LISTA 13 – 23/06

Questão 1

A programação orientada por objeto nada mais é do que uma programação baseada em classes, objetos e o relacionamento entre os dois. Os objetos são criados pois cada variável possui sua própria característica e comportamento, assim devemos tratar cada uma de forma específica, assim são criados conjuntos de operações para manipular esses dados. Mas os objetos devem também pertencer a uma classe, que é um conjunto de objetos para mesma estrutura de atributo, operação e relacionamento, dentro do contexto em que todos são inseridos.

Questão 2

Uma classe é um conjunto de objetos que partilham características comuns. Um objeto é uma instância de uma classe. Os objetos geralmente fazem alguma coisa e a interação com eles se dá através de mensagens que lhes são passadas, alguma ação que foi requisitada para o objeto realizar. A classe é um conjunto de objetos individuais que partilham características comuns e um comportamento comum. Herança é uma relação hierárquica entre as classes. A classe que possui as funcionalidades mais genéricas é geralmente chamada de classe base e a classe que especializa o comportamento da classe base é chamada de classe derivada.

```
#include <iostream>
using namespace std;
class Vetor2d{
private:
  float x, y;
public:
  void setX(float x_){
  x = x_;
  }
  float getX(void){
  return x;
  }
  void setY(float y_){
  y = y_;
  }
  float getY(void){
  return y;
```

```
  }
};
int main(void){
  Vetor2d v1;
  v1.setX(4.5);
  v1.setY(-1.3);
  cout << v1.getX() << " " << v1.getY() << "\n";
}
```

```
#include <iostream>
using namespace std;
class Base{
public:
  Base(){
    cout << "Construtor da classe Base\n";
  }
};
class Intermediaria : public Base{
public:
  Intermediaria(){
    cout << "Construtor da classe Intermediaria\n";
  }
};
class Derivada : public Intermediaria{
public:
  Derivada(){
    cout << "Construtor da classe Derivada\n";
  }
};
int main(void){
  Derivada d;
}
```

Questão 3

Encapsulamento: organizar os elementos de modo que sejam mostrados apenas o necessário sobre o comportamento dos objetos de uma determinada classe.

Polimorfismo: quando herança e ponteiros operam em conjunto, permitindo operações que relacionam classes derivadas diferentes usando ponteiros de um mesmo tipo ancestral.

Herança: é uma relação hierárquica entre as classes. A classe que possui as funcionalidades mais genéricas é geralmente chamada de classe base e a classe que especializa o comportamento da classe base é chamada de classe derivada.

Questão 4

Os construtores são funções especiais padronizadas que podem ser adicionadas a uma classe, eles permitem construir o estado inicial de um objeto, permitindo que seja definido todos os procedimentos necessários para inicializar o objeto. Existem 3 tipos de construtores:

- **Default**: normalmente é disponibilizado pelo compilador, na ausência de uma implementação fornecida pelo usuário, não possuem argumentos.
- **De cópia:** usado quando se deseja criar um novo objeto com as características de um outro que é passado como referência na forma de um parâmetro.
- **Qualquer outro:** são construtores com argumentos variados, fica a critério do programador especificar quantidade e tipos de argumentos que serão fornecidos ao construtor.

Questão 5

```
#include <iostream>
#include <string>
using namespace std;

class Aluno{
  private:
    string nome;
    int serie;
    int grau;
    static int count;
  public:
    Aluno(){
      nome="";
      serie=0;
      grau=0;
    }
    Aluno (string nome,int serie,int grau){
      this->nome=nome;
      this->serie=serie;
      this->grau=grau;
    }
    void setNome (string nome){
      this->nome=nome;
    }
    string getNome(){
      return nome;
    }
    void setSerie(int serie){
      this->serie=serie;
```

```
    }
    int getSerie(){
      return serie;
    }
    void setGrau(int grau){
      this->grau=grau;
    }
    int getGrau(){
      return grau;
    }
    void cadastrarAluno(){
      cout<<"Entre com o nome do aluno"<<endl;
      getline(cin,nome);
      cout<<"Entre com a serie do aluno"<<endl;
      cin>>serie;
      cout<< "Entre com o grau do aluno"<<endl;
      cin>>grau;
      count++;
    }
    void mostrarAluno(){
      cout<<"Nome: "<<nome<<"\t";cout<<"Serie: "<<serie<<"\t";
      cout<<"Grau: º"<<grau<<endl;
    }
    void conutAlunos(){
      cout<<"Numero de alunos cadastrados: "<<count<<endl;
    }
};
int Aluno::count=0;
void menu(){
  system("cls||clear");
  cout<<"1 - para cadastrar aluno"<<endl;
  cout<<"2 - para exibir os alunos cadastrados"<<endl;
  cout<<"3 - para sair"<<endl;
}
int main(){
  int opcao;
  Aluno *aluno[10];
  int qtde=0;
  menu();
  cin>>opcao;
  cin.ignore();
  while(opcao!=3){
    switch(opcao){
      case 1:
        aluno[qtde]=new Aluno();
        aluno[qtde]->cadastrarAluno();
        qtde++;
      break;
      case 2:
```

```
        for(int i=0;i<qtde;i++)
        aluno[i]->mostrarAluno();
      break;
    }
    getchar();
    menu();
    cin>>opcao;
    cin.ignore();
  }
  return 0;
}
```