UFPB
João Pessoa
Aluno: Daniel de Queiroz Cavalcanti
Matrícula:
Assunto: Resposta da Atividade 2.

1)

```
#include <iostream>  /* inclui diretivas  de     entrada-saída */
#include <cmath>  /*  inclui diretivas das funções matemáticas */

using namespace std;

#define  PI  3.14159

int main( )
{
    /* Definir variaveis */
       int Raio;
       float  Perim, Area;

    /* Obter Raio da circunferencia */
       cout << "Entre com o valor do raio: ";
       cin >> Raio;

    /* Calcular Perimetro do Circulo */
     Perim = 2 * PI * Raio;

    /* Calcular Area da Circunferencia */
       Area  = PI * pow(Raio, 2);

    /* Exibir Resultados */
       cout << "O perimetro da circunferencia de raio " << Raio
            << " eh  " << Perim << endl;
       cout << "e a area eh " << Area << endl;
}
```

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

int main( ){

  int lado1, lado2, lado3;
  int s1, s2, s3;

  cout << "Entre com o tamanho dos lados do triangulo: ";
```

```
  cin >> lado1 >> lado2 >> lado3;

  // calcula o quadrado dos lados
  s1 = lado1*lado1;
  s2 = lado2*lado2;
  s3 = lado3*lado3;

  // testa a condicao para um triangulo reto

  if ( lado1>0 && lado2>0 && lado3 > 0 ) {
    if (s1==s2+s3 || s2==s1+s2 || s2==s1+s3) ) {
    cout << "Triangulo reto!\n";
  }
  else {
    cout << "Nao pode ser um triangulo!\n";
  }
}

 // programa que verifica se 3 numeros podem ser os lados de um
     // triangulo reto.
     //
     #include <iostream>
     using namespace std;

     // funcao que calcula o quadrado de um numero
     int quadrado(int n)
     {
        return n * n;
     }


     int main()
     {
        int s1, s2, s3;

        cout << "Entre tres inteiros: ";
        cin >> s1 >> s2 >> s3;

        if ( s1 > 0 && s2 > 0 && s3 > 0 &&
           (quadrado(s1) + quadrado(s2) == quadrado(s3) ||
            quadrado(s2) + quadrado(s3) == quadrado(s1) ||
            quadrado(s3) + quadrado(s1) == quadrado(s2)) )
        {
             cout << " " << s1 << " " << s2 << " " << s3
                  << " podem formar um triangulo reto\n";
        }
        else
        {
             cout << " " << s1 << " " << s2 << " " << s3
                  << " nao podem formar um triangulo reto\n";
        }
}
```

```
/* programa que calcula o perímetro e a área de uma
   circunferência de raio R (fornecido pelo usuário) */

#include <iostream>  /* inclui diretivas  de     entrada-saída */
#include <cmath>  /*  inclui diretivas das funções matemáticas */

using namespace std;

#define  PI  3.14159

int main( )
{
    /* Definir variaveis */
       int Raio;
       float  Perim, Area;

    /* Obter Raio da circunferencia */
       cout << "Entre com o valor do raio: ";
       cin >> Raio;

    /* Calcular Perimetro do Circulo */
     Perim = 2 * PI * Raio;

    /* Calcular Area da Circunferencia */
       Area  = PI * pow(Raio, 2);

    /* Exibir Resultados */
       cout << "O perimetro da circunferencia de raio " << Raio
            << " eh  " << Perim << endl;
       cout << "e a area eh " << Area << endl;
}
```

QUESTÃO 2)

conta.h

```
#ifndef CONTA_H
#define CONTA_H
#include <iostream>
using namespace std;
#ifndef


class Iconta


 { public: //construtor virtual void sacar(double valor)=0;
virtual void depositar(double valor)=0;
virtual ~Iconta() { } };
class conta: public IConta
```

```
{ public: void nomeCliente(){
std::string getnomeCliente();
 } void salarioMensal(){ std::string setsalarioMensal();
} void numeroConta(){ std::string setsalarioMensal();
} void saldo(){ std::string setsaldo();
} };
//destrutor ~conta();
char cliente[80];
double limite;
 virtual void sacar(double valor);
virtual void depositar(double valor);
void imprimirSaldo();
void definirLimite();
 void salarioMensal::setdefinirLimite(std::string DefinirLimite){ this->DefinirLimite =
2*salarioMensal;
} private : double saldo; double salario;
}; #endif conta.cpp #include "conta.h" #include
// class's header file // class constructor conta::conta() {
saldo = 0;
// insert your code here } // class destructor conta::~conta() { } void conta :: sacar(double
valor)
 { saldo = saldo - valor;
} void conta :: depositar(double valor)
{ saldo = saldo + valor;
} void conta :: imprimirSaldo() {
printf("%lf", saldo);
}
```

ContaEspecial.cpp

```
#ifndef CONTA_H
#define CONTA_H
 #ifndef class ContaEspecial {
public: //construtor
contaEspecial::conta();
definirLimite() ;
public class definirLimite extends salarioMensal{
double salarioMensal;
double definirLimite;
public double getDefinirLimite() {
return this.salarioMensal * 3; } }
void salarioMensal::setdefinirLimite(std::string DefinirLimite){
this->DefinirLimite = 3*salarioMensal;
 } }
```

main.h

```
#include <iostream>
#include <cstdlib>
#include "conta.h"
using namespace std;
int main(int argc, char *argv[]) {
```

```
conta c;
ContaEspecial d;
c.depositar(60);
c.sacar(30);
c.imprimirSaldo();
c.definirLimite();
d.depositar();
d.sacar();
d.definirLimite();
system("PAUSE");
 return EXIT_SUCCESS;
}
```

QUESTÃO 3)

Funcionario.cpp

```
#include <iostream>
using namespace std;
class Funcionario {
public: virtual String nome() = 0;
std::string getnome();
virtual int matricula() = 0;
 std::string getmatricula();
int CalcularSalario(); // Superclasses devem ter o destrutor virtual

virtual ~Funcionario() { } };
```

Assalariado.cpp

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

class Assalariado {
public: double salario();
virtual double CalcularSalario()=0;
}
```

Comissionado.cpp

```
#include <iostream>
 using namespace std;
class Comissionado {
 public: double vendasSemanais();
double PercentualComissao();
virtual double calcularSalario()=0;
 }
```

main.h

```
#include <iostream>
```

```
#include <cstdlib>
#include "conta.h"
using namespace std;
int main(int argc, char *argv[])
{ Funcionario c;
 SistemaGerenciaFolha d;
Assalariado e;
Comissionado f;
c.nomedofuncionario();
c.matricula();
c.salario();
d.funcionarios();
d.CalculaValorTotalFolha();
d.ConsultaSalario();
e.salariof();
e.CalculaSalario();
f.vendas();
f.comissao();
f.calculaSalario();
system("PAUSE");
return EXIT_SUCCESS;
 }
```

QUESTÃO 4)

```
  class medico {
     char* nome;
  double altura;
  double peso;
public:
  char* nomedomedico();
  void altura(int num);
  void peso(int num1);
};

class Cirurgiao : public medico{
  char* especializacao;
  char* metodooufunc;
 public:
  void nomeespecializacao(char*);
  void metodooufuncao(char*);
  void mostrar();
};

class Oftalmologista : public medico{
  char* espec;
  char* metodo;
 public:
  void nomeesp(char*);
```

```
  void metodoouf(char*);
  void mostrar();
};

class Otorrino : public medico{
  char* especi;
  char* metodooufu;
 public:
  void nomeespe(char*);
  void metodooufu(char*);
  void mostrar();
};

class Ginecologista : public medico{
  char* especia;
  char* metodooufun;
 public:
  void nomeespec(char*);
  void metodooufun(char*);
  void mostrar();
};


main(){

Cirurgiao a1;
Oftalmologista a2;
Otorrino a3;
Ginecologista a4;
a1.altura();
a1.peso();
a1.nomeespecializacao();
a1.metodooufuncao();
a1.mostrar();
a2.nomeesp();
a2.metodoouf();
a2.mostrar();
a3.nomeespe();
a3.metodooufu()
a3.mostrar();
a4.nomeespec();
a4.metodooufun();
a4.mostrar();

}
```

QUESTÃO 5)

```
#include <iostream>
 #include <string>

 using namespace std;
```

```
class TestaValidaNumero{
  {
   public:
     int num;
     void  validaNumero(int num);
  void setvalidaNumero(int num);
  void getValorAbaixoException() const;
   void setValorAbaixoException() const;
  void Valor(string);

   void getValorAcimaException() const;
   void setValorAcimaException() const;
   void getValorMuitoAcimaException() const;
   void setValorMuitoAcimaException() const;
  };

  string num::getvalidaNumero() const
  { return num; }

  void num::setnum(int s)
  {
try
  {
    if (100>s.num() <1000)
       valor = "ValorAcimaException";
  }
if (s.num() <== 0)
       valor = "ValorAbaixoException";
  }
if (s.num() <== 1000)
       valor = "ValorMuitoAcimaException";
  }


  return 0;
 }
}
catch(void getValor(const num))
{


 {
  cout << "num: " << validaNumero.getvalidaNumero () << endl;
  cout << "validaNumero " << num.setvalidaNumero() << endl;
 }
}
```

QUESTÃO 6)

```
#include <iostream>
 #include <stdexcept> // std::invalid_argument
 #include <bitset>
 #include <string>
using namespace std;

class SaldoNaoDisponivelException
{
  int numero;     // São atributos
  string nome;    // privados por
  float SaldoNaoDisponivelException;    // default
public:
  void inicializa(string n, float s);
  void deposita(float valor);
  void consulta();
  int saque(float valor);
};
```

```
#include <iostream>
 #include <stdexcept> // std::invalid_argument
 #include <bitset>
 #include <string>
#include "exception.h"
int main (void)
 {
   try
   {

void Conta::inicializa(string n, float s)
{
  nome = n;
  SaldoNaoDisponivelException = s;
  if ( SaldoNaoDisponivelException < 0)
    cout << "Erro na Criação da Conta!!!" << endl;
}


void Conta::deposita(float valor)
{
  SaldoNaoDisponivelException = SaldoNaoDisponivelException + valor;
}
}
catch ( void Conta::consulta() )
   {
void Conta::consulta()
```

```
{
  cout << "Cliente: " << nome << endl;
  cout << "Saldo Atual: " << SaldoNaoDisponivelException << endl;
  cout << "Numero da Conta: " << numero << endl;
}

int Conta::saque(float valor)
{
  if ( SaldoNaoDisponivelException < valor)
    return 0;
  else
  {
    SaldoNaoDisponivelException = SaldoNaoDisponivelException - valor;
    return 1;
  }
}
}



#include <iostream>
 #include <stdexcept> // std::invalid_argument
 #include <bitset>
 #include <string>
#include "conta.h"


   try
   {
void main()
{
  Conta MinhaConta;
  Conta *OutraConta;

  MinhaConta. SaldoNaoDisponivelException = 10; // ERRO!!!
{
{
catch(MinhaConta.inicializa("Fulano", 10.25))
{
OutraConta->inicializa("Beltrano", 220.00);


  MinhaConta.deposita(12.75);
  MinhaConta.consulta();
  MinhaConta.saque(15.00);
  MinhaConta.consulta();

  OutraConta->consulta();
}
}
```

QUESTÃO 7)

```
int main(void) {
 void copiaConteudo(FILE *texte.txt, FILE *teste_bkp.txt); //texte.txt
FILE *texte.txt = fopen("tmp/exercicio.txt","r");
if (texte.txt == NULL)
{

cout<< ("Não foi possível abrir o arquivo");
return 1; } FILE *teste_bkp.txt = fopen("home/novo.txt","w");
 copiaConteudo(texte.txt,teste_bkp.txt);
fclose(texte.txt); fclose(teste_bkp.txt);
 return 0; } void copiaConteudo(FILE *texte.txt, FILE *teste_bkp.txt) {
string teste; teste = "Teste";
char ler[100];
while(fgets(ler,100,texte.txt) != NULL) if (ler.find("Teste")) { fputs("0/",texte.txt);
} else { fputs(ler, teste_bkp.txt);
 }
 }
```