2ª Lista de Exercício de Linguagens de Programação para Engenharia Elétrica – Turma B1

1) A função abaixo apresenta um erro. Aponte-o, sabendo que a chamada no programa principal foi esta: *funcao (&a, &b)*, feita após a declaração: *int a(3), b(5);*.

```
void funcao(int *p1, int *p2){
    int *aux;
    *aux = *p1;
    *p1 = *p2;
    *p2 = *aux;
}
```

2) Crie um função `float potencia(float r, int n)` que eleve o valor de *r* à *n-ésima* potência. Por exemplo, `166.375 = potencia(5.5, 3)`, `4.84 = potencia(2.2, 2)`.Rescreva esta função `potencia()` para que receba o endereço da variável da base. A nova função deve ter a seguinte declaração `void potencia(float *r, int n)` e deve retornar a solução na própria variável apontada por *r*. Reescreva a função passando o valor por referência `void potencia(float &r, int n)`.

3) Escreva uma função `void extremos(float vet, int N, float *max, float *min)` que receba um vetor de números reais `vet`, seu tamanho `N` e determine o maior e o menor (`max` e `min`) destes valores.

4) Escreva um programa que leia um vetor de `N` números inteiros, `(N<=100)`, inverta a ordem dos elementos do vetor e imprima o vetor invertido. Por exemplo o vetor: `{1, 3, 5, 7}` terá seus elementos invertidos: `{7, 5, 3, 1}`. **Observação:** É necessário inverter os elementos do vetor. Não basta imprimi-los em ordem inversa! Reescreva o programa para que, usando alocação dinâmica de memória, construa um vetor de `N` elementos, sendo `N` digitado pelo usuário.

5) Qual a diferença em C++ entre classe, estrutura e união?

6) Crie uma classe CComplex que permita fazer as seguintes operações com números complexos: (+, -, *, /). Sobrecarregue ainda operadores para permitir operações entre número complexo e número real. Crie um construtor que permita entrar com valores da parte real e imaginária do número complexo e funções que permitam mostrar o valor do complexo na forma retangular ou polar. Crie uma função main() que exemplifique o uso das operações.

7) Utilize o exemplo apresentado em aula da classe "formas" e acrescente as classes derivadas esfera e prisma que manipulem formas tridimensionais. Acrescente a essas classes funções para o cálculo do volume. Exemplifique o uso das classes no programa principal.

8) Implemente uma classe base abstrata chamada *polinomio* que contenha as seguintes funções: cria o polinômio *void criaPol(int grau, int *coef)*; calcula P(x) *float calcPx(float x)*; e uma função virtual pura chamada *metodo()*. Implemente então duas classes derivadas com métodos de cálculo das raízes do polinômio, o método da bissecção e de Newton, implementados através da função *metodo()*. Defina em cada classe, *bisseccao* e *newton*, as funções necessárias para executar cada método (leitura da inicialização, cálculo da derivada para o Newton, etc.). Finalmente, no programa principal use ponteiros da classe base para manipular polinômios e calcular uma raiz usando os dois métodos. Dicas: use o operador **new** para inicializar o ponteiro da classe base com um dos métodos das classes derivadas, como ilustrado no exemplo de polimorfismo da classe formas apresentado em aula; os métodos da bissecção e de Newton estão nas páginas 11 e 23 do arquivo "raízes de funções.pdf".

9) A pilha é uma estrutura de dados semelhante a uma pilha de objetos, em que o último a ser incluído é o primeiro a ser retirado ou o primeiro item empilhado é o último a ser retirado (mais informações em http://www.cos.ufrj.br/~rfarias/cos121/pilhas.html ou http://gauss.ececs.uc.edu/Courses/C321/html/stack++.html).Crie uma classe CPilha de números inteiros com as seguintes funções: criação da pilha, inclusão de elementos (empilhar ou *push*), retirada elementos (desempilhar ou *pop*), verificação se a pilha está vazia e verificação se está cheia. A criação da pilha deve receber o tamanho máximo da pilha e utilizar alocação dinâmica de memória. Crie uma função main() que leia um conjunto de valores inteiros e armazene os números pares numa lista e os ímpares em outra. Por exemplo:

```
CPilha piImp;
piImp.criaPilha(20);
cin >> i;
piImp.push(i); …
```

10) Modifique a classe anterior para ser uma pilha genérica, ou seja, que pode conter dados de qualquer classe, utilizando o recurso de Template.