Roteiro para elaboração material de apoio

1- Colocar a descrição do problema e um exemplo de entrada e saída

Exemplo:

Faça um programa que tenha um menu com as opções para ler vários nomes de funcionários, imprimir os funcionários e finalizar o programa. A função de leitura dos funcionários só irá parar caso o nome digitado seja -1. O vetor de funcionários terá capacidade inicial igual a 5.

Entradas:

- Nome do funcionário (string).
- Comando (char).

Saídas:

- A média aritmética dos elementos (float).

Exemplos de Entradas e Saídas:

Entradas:

L
Joao
Julio
Rafael
Cleber
Melody
Amanda
-1
i
s

Saídas:

Capacidade de funcionarios: 10
Quantidade de funcionarios: 6

ID	NOME
0	Joao
1	Julio
2	Rafael
3	Cleber

4 Melody

5 Amanda

Finalizando!

2- SUGESTÃO DE SOLUÇÃO:

Ideia geral:

Construir um vetor alocado dinamicamente com ponteiros.

Passo 1: Armazenamento dos dados de entrada: O armazenamento dos dados dos funcionários será feito em um vetor de struct. Para podermos aumentar dinamicamente o vetor usaremos ponteiros.

```
int capacidade = 5;
int tamanho = 0;
funcionarios* vetor_de_funcionarios;
vetor_de_funcionarios = new funcionarios[capacidade];
```

Passo 2: Ler funcionários: A leitura de funcionários so deve ser parada quando o nome digitado for igual a "-1", para isso usaremos a estrutura de repetição while, que terá como condição (nome != '-1'). Usaremos dentro desse while uma condição para ver se o tamanho do vetor já é igual a capacidade ou tamanho. Caso seja, realocaremos o vetor dinamicamente.

```
funcionarios* lerFuncionarios( int& capacidade, int& tamanho, funcionarios* vetor_de_funcionarios ) {
   string nome;
   cin >> nome;
   while ( nome != "-1" ) {
      if ( tamanho >= capacidade ) {
                        vetor_de_funcionarios = RealocaVetorDeFuncionarios( capacidade, vetor_de_funcionarios );
      }
      vetor_de_funcionarios[tamanho].id = tamanho;
      vetor_de_funcionarios[tamanho].nome = nome;
```

```
        tamanho++;
        cin >> nome;
    }
    return vetor_de_funcionarios;
}
```

Passo 3: Realocação do vetor. Por fim, deve-se realizar a realocação do vetor, caso o tamanho atinja a capacidade. Para isso criamos uma função do tipo funcionarios que irá retornar o vetor com a nova capacidade. Dentro da função criaremos um vetor auxiliar para copiar os dados do nosso vetor principal. Logo após isso desalocamos nosso vetor principal e alocamos ele com a nova capacidade. Logo após copiamos os dados do vetor auxiliar para o nosso principal e retornamos o principal na função.

```
funcionarios* RealocaVetorDeFuncionarios( int& capacidade, funcionarios* vetor_de_funcionarios ) {
    funcionarios* vetor_auxiliar = new funcionarios[capacidade];
    for ( int i = 0; i < capacidade; i++ ) {
        vetor_auxiliar[i] = vetor_de_funcionarios[i];
    }
    delete[] vetor_de_funcionarios;
    vetor_de_funcionarios = new funcionarios[capacidade + 5];
    for ( int i = 0; i < capacidade; i++ ) {
        vetor_de_funcionarios[i] = vetor_auxiliar[i];
    }
    delete[] vetor_auxiliar;
    capacidade += 5;
    return vetor_de_funcionarios;
}
```

Passo 4: Impressão da saída de dados: o enunciado pede que a saída seja impressa na tela. Assim, o vetor é impresso.

```
void imprimirFuncionarios( int capacidade, int tamanho, funcionarios* vetor_de_funcionarios ) {
   cout << "\nCapacidade de funcionarios: " << capacidade
      << "\nQuantidade de funcionarios: " << tamanho
      << "\nID\tNOME\n";
   for ( int i = 0; i < tamanho; i++ ) {
      cout << vetor_de_funcionarios[i].id << "\t" << vetor_de_funcionarios[i].nome << endl;
   }
}
```

Exemplo de código:

```
#include <iostream>

using namespace std;

struct funcionarios {
    int id;
    string nome;
};



void imprimirFuncionarios( int capacidade, int tamanho, funcionarios* vetor_de_funcionarios ) {
    cout << "\nCapacidade de funcionarios: " << capacidade
        << "\nQuantidade de funcionarios: " << tamanho
        << "\nID\tNOME\n";
    for ( int i = 0; i < tamanho; i++ ) {
        cout << vetor_de_funcionarios[i].id << "\t" << vetor_de_funcionarios[i].nome << endl;
    }
}



funcionarios* RealocaVetorDeFuncionarios( int& capacidade, funcionarios* vetor_de_funcionarios ) {
    funcionarios* vetor_auxiliar = new funcionarios[capacidade];
    for ( int i = 0; i < capacidade; i++ ) {
        vetor_auxiliar[i] = vetor_de_funcionarios[i];
    }
    delete[] vetor_de_funcionarios;
    vetor_de_funcionarios = new funcionarios[capacidade + 5];
    for ( int i = 0; i < capacidade; i++ ) {
        vetor_de_funcionarios[i] = vetor_auxiliar[i];
```

```
    }
    delete[] vetor_auxiliar;
    capacidade += 5;
    return vetor_de_funcionarios;
}

funcionarios* lerFuncionarios( int& capacidade, int& tamanho, funcionarios* vetor_de_funcionarios ) {
    string nome;
    cin >> nome;
    while ( nome != "-1" ) {
        if ( tamanho >= capacidade ) {
                        vetor_de_funcionarios = RealocaVetorDeFuncionarios( capacidade, vetor_de_funcionarios );
        }
        vetor_de_funcionarios[tamanho].id = tamanho;
        vetor_de_funcionarios[tamanho].nome = nome;
        tamanho++;
        cin >> nome;
    }
    return vetor_de_funcionarios;
}


int main() {
    char comando;
    int capacidade = 5;
    int tamanho = 0;
    funcionarios* vetor_de_funcionarios;
    vetor_de_funcionarios = new funcionarios[capacidade];
    do {
        cin >> comando;
        switch ( comando ) {
            case 'L':
            case 'l':
```

```
                        vetor_de_funcionarios = lerFuncionarios( capacidade, tamanho,
vetor_de_funcionarios );
            break;

        case 'I':
        case 'i':
            imprimirFuncionarios( capacidade, tamanho, vetor_de_funcionarios );
            break;

        case 'S':
        case 's':
            cout << "Finalizando!";
            break;

        default:
            cout << "Opcao Invalida!\n";
            break;
      }
   } while ( comando != 's' );
   return 0;
}
```

Passo 5: Teste do programa:

<table>
<tr><th colspan="4">Menu</th></tr>
<tr><th>Capacidade</th><th>Tamanho</th><th colspan="2">Comando</th></tr>
<tr><td>5</td><td>0</td><td colspan="2">L</td></tr>
<tr><th colspan="4">lerFuncionarios()</th></tr>
<tr><th>Capacidade</th><th>Tamanho</th><th>vetor_de_funcionarios[ ]</th><th>Nome</th></tr>
<tr><td>5</td><td>0</td><td></td><td>Joao</td></tr>
<tr><td>#</td><td>1</td><td>vetor_de_funcionarios[0] =
{
id = 0;
nome = Joao;
}</td><td>Julio</td></tr>
<tr><td>#</td><td>2</td><td>vetor_de_funcionarios[1] =
{
id = 1;
nome = Julio;
}</td><td>Rafael</td></tr>
<tr><td>#</td><td>3</td><td>vetor_de_funcionarios[2] =
{
id = 2;
nome = Rafael;
}</td><td>Cleber</td></tr>
<tr><td>#</td><td>4</td><td>vetor_de_funcionarios[3] =
{
id = 3;
nome = Cleber;
}</td><td>Melody</td></tr>
<tr><td>#</td><td>5</td><td>vetor_de_funcionarios[4] =
{
id = 4;
nome = Melody;
}</td><td>Amanda</td></tr>
<tr><th colspan="4">RealocaVetorDeFuncionarios()</th></tr>
<tr><th>Capacidade</th><th>Contador</th><th>vetor_auxiliar[ ]</th><th>vetor_de_funcionarios[ ]</th></tr>
<tr><td>5</td><td></td><td></td><td></td></tr>
<tr><td></td><td>0</td><td>vetor_auxiliar[0] =
{
id = 0;
nome = Joao;
}</td><td>vetor_de_funcionarios[0] =
{
id = 0;
nome = Joao;
}</td></tr>
</table>

	1	vetor_auxiliar[1] =
		{
		id = 1;
		nome = Julio;
		}
			vetor_de_funcionarios[1] =
			{
			id = 1;
			nome = Julio;
			}

	2	vetor_auxiliar[2] =
		{
		id = 2;
		nome = Rafael;
		}
			vetor_de_funcionarios[2] =
			{
			id = 2;
			nome = Rafael;
			}

	3	vetor_auxiliar[3] =
		{
		id = 3;
		nome = Cleber;
		}
			vetor_de_funcionarios[3] =
			{
			id = 3;
			nome = Cleber;
			}

	4	vetor_auxiliar[4] =
		{
		id = 4;
		nome = Melody;
		}
			vetor_de_funcionarios[4] =
			{
			id = 4;
			nome = Melody;
			}

			delete[] vetor_de_funcionarios
			vetor_de_funcionarios = new funcionarios[capacidade+5]

5	0	vetor_auxiliar[0] =
		{
		id = 0;
		nome = Joao;
		}
			vetor_de_funcionarios[0] =
			{
			id = 0;
			nome = Joao;
			}

	1	vetor_auxiliar[1] =
		{
		id = 1;
		nome = Julio;
		}
			vetor_de_funcionarios[1] =
			{
			id = 1;
			nome = Julio;
			}

	2	vetor_auxiliar[2] =
		{
		id = 2;
		nome = Rafael;
		}
			vetor_de_funcionarios[2] =
			{
			id = 2;
			nome = Rafael;
			}

	3	vetor_auxiliar[3] =
		{
		id = 3;
		nome = Cleber;
		}
			vetor_de_funcionarios[3] =
			{
			id = 3;
			nome = Cleber;
			}

	4	vetor_auxiliar[4] =
		{
		id = 4;
		nome = Melody;
			vetor_de_funcionarios[4] =
			{
			id = 4;
			nome = Melody;

		}	}
capacidade += 5;		delete[] vetor_auxiliar;	return vetor_de_funcionarios;
lerFuncionarios()			
Capacidade	**Tamanho**	**vetor_de_funcionarios[ ]**	**Nome**

10	6	vetor_de_funcionarios[5] =
		{
		id = 5;
		nome = Amanda;
		}
			-1

10	6	return vetor_de_funcionarios;	
Menu			
Capacidade	**Tamanho**	**Comando**	
10	6	i	
ImprimirFuncionarios()			
Contador	**Tamanho**	**Cout:**	**vetor_de_funcionarios[ ]**
	6	Capacidade de funcionarios: 10 Quantidade de funcionarios: 6 ID NOME	

0		0 Joao	vetor_de_funcionarios[0] =
			{
			id = 0;
			nome = Joao;
			}

1		1 Julio	vetor_de_funcionarios[1] =
			{
			id = 1;
			nome = Julio;
			}

2		2 Rafael	vetor_de_funcionarios[2] =
			{
			id = 2;
			nome = Rafael;
			}

3		3 Cleber	vetor_de_funcionarios[3] =
			{
			id = 3;
			nome = Cleber;
			}

4		4 Melody	vetor_de_funcionarios[4] =
			{
			id = 4;

			nome = Melody; }

5		5 Amanda	vetor_de_funcionarios[5] =
			{
			id = 5;
			nome = Amanda;
			}

Menu			
Capacidade	Tamanho	Comando	
10	6	s	
cout: Finalizando!			