INSTITUTO FEDERAL
São Paulo

Campus
Caraguatatuba

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

PROGRAMAÇÃO DE COMPUTADORES 1

Prof.ª Juliana / Prof. Mário

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

Atividade Avaliativa 1

Correção e Recuperação

Faça uma função que recebe, por parâmetros, a hora de inicio e a hora de término de um jogo, ambas subdivididas em 2 valores distintos: horas e minutos. A função deve retornar a duração do jogo em minutos. Considerar que o jogo começa e termina no mesmo dia.

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

PASSOS PARA RESOLUÇÃO:

1. **Identificar as entradas:** O problema fornece a hora e os minutos de início e término do jogo.
2. **Converter tudo para minutos:** Multiplicar a hora por 60 e somar os minutos para obter o total de minutos desde o início do dia.
3. **Calcular a duração do jogo:** Subtrair o tempo inicial do tempo final.
4. **Retornar o resultado:** Exibir a duração total em minutos.
5. **Consideração importante:** O jogo começa e termina no mesmo dia, então não há necessidade de tratar viradas de dia.

Exercício 1

PASSOS PARA RESOLUÇÃO:

As tarifas de certo parque de estacionamento são as seguintes:

- 1a e 2a hora - R$ 1,00 cada
- 3a e 4a hora - R$ 1,40 cada
- 5a hora e seguintes - R$ 2,00 cada

O numero de horas a pagar é sempre inteiro e arredondado por excesso. Deste modo, quem estacionar durante 61 minutos pagará por duas horas, que é o mesmo que pagaria se tivesse permanecido 120 minutos. Os momentos de chegada ao parque e partida deste são apresentados na forma de pares de inteiros, representando horas e minutos. Por exemplo, o par 12 50 representara "dez para a uma da tarde". Pretende-se criar um programa que, lidos pelo teclado os momentos de chegada e de partida, escreva na tela o preço cobrado pelo estacionamento. Admite-se que a chegada e a partida se dão com intervalo não superior a 24 horas. Portanto, se uma dada hora de chegada for superior à da partida, isso não é uma situação de erro, antes significará que a partida ocorreu no dia seguinte ao da chegada.

1. **Identificar as entradas:** São fornecidas a hora e os minutos de chegada e de saída.
2. **Converter tudo para minutos:** Assim como na questão anterior, transformar as horas em minutos e somá-las aos minutos fornecidos.
3. **Calcular o tempo total estacionado:** Fazer a subtração entre o tempo de saída e o tempo de chegada.
4. **Arredondar o tempo para cima:** O tempo é sempre contado em horas inteiras, então deve-se considerar o arredondamento para o próximo número inteiro caso haja minutos adicionais.
5. **Definir as tarifas:** Aplicar as regras do estacionamento conforme as faixas de tempo descritas no enunciado.
6. **Calcular o preço final:** Aplicar a tarifa correta conforme o número total de horas arredondadas.
7. **Exibir o valor final ao usuário.**

Exercício 2

PASSOS PARA RESOLUÇÃO:

Implemente um programa que leia um número inteiro positivo n e verifique se ele é um número de Armstrong (ou seja, se a soma dos seus dígitos elevados ao número de dígitos é igual ao próprio número). Utilize uma estrutura de repetição while. Um número de Armstrong é um número que é igual à soma de seus próprios dígitos, cada um elevado à potência do número de dígitos. Por exemplo: 153 é um número de Armstrong porque (1^3 + 5^3 + 3^3 = 153) e 370 também é um número de Armstrong porque (3^3 + 7^3 + 0^3 = 370). O código poderá receber até 4 dígitos, elaborar a função para calcular o quadrado, cubo e elevado a quarta potência.

1. **Identificar a entrada:** O problema recebe um número inteiro positivo de até quatro dígitos.
2. **Determinar a quantidade de dígitos:** Um número pode ter 1, 2, 3 ou 4 dígitos, e essa informação será utilizada nos cálculos.
3. **Separar os dígitos do número:** Extraí-los individualmente para realizar os cálculos necessários.
4. **Elevar cada dígito à potência correspondente ao número total de dígitos:**
    - 4.1. Se o número tem 2 dígitos, cada dígito deve ser elevado ao quadrado.
    - 4.2. Se tem 3 dígitos, cada dígito deve ser elevado ao cubo.
    - 4.3 Se tem 4 dígitos, cada dígito deve ser elevado à quarta potência.
5. **Somar os resultados obtidos:** Somar os valores obtidos na etapa anterior.
6. **Comparar o resultado com o número original:** Se forem iguais, o número é um número de Armstrong.
7. **Exibir o resultado ao usuário.**

Exercício 3

Crie um programa em Java para ler um vetor VET do tipo inteiro com 20 posições, onde podem existir elementos repetidos. Gere um vetor VET2 ordenado a partir do vetor VET e que terá apenas os elementos não repetidos.

INSTITUTO FEDERAL
São Paulo
Campus Caraguatatuba

PASSOS PARA RESOLUÇÃO:

1. **Identificar a entrada:** O problema fornece um vetor de 20 números inteiros, podendo haver elementos repetidos.
2. **Criar um segundo vetor para armazenar os números únicos:** Esse vetor será preenchido apenas com valores que ainda não foram adicionados.
3. **Percorrer o vetor original:** Para cada elemento do vetor original, verificar se ele já está presente no novo vetor.
4. **Se o número não estiver no novo vetor, adicioná-lo:** Isso garante que não haja repetição no segundo vetor.
5. **Ordenar o vetor resultante:** Após eliminar os elementos duplicados, aplicar um método de ordenação.
6. **Exibir o vetor final ao usuário.**

Exercício 4

INSTITUTO FEDERAL
São Paulo
Campus
Caraguatatuba

Praticando...

Recuperação

INSTITUTO FEDERAL
São Paulo

Campus
Caraguatatuba