Tinnova
software on demand

QUESTÕES PARA AVALIAÇÃO

Desenvolvedores C++

Rua Jurema Vieira Medrado, 88 | Sala 405 | São José dos Campos-SP | CEP 12246-180
(12) 3322-4656 | www.tinnova.com.br

O que devo fazer?

1. Crie um projeto em sua conta GitHub
2. Implemente os desafios descritos nos tópicos abaixo
3. Desenvolva utilizando a linguagem C++
4. Faça um push para seu repositório com os desafios implementados
5. Envie um e-mail para (recrutamento@tinnova.com.br) avisando que finalizou o desafio, adicionando a URL do seu projeto
6. Aguarde nosso contato

Exercício 1

Faça um programa C para calcular o número de lâmpadas 60 watts necessárias para um determinado cômodo. O programa deverá ler um conjunto de informações, tais como: tipo, largura e comprimento do cômodo. O programa termina quando o tipo de cômodo for igual -1. A tabela abaixo mostra, para cada tipo de cômodo, a quantidade de watts por metro quadrado.

Dica: Use uma estrutura **struct** para agrupar logicamente as informações de um comodo (int tipo de comodo, float largura e float altura). Usar uma função (float CalulaArea) para calcular a área do cômodo. Os atributos de entrada serão a largura e comprimento do cômodo. Usar uma função (float Lampada) para calcular a quantidade de lâmpadas necesárias para o cômodo. Os atributos de entrada serão o tipo de cômodo e a metragem (em m^2) do cômodo.

Obs: Utilize a função **ceil(numero)** em **#include math.h** para realizar o arrendondamento para cima.

Tipo Cômodo	Potência (watt/m^2)
0	12
1	15
2	18
3	20
4	22

Exercício 2

Em uma competição de salto em distância cada atleta tem direito a cinco saltos. No final da série de saltos de cada atleta, o melhor e o pior resultados são eliminados.

O seu resultado fica sendo a média dos três valores restantes. Você deve fazer um programa que receba o nome e as cinco distâncias alcançadas pelo atleta em seus saltos e depois informe a média dos saltos conforme a descrição acima informada (retirar o melhor e o pior salto e depois calcular a média).

Faça uso de uma lista para armazenar os saltos. Os saltos são informados na ordem da execução, portanto não são ordenados. O programa deve ser encerrado quando não for informado o nome do atleta. A saída do programa deve ser conforme o exemplo abaixo: Atleta: Joãozinho da Silva

Exemplo
Primeiro Salto: 6.5 m Segundo Salto: 6.1 m Terceiro Salto: 6.2 m Quarto Salto: 5.4 m Quinto Salto: 5.3 m Melhor salto: 6.5 m Pior salto: 5.3 m Média dos demais saltos: 5.9 m Resultado final: Joãozinho da Silva: 5.9 m

Exercício 3

Crie um programa C que simplesmente siga os seguintes passos:

a) crie/abra um arquivo texto de nome "arq.txt",

b) permita que o usuario entre com diversos caracteres nesse arquivo, até que o usuario entre com o caractere '0' (fim da entrada de dados),

c) Feche o arquivo e abra novamente o arq.txt, e

d) lendo-o caractere por caractere, e escrevendo na tela (printf) todos os caracteres armazenados.

Tinnova
software on demand

Exercício 4

Imagine o seguinte vetor.

```
v = {5, 3, 2, 4, 7, 1, 0, 6}
```

Faça um algoritmo que ordene o vetor acima utilizando o **Bubble Sort**.

O Bubble Sort ordena de par em par. Ele pega os dois primeiros elementos e pergunta se o primeiro é maior que o segundo. Se sim, os elementos são trocados (swap), se não, são mantidos. Vai repetindo o processo até o final do vetor.

Obviamente que ele não consegue ordenar todo o vetor em uma única rodada, ele terá que passar pelo vetor um certo número de vezes.

De maneira mais formal podemos destacar:

1. Percorra o vetor inteiro comparando elementos adjacentes (dois a dois)
2. Troque as posições dos elementos se eles estiverem fora de ordem
3. Repita os dois passos acima (n - 1) vezes, onde n é igual ao tamanho do vetor

OK, vamos fazer um exemplo para facilitar o entendimento.

Voltemos ao nosso vetor.

```
5, 3, 2, 4, 7, 1, 0, 6
```

Sabemos que iremos repetir o vetor `n - 1` vezes. O tamanho do vetor é 8, logo iremos repetir 7 vezes o vetor (8-1).

Vamos chamar cada repetição de iteração.

Então, na **primeira iteração**, pegamos os dois primeiros valores e trocamos se estiverem fora de ordem.

```
(5  3) 2  4  7  1  0  6   pegamos o primeiro par
 3--5  2  4  7  1  0  6   trocamos

 3 (5  2) 4  7  1  0  6   pegamos o próximo par
 3  2--5  4  7  1  0  6   trocamos

 3  2 (5  4) 7  1  0  6   pegamos o próximo par
 3  2  4--5  7  1  0  6   trocamos

 3  2  4 (5  7) 1  0  6   pegamos o próximo par
 3  2  4  5--7  1  0  6   mantemos <----

 3  2  4  5 (7  1) 0  6   pegamos o próximo par
```

```
3  2  4  5  1--7  0  6   trocamos


3  2  4  5  1 (7  0) 6   pegamos o próximo par
3  2  4  5  1  0--7  6   trocamos


3  2  4  5  1  0  (7  6) pegamos último par
3  2  4  5  1  0   6  7  trocamos
```

Chegamos ao fim da primeira iteração e, como dito, não foi suficiente para ordenar o vetor.

Teremos que reiniciar, só que agora sabemos que, pelo menos, o último valor (7) já está em seu devido lugar

Então iremos marcá-lo e não precisaremos percorrer todo o vetor na segunda iteração.

```
3  2  4  5  1  0   6  [7]
```

Esse detalhe é importante é fará toda a diferença no entendimento do algoritmo.

Todo esse processo se repetirá até que todos os itens estejam devidamente ordenados.

Critérios de avaliação

- Facilidade de configuração do projeto
- Performance
- Código limpo e organização
- Documentação de código
- Documentação do projeto (readme)
- Boas práticas de desenvolvimento
- Design Patterns