- Prova AMC, 21/05, terça, início às 7h40:

Nota AMC Projeto Integrador 1 Lógica de Programação

SS Sobe 2 menções na P2 Sobe 2 menções na P2

MS Sobe 2 menções na P2 Sobe 1 menções na P2

MM Sobe 1 menções na P2 Sobe 1 menções na P2

- Quantidade de avaliações das duas disciplinas no semestre:

Projeto Integrador 1 (PI1): P1 P2 Projeto final (total de 3 avaliações)

Lógica de Programação (LP): P1 P2 AMC P3 (total de 4 avaliações)

**- Algoritmo**

Um algoritmo é um **conjunto** de passos necessários para realizar uma **tarefa**.

O algoritmo é um conjunto finito de instruções objetivas e não ambíguas que são executadas sequencialmente.

Um algoritmo é uma lista de procedimentos bem definida, na qual as instruções são executadas passo a passo a partir do começo da lista.

Diferentes algoritmos podem realizar a mesma tarefa usando um conjunto diferenciado de instruções.

**Exemplos** de algoritmos no nosso cotidiano, pois a lógica está presente no nosso dia-a-dia:

Indicações de montagens;

Receitas de cozinha;

Fazer um bolo a partir de uma receita.

Fritar um ovo.

Fazer pipocas num micro-ondas;

Trocar uma lâmpada;

**- Algoritmo de criptografia**

Em criptografia, a Cifra de César, também conhecida como cifra de troca, código de César ou troca de César, é uma das mais simples e conhecidas técnicas de criptografia. É um tipo de cifra de substituição na qual cada letra do texto é substituída por outra, que se apresenta no alfabeto abaixo dela um número fixo de vezes. O nome do método é em homenagem a Júlio César, rei de Roma, que o usou para se comunicar com os seus generais.

Exemplo 1:

Palavra original: abc

Palavra criptografada: bcd

Exemplo 2:

Palavra original: abc

Palavra criptografada: cde

**- Compilador**

O compilador é o processo que efetua integralmente a **tradução** de um programa fonte para o código de máquina, podendo assim ser executado diretamente.

Compilador

**Traduz** um programa em uma linguagem de alto nível para uma linguagem de máquina, para que suas instruções sejam executadas pelo processador, ou seja, cria o **executável** de um programa escrito em uma linguagem de alto nível.

**Linguagens** de baixo nível x alto nível

As **linguagens de baixo nível** são interpretadas diretamente pelo computador, tendo um resultado rápido.

As **linguagens de alto nível** são mais fáceis de se trabalhar, escritas geralmente em inglês, facilitando a memorização e a lógica.

Exemplos:

C e C++ (C e C++ são consideradas por muitos como baixo nível e por outros como um nível médio de programação).

Assembly é a linguagem de baixo nível mais próxima da linguagem de máquina.

- Declaração de variáveis

Os principais tipos de dados da linguagem C são int, float e char.

O tipo de dados int é capaz de armazenar números inteiros negativos ou positivos.

O tipo de dados float é capaz de armazenar números reais, com ou sem casas decimais, positivo ou negativo.

O tipo de dados char é capaz de armazenar um caractere: letras, números ou sinais.

Exemplos:

main() {

int idade;

float peso, altura;

char sexo;

}

**- if ... elif ... else ... (se ... então ... senão ...)**

A estrutura de decisão SE/ENTÃO/SENÃO, ou IF/THEN/ELSE, permite que seja sempre executado um comando.

Isso porque, caso a condição seja verdadeira, o comando da condição SE/ENTÃO será executado; caso contrário, o comando da condição SENÃO (falsa) será executado.

Analise este código na lingugem Java e mostre a saída correta.

public class Teste{

private String a = "unipam"; // a = "unipam"

private Integer b = 0;

b -- // b -= 1 ou b = b - 1

if ( b < 0 ) {

System.out.println ("a");

}

else {

System.out.println (b);

}

}

Saída: a

**- if ... elif ... else e os operadores lógicos and (&&, e) e or (||, ou)**

Analise este código na linguagem C e mostre qual a saída se a entrada for 5 (cinco). Prova

int main(){

int x; // Declaração de variável, em Python não precisa.

scanf("%d", &x); // leia (x) ou x = int(input('Valor: '))

if(x > 5 && x < 10){ // && - and - e

printf("%c",'A');

}else if((x \* 2) >= 10 && x < 20){ // else if - elif

printf("%c",'B');

}

if(x > 2 || x == 6){ // || - or - ou

printf("%c",'C');

}else{

printf("%c",'D');

}

return 0; // Sai da função main (do programa)

}

Saída: BC

Analise o algoritmo em pseudocódigo a seguir e complete a condição do se.

Considerando-se que uma nota válida deve possuir valores entre 0 e 10 (inclusive),

a lacuna que corresponde à condição do comando SE é CORRETAMENTE

preenchida por: Prova

início

real: n1, n2;

imprima("Digite a primeira nota: ");

leia (n1);

imprima("Digite a segunda nota: ");

leia (n2);

se \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

então

media ← (n1+n2)/2;

imprima("A média das notas é ", media);

senão

imprima("Alguma nota fornecida é inválida.");

fim se;

fim.

Respostas:

se n1 >= 0 E n1 <= 10 E n2 >= 0 E n2 <= 10 # algoritmo em pseudocódigo

if n1 >= 0 and n1 <= 10 and n2 >= 0 and n2 <= 10: # Python

''' Lista3 - 15. Dados três valores A, B e C. Verifique se eles podem ser o comprimento dos lados

de um triângulo. Se forem, verifique se compõem um triângulo equilátero, isósceles ou

escaleno. Informe se não compuserem nenhum triângulo. Relembre as seguintes definições:

- No plano, triângulo (também aceito como trilátero) é a figura geométrica que ocupa o

espaço interno limitado por três linhas retas que concorrem, duas a duas, em três pontos

diferentes formando três lados e três ângulos internos. Para ser triângulo, qualquer lado

tem medida menor que a soma das medidas dos outros dois.

- Triângulo equilátero: possui três lados iguais;

- Triângulo isósceles: possui dois lados iguais;

- Triângulo escaleno: tem todos os lados diferentes.

leia os valores e faça o teste para verificar se os lados não formam um triângulo.

Testes: Entrada: 1, 2 e 3 Resposta: Não é um triângulo.

Entrada: 2, 3 e 4 Resposta: É um triângulo escaleno.

Entrada: 3, 3 e 3 Resposta: É um triângulo equilátero.

Entrada: 2, 3 e 3 Resposta: É um triângulo isósceles.

a = float(input("Digite o comprimento do primeiro lado: "))

b = float(input("Digite o comprimento do segundo lado: "))

c = float(input("Digite o comprimento do terceiro lado: "))

# Verifica se NÃO é triângulo, se um dos lados é maior ou igual a soma dos outros dois lados.

if a >= b + c or b >= a + c or c >= a + b:

print("Os lados NÃO formam um triângulo")

elif a == b and b == c: # Todos os lados são iguais

print("Triângulo equilátero")

elif a != b and b != c and c != a: # Todos os lados são diferentes

print("Triângulo escaleno")

else: # 2 lados iguais

print("Triângulo isósceles")

**- while (enquanto)**

Analise o trecho de código na linguagem C abaixo:

main (){  
   int i; // Declaração da variável i  
   scanf ("%d", &i); // leia (i) ou i = int (input(‘Valor: ‘))  
   while (i < 2) { // O abre chave indica início do while  
       return; // Sai do método main (do programa)  
    } // O fecha chave indica fim do while  
    i ++; // i = i + 1  
    printf ("%d", i);  
}

Qual a saída para estas entradas:

- Ao ler 2 na entrada

- Ao ler 1 na entrada

Ao ler 2 na entrada, o algoritmo imprimirá 3 na saída.

Ao ler 1 na entrada, o algoritmo não imprimirá nada. Pois ele sai do método main.

Troque o return pelo break

Troque o return pelo continue

Analise o trecho de código na linguagem C abaixo:

int main(){

    int numero = 0;

    int resultado = 0;

    scanf ("%d", &numero); // leia (numero) ou numero = int (input(‘Valor: ‘))

    while (numero > -1){ // O abre chave indica início do while

        resultado = resultado + numero;

        scanf("%d", &numero); // leia (numero) ou numero = int (input(‘Valor: ‘))

    } // O fecha chave indica fim do while  
    printf ("%d", resultado);

    return 0; // Sai do método main (do programa)

}

Qual a saída para as seguintes entradas:

a. Menor ou igual a -1

b. Igual a -1

c. Está sequência: 1, 2, 3, 0, 2, -1

d. Está sequência: último dígito RA, penúltimo dígito RA, antepenúltimo dígito RA, -1

Respostas dos itens acima:

0

0

8

?

Supondo que em um programa que leia o preço de uma quantidade indeterminada de produtos e ao final imprima o somatório de todos os preços, utilizando como flag de saída, um preço negativo ou igual a 0 (zero). Implemente o programa que executa a leitura e a soma dos preços, desconsiderando na soma, os preços negativos ou iguais a zero?

Obs.: a variável soma é inicializada com zero, e a variável preço é inicializada com uma leitura antes da estrutura de repetição.

Resulva sem usar o break e o if dentro do while.

Obs.: a variável soma é inicializada com zero, e a variável preço é inicializada com

uma leitura antes da estrutura de repetição.

soma = 0

preco = float(input("Digite o preço: R$ ")) # Faz somente a primeira leitura.

while preco > 0: # Flag de saída, um preço negativo ou zero

soma += preco

preco = float(input("Digite o preço: R$ ")) # Faz as outras leituras.

print ("A soma dos preços: ", soma)

Analise o algoritmo em pseudocódigo:

r ← 0

cont ← 1

enquanto (cont <= 5) faça

leia (n)

r ← r \* n

cont ← cont + 1

fim\_enquanto

imprima (r)

5a. Qual a saída se forem lidos os seguintes valores: 2, 5, 7, 3 e 4

5b. Troque o r ← 0 por r ← 1 Valores:2, 5, 7, 3 e 4

5c. Troque o r ← 0 por r ← 1

Valores; Está sequência: último dígito RA, penúltimo dígito RA, 6, 1, 1

Saídas:

a. A saída será 0 (pegadinha)

b.

c.

Analise o algoritmo em pseudocódigo:

var res, cont, x, n : inteiro

início

cont ← 1

res ← 0

x ← 2

n ← 4

enquanto (cont >= 4) faça

res ← res \* x

cont ← cont + 1

imprima (res)

fim\_enquanto

imprima (res)

fim

a. A saída na tela será

Saída: zero (pegadinha)

b. Troque o >= pelo <=

Saída: 0 0 0 0 0

c. Além da troca anterior, troque também o res = 0 pelo res = 1

Saída: 2 4 8 16 16

**- for (para)**

Considere esse trecho de código e responda a afirmativa abaixo.

for (int i = 0 ; i < 100; i = i + 2) { # for i in range (0, 100, 2): # Em Python

printf (“%d”, i);

}

Qual a saída?

0 2 4 ... 98

Considere esse trecho de código e responda a afirmativa abaixo.

for (int i = 0 ; i < 100; i = i + 2) { # for i in range (0, 100, 2): # Em Python

printf (“%d”, i);

if (i < 10){

printf (“Sistemas”);

}

}

Quantas vezes a palavra “Sistemas” será impressa?

Resposta: 5 vezes

Qual a saída desse programa?

0 Sistemas 2 Sistemas 4 Sistemas 6 Sistemas 8 Sistemas 10 12 14 ... 98

Considere esse trecho de código e responda a afirmativa abaixo.

for (i = 1; i < 9; i ++) # for i in range (1, 9, 1): # Em Python

printf (“%d”, i);

Qual a saída?

1 2 3 4 5 6 7 8

Observe o fragmento de código abaixo.

x = 3;

y = 4;

z = 5;

if ( (x-1) > 2 )

y = y + 1;

else

y = y - 1;

z = x + y;

for (i = 1; i < 9; i ++) # for i in range (1, 9, 1): # Em Python

y = y + 1;

z = z + y;

Ao final da execução desse código, qual o valor de z?

z = 17

Troque o x = 3 pelo x = 4

z = 22

Considere o algoritmo abaixo:

main()

{

int i;

for (i = 0; i <= 4; i++){

printf (“%d\n”, i);

}

}

A saída do algoritmo será:

for i in range (0, 4 + 1, 1): # Em Python

Saída:

0 1 2 3 4

Considere o algoritmo abaixo:

main()

{

int x = 3, y = 2, i;

for (i = 0; i <= 4; i++){ # for i in range (0, 4 + 1, 1): # Em Python

x = i + 4;

y = x \* 2;

}

printf (“%d\n”, y);

}

A saída do algoritmo será:

Saída: 16

**- fors (paras) encadeados ou aninhados**

Qual o valor da variável, ao final da execução do programa a seguir (antes)

int n = 5;

int a = 0;

for (int b = 1; b <= n; b++) # for b in range (1, n + 1, 1): # Python

for (int c = 1; c <= n; c++) { # for c in range (1, n + 1, 1): # Python

a = a + 1

}

printf (a);

Saída: 25

Qual o valor da variável, ao final da execução do programa a seguir

int n =5;

int a =0;

for (int b = 1; b <= n; b++) # for b in range (1, n + 1, 1): # Python

for (int c = b; c <= n; c++) { # for c in range (b, n + 1, 1): # Python

a = a + 1

}

printf (a);

Saída: 15

**- Lista (vetor)**

Vetores são utilizados quando estruturas indexadas necessitam de **um índice** para identificar um de seus elementos.

Vetores podem ser considerados como **listas** de informações armazenadas em posição contígua na memória.

**Matriz**

Matrizes são utilizados quando estruturas indexadas necessitam **de mais que um índice** para identificar um de seus elementos.

Matrizes com dois índices podem ser considerados como **tabelas** de informações armazenadas em posição na memória.

Considere o algoritmo abaixo:

main ()

{

int soma;

int v[6];

for (i = 1; i < 6; i++){

v[i] = 2 \* i ;

}

soma = v[1] + v[3]

printf (“Resultado: %d\n”, soma);

}

Lista v, antes do for:

v = [0 , 0, 0, 0, 0, 0] # Valores

0 1 2 3 4 5 # Índices ou posições

Lista v, depois do for:

v = [0 , 2, 4, 6, 8, 10] # Valores

0 1 2 3 4 5 # Índices ou posições

Considere o algoritmo abaixo:

main ()

{

int x, y, i;

int v[6];

scanf (“%d”, &x);

scanf (“%d”, &y);

for (i = 1; i < 6; i++){

v[i] = x \* i + y;

}

printf (“r: %d\n”, v[1] + v[3]);

}

Caso os valores lidos para x e y sejam, respectivamente, 3 e 7, então o valor impresso será

v = [0 , 10, 13, 16, 19, 22]

26

Abaixo, implemente o código equivalente na linguagem Python.

Considere as seguintes declarações de variáveis na linguagem C.

int a = 2; # a = 2 Em Python.

double b = 2; # b = 2

int c[] = {1,2}; # c = [1, 2]

A estrutura condicional retornará verdadeiro ou falso?

a. if ( a > c[0])

b. if (b >= c)

c. if (a == c)

d. if ( a > c[1])

e. if ( a == b && a == c[0])

f. if ( (a = b) > c[0])

- Respostas:

Somente as letras a. e f. são verdadeiras ----- '''

a = 2

b = 2

c = [1, 2]

if a > c[0]: # a.

print("A estrutura condicional retorna verdadeiro.")

else:

print("A estrutura condicional retorna falso.")

# if b > c: # b. Não posso comparar uma variável com uma lista.

# print("A estrutura condicional retorna verdadeiro.")

# else:

# print("A estrutura condicional retorna falso.")

if a == c: # c. Não posso comparar uma variável com uma lista.

print("A estrutura condicional retorna verdadeiro.")

else:

print("A estrutura condicional retorna falso.")

if a > c[1]: # d.

print("A estrutura condicional retorna verdadeiro.")

else:

print("A estrutura condicional retorna falso.")

if a == b and a == c[0]: # e.

print("A estrutura condicional retorna verdadeiro.")

else:

print("A estrutura condicaional retorna falso.")

a = b #if (a = b) > c[0]: # f.

if a > c[0]:

print("A estrutura condicional retorna verdadeiro.")

else:

print("A estrutura condicional retorna falso.")