1. **Criar um vetor de tamanho específico**: Comece criando um vetor chamado "tamanhoVetor" com um determinado tamanho que você escolher. Isso é basicamente como reservar espaço na memória para armazenar um certo número de valores.

2. **Inserir valores aleatórios no vetor sem repetições**: Agora, vamos preencher esse vetor com números aleatórios. Mas aqui está a pegadinha: os números não podem se repetir. Ou seja, cada número no vetor deve ser único.

3. **Imprimir os 20 primeiros valores do vetor para testar**: Para ter uma ideia de quais números estão no vetor, vamos imprimir os primeiros 20 valores.

4. **Utilizar o método de busca linear para procurar números aleatórios**: Vamos verificar se alguns números aleatórios estão ou não no vetor. A busca linear é um método simples: começamos no início do vetor e verificamos cada elemento até encontrar o número que estamos procurando ou chegarmos ao final do vetor.

5. **Aumentar gradualmente o tamanho do vetor para testar**: Repita os passos anteriores, mas aumente gradualmente o tamanho do vetor. Isso nos ajudará a entender como o desempenho é afetado à medida que o vetor cresce.

6. **Mudar para o método de busca binária e buscar valores aleatórios novamente**: Vamos alterar o método de busca binária e realizar a busca pelos mesmos valores aleatórios novamente.

7. **Implementar o cálculo de tempo de execução e verificar a velocidade da busca binária em relação à busca linear**: Finalmente, vamos medir o tempo que cada método leva para encontrar os valores no vetor. Para isso, usaremos a função que calcula o tempo de execução do algoritmo. Vamos comparar a busca binária com a busca linear para ver qual é mais rápida.

Perguntas Bônus:

1. **Existe diferença no tempo de execução em uma busca linear utilizando loop e busca linear utilizando recursão?

2. **Existe diferença no tempo de execução em uma busca binária implementada usando loop e uma busca binária utilizando recursão?

O exemplo abaixo contém um trecho que calcula o tempo de execução da função. Lembre-se de implementar a biblioteca time.h e as variáveis clock_t inicio, fim.

```c
#include <stdio.h>
#include <time.h>

void funcao_demorada() {
    // Simula uma operação demorada
    for (int i = 0; i < 1000000000; i++) {
        // Faz algo aqui, pode ser apenas uma operação de adição ou outra coisa
        int x = i + 1;
    }
}

int main() {
    clock_t inicio, fim;
    double tempo_decorrido;

    // Marca o tempo de início
    inicio = clock();

    // Chama a função demorada
    funcao_demorada();

    // Marca o tempo de fim
    fim = clock();

    // Calcula o tempo decorrido em segundos
    tempo_decorrido = ((double)(fim - inicio)) / CLOCKS_PER_SEC;

    printf("Tempo de execução: %.2f segundos\n", tempo_decorrido);

    return 0;
}
```