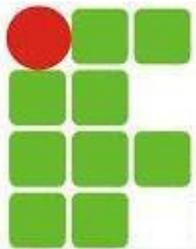


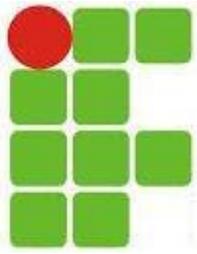
ALGORITMOS

Professor: Diego Oliveira



**Apresentação da
Disciplina**

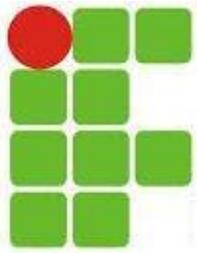




Agenda da Aula

- Apresentação do Professor
- Apresentação dos Alunos
- Apresentação da Disciplina
- Bibliografia Recomendada
- Metodologia Empregada
- Avaliação
- Orientações

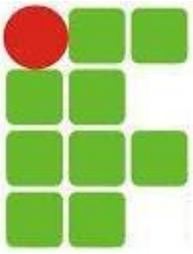




Apresentação do Professor

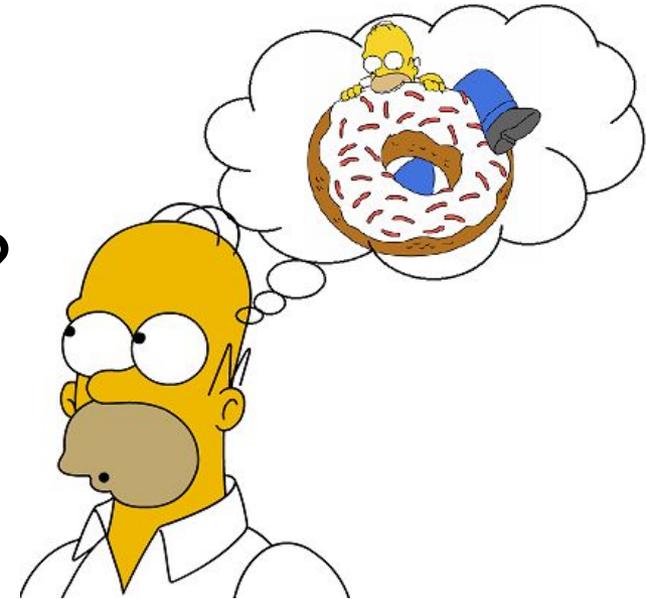
- Formação Acadêmica
 - Ensino Médio – CEFET-RN
 - Ciência da Computação – UERN
 - TADS – IFRN
 - Mestrado – UFRN
- Experiência Profissional
 - DETRAN-RN, SENAC-RN, CAIXA-SRRN, SIEP, PETROBRAS e TJ-RN
- Concursos
 - EXÉRCITO, DATAPREV, SERPRO e **IFRN**

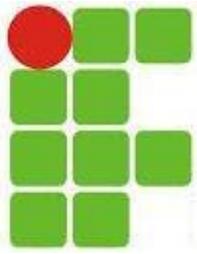




Apresentação dos Alunos

- Nome, idade e cidade?
- Porque escolheu o IFRN?
- Porque escolheu este curso?
- Pretende continuar na área?
- Qual seu sonho?

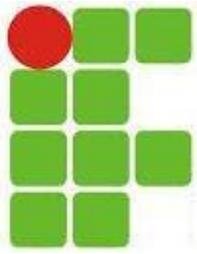




Apresentação da Disciplina

1. Fundamentos de Lógica de Programação
 - 1.1. Algoritmos(metalinguagem)
 - 1.2. Conceitos de memória, variáveis e constantes
 - 1.3. Tipos básicos de dados em C
 - 1.4. Operadores Aritméticos, relacionais e lógicos
 - 1.5. Comandos básicos de entrada e saída e atribuição
 - 1.6. Conceito de bloco de comandos
 - 1.7. Estruturas de controle de fluxo
 - 1.7.1. Condicionais: if, if-else e case
 - 1.7.2. Repetição: for, while e do-while
2. Estrutura de Dados Homogêneas
 - 2.1. Vetores e matrizes
 - 2.2. Cadeia de caracteres





Apresentação da Disciplina

3. Modularização

3.1. Variáveis locais e globais

3.2. Funções

3.3. Passagem de Parâmetro por valor e referência

3.4. Biblioteca de Funções

4. Ponteiros

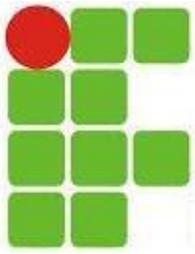
4.1. Alocação dinâmica de vetores e matrizes

4.2. Chamadas de sistemas malloc e free

5. Configuração de robôs móveis usando linguagem de blocos

6. Implementação de programas em Linguagem C aplicados à Mecatrônica





Bibliografia Recomendada

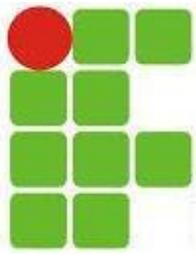
- Livros

1. Medina, Marco e Fertig, Cristina. Algoritmos e Programação: Teoria e Prática, 2005, Novatec
2. José Augusto N. G. Mazano e Jayr Figueiredo de Oliveira: Algoritmos: Lógica para o desenvolvimento de Programação de Computadores, Érica
3. Paulo Felioff, Algoritmos em Linguagem C

- Softwares

- Dev-Cpp

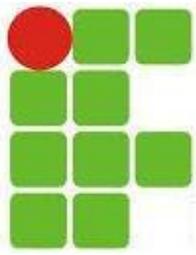




Metodologia

- Aulas expositivas e práticas em laboratório
- Avaliação
 - Participação do Aluno
 - Trabalhos
 - Seminários
 - Provas
- Área e e-mail do Professor
 - <http://docente.ifrn.edu.br/diegooliveira/>
 - diego.oliveira@ifrn.edu.br

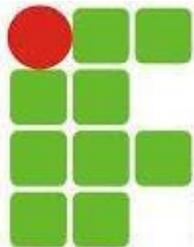




Avaliação

- Haverá pelo menos duas avaliações diferentes por bimestre
- Os trabalhos valerão 3 pontos
- As provas valerão 7 pontos
- Cada questão terá a pontuação indicada ao lado





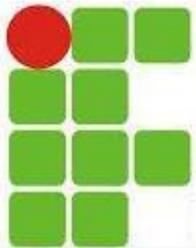
Orientações

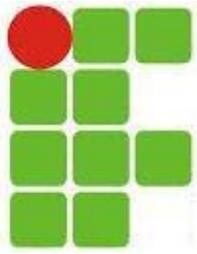
- A turma deve criar um e-mail
- A prova deve ser realizada de caneta
- Não é permitido o uso de celular na sala
- Não é permitida a entrada do aluno sem uniforme
- A tolerância para entrar na sala é de 15 minutos



OVERVIEW DA DISCIPLINA

Professor: Diego Oliveira

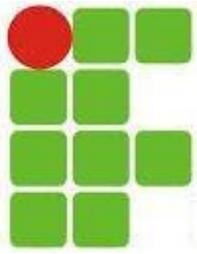




Implementação de Algoritmos

- Conceitos Fundamentais
- Tipos básicos de dados
- Memória, constantes e variáveis
- Operadores Aritméticos, Lógicos e Relacionais
- Comandos básicos de atribuição, entrada e saída
- Funções primitivas
- Estruturas Condicionais
- Estruturas de Repetição

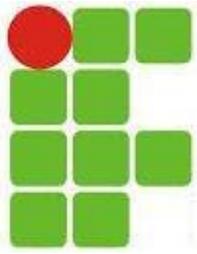




Conceitos Fundamentais

- Um **Algoritmo** serve para representar uma solução para um problema
- É uma linguagem intermediária entre a humana e as de programação
- Pode ser representado como:
 - Narrativa
 - Fluxograma
 - Pseudocódigo

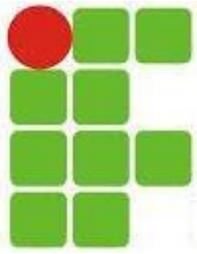




Conceitos Fundamentais

- **Narrativa:** nesta forma de representação, os algoritmos são expressos em linguagem natural
- Exemplo: trocar um pneu
 - 1: Afrouxar as porcas
 - 2: Levantar o carro
 - 3: Retirar as porcas
 - 4: Trocar o pneu pelo estepe
 - 5: Apertar as porcas
 - 6: Abaixar o carro

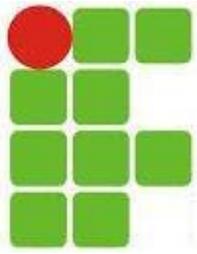




Conceitos Fundamentais

- **Fluxograma:** é uma representação gráfica dos algoritmos
- Cada figura geométrica representa diferentes ações
- Facilita o entendimento das idéias contidas no algoritmo



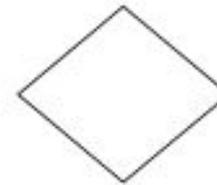


Conceitos Fundamentais

- Elementos do fluxograma:



Início e Fim de Programa



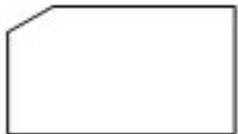
Decisão



Operação de Atribuição

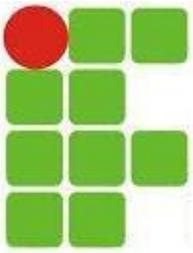


Operação de Saída



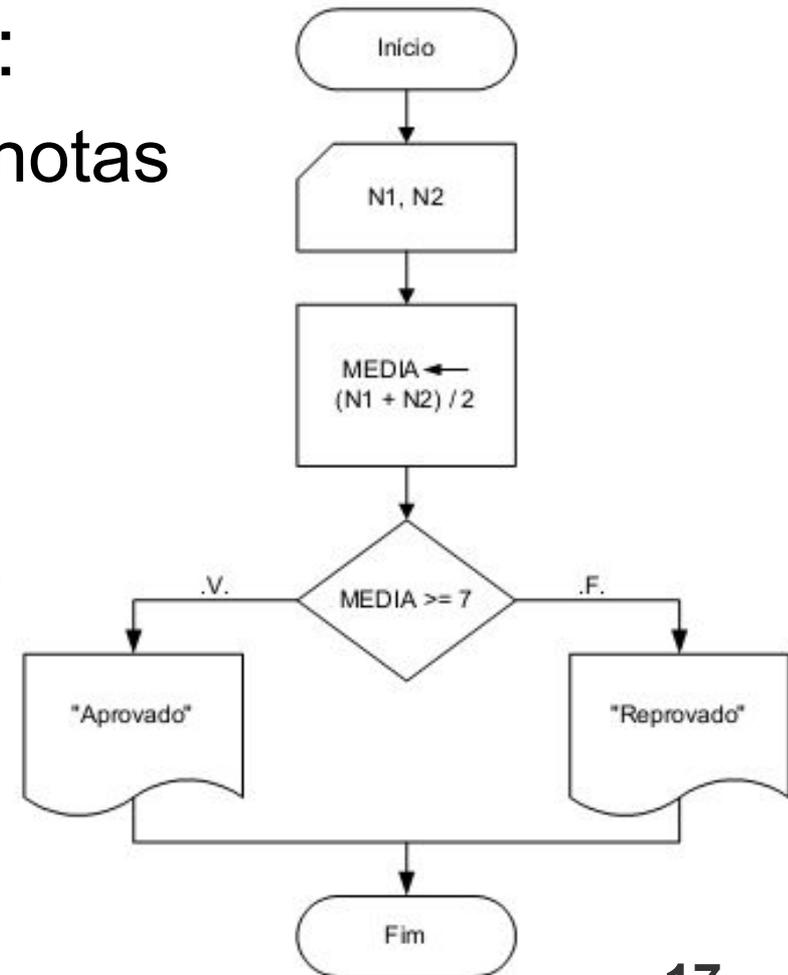
Operação de Entrada de Dados

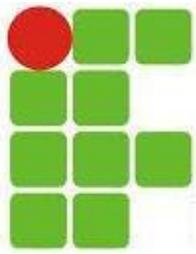




Conceitos Fundamentais

- Exemplo de fluxograma:
 - Calcular média de duas notas
 - A média para passar é 7
 - Não há recuperação
 - Indicar “Aprovado” ou “Reprovado” como saída

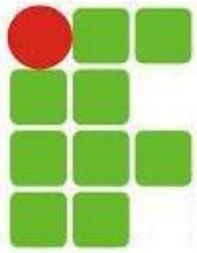




Conceitos Fundamentais

- **Pseudocódigo:** forma de representação de algoritmos rica em detalhes
 - É uma aproximação do código final a ser escrito em uma linguagem de programação
- **Algoritmo** é uma palavra que indica o início da definição de um algoritmo em forma de pseudocódigo.
 - **<nome_do_algoritmo>** é um nome simbólico dado ao algoritmo com a finalidade de distingui-los dos demais.
 - **<declaração_de_variáveis>** consiste em uma porção opcional onde são declaradas as variáveis globais usadas no algoritmo principal e, eventualmente, nos subalgoritmos.
 - **<subalgoritmos>** consiste de uma porção opcional do pseudocódigo onde são definidos os subalgoritmos.
 - **Início e Fim** são respectivamente as palavras que delimitam o início e o término do conjunto de instruções do corpo do algoritmo.





Conceitos Fundamentais

- Algoritmo da média de duas notas em pseudocódigo:

Algoritmo Media;

Var N1, N2, MEDIA: real;

Início

Leia (N1, N2);

MEDIA \leftarrow (N1 + N2) / 2;

Se MEDIA \geq 7 então

Escreva “Aprovado”

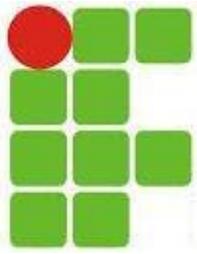
Senão

Escreva “Reprovado”;

Fim_se

Fim

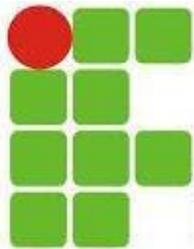




Tipos Básicos de Dados

- Dados Numéricos Inteiros
 - São os números positivos e negativos sem casas decimais
- Dados Numéricos Reais
 - São os números positivos e negativos que possuem casas decimais
- Dados Literais
 - São seqüências de caracteres
- Dados Lógicos ou Booleanos
 - Podem ser verdadeiros ou Falsos, apenas

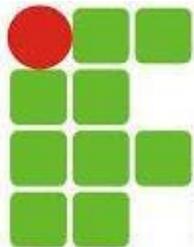




Variáveis

- O armazenamento de informações pelo computador em sua memória, se dá em uma região nomeada através de uma variável
- Uma variável possui:
 - NOME
 - TIPO
 - CONTEÚDO
- As regras para nomes de variáveis mudam de uma linguagem para outra

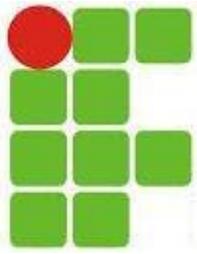




Variáveis

- Variáveis devem ser declaradas antes de serem utilizadas
- Ao declarar uma variável, o computador reserva um espaço na memória para ela
- A memória é constituída de bytes, que são conjuntos de 8 bits
- Cada tipo de variável ocupa um tamanho diferente na memória, isso varia para cada linguagem de programação

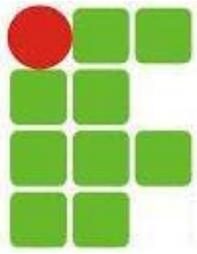




Operadores

- Os operadores podem ser:
 - Lógicos
 - Aritméticos
 - Relacionais
- Cada tipo de operador tem sua função específica e uma ordem de precedência



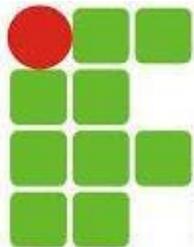


Operadores

- Operadores Lógicos:

Lista de Operadores Lógicos			
Operador	QTD de Operadores	Operação	Prioridade
.OU.	binário	disjunção	3
.E.	binário	conjunção	2
.NAO.	unário	negação	1



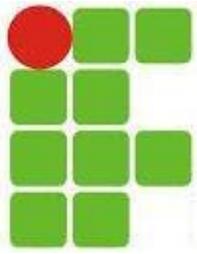


Operadores

- Operadores Aritméticos

Lista de Operadores Numéricos			
Operador	QTD de operadores	Operação	Prioridade
+	binário	adição	4
-	binário	subtração	4
*	binário	multiplicação	3
/	binário	divisão	3
**	binário	exponenciação	2
+	unário	conservação do sinal	1
-	unário	inversão do sinal	1



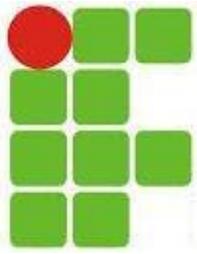


Operadores

- Operadores Relacionais:

Lista de Operadores Relacionais		
Operador	QTD de Operadores	Operação
=	binário	igualdade
<	binário	Menor que
>	binário	Maior que
<=	binário	Menor ou igual
>=	binário	Maior ou igual
<>	binário	diferença

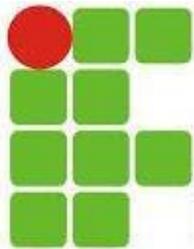




Atribuição

- Permitem colocar um valor em uma variável:
VAR A = 10;
 TEXTO = “Diego”;
- Uma variável só pode receber um valor do seu tipo
- Cada linguagem de programação possui tipos específicos de dados



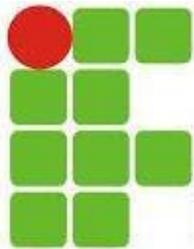


Entrada

- As operações de entrada permitem que o usuário forneça dados ao programa
- A entrada também pode ser dada via programas, scanners, câmeras e outros
- A leitura do teclado em C é feita assim:

```
#include <stdio.h>      // entrada e saída
int main( void ) {     // Programa principal
    int i;
    scanf("%d", &i);
}
```

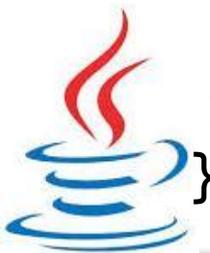


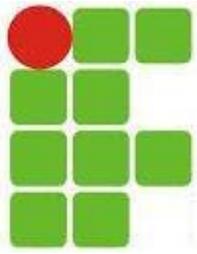


Saída

- As operações de saída permitem que o programa forneça informações ao usuário
- Geralmente a saída é feita na tela, mas também pode ser via rede, impressora, leds, som e outros
- A saída na tela em C é feita assim:

```
#include <stdio.h>          // entrada e saída
int main( void ) {         // Programa principal
    int ano = 2014;        // variável ano
    printf("Estamos no ano %d", ano);
}
```

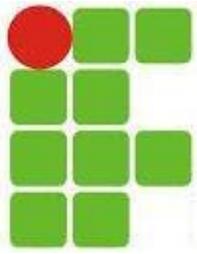




Funções

- Conjuntos de comandos agrupados em um bloco que recebe um nome
- A função pode ser chamada pelo seu nome
- Permitem o reaproveitamento de código
- Facilitam a manutenção do código
- Facilitam a leitura e entendimento do código
- Proporcionam a modularização do programa





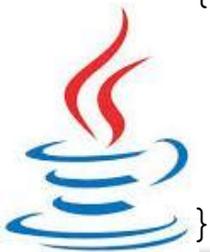
Funções

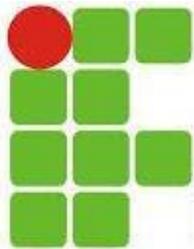
- **Função SOMA:**

```
void SOMA(float a, int b)
{
    float result;
    result = a+b;
    printf("A soma de %6.3f com %d é %6.3f\n", a,b,Result);
}
```

- **Chamando a função SOMA:**

```
#include <stdio.h>
void SOMA(float a, int b); //Protótipo da função SOMA
void main()
{
    float f;
    f = 20.0
    SOMA(16, f);
}
```

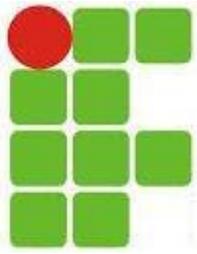




Escopo de Variáveis

- Uma variável GLOBAL pode ser exergada em qualquer parte do código
- Uma variável LOCAL só pode ser enxergada no escopo em que foi declarada (função)
- PARÂMETROS FORMAIS são variáveis inicializadas no momento da chamada da função
- Tentar ler uma variável fora de seu escopo gera um erro de compilação



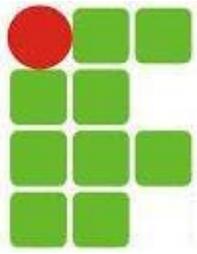


Estruturas Condicionais

- As estruturas condicionais (IF/ELSE) são utilizadas quando é preciso escolher entre mais de um caminho possível
- Para se escolher o caminho, uma estrutura condicional é analisada:

```
if (numero%2 == 0) //se for verdadeiro imprime O numero eh PAR
{
    printf("O numero eh PAR \n");
}
else
{
    printf("O numero eh IMPAR \n");
}
```



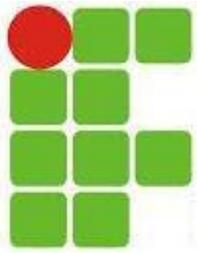


Estruturas Condicionais

- Outro exemplo de IF/ELSE encadeado:

```
if (numero%2 == 0) //se for verdadeiro imprime 0 numero eh PAR
{
    printf("0 numero é multiplo de 2 \n");
}
else if(numero%3 == 0)
{
    printf("0 numero é multiplo de 3 \n");
}
else if(numero%5 == 0)
{
    printf("0 numero é multiplo de 5 \n");
}
else
{
    printf("0 numero nao é multiplo de 2,3 ou 5 \n");
}
```



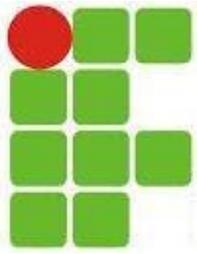


Estruturas Condicionais

- O SWITCH é utilizado quando o range de opções é conhecido, como em um menu:

```
switch (opcaoMenu)
{
    case 1: calcularNota(); break;
    case 2: calcularNotaRecuperacao(); break;
    case 3: calcularNotaParaPassar(); break;
    default: sair();
}
```

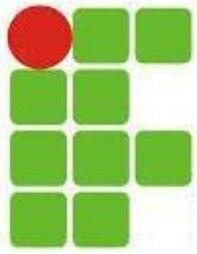




Estruturas de Repetição

- As estruturas de repetição permitem que um trecho de código seja repetido até que uma condição seja satisfeita
- Os laços, ou loops, podem ser:
 - FOR
 - WHILE
 - DO WHILE



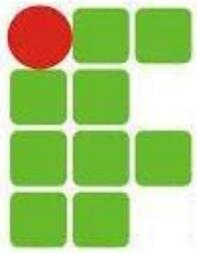


Estruturas de Repetição

- O laço FOR é utilizado quando a quantidade de repetições desejada é conhecida:

```
#include<stdio.h>
int main() {
    int i;
    for (i=0; i<10; i++)    //de 0 a 9
    {
        printf("%d\n", i);
    }
}
```



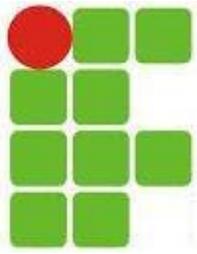


Estruturas de Repetição

- O laço WHILE é utilizado para que a repetição aconteça enquanto uma condição permaneça verdadeira:

```
#include<stdio.h>
int main() {
    int i=0;
    while(i < 10)
    {
        i = i+1; printf ("%d\n", i);
    }
}
```



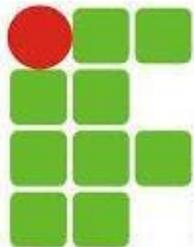


Estruturas de Repetição

- O laço DO WHILE é semelhante ao WHILE, a diferença é que ele primeiro executa a repetição e depois verifica a condição:

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

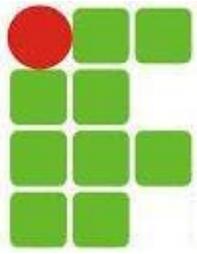




Indicações

- Filmes Indicados:
 - Piratas do Vale do Silício (MS)
 - Jobs (Apple)
 - O Quinto Poder (Wikileaks)
 - A Rede Social (Facebook)
 - Hackers 2 (Kevin Mitnick)
- Livros Indicados:
 - Fortaleza Digital
 - Universidade H4CK3R
 - A Indecifrável Enigma





Perguntas?

