6º Laboratório ECOP13A - Herança – 07 de outubro 2022

1ª Construir as classes para representar uma hierarquia Politico / Presidente / Governador / Prefeito, conforme discutida em sala de aula.

- Acrescente uma função Imprime() em cada uma das classes.
- No construtor de cada classe, acrescente mensagens de depuração para saber por onde o programa está passando enquanto é executado.
- Utilize as funções definidas nas classes bases dentro das classes derivadas.

2ª Crie uma hierarquia de classes para representar a hierarquia Ponto/Circulo/Cilindro, considere que o Cilindro *é um* Circulo com altura diferente de zero e que o Circulo *é um* Ponto com raio diferente de zero. Além dos construtores, métodos de acesso, operadores de leitura (>>) e impressão (<<), implemente as funções area() e volume() para a hierarquia.

3ª Utilizar a classe polinômio do laboratório 5 para implementar uma função que encontre pelo menos uma raiz real dele, se ela existir utilizando o método de Newton:

Para encontrar uma raiz real de um polinômio $p(x)=a_0+a_1x+...+a_nx^n$, ($n \geq 2$), pode-se aplicar o método de Newton, que consiste em refinar uma aproximação inicial x_0 dessa raiz através da expressão:

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)}$$

onde:
n= 0,1,2,...,
$p'(x)$ é a primeira derivada de $p(x)$.

- Usualmente, repete-se esse refinamento até que $|x_{n+1} - x_n| < \varepsilon$, $\varepsilon \geq 0$, ou até que m iterações tenham sido executadas.

Implemente na classe Polinômio as seguintes funções:

- Dado um polinômio $p(x)$, calcule e retorne a sua derivada $p'(x)$.
- Dado um polinômio $p(x)$, calcule seu valor em um ponto. Utilize essa função para calcular $p(x_n)$ e $p'(x_n)$ em cada iteração.
- Dado um polinômio $p(x)$, uma aproximação inicial x_0 e o número máximo m de iterações que devem ser executadas, calcule uma raiz real pelo método de Newton, se ela existir.

4ª Analisar a classe de exemplo do Livro do Deitel que representa um número de telefone formatado, e alterar essa classe para que funcione com o formato utilizado no Brasil.

```
// Fig. 11.3: PhoneNumber.h
// PhoneNumber class definition
#ifndef PHONENUMBER_H
#define PHONENUMBER_H

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

```
class PhoneNumber
{
   friend ostream &operator<<( ostream &, const PhoneNumber & );
   friend istream &operator>>( istream &, PhoneNumber & );
private:
   string areaCode; // 3-digit area code
   string exchange; // 3-digit exchange
   string line; // 4-digit line
}; // end class PhoneNumber

#endif
```

```
// Fig. 11.4: PhoneNumber.cpp
// Overloaded stream insertion and stream extraction operators
// for class PhoneNumber.
#include <iomanip>
#include "PhoneNumber.h"
using namespace std;

// overloaded stream insertion operator; cannot be
// a member function if we would like to invoke it with
// cout << somePhoneNumber;
ostream &operator<<( ostream &output, const PhoneNumber &number )
{
   output << "(" << number.areaCode << ") "
      << number.exchange << "-" << number.line;
   return output; // enables cout << a << b << c;
} // end function operator<<

// overloaded stream extraction operator; cannot be
// a member function if we would like to invoke it with
// cin >> somePhoneNumber;
istream &operator>>( istream &input, PhoneNumber &number )
{
   input.ignore(); // skip (
   input >> setw( 3 ) >> number.areaCode; // input area code
   input.ignore( 2 ); // skip ) and space
   input >> setw( 3 ) >> number.exchange; // input exchange
   input.ignore(); // skip dash (-)
   input >> setw( 4 ) >> number.line; // input line
   return input; // enables cin >> a >> b >> c;
} // end function operator>>
```

```
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

```
// Fig. 11.5: fig11_05.cpp
// Demonstrating class PhoneNumber's overloaded stream insertion
// and stream extraction operators.
#include <iostream>
#include "PhoneNumber.h"
using namespace std;

int main()
{
   PhoneNumber phone; // create object phone

   cout << "Enter phone number in the form (123) 456-7890:" << endl;

   // cin >> phone invokes operator>> by implicitly issuing
   // the global function call operator>>( cin, phone )
   cin >> phone;

   cout << "The phone number entered was: ";

   // cout << phone invokes operator<< by implicitly issuing
   // the global function call operator<<( cout, phone )
   cout << phone << endl;
} // end main

/**************************************************************************
 * (C) Copyright 1992-2010 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```