguagem C

- 4) Leia o nome, número de horas trabalhadas e número de dependentes de um funcionário. Após a leitura, escreva qual o Nome, salário bruto, os valores descontados para cada tipo de imposto e finalmente qual o salário líquido do funcionário. Considerando que:
- a) A empresa paga R\$12 por hora e R\$40 por dependentes.
  - b) Sobre o salário são descontados 8,5% p/ o INSS e 5% p/ IR



## Exercícios – Faça em Linguagem C

- 5) Leia uma distância em km, o preço da gasolina em reais e exiba quantos litros de gasolina o carro irá consumir e quanto será gasto em reais. Considere que o carro faz 12 km/l de gasolina.