* 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 inicio 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\_variaveis> 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
* Inicio e Fim são respectivamente as palavras que delimitam o inicio e o término do conjunto de instruções do corpo do algoritmo.
* Algoritmo da Média de duas notas em pseudocódigo:

Algoritmo média;

Var N1, N2, MEDIA: real;

Inicio

Leia (N1, N2);

MEDIA (N1+N2) / 2;

Se media >=7 então

Escreva “Aprovado”

Senão

Escreva “Reprovado”

Fim\_se

Fim

* 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 seqüênciais de caracteres

* Dados Lógicos ou Booleanos

- Podem ser verdadeiros ou Falsos, apenas.

* 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 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
* Os operadores podem ser:

-Lógicos

-Aritméticos

-Relacionais

* Cada tipo de operador tem sua função especifica e uma ordem de precedência
* 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 Aritméticos

|  |  |  |  |
| --- | --- | --- | --- |
| **Lista de Operadores Numéricos** | | | |
| **Operador** | **QTD de Operadores** | **Operação** | **Prioridade** |
| + | binário | Adição | 4 |
| - | Binário | Subtração | 4 |
| \* | Binário | Multiplicação | 3 |
| / | Binário | Divisão | 3 |
| \*\* | Unário | Exponenciação | 2 |
| + | Unário | Conservação do sinal | 1 |
| - | unário | Inversão do sinal | 1 |

* 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 |

* 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
* As operações de entradas 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) ;}

* 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) ;

* 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 leitura e entendimento do código
* Facilitam a manutenção do código
* Proporcionam a modularização do programa
* 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 SOMAvoid main ()

{

Float f;

F = 20.0

SOMA (16,f);

* 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
* Tenta ler uma variável fora de seu escopo gera um erro de compilação
* 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”);

}

* Outro exemplo de IF/ELSE encadeado:

If (numero%2 = 0) //se for verdadeiro imprime o numero eh PAR

{

printf (“O numero eh múltiplo de 2/n”);

}

Else if (numero%3 == 0)

{

printf (“O numero eh múltiplo de 3/n”);

}

Else if (numero%5 == 0)

{

Printf (“ O numero eh múltiplo de 5 /n”);

}

Else

{

Printf (“ O numero não eh múltiplo de 2,3 ou 5 /n”);

}

* 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 ();

}

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

-FOR

-WHILE

- DO WHILE

* O laço FOR é utilizado quando a quantidade de repetições desejada é reconhecida:

#include<stdio.h>

Int main () {

Int i;

For (i=0; i=10; i++) // de 0 a 9

{

Printf (“%d/n”, i);

}

}

* 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);

}