

IAL-003 – Algoritmos e Programação de Computadores

Prof. Me. Anderson Vanin

Quem sou eu

- Técnico Eletrônico
- Bacharel em Ciência da Computação
- Pós Graduado em Banco de Dados
- Mestre em Gestão do Conhecimento e Informática (Aplicado a VC)
- Atuação no CPS desde 2006

Quem são meus alunos?

Nome, idade e cidade?
Porque escolheu a FATEC?
Porque escolheu este curso?
Pretende continuar na área?
Qual seu sonho?



IAL-003 – Algoritmos e Programação de Computadores

Ementa da Disciplina

Método para desenvolvimento de algoritmos. Modularidade e abstração. Tipos de dados básicos e representações gráficas dos principais comandos em uma linguagem procedural. Expressões aritméticas, lógicas e literais. Estruturas básicas de programas (sequência, iteração, seleção simples e múltipla). Desenvolvimento de programas estruturados para a manipulação de estruturas básicas.

Material e recursos para as aulas

- github.com/profandersonvanin
- Email: ...
- Todos tem um *Github*?

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

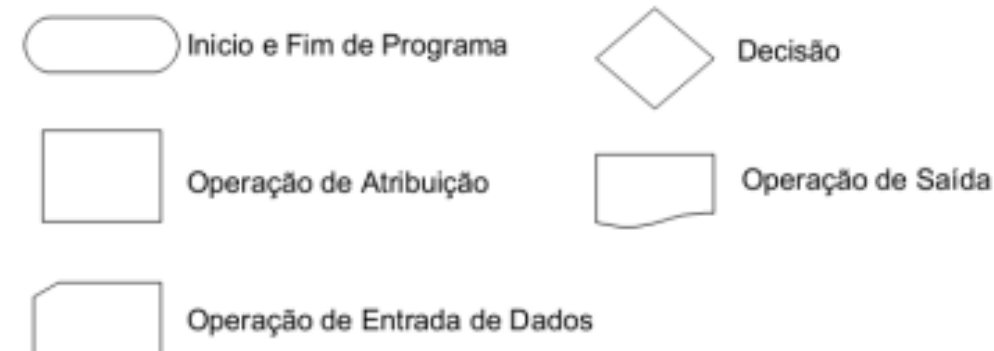
- **Narrativa:** nesta forma de representação, os algoritmos são expressos em linguagem natural
- Exemplo: trocar um pneu
 - 1: Afrouxar as porcas
 - 2: Levantar o carro
 - 3: Retirar as porcas
 - 4: Trocar o pneu pelo estepe
 - 5: Apertar as porcas
 - 6: Abaixar o carro

Conceitos Fundamentais

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

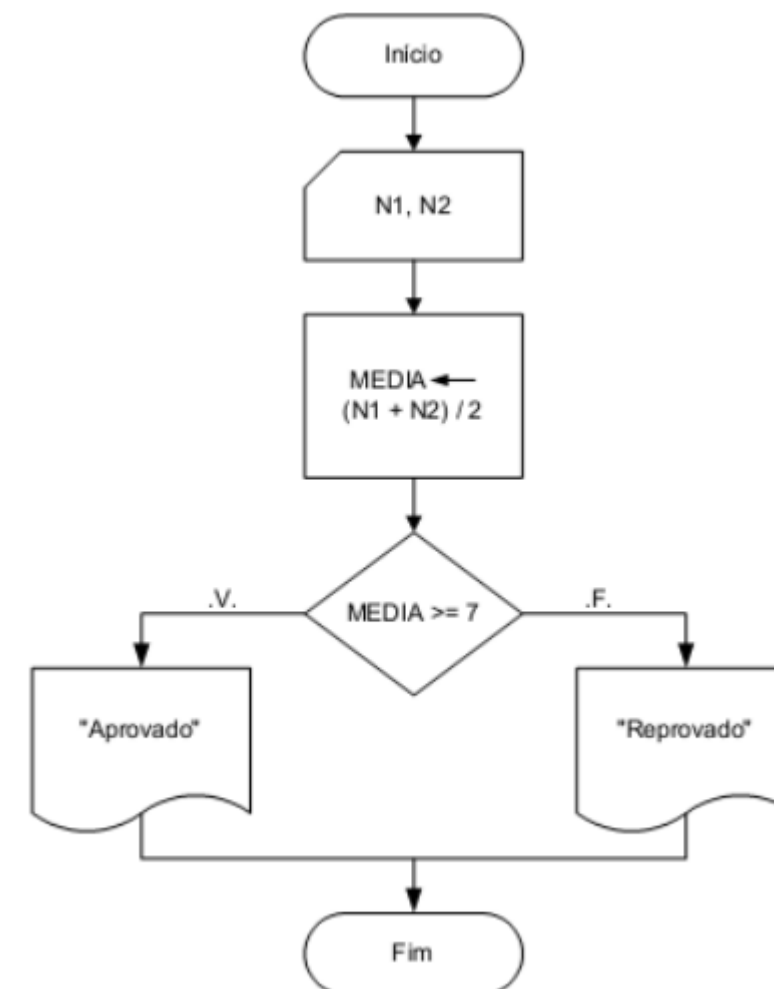
Conceitos Fundamentais

- Elementos do fluxograma:
 - Início e fim de programa
 - Representados por uma elipse
 - Operação de Atribuição
 - Representada por um retângulo
 - Operação de Entrada de Dados
 - Representada por um retângulo com um dos cantos dobrados (como em uma folha de papel)
 - Decisão
 - Representada por um losango
 - Operação de Saída
 - Representada por um retângulo com um dos lados recordado de maneira ondulada



Conceitos Fundamentais

- Exemplo de fluxograma:
 - Início (dentro de uma elipse)
 - Calcular média de duas notas (dentro de um retângulo com um dos cantos dobrados)
 - A média para passar é 7 (dentro de um retângulo)
 - Indicar “Aprovado” ou “Reprovado” como saída (verifica se a média é maior ou igual a 7 dentro de um losango)
 - Se a média for maior ou igual a 7 imprime “Aprovado” dentro de um retângulo com um dos lados recortado de maneira ondulada
 - Se a média for menor do que 7 imprime “Reprovado” dentro de um retângulo com um dos lados recortado de maneira ondulada
 - Fim de programa (dentro de uma elipse)



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 de 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 sequências 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 saida
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 enxergada em qualquer parte do código
- Uma variável LOCAL só pode ser enxergada no escopo em que foi declarada (função)
- PARAMETROS 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 O numero eh PAR
{
    printf("O numero eh multiplo de 2 \n");
}
else if(numero%3 == 0)
{
    printf("O numero eh multiplo de 3 \n");
}
else if(numero%5 == 0)
{
    printf("O numero eh multiplo de 5 \n");
}
else
{
    printf("O numero nao eh 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);
}
```