Atividades de Paradigmas de Programação
Programa de monitoria

Aluno Monitor: Victor Manoel Fernandes de Souza / Professor Orientador: Delair Martinelli

Vale lembrar que, por questões óbvias, o uso de STL é proibido por enquanto!!
É importante que você resolva a lista de exercícios para melhor absorção do conteúdo.

Para treinar a sintaxe, vamos fazer alguns exercícios simples

1 - implemente uma classe para representar números inteiros. essa classe deve calcular a raiz quadrada, a potência de 2 e calcular a soma entre esses números e retornar o objeto da soma. implemente Getter e setter.

2 - Construa uma classe para representação de uma função do segundo grau (ax^2+bx+c). Crie métodos para calcular a raiz da função e o vértice da parábola.

3- Implemente uma classe para a representação de pontos em duas dimensões (X,Y). Sua classe deve realizar as seguintes operações:

a) Alteração das coordenadas do ponto;
b) Translação do ponto;
c) Distância entre dois pontos.

4 - Utilizando a classe implementada do exercício 3, implemente as seguintes formas geométricas:

a) Triângulo. Esta classe deve calcular o ângulo de um vértice determinado, também deve calcular a área. Deve-se verificar se é um triângulo retângulo.
b) Círculo. Implemente métodos para: cálculo do raio, diâmetro, área.
c) Retângulo. Construa métodos para calcular o lado da figura. Calcule também quantos metros quadrados a mesma tem.

5- Implemente uma classe para representar matrizes, de ponto flutuante, 2X2. Implemente métodos para:

a) calcular o determinante
b) soma de duas matrizes;
c) multiplicação de duas matrizes;
d) cálculo de matriz inversa.

6 - Vamos imaginar que você foi contratado por uma empresa de games, e a mesma deseja que você faça um jogo estilo RPG, com os seguintes requisitos:

a)Todo jogo deve ter um personagem. Implemente uma classe para a representação do personagem principal. Ele deve ter um nome, nível, vida, mana, status base(ataque e defesa) e sua posição no mapa(2d). Implemente geters e seters para cada um desses atributos.

b) Um jogo sem inimigos não tem graça. Da mesma maneira, crie uma classe para a representação de inimigos. Seus inimigos deverão ter nome, vida, nível, poder de ataque e posição do mapa. Implemente geters e seters aqui também.

c) Agora vamos tratar do ambiente. Vamos implementar uma classe representando uma arena. Nessa classe iremos determinar o limite do mapa da arena. Nesse mapa temos um único personagem e vários inimigos(especifique um limite de inimigos no mapa). Faça com que ambos lutem entre si até todos os inimigos morrerem ou até o personagem morrer. OBS: Nenhum dos personagens podem estar na mesma coordenada.

7 - implemente uma classe para representar o sistema acadêmico da UEMS (SAU). A classe deve ter alunos e professores. Para tanto, implemente Classes que represente o professor e o aluno. O aluno deve ter seu nome, nome de usuário, senha, dados básicos, nota e matérias que ele cursa no ano. O professor deve ter seu nome, nome de usuário, senha, dados básicos, Matéria(s) que ele leciona. A Classe SAU deve ter dois métodos de login: um para aluno e outro para professor. Ambos os métodos devem receber por parâmetro o nome de usuário e a senha. No método para o professor, implemente uma listagem de alunos de acordo com a matéria que o professor leciona. Permita que o professor modifique a nota do aluno que ele deseja. No método aluno, permita a listagem das notas das matérias que o aluno cursa. Faça uma interface para a classe SAU. OBS: lembrando que não é recomendável printar mensagens dentro de uma classe, porém como estamos construindo uma aplicação para testar as classes professor e aluno, iremos fazer esse tipo de coisa nesse exercício em específico.

8 - Implemente uma classe para a representação de strings. a Classe deve admitir métodos para operações de concatenação (com sobrecarga do operador +), atribuição (com sobrecarga do operador =), modificação de caracteres (com sobrecarga do operador [ ]), cálculo do tamanho da string, verificar se a string é vazia, comparação de strings (com sobrecarga do operador ==). implemente um construtor de cópia para a classe.(similar à biblioteca string). não se esqueça do destrutor.

9 - Construa uma classe para armazenar um número decimal. Seus métodos devem retornar strings (da classe implementada no exercício anterior) desse número decimal para:

a) hexadecimal
b) octal
c) binário

OBS: Caso falte algo da classe string, volte para o exercício 8 e implemente e implemente.

10- Implemente uma classe capaz de armazenar oito bits. O único dado membro permitido é uma variável do tipo byte (ou char se preferir, pois são análogas). Implemente métodos para:

a) um construtor capaz de receber uma cadeia de caracteres (ou string) contendo uma sequência de 8 0's e 1's;
b) que retorne o i-ésimo bit (booleano – bool);
c) que modifique o i-ésimo bit (booleano – bool);
d) que retorne a paridade do byte (par ou ímpar de acordo com número de 1's nesse byte).

11 - Implemente um template para representar duas pilhas, não obstante elas deverão estar armazenadas no mesmo vetor. Caso exceda a memória, dobre seu tamanho e desloque a pilha mais a direita no final do vetor, sobrecarregue o operador [ ].

12 - Implemente um template para representar uma lista. Seu template terá fins didáticos. O template deve ter inserção e remoção de elementos, deverá também ordenar, e embaralhar aleatoriamente.Use o bubble Sort para ordenar. Agora implemente duas buscas: Uma busca varrendo todo vetor e outra uma busca binária. Caso tenha tempo use a biblioteca time.h para ver a diferença de tempo.

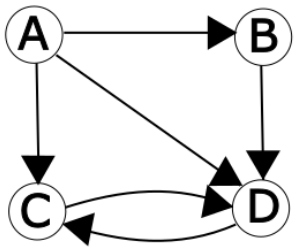
aqui o bixo pega!

13 - Construa um template para representar uma matriz nxn. sobrecarregue o operador [ ] para leitura e alteração dos elementos. Implemente um métodos para a multiplicação de matrizes, soma e cálculo do determinante.

14 - Escreva uma classe que represente um array ordenado (ordem crescente) de charInts, para um número qualquer de elementos. Um charInt é um tipo composto de dados que utiliza a mesma quantidade de bytes de uma variável inteira. No byte menos significativo de um charInt temos um objeto do tipo char e no restante um objeto do tipo int (com um byte a menos). A ordem de um charInt é dada pela parte inteira do objeto. Implemente métodos para a inserção ordenada, um para a exclusão por valor int e um para a exclusão por valor char. Escreva uma aplicação para testar as funcionalidades implementadas.

Restante	Byte menos significativo
Int com um byte a menos	char

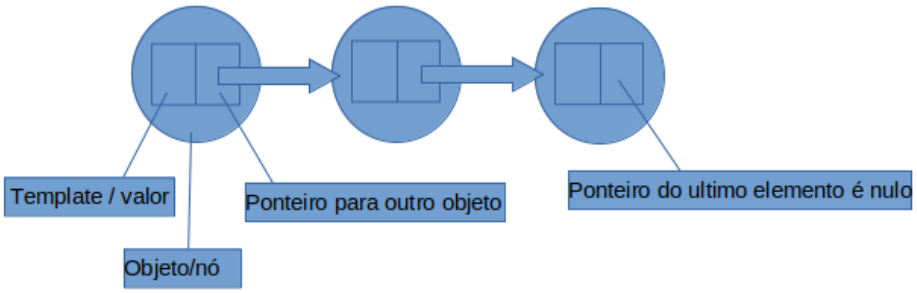
15 - Um grafo é uma estrutura matemática usada para representar um conjunto de objetos (chamados de vértices ou nós) e as conexões entre eles (chamados de arestas ou arcos). O grafo é composto por um conjunto finito de vértices e um conjunto de arestas, que são pares de vértices que representam uma conexão entre eles.Os grafos podem ser direcionados (onde as arestas têm uma direção) ou não-direcionados (onde as arestas não têm uma direção). Implemente uma classe para grafos direcionados. Um objeto da classe irá representar um vértice e as arestas são representadas por um vetor de ponteiros que deve ser membro dessa classe. Sua classe deverá ter um membro do tipo char que representa o nome do vértice. Tente não usar STL.



Grafo B

teste sua classe implementando esse grafo.

16 - Uma lista simplesmente encadeada é uma estrutura de dados linear composta por uma sequência de elementos, onde cada elemento é chamado de nó (ou nodo) e armazena um valor e uma referência (ou ponteiro) para o próximo nó na lista. Implemente um template para lista simplesmente encadeada com inserção e remoção em ambas as extremidades da lista. perceba que é semelhante ao exercício anterior.



17 - Use o exercício anterior como base: implemente uma fila e uma pilha usando listas encadeadas.

18 - Escreva classes para representar medidas de comprimento em metros e milhas. Você deve prover funcionalidades que permitam a conversão automática entre objetos destas classes (ambas as “direções” de conversão devem estar na mesma classe). Implemente, em uma das classes, a sobrecarga dos operadores de soma, subtração, incremento (pós e pré fixados) e operadores de comparação (==, !=, <, <=, > e >=). Obs: uma milha equivale a 1609,34 metros. Escreva uma aplicação para testar as funcionalidades implementadas.

Algoritmos para auxílio:

- **Algoritmo 2.4** **Busca binária**

```
função busca-bin(x)
    inf := 1;   sup := n;   busca- bin := 0
    enquanto inf ≤ sup faça
        meio := ⌊(inf + sup)/2⌋        % índice a ser buscado
        se L[meio] . chave = x então
            busca- bin := meio         % elemento encontrado
            inf := sup + 1
        senão se L[meio] . chave < x então
                inf := meio + 1
        senão sup := meio – 1
```

- **Algoritmo 7.1** **Ordenação bolha de uma tabela com n elementos**

```
para i = 1, … n faça
    para j = 1, … n – 1 faça
```

```
        se L[ j ] . chave > L[ j + 1] . chave então
            trocar(L[ j ], L[ j + 1])
```

Referência dos algoritmos acima:

Estrutura de dados e seus algoritmos, 3ª Edição.