Exercicio 1.

```C/C++
#include <iostream>
using namespace std;

int main () {
   cout << "Hello World";

   return 0;
}
```

Exercício 2.

C/C++

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

int main() {
   int inteiro;
   float flutuante;
   string texto;

   cout << "Digite um número inteiro: ";
   cin >> inteiro;

   cout << "Digite um número um ponto flutuante: ";
   cin >> flutuante;

   cin.ignore();

   cout << "Digite uma string: ";
   getline(cin, texto);

   cout << inteiro << " , " << flutuante <<" , " << texto <<
endl;

   return 0;
}
```

Exercício 3.

C/C++

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

class classe {
   private:
       int atr1;
   public:
       int getAtr1() {return this->atr1;};
       void setAtr1(int atr1) {this->atr1 = atr1;};
};

int main () {
   classe teste;
   teste.setAtr1(2);
   cout << teste.getAtr1() << endl;

   return 0;
}
```

Exercício 4.

```C/C++
#include <iostream>
using namespace std;

class Pessoa {
   private:
       int atr1;
   public:
       Pessoa(int atr1) {this->atr1 = atr1;}
       int getAtr1() {return atr1;}
       void setAtr1(int atr1) {this->atr1 = atr1;}
};

int main () {
   Pessoa jorge(12);
   cout << jorge.getAtr1() << endl;
   jorge.setAtr1(2);
   cout << jorge.getAtr1() << endl;

   return 0;
}
```

Exercício 5.

```C/C++
#include <iostream>
using namespace std;

class Person {
private:
   string text;
public:
   Person(string text = "Unicamp") {this->text = text;}

   string getText() {return text;}
   void setText(string text) {this->text = text;}
};

int main(){
   Person *estudante = new Person();
   cout << estudante->getText() << endl;
   Person *estudante1 = new Person("Fapesp");
   cout << estudante1->getText() << endl;
   delete estudante;
   delete estudante1;
   return 0;
}
```

Exercício 6.

C/C++

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

class Tabuada1 {
private:
 int atr;
public:
 Tabuada1(int atr) { this->atr = atr; }
 int getAtr() { return atr; }
 void setAtr(int atr) { this->atr = atr; }
 void soma() {
   for (int x = 1; x <= 10; x++)
     cout << atr << " + " << x << " = " << atr + x << endl;
 }
 void subtracao() {
   for (int x = 1; x <= 10; x++)
     cout << atr << " - " << x << " = " << atr - x << endl;
 }
 void multiplicacao() {
   for (int x = 1; x <= 10; x++)
     cout << atr << " * " << x << " = " << atr * x << endl;
 }
 void divisao() {
   if (atr == 0) {
     cout << "Erro divisão com zero" << endl;
   } else {
     for (int x = 1; x <= 10; x++)
       cout << atr << " / " << x << " = "
            << ((atr / x) > 0 ? to_string(atr / x) : "Erro número não
inteiro")
            << endl;
   }
 };
};

int main() {
 Tabuada1 *teste = new Tabuada1(0);
 teste->setAtr(2);
 teste->soma();
 teste->subtracao();
 teste->multiplicacao();
 teste->divisao();

 delete teste;
 return 0;
}
```

Exercício 7.

C/C++

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

enum OPERACAO {
   SOMA,
   SUBTRACAO,
   MULTIPLICACAO,
   DIVISAO,
};

class Tabuada2 {
   private:
       int n;
       OPERACAO op;
   public:
       Tabuada2 (int n = 1, OPERACAO op = SOMA) {
       this->n = n;
       this->op = op;
   }
       int getN () {return n;};
       void setN (int n) {this->n = n;}
       void setOp (OPERACAO op) {this->op = op;}
       OPERACAO getOp () {return op;}
       void Servico();
};

void Tabuada2::Servico () {
   if (op == DIVISAO && n == 0) {
       cout << "Erro impossível dividir por zero" << endl;
       return;
   }
   for (int x = 1; x <= 10; x++)
       switch (op) {
           case SOMA:
               cout << n << " + " << x << " = " << n + x << endl;
               break;
           case SUBTRACAO:
               cout << n << " - " << x << " = " << n - x << endl;
               break;
           case MULTIPLICACAO:
               cout << n << " * " << x << " = " << n * x << endl;
               break;
           case DIVISAO:
               cout << n << " / " << x << " = " << n / x << endl;
               break;
       }
```

```
}

int main () {
   Tabuada2 * tab = new Tabuada2(2, SUBTRACAO);
   tab->Servico();
   tab->setOp(MULTIPLICACAO);
   tab->setN(5);
   cout << tab->getN() << tab->getOp() << endl;
   tab->Servico();

   delete tab;

}
```

Exercício 8.

C/C++

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

class Forno {
private:
   float temperatura;
public:
   Forno(float temperatura = 0.0f) {
       setTemperatura(temperatura);
   }
   float getTemperatura() { return temperatura; }
   void setTemperatura(float temperatura) {
       if (!(temperatura > 280.0f || temperatura < 0.0f)) {
           this->temperatura = temperatura;
       } else {
           this->temperatura = 280.0f;
           cout << "Ajustado para o limite de 280°C" << endl;
       }
   }
   void getStatus() {
       cout << "Forno cozinhando a " << temperatura << " graus" << endl;
   }
};

class FornoEletrico : public Forno {
private:
   int potencia;
public:
   FornoEletrico(float temperatura, int potencia) : Forno(temperatura) {
       this->potencia = potencia;
   }
   void setPotencia(int potencia) { this->potencia = potencia; }
   int getPotencia() { return potencia; }

   void getStatus()  {
       cout << "Forno Elétrico trabalhando a " << getTemperatura() << " graus
a uma potência de " << potencia << " watts" << endl;
   }
};

class FornoResistivo : public FornoEletrico {
private:
   int resistencia;
public:
   FornoResistivo(float temperatura, int potencia, int resistencia) :
FornoEletrico(temperatura, potencia) {
```

```
        this->resistencia = resistencia;
    }

    void setResistencia(int resistencia) { this->resistencia = resistencia; }
    int getResistencia() { return resistencia; }

    void getStatus() {
        cout << "Forno Resistivo trabalhando a " << getTemperatura() << "
graus a uma potência de " << getPotencia() << " watts com resistência de " <<
resistencia << " ohms" << endl;
    }
};

class FornoInducao : public FornoEletrico {
private:
    float amperagem;
public:
    FornoInducao(float temperatura, int potencia, float amperagem) :
FornoEletrico(temperatura, potencia) {
        this->amperagem = amperagem;
    }

    void setAmperagem(float amperagem) { this->amperagem = amperagem; }
    float getAmperagem() { return amperagem; }

    void getStatus() {
        cout << "Forno a Indução trabalhando a " << getTemperatura() << "
graus a uma potência de " << getPotencia() << " watts com corrente de " <<
amperagem << " amperes" << endl;
    }
};

class FornoGas : public Forno {
private:
    float pressao;
public:
    FornoGas(float temperatura, float pressao) : Forno(temperatura) {
        this->pressao = pressao;
    }

    void setPressao(float pressao) { this->pressao = pressao; }
    float getPressao() { return pressao; }

    void getStatus() {
        cout << "Forno a Gás trabalhando a " << getTemperatura() << " graus a
uma pressão de " << pressao << " psi" << endl;
    }
};
```

```
class FornoGasNatural : public FornoGas {
private:
   float fluxoGas;
public:
   FornoGasNatural(float temperatura, float pressao, float fluxoGas) :
FornoGas(temperatura, pressao) {
        this->fluxoGas = fluxoGas;
   }

   void setFluxoGas(float fluxoGas) { this->fluxoGas = fluxoGas; }
   float getFluxoGas() { return fluxoGas; }

   void getStatus() {
        cout << "Forno a Gás Natural trabalhando a " << getTemperatura() << "
graus a uma pressão de " << getPressao() << " psi e um fluxo de " << fluxoGas
<< " l/s" << endl;
   }
};

class FornoPetroleo : public FornoGas {
private:
   float fluxoPetroleo;
public:
   FornoPetroleo(float temperatura, float pressao, float fluxoPetroleo) :
FornoGas(temperatura, pressao) {
        this->fluxoPetroleo = fluxoPetroleo;
   }

   void setFluxoPetroleo(float fluxoPetroleo) { this->fluxoPetroleo =
fluxoPetroleo; }
   float getFluxoPetroleo() { return fluxoPetroleo; }

   void getStatus() {
        cout << "Forno a Petróleo trabalhando a " << getTemperatura() << "
graus a uma pressão de " << getPressao() << " psi e um fluxo de " <<
fluxoPetroleo << " l/s" << endl;
   }
};

class FornoLenha : public Forno {
private:
   int qntTroncos;
public:
   FornoLenha(float temperatura, int qntTroncos) : Forno(temperatura) {
        this->qntTroncos = qntTroncos;
   }
   void getStatus() {
```

```
        cout << "Forno a Lenha trabalhando a " << getTemperatura() << " graus
com " << qntTroncos << " troncos" << endl;
    }
};

int main() {
    Forno* FornoClassico = new Forno(100.0f);
    cout << "Forno Clássico Temperatura Inicial: " <<
FornoClassico->getTemperatura() << endl;
    FornoClassico->setTemperatura(3000.0f);
    FornoClassico->getStatus();
    delete FornoClassico;

    FornoEletrico* FornoEletrico1 = new FornoEletrico(100.0f, 230);
    FornoEletrico1->getStatus();
    delete FornoEletrico1;

    FornoResistivo* FornoResistivo1 = new FornoResistivo(110.0f, 200, 30);
    FornoResistivo1->getStatus();
    delete FornoResistivo1;

    FornoInducao* FornoInducao1 = new FornoInducao(140.0f, 200, 30.0f);
    FornoInducao1->getStatus();
    delete FornoInducao1;

    FornoGas* FornoGas1 = new FornoGas(230.0f, 23.0f);
    FornoGas1->getStatus();
    delete FornoGas1;

    FornoGasNatural* FornoGasNatural1 = new FornoGasNatural(220.0f, 20.0f,
5.0f);
    FornoGasNatural1->getStatus();
    delete FornoGasNatural1;

    FornoPetroleo* FornoPetroleo1 = new FornoPetroleo(210.0f, 25.0f, 3.0f);
    FornoPetroleo1->getStatus();
    delete FornoPetroleo1;

    FornoLenha* FornoLenha1 = new FornoLenha(250.0f, 500);
    FornoLenha1->getStatus();
    delete FornoLenha1;

    return 0;
}
```

Exercício 9.

C/C++

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

class Potencia {
   private:
       double resultado;
   public:

   double calcula(int base, int expoente) {
       cout << "Método calcula(int, int) chamado.\n";
       resultado = pow(base, expoente);
       return resultado;
   }

   // 2. Base inteira e expoente real
   double calcula(int base, double expoente) {
       cout << "Método calcula(int, double) chamado.\n";
       resultado = pow(base, expoente);
       return resultado;
   }

   // 3. Base e expoente reais
   double calcula(double base, double expoente) {
       cout << "Método calcula(double, double) chamado.\n";
       resultado = pow(base, expoente);
       return resultado;
   }

};


int main () {
   Potencia *teste = new Potencia();
   cout << teste->calcula(2,2.5) << endl;
}
```

Exercício 10.

C/C++

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

class Forno {
private:
   float temperatura;
public:
   Forno(float temperatura = 0.0f) {
       setTemperatura(temperatura);
   }
   float getTemperatura() { return temperatura; }
   void setTemperatura(float temperatura) {
       if (!(temperatura > 280.0f || temperatura < 0.0f)) {
           this->temperatura = temperatura;
       } else {
           this->temperatura = 280.0f;
           cout << "Ajustado para o limite de 280°C" << endl;
       }
   }
   virtual ~Forno() {}

   virtual void getStatus() {
       cout << "Forno cozinhando a " << temperatura << " graus" << endl;
   }
};

class FornoEletrico : public Forno {
private:
   int potencia;
public:
   FornoEletrico(float temperatura, int potencia) : Forno(temperatura) {
       this->potencia = potencia;
   }
   void setPotencia(int potencia) { this->potencia = potencia; }
   int getPotencia() { return potencia; }

   void getStatus() {
       cout << "Forno Elétrico trabalhando a " << getTemperatura() << " graus
a uma potência de " << potencia << " watts" << endl;
   }
};

class FornoResistivo : public FornoEletrico {
private:
   int resistencia;
public:
```

```
    FornoResistivo(float temperatura, int potencia, int resistencia) :
FornoEletrico(temperatura, potencia) {
        this->resistencia = resistencia;
    }

    void setResistencia(int resistencia) { this->resistencia = resistencia; }
    int getResistencia() { return resistencia; }

    void getStatus() override {
        cout << "Forno Resistivo trabalhando a " << getTemperatura() << "
graus a uma potência de " << getPotencia() << " watts com resistência de " <<
resistencia << " ohms" << endl;
    }
};

class FornoInducao : public FornoEletrico {
private:
    float amperagem;
public:
    FornoInducao(float temperatura, int potencia, float amperagem) :
FornoEletrico(temperatura, potencia) {
        this->amperagem = amperagem;
    }

    void setAmperagem(float amperagem) { this->amperagem = amperagem; }
    float getAmperagem() { return amperagem; }

    void getStatus() override {
        cout << "Forno a Indução trabalhando a " << getTemperatura() << "
graus a uma potência de " << getPotencia() << " watts com corrente de " <<
amperagem << " amperes" << endl;
    }
};

class FornoGas : public Forno {
private:
    float pressao;
public:
    FornoGas(float temperatura, float pressao) : Forno(temperatura) {
        this->pressao = pressao;
    }

    void setPressao(float pressao) { this->pressao = pressao; }
    float getPressao() { return pressao; }

    void getStatus() override {
        cout << "Forno a Gás trabalhando a " << getTemperatura() << " graus a
uma pressão de " << pressao << " psi" << endl;
```

```
    }
};

class FornoGasNatural : public FornoGas {
private:
    float fluxoGas;
public:
    FornoGasNatural(float temperatura, float pressao, float fluxoGas) :
FornoGas(temperatura, pressao) {
        this->fluxoGas = fluxoGas;
    }

    void setFluxoGas(float fluxoGas) { this->fluxoGas = fluxoGas; }
    float getFluxoGas() { return fluxoGas; }

    void getStatus() override {
        cout << "Forno a Gás Natural trabalhando a " << getTemperatura() << "
graus a uma pressão de " << getPressao() << " psi e um fluxo de " << fluxoGas
<< " l/s" << endl;
    }
};

class FornoPetroleo : public FornoGas {
private:
    float fluxoPetroleo;
public:
    FornoPetroleo(float temperatura, float pressao, float fluxoPetroleo) :
FornoGas(temperatura, pressao) {
        this->fluxoPetroleo = fluxoPetroleo;
    }

    void setFluxoPetroleo(float fluxoPetroleo) { this->fluxoPetroleo =
fluxoPetroleo; }
    float getFluxoPetroleo() { return fluxoPetroleo; }

    void getStatus() override {
        cout << "Forno a Petróleo trabalhando a " << getTemperatura() << "
graus a uma pressão de " << getPressao() << " psi e um fluxo de " <<
fluxoPetroleo << " l/s" << endl;
    }
};

class FornoLenha : public Forno {
private:
    int qntTroncos;
public:
    FornoLenha(float temperatura, int qntTroncos) : Forno(temperatura) {
        this->qntTroncos = qntTroncos;
```

```
    }
    void getStatus() override {
        cout << "Forno a Lenha trabalhando a " << getTemperatura() << " graus
com " << qntTroncos << " troncos" << endl;
    }
};

int main() {
    Forno* fornos[8];
    fornos[0] = new Forno(100.0f);
    fornos[1] = new FornoEletrico(120.0f, 250);
    fornos[2] = new FornoResistivo(150.0f, 300, 50);
    fornos[3] = new FornoInducao(180.0f, 350, 15.5f);
    fornos[4] = new FornoGas(200.0f, 20.0f);
    fornos[5] = new FornoGasNatural(220.0f, 18.0f, 2.5f);
    fornos[6] = new FornoPetroleo(250.0f, 22.0f, 3.0f);
    fornos[7] = new FornoLenha(200.0f, 100);

    for (Forno * forno : fornos){
        forno->getStatus();
    }
    for (Forno * forno : fornos){
        delete forno;
    }

    return 0;
}
```