UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
LABORATÓRIO DE ENGENHARIA E EXPLORAÇÃO DE PETRÓLEO

CORRELAÇÕES PVT

ALEXSANDRO DA SILVA ALVES
ALINE KARL ARAUJO

MACAÉ - RJ
JANEIRO - 2014

Sumário

Capítulo 1

Introdução

No presente trabalho desenvolve-se um software contendo diversos métodos para estimar as principais propriedades dos fluidos presentes em um reservatório, referentes aos fluidos oléo e gás. Para a criação deste banco de funções foi necessária a pesquisa das melhores correlações existentes em livros, boletins tecnológicos e papers .

1.1 Escopo do Problema

- A objetivo do programa é criar um banco de correlações PVT e desta forma poder simular o comportamento das propriedades dos fluidos dentro de um reservatório.
- O banco de correlações é composto por diversas funções implementadas com variados métodos existentes na literatura.

1.2 Objetivos

Os objetivos deste trabalho são:

- Objetivo geral:
  - Criar um banco virtual de correlações onde seja possível estimar diversas propriedades relacionadas ao gás e ao óleo presentes em um reservatório. Visando a empregá-lo no ensino e pesquisa.
- Objetivos específicos:
  - Validar os códigos implementados com exemplos obtidos da literatura.
  - Simular o comportamento das propriedades dos fluidos óleo e gás presentes em um reservatório.

Capítulo 2

Especificação

Apresenta-se neste capítulo a concepção, a especificação do sistema a ser modelado e desenvolvido.

2.1 Especificação do programa - descrição dos requisitos

Para cada correlação é criada uma função que recebe um determinado número de parâmetros referentes ao fluido e um parâmetro no qual o usuário poderá escolher por qual método ele irá estimar a propriedade desejada.

2.2 Casos de uso do Programa

A Tabela 2.1 inclui um protótipo dos casos de uso para esse programa, mostrando quais são as etapas de interação entre o usuário e o programa.

O diagrama de casos de uso específico, especifica uma das ações do usuário. Neste caso, o usuário ao acessar o software "CORRELAÇÕES PVT", para calcular a propriedade de sua escolha, seleciona o fluido, a seguir, seleciona a propriedade que deseja calcular, caso haja mais de um método para cálculo também deverá selecionar o método com a qual deseja realizar o cálculo, em seguida entra com os dados necessários para o cálculo. Então, o software calcula e gera um valor numérico que é exibido via tela para o usuário. Este, agora, realiza comparações entre o valor gerado e o valor literário através da análise dos resultados gerados.

Tabela 2.1: Caso de uso.

Nome do caso de uso:	Correlações PVT dos fluidos óleo e gás
Resumo/descrição:	Cálculo de propriedades do gás e do óleo.
Etapas:	1. Definir o tipo de fluido. 2. Definir a propriedade que se deseja calcular. 3. Definir, se necessário, o método pelo qual se deseja calcular a propriedade. 4. Entrar com os dados, via teclado. 5. Receber resultado da propriedade, via tela.
Cenários alternativos:	Um cenário alternativo envolve uma entrada errada do usuário (por exemplo, entrar com dados negativos).

2.3 Diagrama de caso de uso geral do programa

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário acessando os sistemas do software e assim entrando com o valor dos atributos para cálculo das propriedades do fluido de escolha.

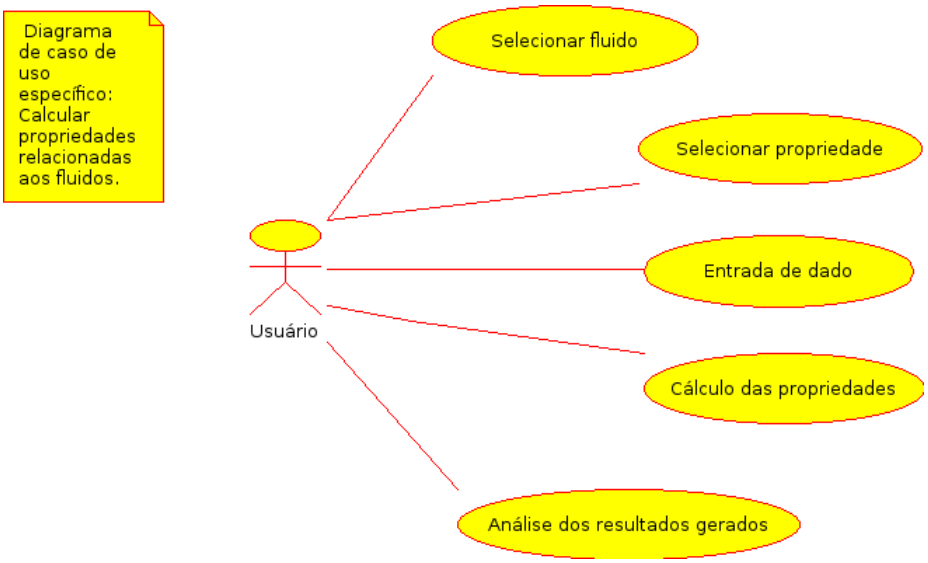

Figura 2.1: Diagrama de caso de uso – Caso de uso geral.

2.4 Diagrama de caso de uso específico do programa

O diagrama de caso de uso específico na Figura 2.2.

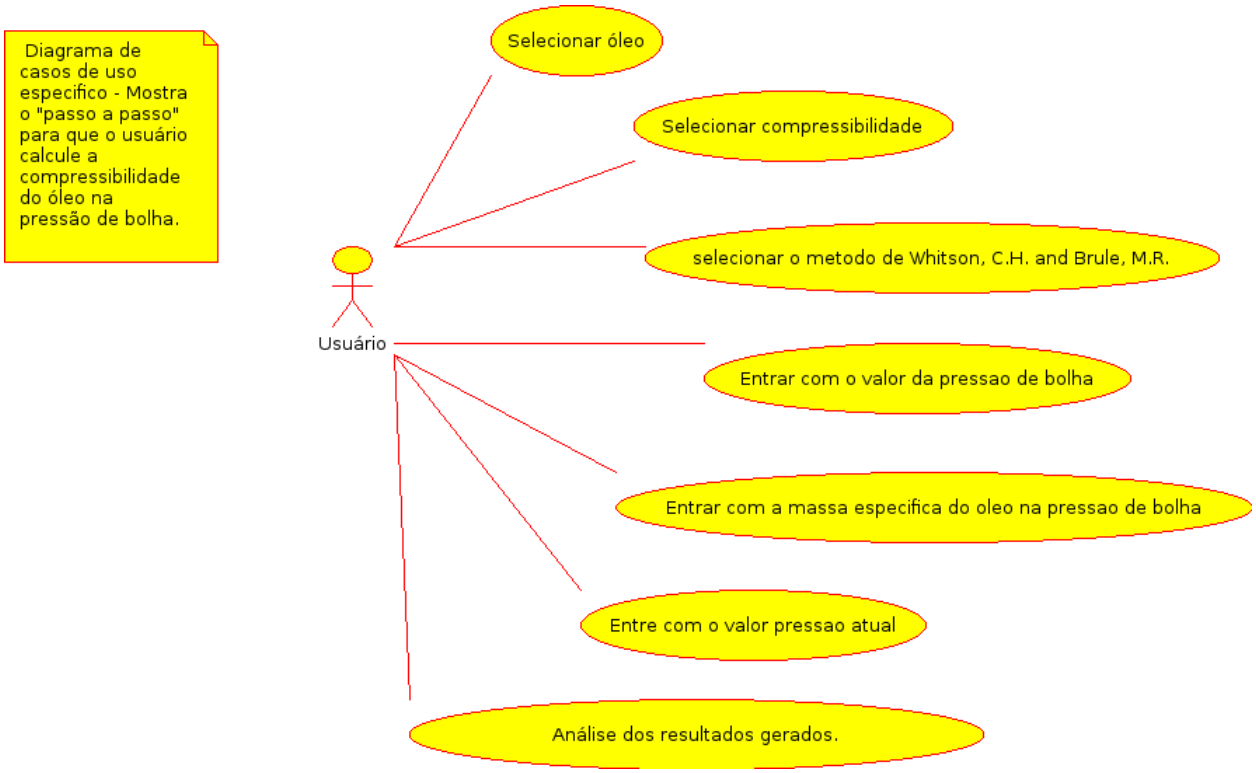

Figura 2.2: Diagrama de caso de uso – Caso de uso específico.

Capítulo 3

Elaboração

3.1 Análise de domínio

O sistema envolve conceitos de Engenharia de Reservatórios. Esses conceitos são vistos inicialmente na disciplina de Introdução à Engenharia de Reservatório e tem sido aprofundados nos estudos direcionados à Pesquisa.

As propriedades dos fluidos do reservatório, são os dados de entrada da maioria dos simuladores de reservatório, ferramenta esta que é utilizada para conhecer e prever o comportamento de um campo de petróleo, além disso, o conhecimento das propriedades dos fluidos do reservatório é um fator de grande importância no momento da criação e implementação de um projeto de produção de um campo.

As funções foram implementadas em Matlab 7.10.0 (R2010b), que apesar de ser um software proprietário, possui linguagem de fácil assimilação, e com pequenas adaptações, os scripts podem ser convertidos para serem utilizados em softwares livres.

Os resultados obtidos com as correlações implementadas serão comparados com os exemplos tirados da literatura, para validação dos códigos. Nas propriedades em que há mais de um método de correlação para se estimar o seu valor, os resultados serão também comparados entre si. Os resultados serão apresentados: pelo tipo de fluido (gás ou óleo) e dentro de cada tipo de fluido. Os resultados serão divididos por propriedades.

3.2 Identificação de pacotes – assuntos

- Propriedades Fluido: Englobam as classes de propriedades comuns entre os fluidos, óleo e gás, presentes em um reservatório. Como por exemplo: Compressibilidade, Fator volume formação e viscosidade.

- Propriedades Gas: Englobam as propriedades relacionadas somente ao gás. Como por exemplo: Propriedades pseudocríticas (temperatura e pressão pseudocríticas) e fator de compressibilidade Z.

- Propriedades OleoGás: Pacote que englobam as propriedades relacionadas concomitantemente ao óleo e ao gás. Como por exemplo: Pressão de bolha, Razão de solubilidade, Densidade dos gases em superfície e Massa específica do óleo.

3.3 Diagrama de pacotes – assuntos

O diagrama de pacotes a seguir apresenta os principais assuntos abordados no programa. Veja Figura 3.1.

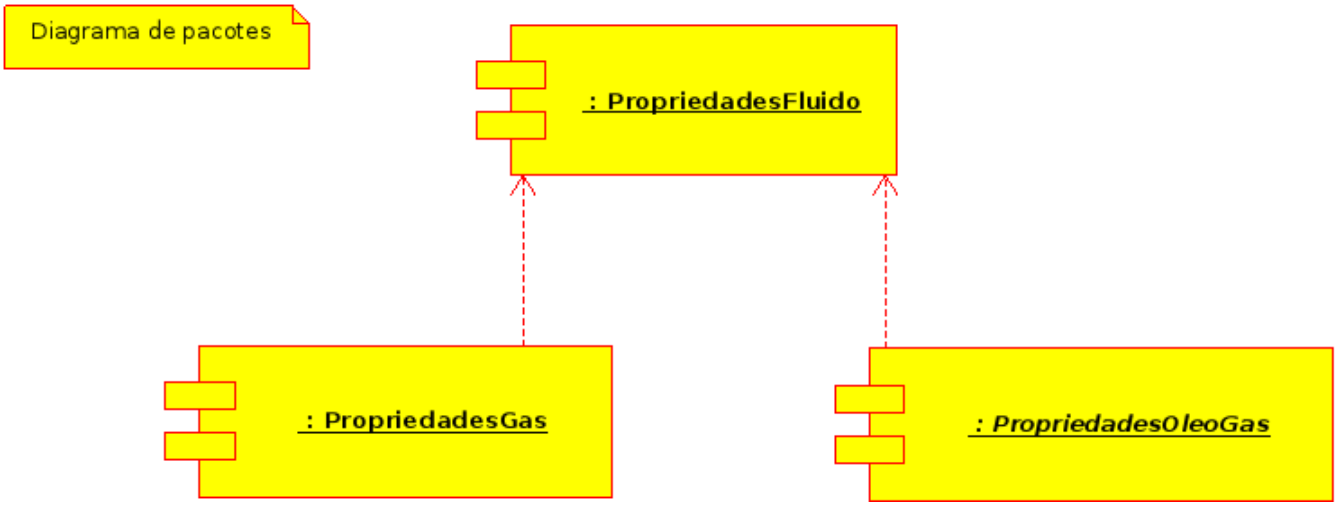

Figura 3.1: Diagrama de Pacotes - CORRELACOES PVT.

Capítulo 4

AOO – Análise Orientada a Objeto

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1 e Figura 4.2. Por ser muito grande, o mesmo foi particionado em 6 imagens que estão listadas abaixo.

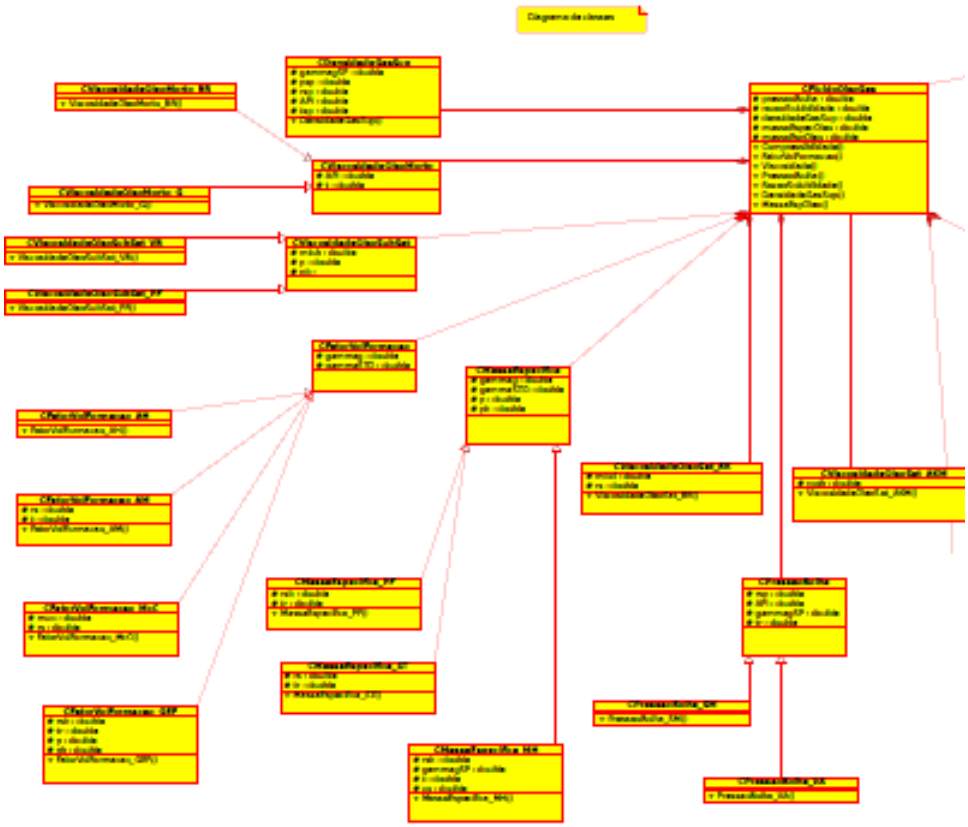

Figura 4.1: Diagrama de classes geral - Parte A.

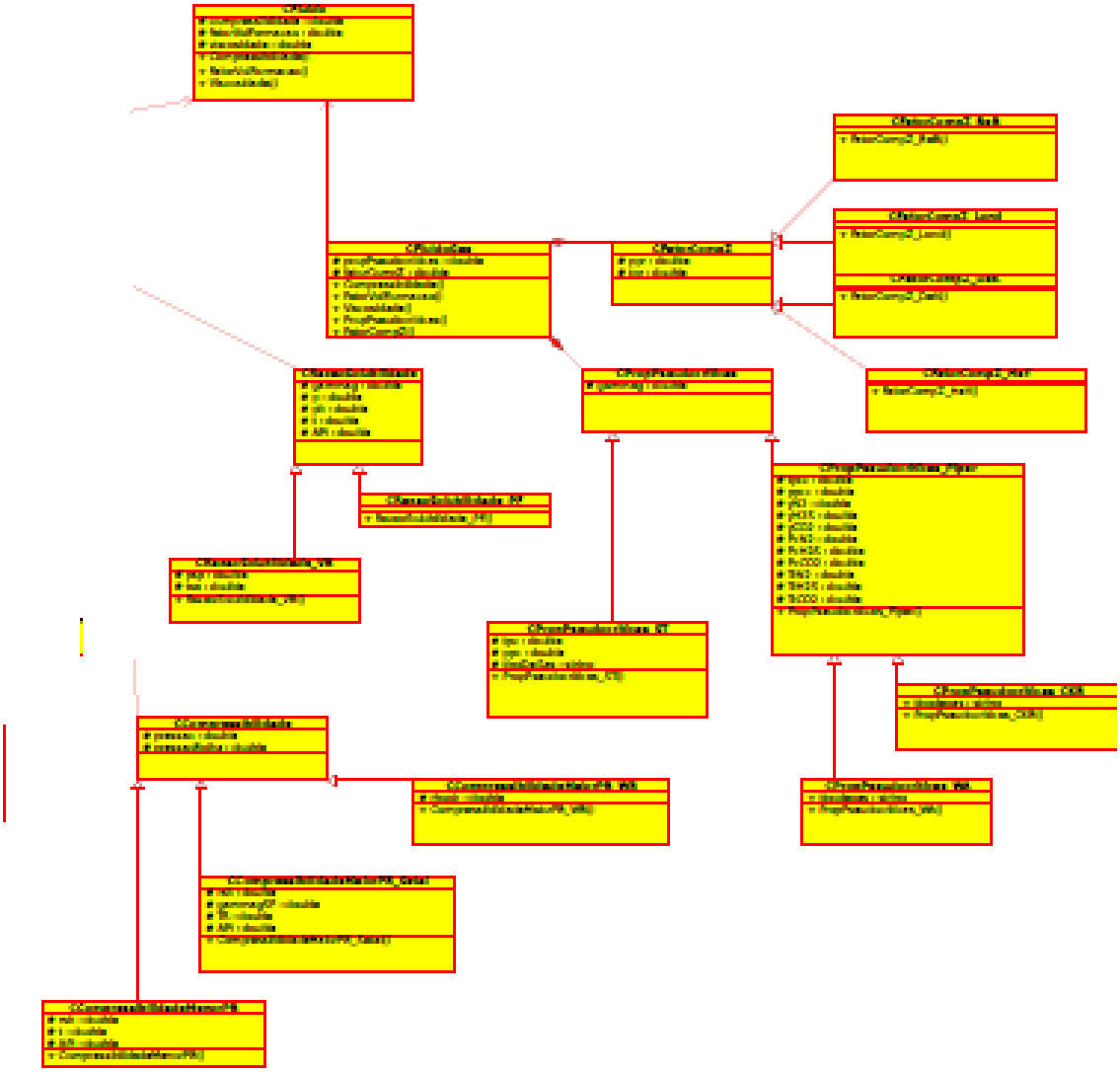

Figura 4.2: Diagrama de classes geral - Parte B.

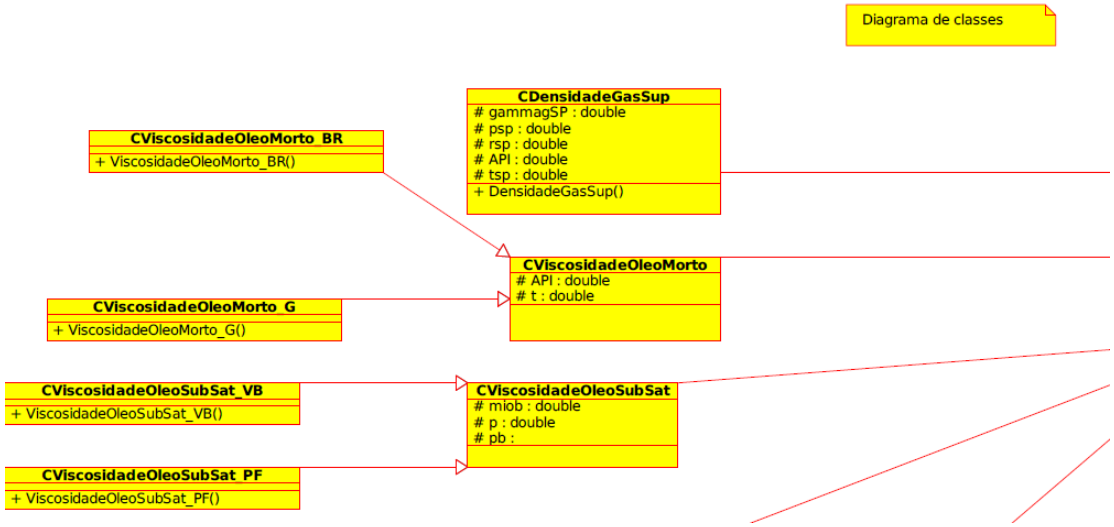

Figura 4.3: Diagrama de classes - Parte 1

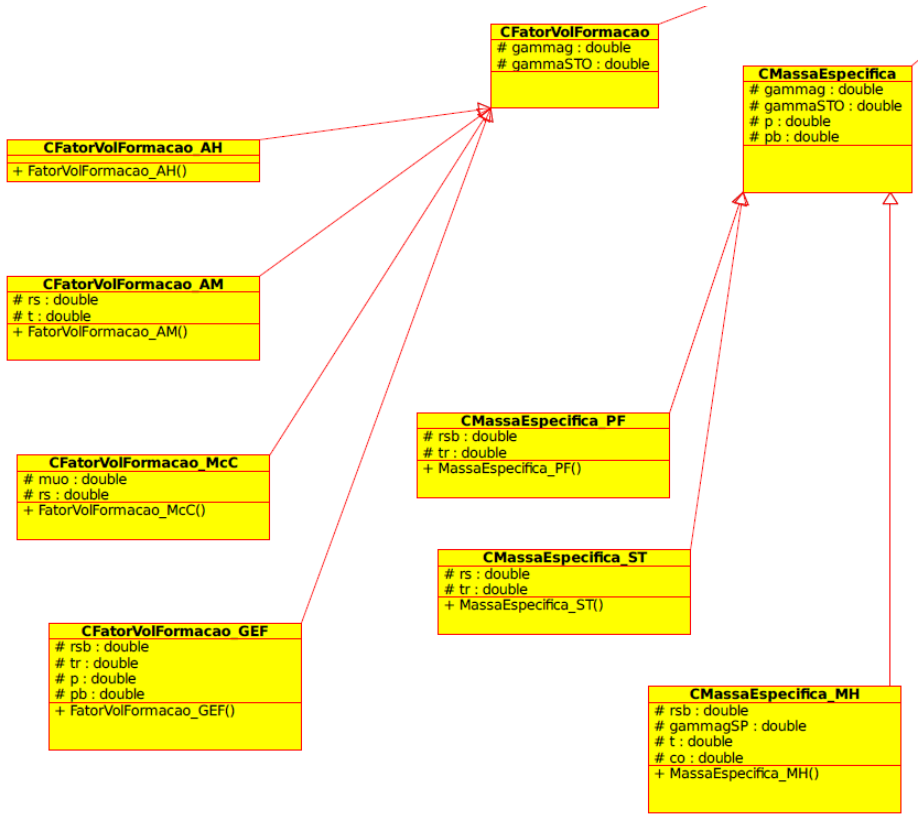

Figura 4.4: Diagrama de classes -Parte 2

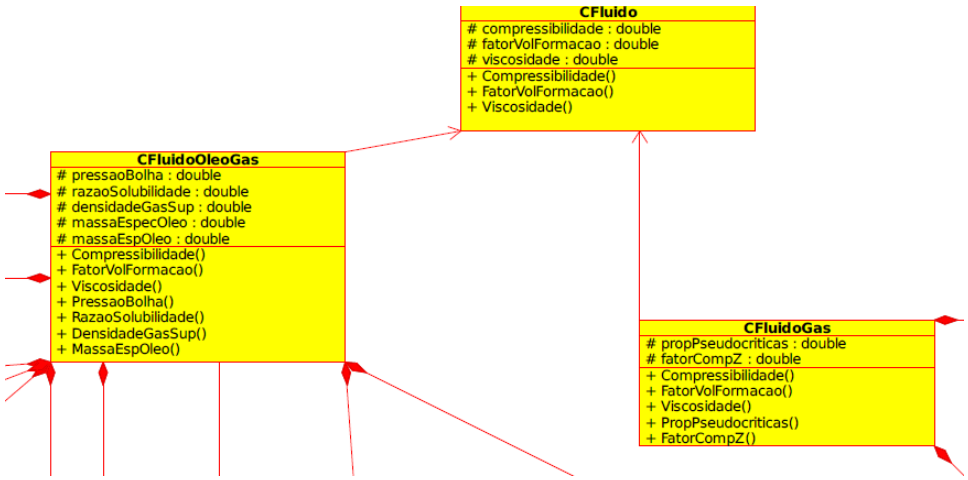

Figura 4.5: Diagrama de classes, CFluidoGas. Parte 3

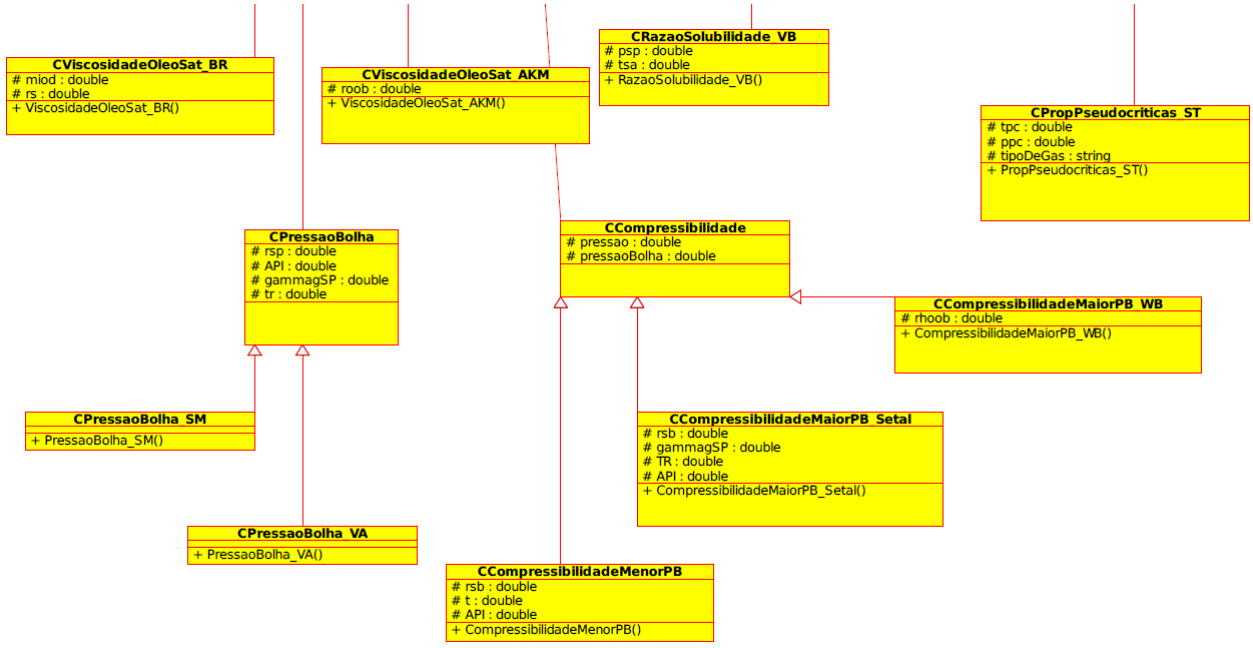

Figura 4.6: Diagrama de classes, CFluidoGas. Parte 4

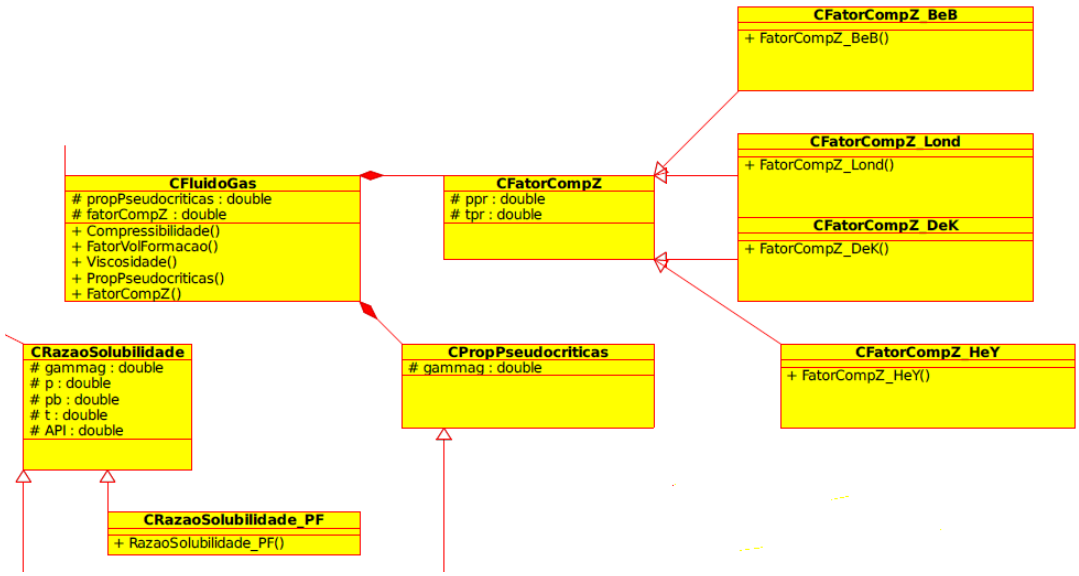

Figura 4.7: Diagrama de classes, CFluidoGas. Parte 5

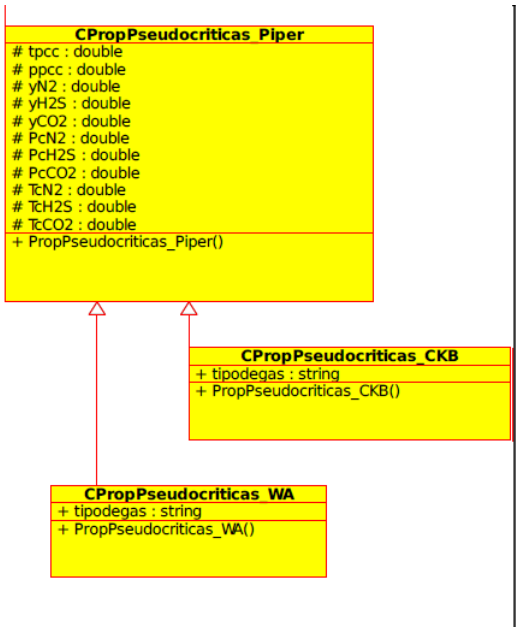

Figura 4.8: Diagrama de classes, CFluidoGas. Parte 6

4.1.1 Dicionário de classes

- Classe CFluido: representa a classe onde há atributos e métodos semelhantes tanto ao óleo quanto ao gás.
  - atributo compressibilidade: representa a compressibilidade tanto do óleo quanto do gás.
  - atributo fatorVolFormacao: representa o fator volume formação tanto do óleo quanto do gás.
  - atributo viscosidade: representa a viscosidade tanto do óleo quanto do gás.
  - método Compressibilidade: representa o método que calcula a compressibilidade ou do gás ou do óleo a partir dos dados fornecidos pelo usuário.
  - método FatorVolFormacao: representa o método que calcula o Fator Volume Formação ou do gás ou do óleo a partir dos dados fornecidos pelo usuário.
  - método Viscosidade: representa o método que calcula a Viscosidade ou do gás ou do óleo a partir dos dados fornecidos pelo usuário.
- Classe CFluidoOleoGas: representa a classe onde há atributos e métodos característicos ao óleo e ao gás presentes no reservatório e/ou na superfície.
  - atributo pressaoBolha: representa a pressão de bolha do óleo, (Psia).
  - atributo razaoSolubilidade: representa a razão de solubilidade do óleo.
  - atributo densidadeGasSup: representa a densidade do gás na superfície.
  - atributo massaEspOleo: representa a massa específica do óleo.
  - método Compressibilidade: representa o método que calcula a compressibilidade do óleo a partir dos dados fornecidos pelo usuário.
  - método FatorVolFormacao: representa o método que calcula o Fator Volume Formação do óleo a partir dos dados fornecidos pelo usuário.
  - método Viscosidade: representa o método que calcula a Viscosidade do óleo a partir dos dados fornecidos pelo usuário.
  - método PressaoBolha: representa o método que calcula a pressão de bolha do óleo a partir dos dados fornecidos pelo usuário.
  - método RazaoSolubilidade: representa o método que calcula a razão de solubilidade do óleo a partir dos dados fornecidos pelo usuário.
  - método DensidadeGasSup: representa o método que calcula a densidade do gás na superfície a partir dos dados fornecidos pelo usuário.
  - método MassaEspOleo: representa o método que calcula a massa específica do óleo a partir dos dados fornecidos pelo usuário.

- Classe CCompressibilidade: Classe agregada à classe CFluidoOleoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo da compressibilidade do óleo.
  - atributo pressao: representa a pressão do óleo, (Psia).
  - atributo pressaoBolha: representa a pressão de bolha do óleo, (Psia).
- Classe CCompressibilidadeMaiorPB_Setal: Classe herdeira da classe CCompressibilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da compressibilidade do óleo, acima da pressão de bolha, utilizando o método de Spivey, John e Valko, Peter e McCain, William.
  - atributo rsb: razão de solubilidade no ponto de bolha(scf/STB).
  - atributo gammagSP: representa a densidade do gás no separador.
  - atributo TR: representa a Temperatura do reservatório, (°F).
  - atributo API: densidade óleo, grau API, (°API).
  - método CompressibilidadeMaiorPB_Setal: calcula a compressibilidade do óleo, acima da pressão de bolha, utilizando o método de Spivey, John e Valko, Peter e McCain, William.
- Classe CCompressibilidadeMaiorPB_WB: Classe herdeira da classe CCompressibilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da compressibilidade do óleo, acima da pressão de bolha, utilizando o método de Whitson, C.H. e Brulé, M.R.
  - atributo rhoob: representa a massa específica do óleo na pressão de bolha (lb/ cu ft).
  - método CompressibilidadeMaiorPB_WB: calcula a compressibilidade do óleo em pressões de reservatório maiores que a pressão de bolha utilizando o método de Whitson, C.H. e Brulé, M.R.
- Classe CCompressibilidadeMenorPB: Classe herdeira da classe CCompressibilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da compressibilidade do óleo, menores ou igual a pressão de bolha, utilizando o método de Ahmed, T.
  - atributo rsb: razão de solubilidade no ponto de bolha(scf/STB).
  - atributo t: temperatura do óleo (°F).
  - atributo API: densidade óleo, grau API, (°API).

  - método CompressibilidadeMenorPB: calcula a compressibilidade isotérmica do óleo em pressões de reservatório menores ou igual a pressão de bolha utilizando o método de Ahmed, T.

- Classe CFatorVolFormacao: Classe agregada à classe CFluidoOleoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo do Fator Volume Formação do óleo.

  - atributo gammag: densidade do gás em uma determinada pressão e temperatura (p, T).
  - atributo gammaSTO: densidade do óleo em uma determinada pressão e temperatura (p, T).

- Classe CFatorVolFormacao_AM: Classe herdeira da classe CFatorVolFormacao, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do Fator volume Formação do óleo, em pressões menores ou igual ao da pressão de bolha, utilizando o método de Al-Marhoun, M.

  - atributo rs: razão de solubilidade do óleo (scf/STB).
  - atributo t: Temperatura (°F).
  - método FatorVolFormacao_AM: calcula o fator volume formação do óleo em pressões de reservatório menor e igual a pressão de bolha, utilizando o método de Al-Marhoun, M.

- Classe CFatorVolFormacao_McC: Classe herdeira da classe CFatorVolFormacao, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do Fator volume Formação do óleo, em pressões maiores pressão de bolha, utilizando o método de McCain, W.D. e Spivey, J.P. e Lenn, C.P.

  - atributo rs: razão de solubilidade do óleo (scf/STB).
  - atributo muo: massa específica do óleo nas condições p,T.
  - método FatorVolFormacao_McC: calcula o fator volume formação do óleo em pressões de reservatório maiores que a pressão de bolha, utilizando o método apresentado por McCain, W.D. e Spivey, J.P. e Lenn, C.P.

- Classe CFatorVolFormacao_GEF: Classe herdeira da classe CFatorVolFormacao, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do Fator volume Formação do óleo, em pressões maiores pressão de bolha, utilizando o método %apresentado por GE, Petrosky and Farshad, FF.

  - atributo rsb: razão de solubilidade do óleo no ponto de bolha (scf/STB).
  - atributo tr: Temperatura de reservatório(°F) .
  - atributo p: pressão atual(psia).
  - atributo pb: pressão de bolha(psia).
  - método FatorVolFormacao_GEF: calcula o fator volume formação do óleo em pressões de reservatório maiores que a pressão de bolha, utilizando o método apresentado por GE, Petrosky e Farshad, FF.

- Classe CFatorVolFormacao_AH: Classe herdeira da classe CFatorVolFormacao, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do Fator volume Formação do óleo, a uma pressão menor ou igual à pressão de bolha, utilizando o método de Ahmed, T.

  - método FatorVolFormacao_AH: calcula o fator volume formação do óleo em pressões de reservatório menor e igual a pressão de bolha, utilizando o método apresentado por Ahmed, T.
  - atributo gammaSTO: densidade do óleo em uma determinada pressão e temperatura (p, T).

- Classe CRazaoSolubilidade: Classe agregada à classe CFluidoOleoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo da Razão de Solubilidade do óleo.

  - atributo gammag: densidade do gas no separador.
  - atributo p: pressão em que ser quer calcular a razão de solibilidade, (psia).
  - atributo pb: Pressão de bolha (Psia).
  - atributo t: temperatura em que ser quer calcular a razão de solibilidade, (°F).
  - atributo API: densidade óleo, grau API, (°API).

- Classe CRazaoSolubilidade_PF: Classe herdeira da classe CRazaoSolubilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do Fator volume Formação do óleo, em pressões menores que a pressão de bolha, utilizando o método de Petrosky, G. E., e F. Farshad.

  - método RazaoSolubilidade_PF: calcula a razão de solubilidade gás-óleo, para pressão menores que a pressão de bolha, utilizando o método de Petrosky, G. E., e F. Farshad.

- Classe CRazaoSolubilidade_VB: Classe herdeira da classe CRazaoSolubilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do Fator volume Formação do óleo, em pressões menores que a pressão de bolha, utilizando o método de Petrosky, G. E., e F. Farshad.

  - atributo psp: pressão no separador, (psia).
  - atributo tsa: Temperatura no separador, (°F).
  - método RazaoSolubilidade_VB: calcula a razão de solubilidade gás-óleo, para pressão menores que a pressão de bolha, utilizando o método de Vazquez, M. e Beggs, H.D..

- Classe CPressaoBolha: Classe agregada à classe CFluidoOleoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo da Pressão de Bolha do óleo.

  - atributo gammagSP: densidade do gas no separador.
  - atributo rsp: razão de solubilidade no separador(scf/STB).
  - atributo tr: Temperatura do reservatório (°F).
  - atributo API: densidade óleo, grau API, (°API).

- Classe CPressaoBolha_SM: Classe herdeira da classe CRazaoSolubilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da pressão de bolha do óleo através da a correlação de Standing, M.B..

  - método PressaoBolha_SM: Função para estimar a pressão de bolha do óleo utilizando a correlação de Standing, M.B.

- Classe CPressaoBolha_VA: Classe herdeira da classe CRazaoSolubilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da pressão de bolha do óleo através do método de Valko.

  - método PressaoBolha_VA: Função para estimar a pressão de bolha do óleo utilizando o método de Valko.

- Classe CMassaEspecifica: Classe agregada à classe CFluidoOleoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo da massa específica do óleo.

  - atributo gammag: densidade do gás em uma determinada pressão e temperatura (p, T).
  - atributo gammaSTO: densidade do óleo em uma determinada pressão e temperatura (p, T).
  - atributo p: pressão em que ser quer calcular a massa específica, (psia).
  - atributo pb: Pressão de bolha (Psia).

- Classe CMassaEspecifica_PF: Classe herdeira da classe CRazaoSolubilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da massa específica do óleo, em pressões maiores que a pressão de bolha, utilizando o método proposto por GE, P. & Farshad, F.

  - atributo rsb: razão gás-óleo no ponto de bolha , scf/STB.
  - atributo tr: temperatura do reservatório em °F
  - método MassaEspecifica_PF: função calcula a massa específica do óleo em pressões maiores que a pressão de bolha , utilizando o método proposto por GE, P. & Farshad, F.

- Classe CMassaEspecifica_ST: Classe herdeira da classe CRazaoSolubilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da massa específica do óleo, em pressões menores e igual a pressão de bolha, utilizando o método proposto por Standing, M.B.

  - atributo rs: razão gás-óleo, scf/STB.
  - atributo tr: temperatura do reservatório em °F
  - método MassaEspecifica_ST: A função cálcula a massa específica do óleo em pressões menores e igual a pressão de bolha , utilizando o método proposto por Standing, M.B.

- Classe CMassaEspecifica_MH: Classe herdeira da classe CRazaoSolubilidade, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo da massa específica do óleo, na pressão de bolha e em pressões menores que a pressão de bolha, utilizando o método proposto por McCain, Jr, W. and Hill, N.

  - atributo rsb: razão gás-óleo na pressão de bolha, scf/STB.
  - atributo t: temperatura em °F.
  - atributo co: Compressibilidade do óleo.
  - atributo gammagSP: densidade do gas no separador.

  - método MassaEspecifica_MH: estimar a massa específica do óleo em pressões maiores que a pressão de bolha, utilizando o método de McCain e Hill junto com a compressibilidade do óleo.

- Classe CViscosidadeOleoSat_AKM: Classe agregada à classe CFluidoOleoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo da viscosidade do óleo.

  - atributo roob: massa específica do óleo na pressão de bolha, (lb/ cu ft).
  - método ViscosidadeOleoSat_AKM: calcula a viscosidade de óleos saturados pelo métode de Abu-Khamsin, A. e Al-Marhoun,M.

- Classe CViscosidadeOleoSat_BR: Classe agregada à classe CFluidoOleoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo da viscosidade do óleo.

  - atributo miod: viscosidade do óleo morto (cp);
  - atributo rs: razão de solubilidade (scf/STB);
  - método ViscosidadeOleoSat_BR: calcula a viscosidade de óleos saturados pelo métode de Beggs, HD and Robinson, JR.

- Classe CFluidoGas: representa a classe onde há atributos e métodos característicos ao gás presentes no reservatório.

  - atributo propPseudocriticas: representa o atributos das propriedades pseudocríticas do gás.
  - atributo fatorCompZ: representa o fator de compressibilidade (Z) do gás.
  - método Compressibilidade: representa o método que calcula a compressibilidade do ógás a partir dos dados fornecidos pelo usuário.
  - método FatorVolFormacao: representa o método que calcula o Fator Volume Formação do gás a partir dos dados fornecidos pelo usuário.
  - método Viscosidade: representa o método que calcula a Viscosidade do gás a partir dos dados fornecidos pelo usuário.
  - método PropPseudocriticas: representa o método que calcula as propriedades pseudocríticas do gás a partir dos dados fornecidos pelo usuário.
  - método fatorCompZ: representa o método que calcula o fator de compressibilidade (Z) do gás a partir dos dados fornecidos pelo usuário.

- Classe CPropPseudocriticas: Classe agregada à classe CFluidoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo das propriedades pseudocríticas do Gás.

  - atributo gammag: densidade do gás em uma determinada pressão e temperatura (p, T).

- Classe CPropPseudocriticas_ST: Classe herdeira da classe CPropPseudocriticas, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo das propriedades pseudocríticas do Gás.

  - atributo tpc: Temperatura pseudocrítica do gás °R.
  - atributo ppc: Pressão pseudocrítica do gás, Psia.
  - atributo tipoDeGas: Especificar se o gás é úmido ou seco.
  - método PropPseudocriticas_ST: estimar as propriedades pseudocríticas do gás.

- Classe CPropPseudocriticas_Piper: Classe herdeira da classe CPropPseudocriticas, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo das propriedades pseudocríticas do Gás com contaminates utilizando método - Piper et Al.

  - atributo tpcc: temperatura pseudo-crítica com correção de contaminates (°R)
  - atributo ppcc: pressão pseudo-crítica com correção de contaminates (psia)
  - atributo yN2: porcentagem de N2 (ex: 35% entre com 0.35).
  - atributo yH2S: porcentagem de H2S.
  - atributo yCO2: pocentagem de CO2.
  - atributo PcN2: pressão crítica do N2 (psia).
  - atributo PcH2S: pressão crítica do H2S (psia).
  - atributo PcCO2: pressão crítica do CO2 (psia).
  - atributo TcN2: Temperatura crítica de N2 (°R).
  - atributo TcH2S: Temperatura crítica de H2S (°R).
  - atributo TcCO2: Temperatura crítica de CO2 (°R).
  - método PropPseudocriticas_Piper: Correção para calcular as propriedades pseudocríticas de gases com contaminates utilizando método - Piper et Al. Este método correlaciona as Propriedade de pseudocríticas do gás com a densidade do gás, porcentagem de contaminantes, e temperatura e pressão crítica dos contaminantes.

- Classe CPropPseudocriticas_WA: Classe herdeira da classe CPropPseudocriticas_Piper, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo das propriedades pseudocríticas do Gás com contaminates utilizando método de Wichert & Aziz.

  - atributo tipodegas: Especificar se o gás é úmido ou seco.
  - método PropPseudocriticas_WA: Correção para calcular as propriedades pseudocríticas de gases com contaminates utilizando método de Wichert & Aziz. Este método correlaciona as Propriedade de pseudocríticas do gás com a densidade do gás, porcentagem de contaminantes.

- Classe CPropPseudocriticas_CKB: Classe herdeira da classe CPropPseudocriticas_Piper, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo das propriedades pseudocríticas do Gás com contaminates utilizando método de Carr-Kobayashi-Burrows.

  - atributo tipodegas: Especificar se o gás é úmido ou seco.
  - método PropPseudocriticas_CKB: Correção para calcular as propriedades pseudocríticas de gases com contaminates utilizando método de Carr-Kobayashi-Burrows. Este método correlaciona as Propriedade de pseudocríticas do gás com a densidade do gás, porcentagem de contaminantes.

- Classe CFatorCompZ: Classe agregada à classe CFluidoGas, representa uma classe base onde são encontrados alguns atributos necessários ao cálculo do fator de compressibilidade (Z) do Gás.

  - atributo ppr: pressão pseudo-reduzida.
  - atributo tpr: temperatura pseudo-reduzida.

- Classe CFatorCompZ_BeB: Classe herdeira da classe CFatorCompZ, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do fator de compressibilidade (Z) do Gás.

  - método FatorCompZ_BeB: extima o valor do fator de compressibilidade gás Z com ppr e tpr utilizando o método apresentado po Brill e Beggs.

- Classe CFatorCompZ_Lond: Classe herdeira da classe CFatorCompZ, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do fator de compressibilidade (Z) do Gás.

  - método FatorCompZ_Lond: extima o valor do fator de compressibilidade gás Z com ppr e tpr utilizando o método apresentado por Londono et al

- Classe CFatorCompZ_DeK: Classe herdeira da classe CFatorCompZ, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do fator de compressibilidade (Z) do Gás.

  - método FatorCompZ_DeK: extima o valor do fator de compressibilidade gás Z com ppr e tpr utilizando o método apresentado por Dranchuk and Abou-Kassem.

- método FatorCompZ_HeY: Classe herdeira da classe CFatorCompZ, representa uma classe onde são encontrados alguns atributos, além dos herdados da classe base, necessários ao cálculo do fator de compressibilidade (Z) do Gás.

  - método FatorCompZ_HeY: extima o valor do fator de compressibilidade gás Z com ppr e tpr utilizando o método apresentado por Hall-Yarborough.

4.2 Diagrama de seqüência – eventos e mensagens

4.2.1 Diagrama de Sequência geral

Veja o diagrama de seqüência geral na Figura 4.9. Mostra a sequência de passos que o usuário deve realizar para que se possar calcular uma determinada propriedade da biblioteca de CORRELAÇÕES PVT.

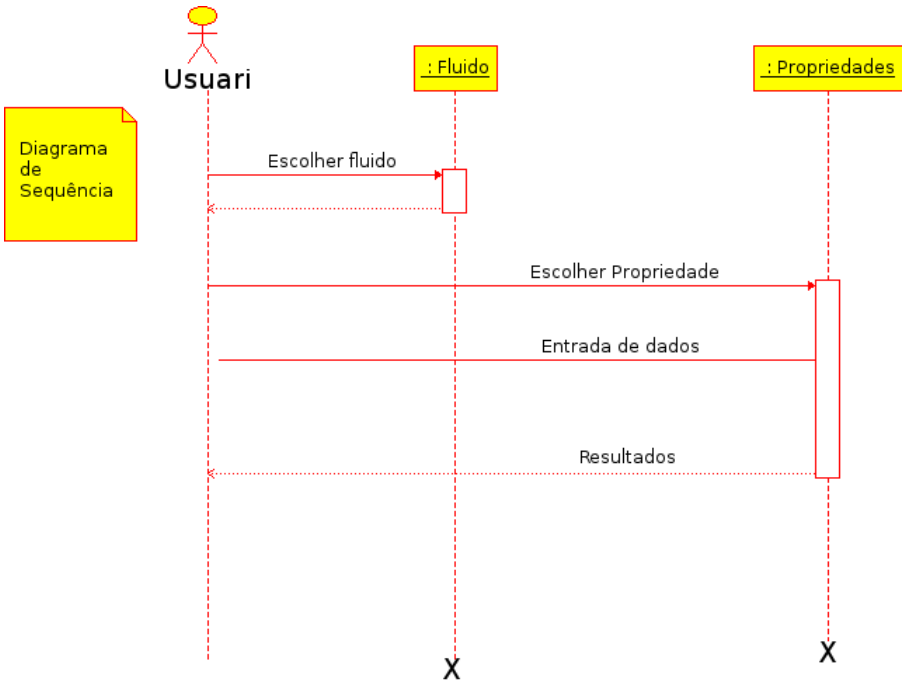

Figura 4.9: Diagrama de sequência-geral.

4.2.2 Diagrama de Sequência específico

Veja o diagrama de seqüência específico na Figura 4.10. Mostra umaa sequência específica de passos para que o usuário calcule a massa específica do óleo utilizando o método proposto por McCain, Jr, W. and Hill, N.

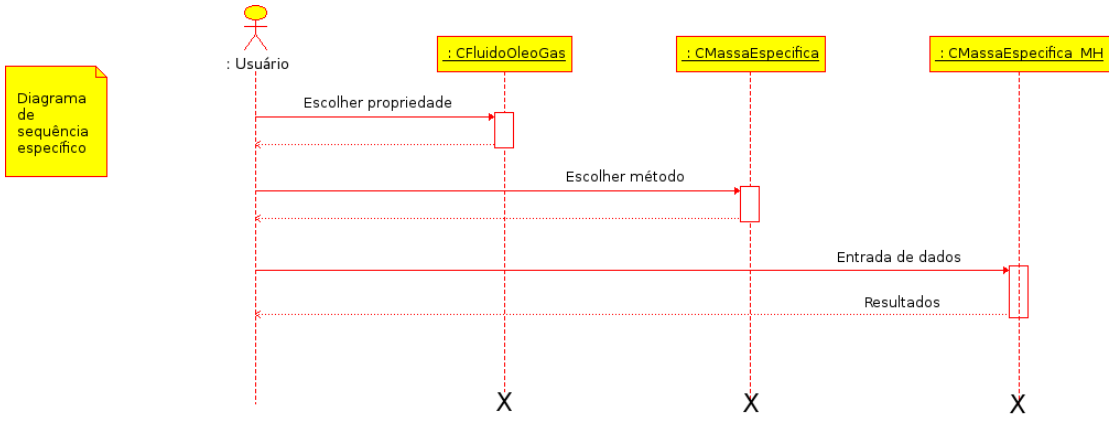

Figura 4.10: Diagrama de sequência específico

4.3 Diagrama de comunicação – colaboração

No diagrama de comunicação o foco é a interação e a troca de mensagens e dados entre os objetos.

Veja na Figura 4.11 o diagrama de comunicação.

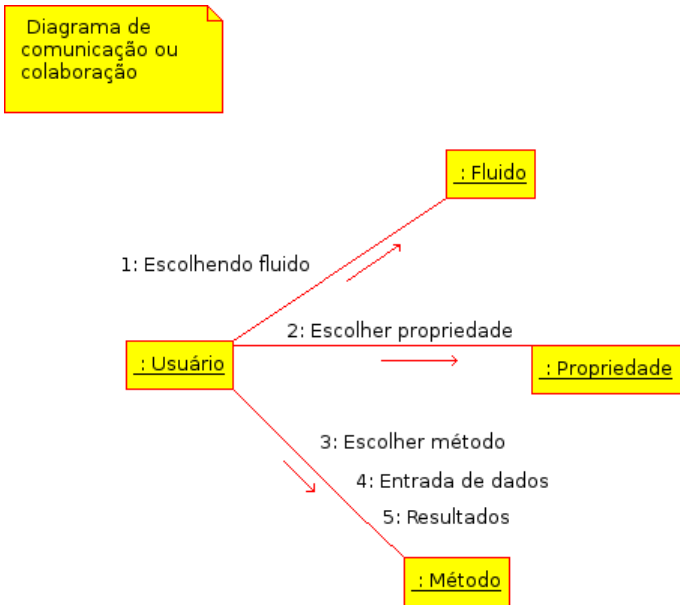

Figura 4.11: Diagrama de comunicação.

4.4 Diagrama de máquina de estado

Veja na Figura 4.12 o diagrama de máquina de estado. Onde se calcula a viscosidade do gás, da classe CFluidoGas, utilizando o método de Lee, A. and Gonzalez, M. and Eakin, B.

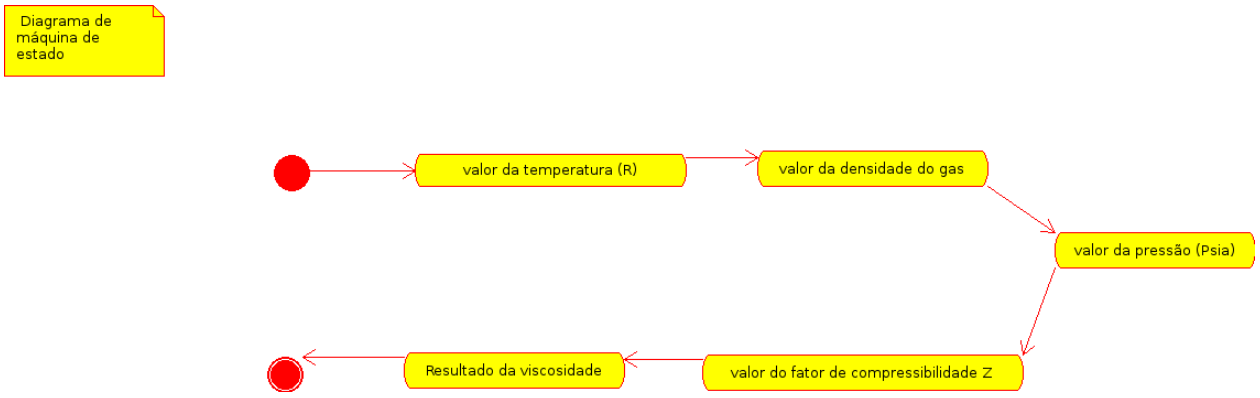

Figura 4.12: Diagrama de máquina de estado - classe CFluidoGas - calculo-viscosidade-gas.

4.5 Diagrama de atividades

Veja na Figura 4.13 o diagrama de atividades correspondente a uma atividade específica do diagrama de máguina de estado. Observe que é descrito em detalhes o processo de obtenção da do Cálculo do fator de compressibilidade "Z" pelo metodo apresentado por Brill e Beggs.

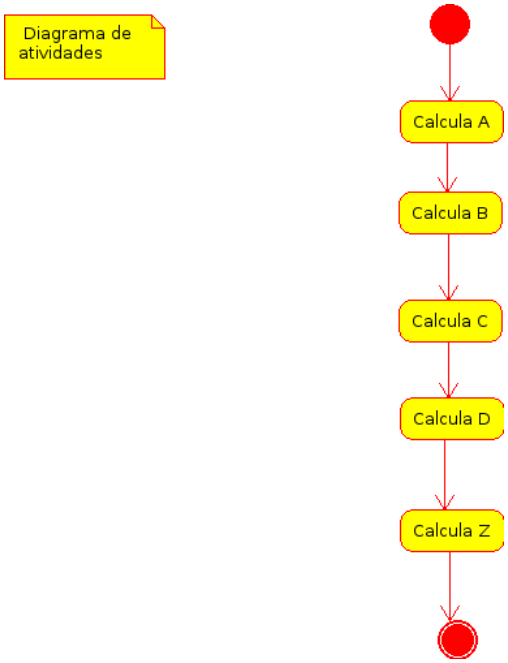

Figura 4.13: Diagrama de atividades - Calculo-do-fator-compressibilidade-Z.

Capítulo 5

Projeto

5.1 Projeto do sistema

1. Protocolos

    - Definição do formato dos arquivos gerados pelo programa: O programa "CORRELAÇÕES PVT" não gera aquivos. A entrada dos dados é feita pelo próprio usuário via teclado e a saída do resultado é via tela. Logo, não há formato de arquivo para os resultados gerados.

2. Plataformas

    - Lnguagem de programação C++ com orientação a objeto.
    - Interface de desenvolvimento IDE o software Kate com o compilador G++ do GNU.
    - O software é executado em uma janela do terminal do linux.

5.2 Projeto orientado a objeto – POO

Efeitos do projeto no modelo estrutural

- Aqui são estabelecidos as dependencias e restrições do software "CORRELAÇÕES PVT".

    – O Software necessita da plataforma GNU/linux para ser executado.

Efeitos do projeto nas heranças

O projeto não interferiu nas classes herdeiras, desde o inicio da elaboração do projeto convencionou-se que seria necessário uma classe regime e que teria como classes herdeiras as classes correspondentes.

Efeitos do projeto nas associações

- Não houve necessidade de rever o projeto nesta etapa.

Efeitos do projeto nas otimizações

- Não houve necessidade de rever o projeto nesta etapa.

5.3 Diagrama de componentes

O diagrama de componente mostra as partes de um projeto para um sistema de software, sendo possível visualizar a estrutura de alto nível do sistema e o comportamento de serviço que essas partes fornecem e consomem através de interfaces.

Capítulo 6

Implementação

Neste capítulo serão implementados tanto os código fonte de todas as classes, assim como o programa que as utiliza.

6.1 Código fonte

Apresenta-se a seguir um conjunto de classes (arquivos .h e .cpp) além do programa main.

Apresenta-se na listagem 6.1 o arquivo com código da classe CFluido.

Listing 6.1: Arquivo de cabeçalho da classe CFluido.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFluido_h
#define CFluido_h

class CFluido
{

//atributos comuns a todas as classes derivadas da classe base CFluido
//os atributos sao comuns aos dois fluidos : oleo e gas
protected:

double compressibilidade; //compressibilidade do oleo ou do gas
double fatorVolFormacao; //fator volume formacao do oleo ou do gas
double viscosidade; //viscosidade do oleo ou do gas


//metodos comuns aos dois fluidos
public:

//metodo calcula a compressibilidade do oleo e do gas
double Compressibilidade();

//metodo calcula o fator volume formacao do oleo e do gas
double FatorVolFormacao();

//metodo calcula a viscosidade do oleo e do gas
double Viscosdade();
```

```
};

#endif
```

Apresenta-se na listagem 6.2 o arquivo com código da classe CFluido.

Listing 6.2: Arquivo de implementação da classe CFluido.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CFluido.h"
```

Apresenta-se na listagem 6.3 o arquivo com código da classe CFluidoOleoGas.

Listing 6.3: Arquivo de cabeçalho da classe CFluidoOleoGas.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFluidoOleoGas_h
#define CFluidoOleoGas_h

class CFluidoOleoGas
{

//atributos comuns a todas as classes derivadas da classe base CFluidoOleoGas
protected:

double pressaoBolha; //pressao de bolha
double razaoSolubilidade; //razao de solubilidade
double densidadeGasSup; //densidade do gas na superficie
double massaEspecifica; //massa especifica do oleo

//metodos comuns a todas as classes derivadas
public:

  //metodo calcula a compressibilidade do oleo
void Compressibilidade();

  //metodo calcula o fator volume formacao do oleo
void FatorVolFormacao();

  //metodo calcula a viscosidade do oleo
void Viscosidade();

  //metodo calcula a pressao de bolha do oleo
void PressaoBolha();

 // metodo calcula a razao de solubilidade do oleo
void RazaoSolubilidade();

  //metodo calcula a densidade do gas na superficie
void DensidadeGasSup();

  //metodo calcula a massa especifica do oleo
void MassaEspecifica();

};

#endif
```

Apresenta-se na listagem 6.4 o arquivo com código da classe CFluidoOleoGas.

Listing 6.4: Arquivo de implementação da classe CFluidoOleoGas.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CFluidoOleoGas.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CFluidoOleoGas::DensidadeGasSup()

{
/*Densidade do gas na superficie

gammag(gammagSP, pSP, RSP,API) calcula a densidade do gas, utilizando os
metodo de McCain, W.D. and Spivey, J.P. and Lenn, C.P., Petroleum
Reservoir Fluid Property Correlations (Oklahoma: PennWell Corporation,
2010) e o metodo de Velarde, J. and Blasingame, T. and McCain, Jr, W.,
"Correlation of Black Oil Properties at Pressures Below Bubble Point
Pressure-A New Approach", in Annual Technical Meeting (1999)
em que:
gammagSP= densidade do gas no separador

*/

fstream fin;
fin.open("dados_fluido_oleogas_dens_gas_sup.dat");


while(!fin.eof())
{
  double gammagSP;
  fin >> gammagSP;
  cout<<gammagSP<<endl;

    //entrada de dados
/*  cout<<"Entre com o valor da densidade do gas no separador."<<endl;
  cin>> gammagSP;  cin.get();
*/
   //calculo da densidade do gas na superficie
  double densidadegassurperficie = 1.066*gammagSP;

   //mostra na tela o resultado do gas em superficie
  cout<<"O valor da densidade do gas em superficie e:" <<densidadegassurperficie << endl;


}
}
```

Apresenta-se na listagem 6.5 o arquivo com código da classe CViscosidadeOleoSubSat.

Listing 6.5: Arquivo de cabeçalho da classe CViscosidadeOleoSubSat.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoSubSat_h
```

```
#define CViscosidadeOleoSubSat_h

class CViscosidadeOleoSubSat

{
/* atributos comuns a todas as classes derivadas da classe base
CViscosidadeOleoSubSat */

protected:
  double miob; //viscosidade do oleo saturado
  double p; //pressao atual
  double pb; //pressao de bolha

};

#endif
```

Apresenta-se na listagem 6.6 o arquivo com código da classe CViscosidadeOleoSubSat.

Listing 6.6: Arquivo de implementação da classe CViscosidadeOleoSubSat.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoSubSat.h"
```

Apresenta-se na listagem 6.7 o arquivo com código da classe CViscosidadeOleoSubSat_PF.

Listing 6.7: Arquivo de cabeçalho da classe CViscosidadeOleoSubSatPF.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoSubSat_PF_h
#define CViscosidadeOleoSubSat_PF_h

#include "CViscosidadeOleoSubSat.h"

class CViscosidadeOleoSubSat_PF : public CViscosidadeOleoSubSat

{

public:

  /*metodo calcula viscosidade do oleo subsaturado pelo
   metodo de Petrosky e Farshad*/
  void ViscosidadeOleoSubSat_PF();

};

#endif
```

Apresenta-se na listagem 6.8 o arquivo com código da classe CViscosidadeOleoSubSat_PF.

Listing 6.8: Arquivo de implementação da classe CViscosidadeOleoSubSatPF.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoSubSat_PF.h"
#include <cmath>
#include <iostream>
```

```
#include <fstream>

using namespace std;

void CViscosidadeOleoSubSat_PF :: ViscosidadeOleoSubSat_PF()

{

fstream fin;
fin.open("dados_viscosidade_oleo_sub_sat_PF.dat");


while(!fin.eof())
{

   double miob;
  fin >> miob;
  cout<<miob<<endl;

   double p;
  fin >> p;
  cout<<p<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;


  //entrada de dados

/*cout << "Entre com o valor da viscosidade do oleo saturado\n";
  cin >> miob;
  cin.get();

  cout << "Entre com o valor da pressao do reservatorio\n";
  cin >> p;
  cin.get();

  cout << "Entre com o valor da pressao de bolha\n";
  cin >> pb;
  cin.get();
*/
  double A;

  //calculo de A
  A= -1.0146 + 1.3322 * (log10(miob)) - 0.4876 * pow(log10(miob),2)- 1.15036 * pow(log10(
      miob),3);

  //calculo viscosidade do oleo subsaturado
double viscosidadedooleosubsaturado_PF = miob + 1.3449 * pow(10,-3) * (p-pb) * pow(10,A);

  //mostra na tela o resultado da viscosidade do oleo subsaturado
cout << "\nO valor da viscosidade do oleo sub saturado pelo metodo de Petrosky e Farshad
    e" << viscosidadedooleosubsaturado_PF << endl;

}
}
```

Apresenta-se na listagem 6.9 o arquivo com código da classe CViscosidadeOleoSubSat_VB.

Listing 6.9: Arquivo de cabeçalho da classe CViscosidadeOleoSubSatVB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoSubSat_VB_h
#define CViscosidadeOleoSubSat_VB_h

#include "CViscosidadeOleoSubSat.h"

class CViscosidadeOleoSubSat_VB : public CViscosidadeOleoSubSat

{
 //a classe possui apenas atributos da classe base CViscosidadeOleoSubSat

public:

  /* metodo calcula viscosidade do oleo subsaturado
   pelo metodo de Vazquez e Beggs */
  void ViscosidadeOleoSubSat_VB();

};

#endif
```

Apresenta-se na listagem 6.10 o arquivo com código da classe CViscosidadeOleoSubSat_VB.

Listing 6.10: Arquivo de implementação da classe CViscosidadeOleoSubSatVB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoSubSat_VB.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CViscosidadeOleoSubSat_VB :: ViscosidadeOleoSubSat_VB()

{

fstream fin;
fin.open("dados_viscosidade_oleo_sub_sat_VB.dat");


while(!fin.eof())
{

   double miob;
  fin >> miob;
  cout<<miob<<endl;

   double p;
  fin >> p;
  cout<<p<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;
```

```
  //entrada de dados
/*  cout << "Entre com o valor da viscosidade do oleo saturado\n";
  cin >> miob;
  cin.get();

  cout << "Entre com o valor da pressao do reservatorio\n";
  cin >> p;
  cin.get();

  cout << "Entre com o valor da pressao de bolha\n";
  cin >> pb;
  cin.get();
*/

  double m;

  //calculo de m
  m= 2.6 * pow(p,1.187) * pow(10,(-3.9 * pow(10,-5) *p -5));

  //calculo viscosidade do oleo subsaturado
double viscosidadedooleosubsaturado_VB = (miob) * pow((p/pb),m);

 //mostra na tela o resultado da viscosidade do oleo subsaturado
cout << "\nO valor da viscosidade do oleo sub saturado pelo metodo de Vazquez-Beggs e" <<
     viscosidadedooleosubsaturado_VB << endl;

}
}
```

Apresenta-se na listagem 6.11 o arquivo com código da classe CViscosidadeOleoMorto.

Listing 6.11: Arquivo de cabeçalho da classe CViscosidadeOleoMorto.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoMorto_h
#define CViscosidadeOleoMorto_h

class CViscosidadeOleoMorto

{
  /*atributos comuns a todas as classes derivadas
  da classe base CViscosidadeOleoMorto */
protected:
  double API; //grau API
  double t; //temperatura atual

};

#endif
```

Apresenta-se na listagem 6.12 o arquivo com código da classe CViscosidadeOleoMorto.

Listing 6.12: Arquivo de implementação da classe CViscosidadeOleoMorto.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoMorto.h"
```

Apresenta-se na listagem 6.13 o arquivo com código da classe CViscosidadeOleoMorto_G.

Listing 6.13: Arquivo de cabeçalho da classe CViscosidadeOleoMortoG.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoMorto_G_h
#define CViscosidadeOleoMorto_G_h

#include "CViscosidadeOleoMorto.h"

class CViscosidadeOleoMorto_G : public CViscosidadeOleoMorto

{
 //a classe possui apenas atributos da classe base CViscosidadeOleoMorto

public:

  /* metodo calcula viscosidade do oleo morto pelo metodo de
   Glaso*/
  void ViscosidadeOleoMorto_G();

};

#endif
```

Apresenta-se na listagem 6.14 o arquivo com código da classe CViscosidadeOleoMorto_G.

Listing 6.14: Arquivo de implementação da classe CViscosidadeOleoMortoG.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoMorto_G.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CViscosidadeOleoMorto_G :: ViscosidadeOleoMorto_G()

{

fstream fin;
fin.open("dados_viscosidade_oleo_morto_G.dat");


while(!fin.eof())
{

   double API;
  fin >> API;
  cout<<API<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;


  //entrada de dados
/*  cout << "Entre com o valor da densidade API\n";
  cin >> API;
```

```
  cin.get();

  cout << "Entre com o valor da temperatura\n";
  cin >> t;
  cin.get();
*/
  //calculo de A
double A= 10.313 * (log10(t)) - 36.447;

  //calculo da viscosidade do oleo morto
double viscosidadedooleomorto_G = (3.141 * (10^10)) * pow(t,-3.444) * pow(log10(API),A);

  //mostra na tela o resultado da viscosidade do oleo morto
cout << "\nO valor da viscosidade do oleo morto pelo metodo de Glaso e" <<
    viscosidadedooleomorto_G << endl;

}
}
```

Apresenta-se na listagem 6.15 o arquivo com código da classe CViscosidadeOleoMorto_BR.

Listing 6.15: Arquivo de cabeçalho da classe CViscosidadeOleoMortoBR.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoMorto_BR_h
#define CViscosidadeOleoMorto_BR_h

#include "CViscosidadeOleoMorto.h"

class CViscosidadeOleoMorto_BR : public CViscosidadeOleoMorto

{
  //a classe possui apenas atributos da classe base CViscosidadeOleoMorto

public:

  /*metodo calcula viscosidade do oleo morto pelo metodo de Beggs
  e Robinson */
  void ViscosidadeOleoMorto_BR();

};

#endif
```

Apresenta-se na listagem 6.16 o arquivo com código da classe CViscosidadeOleoMorto_BR.

Listing 6.16: Arquivo de implementação da classe CViscosidadeOleoMortoBR.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoMorto_BR.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CViscosidadeOleoMorto_BR :: ViscosidadeOleoMorto_BR()
```

```
{

fstream fin;
fin.open("dados_viscosidade_oleo_morto_BR.dat");


while(!fin.eof())
{

   double API;
  fin >> API;
  cout<<API<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;

  //entrada de dados
/*  cout << "Entre com o valor da densidade API\n";
  cin >> API;
  cin.get();

  cout << "Entre com o valor da temperatura\n";
  cin >> t;
  cin.get();
*/
  //calculo de A
double A= pow(10,(3.0324 - 0.02023 * API));

  //calculo da viscosidade do oleo morto
double viscosidadedooleomorto_G = pow(10,(A * pow(t,-1.163)))-1;

 //mostra na tela o resultado da viscosidade do oleo morto
cout << "\nO valor da viscosidade do oleo morto pelo metodo de Beggs-Robbinson e" <<
    viscosidadedooleomorto_G << endl;


}
}
```

Apresenta-se na listagem 6.17 o arquivo com código da classe CViscodidadeOleoSat_BR.

Listing 6.17: Arquivo de cabeçalho da classe CViscodidadeOleoSatBR.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoSat_BR_h
#define CViscosidadeOleoSat_BR_h

class CViscosidadeOleoSat_BR

{

protected:
  double miob; //viscosidade do oleo morto
  double Rs; //razao de solubilidade

public:

  /*metodo calcula viscosidade do oleo saturado
```

```
   pelo metodo de Beggs e Robinson*/
  void ViscosidadeOleoSat_BR();

};

#endif
```

Apresenta-se na listagem 6.18 o arquivo com código da classe CViscodidadeOleoSat_BR.

Listing 6.18: Arquivo de implementação da classe CViscodidadeOleoSatBR.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoSat_BR.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CViscosidadeOleoSat_BR :: ViscosidadeOleoSat_BR()

{

/*Viscosidade do oleo - metodo de Beggs-Robinson para oleos saturados
 A fucao miombr calcula a viscosidade de oleos saturados pelo metodo de
 Beggs, HD and Robinson, JR, "Estimating the viscosity of crude oil
 systems"(1975)

 Entrada:
 Rs = razao de solubilidade(scf/STB);
 miod = viscosidade do oleo morto(cp); */

fstream fin;
fin.open("dados_viscosidade_oleo_sat_BR.dat");


while(!fin.eof())
{

   double miob;
  fin >> miob;
  cout<<miob<<endl;

   double Rs;
  fin >> Rs;
  cout<<Rs<<endl;

//entrada de dados
/*cout << "Entre com o valor da viscosidade do oleo saturado\n";
  cin >> miob;
  cin.get();

  cout << "Entre com o valor da razao de solubilidade\n";
  cin >> Rs;
  cin.get();
*/
  //calculo de a e b
double a = 10.715 * pow((Rs +100),-0.515);
double b = 5.44 * pow((Rs + 150),-0.338);
```

```
 //calculo da viscosidade do oleo saturado
double viscosidadedooleosaturado_BR = a * pow(miob,b);

//mostra na tela o resultado da viscosidade do oleo saturado
cout << "\n0␣valor␣da␣viscosidade␣do␣oleo␣saturado␣pelo␣metodo␣de␣Beggs-Robbinson␣e" <<
    viscosidadedooleosaturado_BR << endl;

}
}
```

Apresenta-se na listagem **??** o arquivo com código da classe `CViscodidadeOleoSat_AKM`.

Listing 6.19: Arquivo de cabeçalho da classe CViscodidadeOleoSatAKM.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CViscosidadeOleoSat_AKM_h
#define CViscosidadeOleoSat_AKM_h

class CViscosidadeOleoSat_AKM

{

protected:
  double roob; //massa especifica do oleo na pressao de bolha

public:

  /*metodo calcula viscosidade do oleo saturado pelo
   metodo de Abu-Khamsin e Al-Marhoun */
  void ViscosidadeOleoSat_AKM();

};

#endif
```

Apresenta-se na listagem 6.20 o arquivo com código da classe `CViscosidadeOleoSat_AKM`.

Listing 6.20: Arquivo de implementação da classe CViscodidadeOleoSatAKM.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CViscosidadeOleoSat_AKM.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CViscosidadeOleoSat_AKM :: ViscosidadeOleoSat_AKM()

{

/*Viscosidade do oleo - metodo de Abu-Khamsim e Al-Marhoun para oleos
  saturados.
 O programa calcula a viscosidade de oleos saturados pelo metode  de
 Abu-Khamsin, A. and Al-Marhoun,M., "Development of a New Correlation for
 Bubllepoint Viscosity"
```

```
roob = massa especifica do oleo na pressao de bolha, (lb/ cu ft)

 */


fstream fin;
fin.open("dados_viscosidade_oleo_sat_AKM.dat");


while(!fin.eof())
{

   double roob;
  fin >> roob;
  cout<<roob<<endl;


//entrada de dados
/*cout << "Entre com o valor da massa especifica do oleo na pressao de bolha\n";
  cin >> roob;
  cin.get();
*/
 //calculo da massa especifica do oleo na pressao de bolha
double roob1 = roob/62.4;

//calculo da viscosidade do oleo saturado
double viscosidadedooleosaturado_AKM = exp(8.484462 * pow(roob1,4)-2.652294);

//mostra na tela o resultado da viscosidade do oleo saturado
cout << "\nO valor da viscosidade do oleo saturado pelo metodo de Abu-Khamsim e Al-
    Marhoun e" << viscosidadedooleosaturado_AKM << endl;

}
}
```

Apresenta-se na listagem 6.21 o arquivo com código da classe CPressaoBolha.

Listing 6.21: Arquivo de cabeçalho da classe CPressaoBolha.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CPressaoBolha_h
#define CPressaoBolha_h

class CPressaoBolha

{
//atributos comuns a todas as classes derivadas da classe base CPressaoBolha
protected:

double rsp; //razao de solubilidade no separador
double API; //grau API do oleo
double gammagSP; //densidade do gas no separador
double tr; //temperatura reduzida

};

#endif
```

Apresenta-se na listagem 6.22 o arquivo com código da classe CPressaoBolha.

Listing 6.22: Arquivo de implementação da classe CPressaoBolha.

```
//trabalho_programacao_pratica
//dupla_alex_aline
#include "CPressaoBolha.h"
```

Apresenta-se na listagem 6.23 o arquivo com código da classe CPressaoBolha_SM.

Listing 6.23: Arquivo de cabeçalho da classe CPressaoBolhaSM.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CPressaoBolha_SM_h
#define CPressaoBolha_SM_h

#include "CPressaoBolha.h"

class CPressaoBolha_SM : public CPressaoBolha

{
//a classe derivada so possui atributos da classe base

public:

//metodo calcula a pressao de bolha pelo metodo de Standing
void PressaoBolha_SM();

};

#endif
```

Apresenta-se na listagem 6.24 o arquivo com código da classe CPressaoBolha_SM.

Listing 6.24: Arquivo de implementação da classe CPressaoBolhaSM.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CPressaoBolha_SM.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CPressaoBolha_SM::PressaoBolha_SM()
{
/* Programa para estimar a pressao de bolha do oleo utilizando a
correlacao de Standing, M.B.(1981), Volumetric and phase behavior
of oil field hydrocarbon systems9th e a troca da densidade do gas pela
densidade do gas no separador proposto por McCain(1991) The properties of
petroleum fluids

Entrada:

rsp= razÃ£o de solubilidade no separador(scf/STB)
API =  densidade oleo, grau API, (API)
gammagSP= densidade do gas no separador
tr= Temperatura do reservatorio (F)
```

```
Saida:

pb=pressao de bolha do oleo(psia) */

fstream fin;
fin.open("dados_pressao_bolha_SM.dat");


while(!fin.eof())
{
  double rsp;
  fin >> rsp;
  cout<<rsp<<endl;

  double API;
  fin >> API;
  cout<<API<<endl;

  double gammagSP;
  fin >> gammagSP;
  cout<<gammagSP<<endl;

  double tr;
  fin >> tr;
  cout<<tr<<endl;

//entrada de dados
/*cout<<"Entre com o valor da razao de solubilidade no separador."<<endl;
cin>>rsp;  cin.get();

cout<<"Entre com o valor do grau API do oleo."<<endl;
cin>>API;  cin.get();

cout<<"Entre com o valor da densidade do gas no separador."<<endl;
cin>>gammagSP;  cin.get();

cout<<"Entre com o valor da temperatura do reservatorio."<<endl;
cin>>tr;  cin.get();
*/
//calculo da pressao de bolha
double pressaodebolha = 18.2*((pow((rsp/gammagSP),0.83))*(pow(10,(0.00091* tr - 0.0125*
    API)))-1.4);

//mostra na tela o resutado da pressao de bolha
cout<<"O valor da pressao de bolha pelo metodo de Standing e McCain e:"<<pressaodebolha<<
    endl;
}
}
```

Apresenta-se na listagem 6.25 o arquivo com código da classe CPressaoBolha_VA.

Listing 6.25: Arquivo de cabeçalho da classe CPressaoBolhaVA.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CPressaoBolha_VA_h
#define CPressaoBolha_VA_h

#include "CPressaoBolha.h"
```

```
class CPressaoBolha_VA : public CPressaoBolha

{
 //a classe so possui os atributos da classe base

public:

  //metodo calcula a pressao de bolha pelo metodo de Valko e Jr
void PressaoBolha_VA();

};

#endif
```

Apresenta-se na listagem 6.26 o arquivo com código da classe CPressaoBolha_VA.

Listing 6.26: Arquivo de implementação da classe CPressaoBolhaVA.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CPressaoBolha_VA.h"

#include <cmath>
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

void CPressaoBolha_VA::PressaoBolha_VA()
{
/* Programa para estimar a pressao de bolha do oleo utilizando o metodo
de Valko(2003)- Reservoir oil bubblepoint pressures revisited; solution
gas--oil ratios and surface gas specific gravities


Entrada:

rsp= razao de solubilidade no separador(scf/STB)
API =  densidade oleo, grau API, (API)
gammagSP= densidade do gas no separador
tr= Temperatura do reservatorio (F)

Saida:

pb=pressao de bolha do oleo(psia) */

fstream fin;
fin.open("dados_pressao_bolha_VA.dat");


while(!fin.eof())
{
  double rsp;
  fin >> rsp;
  cout<<rsp<<endl;

  double API;
  fin >> API;
```

```
  cout<<API<<endl;

  double gammagSP;
  fin >> gammagSP;
  cout<<gammagSP<<endl;

  double tr;
  fin >> tr;
  cout<<tr<<endl;

//entrada de dados
/*cout<<"Entre com o valor da razÃfo de solubilidade no separador."<<endl;
cin>>rsp;  cin.get();

cout<<"Entre com o valor do grau API do oleo."<<endl;
cin>>API;  cin.get();

cout<<"Entre com o valor da densidade do gas no separador."<<endl;
cin>>gammagSP;  cin.get();

cout<<"Entre com o valor da temperatura do reservatorio."<<endl;
cin>>tr;  cin.get();
*/
//atribuicao de variaveis
double VAR_1 = log(rsp);
double VAR_2 = API;
double VAR_3 = gammagSP;
double VAR_4 = tr;

//inicializacao de variaveis
double C0_1 = -5.48;
double C0_2 = 1.27;
double C0_3 = 4.51;
double C0_4 = -0.7835;

double C1_1 = -0.0378;
double C1_2 = -0.0449;
double C1_3 = -10.84;
double C1_4 = 6.23*(pow(10,(-3)));

double C2_1 = 0.281;
double C2_2 = 4.36*(pow(10,(-4)));
double C2_3 = 8.39;
double C2_4 = -1.22*(pow(10,(-5)));

double C3_1 = -0.0206;
double C3_2 = -4.76*(pow(10,(-6)));
double C3_3 = -2.34;
double C3_4 = 1.03*(pow(10,(-8)));

double Z[4];

//calculo dos componentes de Z
Z[0]= C0_1 + C1_1*VAR_1 + C2_1*(VAR_1*VAR_1)+ C3_1*(pow(VAR_1,3));

Z[1]= C0_2 + C1_2*VAR_2 + C2_2*(VAR_2*VAR_2)+ C3_2*(pow(VAR_2,3));

Z[2]= C0_3 + C1_3*VAR_3 + C2_3*(VAR_3*VAR_3)+ C3_3*(pow(VAR_3,3));

Z[3]= C0_4 + C1_4*VAR_4 + C2_4*(VAR_4*VAR_4)+ C3_4*(pow(VAR_4,3));
```

```
double z= Z[0] + Z[1] + Z[2] + Z[3];

//calculo da pressao de bolha
double pressaodebolha = exp(7.475 + 0.713*z + 0.0075*(z*z));

//mostra na tela o resultado da pressao de bolha
cout<<"O valor da pressao de bolha pelo metodo de Valko e: "<<pressaodebolha<<endl;

}
}
```

Apresenta-se na listagem 6.27 o arquivo com código da classe CMassaEspecifica.

Listing 6.27: Arquivo de cabeçalho da classe CMassaEspecifica.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CMassaEspecifica_H
#define CMassaEspecifica_H

class CMassaEspecifica
{

//atributos comuns a todas as classes derivadas das classes bases CMassaEspecifica
protected:

double gammag; //densidade do gas
double gammaSTO; //densidade do oleo
double p; //pressao atual
double pb; //pressao de bolha

};

#endif
```

Apresenta-se na listagem 6.28 o arquivo com código da classe CMassaEspecifica.

Listing 6.28: Arquivo de implementação da classe CMassaEspecifica.

```
#include "CMassaEspecifica.h"
```

Apresenta-se na listagem 6.29 o arquivo com código da classe CMassaEspecifica_PF.

Listing 6.29: Arquivo de cabeçalho da classe CMassaEspecificaPF.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CMassaEspecifica_PF_H
#define CMassaEspecifica_PF_H

#include "CMassaEspecifica.h"

class CMassaEspecifica_PF : public CMassaEspecifica

{
//atributos pertencentes somente a classe derivada CMassaEspecifica_PF
protected:

double rsb; //razao gas-oleo no ponto de bolha
```

```
double tr; //temperatura reduzida


public:

 /* metodo calcula a massa especifica do oleo pelo metodo de McCain Jr e Hill*/
void MassaEspecifica_PF();

};

#endif
```

Apresenta-se na listagem 6.30 o arquivo com código da classe CMassaEspecifica_PF.

Listing 6.30: Arquivo de implementação da classe CMassaEspecificaPF.

```
#include "CMassaEspecifica_PF.h"

#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CMassaEspecifica_PF::MassaEspecifica_PF()
{
/* Massa especifica Oleo - Standing
A funcao calcula a massa especifica do oleo em pressoes maiores que
a pressao de bolha , utilizando o metodo proposto por GE, Petrosky. & Farshad, F.
Pressure-Volume-Temperature Correlations for Gulf of Mexico Crude Oils

Entrada:

rsb = razao gas-oleo no ponto de bolha , scf/STB
gammag= media ponderada da densidade dos gases em superficie
gammaSTO= densidade oleo no tanque
tr= temperatura do reservatorio em F
p= pressao atual (psia)
pb= pressao de bolha (psia)

SAIDA:

massa especifica do oleo( lb/cu ft) */

fstream fin;
fin.open("dados_mass_especifica_PF.dat");


while(!fin.eof())
{
  double rsb;
  fin >> rsb;
  cout<<rsb<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double gammagSTO;
  fin >> gammagSTO;
  cout<<gammagSTO<<endl;
```

```
  double tr;
  fin >> tr;
  cout<<tr<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;

  double mepb1;
  fin >> mepb1;
  cout<<mepb1<<endl;


//entrada de dados
/*cout<<"Entre com o valor da razao gas ole no ponto de bolha"<<endl;
cin>>rsb;  cin.get();

cout<<"Entre com o valor da mÃ©dia ponderada da densidade dos gÃ¡ses em superfÃcie"<<
    endl;
cin>>gammag;  cin.get();

cout<<"Entre com o valor da densidade do oleo no tanque"<<endl;
cin>>gammaSTO;  cin.get();

cout<<"Entre com o valor da temperatura do reservatorio"<<endl;
cin>>tr;  cin.get();

cout<<"Entre com o valor da pressao atual"<<endl;
cin>>p;  cin.get();

cout<<"Entre com o valor da pressao de bolha"<<endl;
cin>>pb;  cin.get();
*/
 //calculo do grau API
 double API=((141.5)/(gammaSTO))-131.5;

 //entrada de dados
cout<<"Entre com o valor da massa especifica calculada pelo metodo de Standing"<<endl;
cin>>mepb1;  cin.get();

  //calculo da massa especifica
 double  A=4.1646*pow(10,-7)*pow(rsb,0.69357)*pow(gammag,0.1885)*pow(API,0.3772)*pow(tr
     ,0.6729);
 double massaespecifica = mepb1* exp(A*(pow(p,0.4094)-pow(pb,0.4094)));

 //mostra na tela o resultado da massa especifica do oleo
 cout<<"O valor da massa especifica do oleo e:"<< massaespecifica <<endl;

}
}
```

Apresenta-se na listagem 6.31 o arquivo com código da classe CMassaEspecifica_ST.

Listing 6.31: Arquivo de cabeçalho da classe CMassaEspecificaST.

```
#ifndef CMassaEspecifica_ST_H
```

```
#define CMassaEspecifica_ST_H

#include "CMassaEspecifica.h"

class CMassaEspecifica_ST : public CMassaEspecifica

{
//atributos pertencentes somente a classe derivada CMassaEspecifica_ST
protected:

double rs; //razao gas-oleo
double tr; //temperatura reduzida

public:

//metodo calcula a massa especifica do oleo pelo metodo de Standing
void MassaEspecifica_ST();

};

#endif
```

Apresenta-se na listagem 6.32 o arquivo com código da classe CMassaEspecifica_ST.

Listing 6.32: Arquivo de implementação da classe CMassaEspecificaST.

```
#include "CMassaEspecifica_ST.h"

#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CMassaEspecifica_ST::MassaEspecifica_ST()

{
/* Massa especifica Oleo - Standing
A funcao calcula a massa especifica do oleo em pressoes menores e igual
a pressao de bolha , utilizando o metodo proposto por Standing, M.B.,
Volumetric and phase behavior of oil field hydrocarbon systems9th(1981).

Entrada:

rs = razao gas-oleo , scf/STB
gammag= media ponderada da densidade dos gases em superficie
gammaSTO= densidade oleo no tanque
tr= temperatura do reservatorio em F
p= pressao atual (psia)
pb= pressao de bolha (psia)

SAIDA:

massa especifica do oleo( lb/cu ft) */

fstream fin;
fin.open("dados_mass_especifica_ST.dat");


while(!fin.eof())
{
```

```
  double rs;
  fin >> rs;
  cout<<rs<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double gammagSTO;
  fin >> gammagSTO;
  cout<<gammagSTO<<endl;

  double tr;
  fin >> tr;
  cout<<tr<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;

//entrada de dados
/*cout<<"Entre com o valor da razao gas-oleo."<<endl;
cin>>rs;  cin.get();

cout<<"Entre com o valor da media ponderada da densidade dos gases em superficie."<<endl;
cin>>gammag;  cin.get();

cout<<"Entre com o valor da densidade do oleo no tanque."<<endl;
cin>>gammaSTO;  cin.get();

cout<<"Entre com o valor da temperatura do reservatorio."<<endl;
cin>>tr;  cin.get();

cout<<"Entre com o valor da pressao atual."<<endl;
cin>>p;  cin.get();

cout<<"Entre com o valor da pressao de bolha."<<endl;
cin>>pb;  cin.get();
*/

double massaespecificaooleopb;

//teste da pressao
if( p<=pb)
{
massaespecificaooleopb= (62.4*gammaSTO + 0.0136*rs*gammag)/
    (0.972+0.000147*pow((rs*pow((gammag/gammaSTO),0.5)+1.25*tr),1.175));
}
    else
 {

 cout<<"Programa so calcula a massa especifica em pressoes de reservatorio menores ou
     iguais a pressao de bolha"<<endl;

 }
}
```

```
}
```

Apresenta-se na listagem 6.33 o arquivo com código da classe CMassaEspecifica_MH.

Listing 6.33: Arquivo de cabeçalho da classe CMassaEspecificaMH.

```
#include "CMassaEspecifica.h"

#ifndef CMassaEspecifica_MH_H
#define CMassaEspecifica_MH_H

class CMassaEspecifica_MH : public CMassaEspecifica

{
//atributos pertencentes somente a classe derivada CMassaEspecifica_MH
protected:

double rsb; //razao gas-oleo na pressao de bolha
double gammagSP; //densidade do gas no separador
double t; //temperatura atual
double co; //compressibilidade do oleo em pressoes maiores que a pressao de bolha

public:

/* metodo calcula a massa especifica do oleo pelo metodo de McCain Jr e Hill*/

void MassaEspecifica_MH();

};

#endif
```

Apresenta-se na listagem 6.34 o arquivo com código da classe CMassaEspecifica_MH.

Listing 6.34: Arquivo de implementação da classe CMassaEspecificaMH.

```
#include "CMassaEspecifica_MH.h"

#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CMassaEspecifica_MH::MassaEspecifica_MH()

{
/*Massa especifica Oleo- McCain e Hill

Programa calcula a massa especifica do oleo, utilizando o metodo proposto
por McCain, Jr, W. and Hill, N., "Correlations for liquid densities and
evolved gas specific gravities for black oils during pressure depletion,
para se estimar a a massa especifica do oleo na pressao de bolha e em
pressoes menores que a pressao de bolha. Para estimar a massa especifica
do oleo em pressoes maiores que a pressao de bolha, este metodo utiliza o
metodo de McCai e Hill junto com a equacao de definicao da
compressibilidade do oleo.

Entrada:

rsb = razao gas-oleo na pressao de bolha, scf/STB
gammagSP = densidade do gas no separador
```

```
gammag= media ponderada da densidade dos gases em superficie
gammaSTO= densidade oleo no tanque
t= temperatura em F
p=pressao que se deseja estimar a massa especifica do oleo(psia)
pb = pressao de bolha(psia)
co= compressibiliadade do oleo em pressoes maiores que a pressao de
bolha, (1/psi)

modo de usar:

1 p=pb,entre com:
muoleom(Rsb,gammagSP, gammag, gammaSTO, T, pb, pb)

2 p<pb, entre com:
muoleom(Rsb,gammagSP, gammag, gammaSTO, T, p, pb)

3 p>pb, entre com:
muoleom(Rsb,gammagSP, gammag, gammaSTO, T, p, pb, co)

SAIDA:

massa especifica do oleo( lb/cu ft) */

fstream fin;
fin.open("dados_mass_especifica_MH.dat");


while(!fin.eof())
{
  double rsb;
  fin >> rsb;
  cout<<rsb<<endl;

  double gammagSP;
  fin >> gammagSP;
  cout<<gammagSP<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double gammagSTO;
  fin >> gammagSTO;
  cout<<gammagSTO<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;

  double co;
  fin >> co;
  cout<<co<<endl;
```

```
//entrada de dados

/*cout<<"Entre com o valor da razao gas-oleo na pressao de bolha."<<endl;
cin>>rsb;  cin.get();

cout<<"Entre com o valor da densidade do gas no separador."<<endl;
cin>>gammagSP;  cin.get();

cout<<"Entre com o valor da media ponderada da densidade dos gases em superficie."<<endl;
cin>>gammag;  cin.get();

cout<<"Entre com o valor da densidade do oleo no tanque."<<endl;
cin>>gammaST0;  cin.get();

cout<<"Entre com o valor da temperatura."<<endl;
cin>>t;  cin.get();

cout<<"Entre com o valor da pressao em que se deseja estimar a massa especifica do oleo
    ."<<endl;
cin>>p;  cin.get();

cout<<"Entre com o valor da pressao de bolha."<<endl;
cin>>pb;  cin.get();

cout<<"Entre com o valor da compressibilidade do oleo em pressoes maiores que a pressao
    de bolha."<<endl;
cin>>co;  cin.get();
*/
//declaracao de variavel
double massaespecifica;

//correlacao  para pressoes igual a pressao de bolha
if (p == pb)
{
//inicializacao das variaveis
double w0= -49.8930;
double w1= 85.0149;
double w2= -3.70373;
double w3= 0.0479818;
double w4= 2.98914;
double w5= -0.0356888;

//consideracoes a fazer
double rsb1=rsb;
double gammaST01= gammaST0;
double gammag2=gammag;
double gammagSP1=gammagSP;
double ropo = (( rsb1*gammag2 + 4600*gammaST01)/(73.71+rsb1*gammag2)
    /(w0+w1*gammag2+w2*gammagSP1+w3*gammagSP1 + w4
    + w5));


//calculo da massa especifica
double deltarop = (0.167 + 16.181*pow(10,(-0.0425*ropo)))*(pb/1000)-0.01
    *(0.299+263*pow(10,(-0.0603*ropo)))*pow((pb/1000),2);


double robs = ropo + deltarop;
```

```
double deltarot = (0.00302 + 1.505* pow(robs,(-0.951)))*pow((t-60),0.938)
    -(0.0216 - 0.0233*pow(10,(-0.0161*robs)))*pow((t-60),0.475);

double massaespecifica = robs - deltarot;
}

//correlacao para pressao de reservatorio menor que a pressao de bolha

if( p<pb)

{
//consideracoes a fazer
double rsb1=rsb;
double gammaST01= gammaST0;
double gammag2=gammag;
double gammagSP1=gammagSP;

//inicializacao de variaveis
double w0= -49.8930;
double w1= 85.0149;
double w2= -3.70373;
double w3= 0.0479818;
double w4= 2.98914;
double w5= -0.0356888;

//calculo da massa especifica
double ropo = (( rsb1*gammag2 + 4600*gammaST01)/(73.71+rsb1*gammag2
    /(w0+w1*gammag2+w2*gammagSP1+w3*gammagSP1 + w4
    + w5)));

double deltarop = (0.167 + 16.181*pow(10,(-0.0425*ropo)))*(p/1000)-0.01
    *(0.299+263*pow(10,(-0.0603*ropo)))*pow((p/1000),2);


double robs = ropo + deltarop;


double deltarot = (0.00302 + 1.505* pow(robs,(-0.951)))*pow((t-60),0.938)
    -(0.0216 - 0.0233*pow(10,(-0.0161*robs)))*pow((t-60),0.475);

  double  massaespecifica = robs - deltarot;
}


//correlacao para pressao de reservatorio maior que a pressao de bollha
if( p>pb )
{
//consideracoes a fazer
double rsb1=rsb;
double gammaST01= gammaST0;
double gammag2=gammag;
double gammagSP1=gammagSP;

//inicializacao de variaveis
double w0= -49.8930;
double w1= 85.0149;
double w2= -3.70373;
double w3= 0.0479818;
```

```
double w4= 2.98914;
double w5= -0.0356888;

//calculo da massa especifica
double ropo = (( rsb1*gammag2 + 4600*gammaSTO1)/(73.71+rsb1*gammag2
    /(w0+w1*gammag2+w2*gammagSP1+w3*gammagSP1 + w4
    + w5)));


double deltarop = (0.167 + 16.181*pow(10,(-0.0425*ropo)))*(p/1000)-0.01
    *(0.299+263*pow(10,(-0.0603*ropo)))*pow((p/1000),2);


double robs = ropo + deltarop;


double deltarot = (0.00302 + 1.505* pow(robs,(-0.951)))*pow((t-60),0.938)
    -(0.0216 - 0.0233*pow(10,(-0.0161*robs)))*pow((t-60),0.475);

    double massaespecificapb = robs - deltarot;


   double massaespecifica = massaespecificapb*exp(co*(p-pb));
}

//mostra na tela o resultado da massa especifica
cout<<"O valor da massa especifica e:"<<massaespecifica<<endl;
}
}
```

Apresenta-se na listagem 6.35 o arquivo com código da classe CViscosidadeOleoSubSat.

Listing 6.35: Arquivo de cabeçalho da classe CViscosidadeOleoSubSat.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorVolFormacao_H
#define CFatorVolFormacao_H

class CFatorVolFormacao

{
//atributos comuns a todas as classes derivadas da classe base CFatorVolFormacao

protected:

double gammag; // densidade do gas
double gammaSTO; //densidade do oleo

};

#endif
```

Apresenta-se na listagem 6.36 o arquivo com código da classe CViscosidadeOleoSubSat.

Listing 6.36: Arquivo de implementação da classe CViscosidadeOleoSubSat.

```
#include "CFatorVolFormacao.h"
```

Apresenta-se na listagem **??** o arquivo com código da classe CFatorVolFormacao_AM.

Listing 6.37: Arquivo de cabeçalho da classe CFatorVolFormacaoAM.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorVolFormacao_AM_H
#define CFatorVolFormacao_AM_H

#include "CFatorVolFormacao.h"

class CFatorVolFormacao_AM : public CFatorVolFormacao

{
//a classe so possui atributos da classe base CFatorVolFormacao

protected:

double rs; //razao de solubilidade
double t; //temperatura atual

public:

/* metodo calcula fator volume formacao do oleo pelo metodo
de Al-Marhoun */

void FatorVolFormacao_AM();

};

#endif
```

Apresenta-se na listagem ?? o arquivo com código da classe CFatorVolFormacao_AM.

Listing 6.38: Arquivo de implementação da classe CFatorVolFormacaoAM.

```
#include "CFatorVolFormacao_AM.h"

#include <cmath>
#include <iostream>
#include <fstream>


using namespace std;

void CFatorVolFormacao_AM::FatorVolFormacao_AM()

{
/* Fator volume formacao - metodo Al-Marhoun

 Programa calcula o fator volume formacao do oleo em pressoes de
reservatorio menores e iguais a pressao de bolha, utilizando o metodo de
Al-Marhoun, M., "PVT correlations for Middle East crude oils"(1988)

Entrada:

rs= razao de solubilidade (scf/STB)
gammag = densidade do gas em p,t
gammaSTO = densidade oleo em p,t
t= Temperatura (F)

Saida:
bo= fator volume formacao do oleo (bbl/STB) */
```

```
//entrada de dados


fstream fin;
fin.open("dados_fator_vol_for_AM.dat");


while(!fin.eof())
{
  double rs;
  fin >> rs;
  cout<<rs<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double gammaSTO;
  fin >> gammaSTO;
  cout<<gammaSTO<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;

  double a= 0.742390;
double b=0.323294;
double c=-1.202040;

//conversao da temperatura para rankine
double Trank= t +459.67;

//calculo de f
double f =pow(rs,a)*pow(gammag,b)*pow(gammaSTO,c);

//calculo do fator volume formacao
double fatorvolumeformacao = 0.497069 + 0.000862963*Trank + 0.00182594*f
    + 0.00000318099*pow(f,2);

    //mostra na tela o valor do fator volume formacao calculado
    cout<<"O valor do fator volume formacao e :"<<fatorvolumeformacao<<endl;
}
}
```

Apresenta-se na listagem 6.39 o arquivo com código da classe CFatorVolFormacao_McC.

Listing 6.39: Arquivo de cabeçalho da classe CFatorVolFormacaoMcC.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorVolFormacao_McC_H
#define CFatorVolFormacao_McC_H

#include "CFatorVolFormacao.h"

class CFatorVolFormacao_McC : public CFatorVolFormacao

{
protected:
```

```
double muo; //massa especifica do oleo
double rs; //razao de solubilidade

public:

/* metodo calcula o fator volume formacao pelo metodo de McCain et al */

void FatorVolFormacao_McC();

};

#endif
```

Apresenta-se na listagem 6.40 o arquivo com código da classe CFatorVolFormacao_McC.

Listing 6.40: Arquivo de implementação da classe CFatorVolFormacaoMcC.

```
#include "CFatorVolFormacao_McC.h"

#include <iostream>
#include <fstream>

using namespace std;

void CFatorVolFormacao_McC::FatorVolFormacao_McC()
{
/*Fator volume formacao - metodo  Ahmed, T

 Programa calcula o fator volume formacao do oleo em pressoes de
reservatorio menores e iguais a pressao de bolha, utilizando o metodo
apresentado por Ahmed, T., Equations of State and PVT Analysis (2007)

Entrada:

rs= razao de solubilidade (scf/STB)
gammag = densidade do gas em p,T
gammaSTO = densidade oleo em p,T
muo= massa especifica do oleo nas condicoes p,T

Saida:

Bo= fator volume formacao do oleo (bbl/STB) */

fstream fin;
fin.open("dados_fator_vol_for_McC.dat");


while(!fin.eof())
{
  double rs;
  fin >> rs;
  cout<<rs<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double gammaSTO;
  fin >> gammaSTO;
  cout<<gammaSTO<<endl;
```

```
   double muo;
  fin >> muo;
  cout<<muo<<endl;


//entrada de dados

/*cout<<"Entre com o valor da razao de solubilidade"<<endl;
cin>>rs;  cin.get();

cout<<"Entre com o valor da densidade do gas"<<endl;
cin>>gammag;  cin.get();

cout<<"Entre com o valor da densidade do oleo"<<endl;
cin>>gammaSTO;  cin.get();

cout<<"Entre com o valor da massa especifica do oleo"<<endl;
cin>>muo;  cin.get();
*/
//calculo do fator volume formacao
double fatorvolumeformacao= ((62.4*gammaSTO)+0.0136*rs*gammag)/muo;

//mostra na tela o resultado do fator volume formacao
cout<<"O valor do fator volume formacao e: "<< fatorvolumeformacao <<endl;
}
}
```

Apresenta-se na listagem 6.41 o arquivo com código da classe CFatorVolFormacao_GEF.

Listing 6.41: Arquivo de cabeçalho da classe CFatorVolFormacaoGEF.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorVolFormacao_GEF_H
#define CFatorVolFormacao_GEF_H

#include "CFatorVolFormacao.h"

class CFatorVolFormacao_GEF : public CFatorVolFormacao

{
protected:

double rsb; //razao de solubilidade no ponto de bolha
double tr; //temperatura reduzida
double p; //pressao atual
double pb; //pressao de bolha

public:

/* metodo calcula o fator volume formacao pelo metodo de
 GE e Farshad*/

void FatorVolFormacao_GEF();

};

#endif
```

Apresenta-se na listagem 6.42 o arquivo com código da classe CFatorVolFormacao_GEF.

Listing 6.42: Arquivo de implementação da classe CFatorVolFormacaoGEF.

```
#include "CFatorVolFormacao_GEF.h"

#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

void CFatorVolFormacao_GEF::FatorVolFormacao_GEF()

{
/*Fator volume formacao - metodo Pettrosky e Farshad

 Programa calcula o fator volume formacao do oleo em pressoes de
reservatorio maiores que a pressao de bolha, utilizando o metodo
apresentado por  Petrosky, GE and Farshad, FF, "Pressure-Volume-Temperature
Correlations for Gulf of Mexico Crude Oils" (1993)

Entrada:

rsb= razao de solubilidade no ponto de bolha (scf/STB)
gammag = densidade do gas em p,T
gammaSTO = densidade oleo em p,T
tr= Temperatura de reservatorio(F)
p= pressao atual(psia)
pb= pressao de bolha(psia)

Saida:

Bo= fator volume formacao do oleo (bbl/STB) */

fstream fin;
fin.open("dados_fator_vol_for_GEF.dat");


while(!fin.eof())
{
  double rsb;
  fin >> rsb;
  cout<<rsb<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double gammaSTO;
  fin >> gammaSTO;
  cout<<gammaSTO<<endl;

   double tr;
  fin >> tr;
  cout<<tr<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;
```

```
  double pb;
  fin >> pb;
  cout<<pb<<endl;

//entrada de dados

/*cout<<"Entre com o valor da razao de solubilidade no ponto de bolha"<<endl;
cin>>rsb;  cin.get();

cout<<"Entre com o valor da densidade do gas"<<endl;
cin>>gammag;  cin.get();

cout<<"Entre com o valor da densidade do oleo"<<endl;
cin>>gammaSTO;  cin.get();

cout<<"Entre com o valor da temperatura do reservatorio"<<endl;
cin>>tr;  cin.get();

cout<<"Entre com o valor da rpressao atual"<<endl;
cin>>p;  cin.get();

cout<<"Entre com o valor da pressao de bolha"<<endl;
cin>>pb;  cin.get();
*/
//calculo do grau API do oleo
double API=((141.5)/(gammaSTO))-131.5;

//calculo da variavel a
double A=4.1646*pow(10,-7)*pow(rsb,0.69357)*pow(gammag,0.1885)*
    pow(API,0.3272)*(tr,0.6729);

//calculo do fator volume formacao
double fatorvolumeformacao= exp(-A*(pow(p,0.4094)-pow(pb,0.4094)));

//mostra na tela o resultado do fator volume formacao
cout<<"O valor do fator volume formacao e :"<<fatorvolumeformacao<<endl;

}
}
```

Apresenta-se na listagem 6.43 o arquivo com código da classe CFatorVolFormacao_AM.

Listing 6.43: Arquivo de cabeçalho da classe CFatorVolFormacaoAM.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorVolFormacao_AM_H
#define CFatorVolFormacao_AM_H

#include "CFatorVolFormacao.h"

class CFatorVolFormacao_AM : public CFatorVolFormacao

{
//a classe so possui atributos da classe base CFatorVolFormacao

protected:

double rs; //razao de solubilidade
double t; //temperatura atual
```

```
public:

/* metodo calcula fator volume formacao do oleo pelo metodo
de Al-Marhoun */

void FatorVolFormacao_AM();

};

#endif
```

Apresenta-se na listagem 6.44 o arquivo com código da classe CFatorVolFormacao_AM.

Listing 6.44: Arquivo de implementação da classe CFatorVolFormacaoAM.

```
#include "CFatorVolFormacao_AM.h"

#include <cmath>
#include <iostream>
#include <fstream>


using namespace std;

void CFatorVolFormacao_AM::FatorVolFormacao_AM()

{
/* Fator volume formacao - metodo Al-Marhoun

 Programa calcula o fator volume formacao do oleo em pressoes de
reservatorio menores e iguais a pressao de bolha, utilizando o metodo de
Al-Marhoun, M., "PVT correlations for Middle East crude oils"(1988)

Entrada:

rs= razao de solubilidade (scf/STB)
gammag = densidade do gas em p,t
gammaSTO = densidade oleo em p,t
t= Temperatura (F)

Saida:
bo= fator volume formacao do oleo (bbl/STB) */

//entrada de dados


fstream fin;
fin.open("dados_fator_vol_for_AM.dat");


while(!fin.eof())
{
  double rs;
  fin >> rs;
  cout<<rs<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;
```

```
  double gammaSTO;
  fin >> gammaSTO;
  cout<<gammaSTO<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;

  double a= 0.742390;
double b=0.323294;
double c=-1.202040;

//conversao da temperatura para rankine
double Trank= t +459.67;

//calculo de f
double f =pow(rs,a)*pow(gammag,b)*pow(gammaSTO,c);

//calculo do fator volume formacao
double fatorvolumeformacao = 0.497069 + 0.000862963*Trank + 0.00182594*f
    + 0.00000318099*pow(f,2);

    //mostra na tela o valor do fator volume formacao calculado
    cout<<"O valor do fator volume formacao e :"<<fatorvolumeformacao<<endl;
}
}
```

Apresenta-se na listagem 6.45 o arquivo com código da classe CRazaoSolubilidade.

Listing 6.45: Arquivo de cabeçalho da classe CRazaoSolubilidade.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CRazaoSolubilidade_h
#define CRazaoSolubilidade_h

class CRazaoSolubilidade

{
//atributos comuns a todas as classes derivadas da classe base CRazaoSolubilidade

protected:

double gammag; //densidade do gas
double p; //pressao atual
double pb; //pressao de bolha
double t; //temperatura atual
double API; //grau API

};


#endif
```

Apresenta-se na listagem 6.46 o arquivo com código da classe CRazaoSolubilidade.

Listing 6.46: Arquivo de implementação da classe CRazaoSolubilidade.

```
#include "CRazaoSolubilidade.h"
```

Apresenta-se na listagem 6.47 o arquivo com código da classe CRazaoSolubilidade_PF.

Listing 6.47: Arquivo de cabeçalho da classe CRazaoSolubilidadePF.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CRazaoSolubilidade_PF_H
#define CRazaoSolubilidade_PF_H

#include "CRazaoSolubilidade.h"

class CRazaoSolubilidade_PF : public CRazaoSolubilidade

{
 //a classe so possui atributos da classe base CRazaoSolubilidade

public:

  //metodo calcula razao de solubilidade pelo metodo de GE e Farshad
  void RazaoSolubilidade_PF();

};

#endif
```

Apresenta-se na listagem 6.48 o arquivo com código da classe CRazaoSolubilidade_PF.

Listing 6.48: Arquivo de implementação da classe CRazaoSolubilidadePF.

```
#include "CRazaoSolubilidade_PF.h"

#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CRazaoSolubilidade_PF::RazaoSolubilidade_PF()

{
/* Razao de solubilidade gas-oleo
 Programa calcula a razao de solubilidade gas-oleo utilizando o metodo
 de Petrosky, G. E., and F. Farshad. Pressure Volume Temperature
Correlations for Gulf of Mexico Crude Oils.(1993).

Entrada:

gammag = densidade do gas ou densidade do gas no separador;
p = pressao em que ser quer calcular a razao de solibilidade, (psia)
pb = pressao de bolha, (psia)
t = temperatura em que ser quer calcular a razao de solibilidade, (F)
API =  densidade oleo, grau API, (API)

SAIDA:

Razao de solubilidade (scf/STB) */

fstream fin;
fin.open("dados_razao_solub_PF.dat");


while(!fin.eof())
{
```

```
  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double API;
  fin >> API;
  cout<<API<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;

   double p;
  fin >> p;
  cout<<p<<endl;


//entrada de dados
/*cout<<"Entre com o valor a densidade do gas"<<endl;
cin>>gammag;  cin.get();

cout<<"Entre com o valor do grau API do oleo"<<endl;
cin>>API;  cin.get();

cout<<"Entre com o valor da temperatura em que se quer calcular a razao de solubilidade
    "<<endl;
cin>>t;  cin.get();

cout<<"Entre com o valor da pessao de bolha"<<endl;
cin>>pb;  cin.get();

cout<<"Entre com o valor da pressao em que se quer calcular a razao de solubilidade"<<
    endl;
cin>>p;  cin.get();
*/
double razaodesolubidade;

//teste da pressao
if (p>pb)
{
  double x = 7.916*pow(10,-4)*pow(API,1.5410)- 4.561*pow(10,-5)*pow(t,1.3911);

   razaodesolubidade = ((pb/112.727)+12.340)*pow(gammag,0.8439)*pow(pow(10,x),1.73184);
}
else
{
 double x = 7.916*pow(10,-4)*pow(API,1.5410)- 4.561*pow(10,-5)*pow(t,1.3911);

  razaodesolubidade = ((p/112.727)+12.340)*pow(gammag,0.8439)*pow(pow(10,x),1.73184);
}

//mostra na tela o resultado da razao de solubilidade
cout<<"O␣valor␣da␣razao␣de␣solubilidade␣e:"<<razaodesolubidade<<endl;
}
}
```

Apresenta-se na listagem 6.49 o arquivo com código da classe CRazaoSolubilidade_VB.

Listing 6.49: Arquivo de cabeçalho da classe CRazaoSolubilidadeVB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CRazaoSolubilidade.h"

#ifndef CRazaoSolubilidade_VB_H
#define CRazaoSolubilidade_VB_H

class CRazaoSolubilidade_VB : public CRazaoSolubilidade
{
//atributos pertencentes somente a classe derivada CRazaoSolubilidade_VB

protected:

double psp; //pressao no separador
double tsp; //temperatura no separador

public:

//metodo calcula razao de solubilidade pelo metodo de Vazquez e Beggs
void razaoSolubilidade_VB();

};

#endif
```

Apresenta-se na listagem 6.50 o arquivo com código da classe CRazaoSolubilidade_VB.

Listing 6.50: Arquivo de implementação da classe CRazaoSolubilidadeVB.

```
#include  "CRazaoSolubilidade_VB.h"

#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CRazaoSolubilidade_VB::razaoSolubilidade_VB()

{
/* Razao de solubilidade gas-oleo

Programa cacula a razao desolubilidade gas-oleo utilizando o metodo
de Vazquez, M. and Beggs, H.D., "Correlations for fluid physical property
prediction (1980).

Entrada:

gammag = densidade do gas ou densidade do gas no separador;
p = pressao em que ser quer calcular a razao de solibilidade, (psia)
pb = pressao de bolha, (psia)
t= temperatura em que ser quer calcular a razao de solibilidade, (F)
API =  densidade oleo, grau API, (API)
psp = pressao no separador, (psia)
tsp = Temperatura no separador, (F)

SAIDA:
```

```
Razao de solubilidade (scf/STB) */

fstream fin;
fin.open("dados_razao_solub_VB.dat");


while(!fin.eof())
{
  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;

   double API;
  fin >> API;
  cout<<API<<endl;

  double psp;
  fin >> psp;
  cout<<psp<<endl;

  double tsp;
  fin >> tsp;
  cout<<tsp<<endl;

//entrada de dados
/*cout<<"Entre com o valor da densidade do gas"<<endl;
cin>>gammag;  cin.get();

cout<<"Entre com o valor da pressao em que se quer calcular a razao de solubilidade"<<
    endl;
cin>>p;  cin.get();

cout<<"Entre com o valor da pressao de bolha"<<endl;
cin>>pb;  cin.get();

cout<<"Entre com o valor da temperatura em que se quer calcular a razao de solubilidade
    "<<endl;
cin>>t;  cin.get();

cout<<"Entre com o valor do grau API do oleo"<<endl;
cin>>API;  cin.get();

cout<<"Entre com o valor da pressao no separador"<<endl;
cin>>psp;  cin.get();

cout<<"Entre com o valor da temperatura no separador"<<endl;
cin>>tsp;  cin.get();
*/
```

```
double C1;
double C2;
double C3;

//contantes do grau API
 if(API <= 30)
     {
     C1=0.0362;
     C2= 1.0937;
     C3= 25.7240;
     }
 else
     {
     C1=0.0178;
     C2= 1.1870;
     C3= 23.931;
     }

     double razaodesolubilidade;
     double gammagsp;

 //teste da pressao
 if(p>pb)
{
     gammagsp= gammag*(1 + 5.912*pow(10,-5)*(API)*tsp*log10(psp/114.7));

       razaodesolubilidade = C1*gammagsp*pow(pb,C2)*exp(C3*(API/(t+460)));
}
        else
        {
        gammagsp= gammag*(1 + 5.912*pow(10,-5)*(API)*tsp*log10(psp/114.7));

        razaodesolubilidade = C1*gammagsp*pow(p,C2)*exp(C3*(API/(t+460)));
        }

        //mostra na tela o resultado da razao de solubilidade
  cout<<"O valor da razao de solubilidade e:"<<razaodesolubilidade<<endl;
}
}
```

Apresenta-se na listagem 6.51 o arquivo com código da classe CCompressibilidade.

Listing 6.51: Arquivo de cabeçalho da classe CCompressibilidade.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CCompressibilidade_h
#define CCompressibilidade_h

class CCompressibilidade

{
protected:

//atributos comuns a todas as classes derivadas da classe base CCompressibilidade

double pressao; //pressao atual
double pressaoBolha;//pressao de bolha

};
```

```
#endif
```

Apresenta-se na listagem 6.52 o arquivo com código da classe CCompressibilidade.

Listing 6.52: Arquivo de implementação da classe CCompressibilidade.

```
#include "CCompressibilidade.h"
```

Apresenta-se na listagem 6.53 o arquivo com código da classe CCompressibilidadeMaiorPB_Setal.

Listing 6.53: Arquivo de cabeçalho da classe CCompressibilidadeMaiorPBSetal.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CCompressibilidadeMaiorPB_Setal_h
#define CCompressibilidadeMaiorPB_Setal_h

#include "CCompressibilidade.h"

class CCompressibilidadeMaiorPB_Setal : public CCompressibilidade

{
protected:

//atributos pertencentes somente a classe derivada CCompressibilidadeMaiorPB_Setal

double rsb; //razao de solubilidade
double gammagSP; //densidade do gas no separador
double tr; //temperatura reduzida
double API; //grau API

public:

/*metodo que calcula a compressibilidade para pressoes maiores que a pressao de
bolha pelo metodo de Spivey et al. */

void CompressibilidadeMaiorPB_Setal();

};

#endif
```

Apresenta-se na listagem 6.54 o arquivo com código da classe CCompressibilidadeMaiorPB_Setal.

Listing 6.54: Arquivo de implementação da classe CCompressibilidadeMaiorPBSetal.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CCompressibilidadeMaiorPB_Setal.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CCompressibilidadeMaiorPB_Setal::CompressibilidadeMaiorPB_Setal()

{
/* Coeficiente de compressibilidade isotermica do oleo(compressibilidade
```

```
do oleo),
Co_maiorpbSetal calcula a compressibilidade do oleo,utilizando o metodo de
Spivey, John and Valko, Peter and McCain, William, "Applications of the
Coefficient of Isothermal Compressibility to Various Reservoir Situations
With New Correlations for Each Situation"(2007)

Entrada:

p = pressao atual (psia)
pb = pressao de bolha, (psia)
rsb= razao de solubilidade no ponto de bolha(scf/STB)
gammagSP= densidade do gas no separador
tr = Temperatura do reservatorio, (F)
API =  densidade oleo, grau API, (API)

SAIDA:

Co= compressibilidade do oleo,(1/psi) */

//Calculo comum a todos os metodos

fstream fin;
fin.open("dados_comp_maior_setal.dat");


while(!fin.eof())
{
  double API;
  fin >> API;
  cout<<API<<endl;

  double gammagSP;
  fin>> gammagSP;
  cout<<gammagSP<<endl;

  double rsb;
  fin >> rsb;
  cout<<rsb<<endl;

  double tr;
  fin >> tr;
  cout<<tr<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;

/*
cout<<"Entre com o valor do grau API do oleo."<<endl;
double API;
cin>>API;  cin.get();

cout<<"Entre com o valor da densidade do gas no separador."<<endl;
double gammagSP;
cin>>gammagSP;  cin.get();
```

```
cout<<"Entre com o valor da razao de solubilidade."<<endl;
double rsb;
cin>>rsb;  cin.get();

cout<<"Entre com o valor da temperatura no reservatorio."<<endl;
double tr;
cin>>tr;  cin.get();

cout<<"Entre com o valor da pressao de bolha."<<endl;
double pb;
cin>>pb;  cin.get();

cout<<"Entre com o valor da pressao atual."<<endl;
double p;
cin>>p;  cin.get(); */

//teste do valor da pressao

if( p <= pb )
{
cout<<"p␣<=␣pb,␣O␣programa␣calcula␣a␣compressibilidade␣do␣oleo␣em␣pressoes␣maiores␣que␣a␣
    pressÃ£o␣de␣bolha"<<endl;
}

//inicializacao das variaveis

double VAR_1 = log (API);
double VAR_2 = log (gammagSP);
double VAR_3= log(pb);
double VAR_4= log(p/pb);
double VAR_5 = log(rsb);
double VAR_6 = log (tr);

double C0_1=3.011;
double C0_2=-0.0835;
double C0_3=3.51;
double C0_4=0.327;
double C0_5=-1.918;
double C0_6=2.52;

double C1_1=-2.6254;
double C1_2=-0.259;
double C1_3=-0.0289;
double C1_4=-0.608;
double C1_5=-0.642;
double C1_6=-2.73;

double C2_1= 0.497;
double C2_2= 0.382;
double C2_3= -0.0584;
double C2_4= 0.0911;
double C2_5= 0.154;
double C2_6=0.429;


//calculo de cofb

double Z[5];

    Z[0]= C0_1+C1_1*VAR_1+C2_1*(VAR_1*VAR_1);
```

```
    Z[1]= C0_2+C1_2*VAR_2+C2_2*(VAR_2*VAR_2);

    Z[2]= C0_3+C1_3*VAR_3+C2_3*(VAR_3*VAR_3);

    Z[3]= C0_4+C1_4*VAR_4+C2_4*(VAR_4*VAR_4);

    Z[4]= C0_5+C1_5*VAR_5+C2_5*(VAR_5*VAR_5);

    Z[5]= C0_6+C1_6*VAR_6+C2_6*(VAR_6*VAR_6);

double z = Z[0] + Z[1] + Z[2] + Z[3] + Z[4] + Z[5];

double cofb = exp(2.434 +0.475*z +0.048*(z*z)-log(pow(10,6)));



            double dzdp = (-0.608 + 0.1822*log(p/pb))/p;

            double  dcofbdp = cofb*(0.475 +0.096*z)*dzdp;

            //calculo da compressibilidade do oleo

            double compressibilidadeoleo = cofb + (p-pb)*dcofbdp;

cout<<"A compressibilidade do oleo para pressoes maiores que a pressao de bolha pelo
    metodo de Spivey, John and Valko, Peter and McCain, William e:"<<
    compressibilidadeoleo <<endl;

}
}
```

Apresenta-se na listagem 6.55 o arquivo com código da classe CCompressibilidadeMaiorPB_WB.

Listing 6.55: Arquivo de cabeçalho da classe CCompressibilidadeMaiorPBWB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CCompressibilidadeMaiorPB_WB_h
#define CCompressibilidadeMaiorPB_WB_h

#include "CCompressibilidade.h"

class CCompressibilidadeMaiorPB_WB : public CCompressibilidade

{
//atibuto pertencente somente a classe derivada CCompressibilidadeMaiorPB_WB

protected:

double rhoob; //massa especifica oleo na pressao de bolha

public:

/*metodo calcula a compressibilidade do oleo para pressoes acima
da pressao de bolha pelo metodo de Whitson e Brule. */

void CompressibilidadeMaiorPB_WB();

};
```

```
#endif
```

Apresenta-se na listagem 6.56 o arquivo com código da classe CCompressibilidadeMaiorPB_WB.

Listing 6.56: Arquivo de implementação da classe CCompressibilidadeMaiorPBWB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CCompressibilidadeMaiorPB_WB.h"

#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CCompressibilidadeMaiorPB_WB::CompressibilidadeMaiorPB_WB()

{
/* Coeficiente de compressibilidade isotermica do oleo(compressibilidade
do oleo) em pressoes de reservatorio maiores que a pressao de bolha
utilizando o metodo de Whitson, C.H. and Brule, M.R., Phase behavior,
(2000), que correlaciona a compressibilidade isotermica com a pressao,
pressao de bolha, e massa especifica na pressao de bolha.

Entrada:

p = pressao atual (psia)
pb = pressao de bolha (psia)
rhoob = massa especifica oleo na pressao de bolha (lb/ cu ft)

SAIDA:

Co= compressibilidade do oleo,(1/psi) */

//entrada de dados

fstream fin;
fin.open("dados_comp_maior_wb.dat");


while(!fin.eof())
{
  double rhoob;
  fin >> rhoob;
  cout<<rhoob<<endl;

  double pb;
  fin >> pb;
  cout<<pb<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;


/* cout<<"Entre com o valor da massa especifica do oleo na pressao de bolha"<<endl;
double rhoob;
cin>>rhoob;  cin.get();
```

```
cout<<"Entre com o valor da pressao de bolha"<<endl;
double pb;
cin>>pb;  cin.get();

cout<<"Entre com o valor pressao atual"<<endl;
double p;
cin>>p;  cin.get(); */

double compressibilidadeoleowb;

//teste do valor da pressao

if( p <= pb)
{
 cout<<"p␣<=␣pb,␣O␣programa␣calcula␣a␣compressibilidade␣do␣oleo␣em␣pressoes␣maiores␣que␣a
     ␣pressao␣de␣bolha"<<endl;
}
else
{
 compressibilidadeoleowb = (pow(10,(-6)))*exp((rhoob + 0.00434*(p-pb)-79.1)/(0.0007141*(p
     -pb)-12.938));
}

cout<<"A␣compressibilidade␣do␣oleo␣para␣pressoes␣maiore␣que␣a␣pressao␣de␣bolha␣pelo␣
    metodo␣de␣Whitson,␣C.H.␣and␣BrulÃ©,␣M.R.␣e:"<<compressibilidadeoleowb<<endl;
}
}
```

Apresenta-se na listagem 6.57 o arquivo com código da classe CCompressibilidadeMenorPB.

Listing 6.57: Arquivo de cabeçalho da classe CCompressibilidadeMenorPB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CCompressibilidadeMenorPB_h
#define CCompressibilidadeMenorPB_h

#include "CCompressibilidade.h"

class CCompressibilidadeMenorPB:public CCompressibilidade

{
protected:

//atributos pertencentes somente a classe derivada CCompressibilidadeMenorPB

double rsb; //razao de solubilidade
double t; //temperatura atual
double API; //grau API

public:

/*metodo calcula a compressibilidade do oleo para pressoes abaixo da pressao
de bolha pelo metodo de McCain Jr et al.*/

void CompressibilidadeMenorPB();

};
```

```
#endif
```

Apresenta-se na listagem 6.58 o arquivo com código da classe CCompressibilidadeMenorPB.

Listing 6.58: Arquivo de implementação da classe CCompressibilidadeMenorPB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CCompressibilidadeMenorPB.h"
#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

/*Coeficiente de compressibilidade isotermica do oleo(compressibilidade
do oleo) em pressoes de reservatorio menores ou igual a pressao de bolha-
utilizando o metodo de McCainJr, W. and Rollins, J. and Lanzi, Alejandro,
J., The coefficient of isothermal compressibility of black oils at
pressures below the bubblepoint, SPE formation evaluation 3, 3 (1988),
pp. 659--662.

Entrada:

pressao = pressao atual (psia)
pressaoBolha= pressao de bolha
rsb= razao de solubilidade no ponto de bolha (scf/STB)
t= temperatura (F)
API= densidade API do oleo

Saida:

Co= compressibilidade do oleo (1/psia)*/


void CCompressibilidadeMenorPB::CompressibilidadeMenorPB()

{

//entrada de dados

fstream fin;
fin.open("dados_comp_menor.dat");


while(!fin.eof())
{
  double pressao;
  fin >> pressao;
  cout<<pressao<<endl;

  double pressaoBolha;
  fin >> pressaoBolha;
  cout<<pressaoBolha<<endl;

  double t;
  fin >> t;
  cout<<t<<endl;

  double API;
```

```
  fin >> API;
  cout<<API<<endl;

  double rsb;
  fin >> rsb;
  cout<<rsb<<endl;

/*cout<<"Entre com o valor da pressao atual."<<endl;
double pressao;
cin>>pressao; cin.get();

cout<<"Entre com o valor da pressao de bolha."<<endl;
double pressaoBolha;
cin>>pressaoBolha; cin.get();

cout<<"Entre com o valor da temperatura."<<endl;
double t;
cin>>t; cin.get();

cout<<"Entre com o valor do grau API."<<endl;
double API;
cin>>API; cin.get();

cout<<"Entre com o valor da razao de solubilidade."<<endl;
double rsb;
cin>>rsb; cin.get(); */

double A;

//teste do valor da pressao

if (pressao >= pressaoBolha)
{
cout<<"error('␣p=>␣pb,␣A␣funÃ§Ã£o␣Co_menorpbMR␣calcula␣a␣compressibilidade␣do␣oleo␣em␣
    pressoes␣menores␣que␣a␣pressao␣de␣bolha')"<<endl;
}
else
{
  A = -7.573-1.45*log(pressao)-0.383*log(pressaoBolha)+1.402*log(t+459.67)+
    0.256*log(API)+0.449*log(rsb);
}

double compressibilidadeoleo = exp(A);

cout<<"O␣valor␣da␣compressibilidade␣do␣oleo␣e:␣"<<"␣␣␣"<<compressibilidadeoleo<<endl;

}
}
```

Apresenta-se na listagem 6.59 o arquivo com código da classe CFluidoGas.

Listing 6.59: Arquivo de cabeçalho da classe CFluidoGas.

```
#ifndef CFluidoGas_h
#define CFluidoGas_h

#include "CFluido.h"

class CFluidoGas : public CFluido

{
```

```
//atributos pertencentes somente a classe derivada CFluidoGas
protected:

double propPseudocriticas; //propriedades pseudocriticas (temperatura e pressao)
double fatorCompZ; //fator de compressibilidade Z

//metodos que possuem uma unica forma de resolver
public:

  //metodo calcula a compressibilidade do gas
void Compressibilidade();

  //metodo calcula o fator volume formacao do gas
void FatorVolFormacao();

 //metodo calcula a viscosidade do gas
void Viscosidade();

 //metodo calcula as propriedades pseudocriticas do gas
void PropPseudocriticas();

 //metodo calcula o fator de compressibilidade Z do gas
void FatorCompZ();

};

#endif
```

Apresenta-se na listagem 6.60 o arquivo com código da classe CFluidoGas.

Listing 6.60: Arquivo de implementação da classe CFluidoGas.

```
#include "CFluidoGas.h"
#include <cmath>

#include <iostream>
#include <fstream>

using namespace std;

void CFluidoGas::Compressibilidade()
{
/* Funcao Cg, calcula a compressibilidade dos gases naturais, utilizando o
 o metodo de Mattar, L. and Brar, G. and Aziz, K., Compressibility of
 natural gases vol. 14, no. 4 (Society of Petroleum Engineers, 1975), que
calcula a compressibilidade reduzida do gas e depois a converte na
compressibilidade do gas multiplicando pela pressao pseudocritica.*/

//inicializacao de variaveis
double a1  = +0.32650;
double a2  = -1.07000;
double a3  = -0.53390;
double a4  = +0.01569;
double a5  = -0.05165;
double a6  = +0.54750;
double a7  = -0.73610;
double a8  = +0.18440;
double a9  = +0.10560;
double a10 = +0.61340;
double a11 = +0.72100;
```

```
fstream fin;
fin.open("dados_fluido_gas_compressibilidade.dat");


while(!fin.eof())
{
  double ppr;
  fin >> ppr;
  cout<<ppr<<endl;

  double Tpr;
  fin >> Tpr;
  cout<<Tpr<<endl;

  double Z;
  fin >> Z;
  cout<<Z<<endl;

   double ppc;
  fin >> ppc;
  cout<<ppc<<endl;



//entrada de dados
/*cout<<"Entre com o valor da pressao pseudoreduzida do gas ."<<endl;
cin>>ppr; cin.get();

cout<<"Entre com o valor da temperatura pseudoreduzida do gas ."<<endl;
cin>>Tpr; cin.get();

cout<<"Entre com o valor do fator de compressibilidade Z."<<endl;
cin>>Z; cin.get();
*/
//calculo da massa especifica  do gas pseudo-reduzida
 double ropr = 0.27*( ppr/( Z* Tpr));


//calculo da derivada (dz/dro)Tpr utilizando o  metodo Dranchuk e Abu-Kassem

   double dzdro_Tro = a1 + a2/Tpr + a3/(Tpr*Tpr*Tpr)+ a4/(Tpr*Tpr*Tpr*Tpr)+ a5/(Tpr*Tpr*
       Tpr*Tpr*Tpr*Tpr)+
    2*ropr*(a6 + a7/Tpr +a8/(Tpr*Tpr)) -
    5*(ropr*ropr*ropr*ropr)*a9*(a7/Tpr + a8/(Tpr*Tpr)) +
    (2*a10*ropr/(Tpr*Tpr*Tpr))*(1+a11*(ropr*ropr)- (a11*a11)*(ropr*ropr*ropr*ropr))*
    exp(-a11*(ropr*ropr));

    //calculo da variavel F_Cgr
 double F_Cgr = 1/ppr - (0.27/((Z*Z)*Tpr))*(dzdro_Tro/(1+(ropr/Z)+ dzdro_Tro));

 //entrada de dado
/* cout<<"Entre com o valor da pressao pseudocritica."<<endl;
 cin>>ppc; cin.get();
*/
 //calculo da compressibiidade
 double Compressibilidadegas = F_Cgr/ppc;

 //mostra na tela o resultado da compressibilidade
cout<<"O valor da compressibilidade do gas e:"<<Compressibilidadegas<<endl;
```

```
}
}

void CFluidoGas::FatorVolFormacao()
{
// Bg calcula o fator volume formacao do gas(Bg), utilizando a equacao que
//define esta propriedade. Esta funcao estima Bg em funcao da temperatura,
//pressao e fator de compressibilidade do gas.


 //inicializacao de variavel
double const c=0.02827;

fstream fin;
fin.open("dados_fluido_gas_fator_vol_form.dat");


while(!fin.eof())
{
  double Z;
  fin >> Z;
  cout<<Z<<endl;

  double T;
  fin >> T;
  cout<<T<<endl;

  double P;
  fin >> P;
  cout<<P<<endl;


//entrada de dados
/*cout<<"Entre com o valor do fator de compressibilidade Z."<<endl;
cin>>Z; cin.get();

cout<<"Entre com o valor da temperatura em rankine."<<endl;
cin>>T; cin.get();

cout<<"Entre com o valor da pressao em psia."<<endl;
cin>>P; cin.get();
*/
//calculo do fator de compressibilidade
double Fator_Volume_Formacao_gas = c*Z*T/P;

//mostra na tela o resultado do fator volume formacao
cout<<"O valor do fator volume formacao do gas e:"<<Fator_Volume_Formacao_gas;

}
}

void CFluidoGas::Viscosidade()

{
//muLGE calcula a viscosidade do gas utilizando o metodo de Lee, A. and
//Gonzalez, M. and Eakin, B., The viscosity of natural gases vol. 18, no.
//8(1966).

fstream fin;
fin.open("dados_fluido_gas_viscosidade.dat");
```

```
while(!fin.eof())
{
  double T;
  fin >> T;
  cout<<T<<endl;

  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double p;
  fin >> p;
  cout<<p<<endl;

  double Z;
  fin >> Z;
  cout<<Z<<endl;


 //entrada de dados
/*cout<<"Entre com o valor da temperatura em rankine."<<endl;
cin>>T; cin.get();
*/
//conversao da temperatura para rankine
double Trank = T + 459.67;

//cÃ¡lculo dp peso molecular aparente

/*cout<<"Entre com o valor da densidade do gas"<<endl;
cin>>gammag; cin.get();
*/
//calculo da variavel pmga
double pmga = gammag*28.96;

//entrada de dados
/*cout<<"Entre com o valor da pressao em psia."<<endl;
cin>>p; cin.get();

cout<<"Entre com o valor do fator de compressibilidade Z."<<endl;
cin>>Z; cin.get();
*/
//calculo da variavel rhog
double rhog= (p*pmga/(10.73*Z*(Trank)));

//parametros do metodo
double X=3.448 + 986.4/Trank + 0.010009*pmga;

double Y=2.447  -0.2224*X;

double K=(9.379 + 0.01607*pmga)*(pow(Trank,1.5))/(209.2 + 19.26*pmga + Trank);

//calculo da viscosidade do gas
double viscosidade_gas= (0.0001)* K * exp((pow(X*(rhog/62.428),Y)));

//mostra na tela o resultado da viscosidade
cout<<"O valor da viscosidade e:"<< viscosidade_gas <<endl;

}
```

```
}
```

Apresenta-se na listagem 6.61 o arquivo com código da classe CPropPseudocriticas.

Listing 6.61: Arquivo de cabeçalho da classe CPropPseudocriticas.

```
#ifndef CPropPseudocriticas_h
#define CPropPseudocriticas_h

class CPropPseudocriticas
{
//atributo comum a todas as classes derivadas da classe base CPropPseudocriticas
protected:

double gammag; //densidade do gas
};

#endif
```

Apresenta-se na listagem 6.62 o arquivo com código da classe CPropPseudocriticas.

Listing 6.62: Arquivo de implementação da classe CPropPseudocriticas.

```
#include "CPropPseudocriticas.h"
```

Apresenta-se na listagem 6.63 o arquivo com código da classe CPropPseudocriticas_ST.

Listing 6.63: Arquivo de cabeçalho da classe CPropPseudocriticasST.

```
#ifndef CPropPseudocriticas_ST_h
#define CPropPseudocriticas_ST_h

#include "CPropPseudocriticas.h"

class CPropPseudocriticas_ST: public CPropPseudocriticas
{
 //atributos pertencentes somente a classe derivada CPropPseudocriticas_ST
protected:

double tpc; //temperatura pseudocritica
double ppc; //pressao pseudocritica


public:

//metodo calcula propriedades pseudocriticas pelo metodo de Standing
void PropPseudocriticas_ST();

};

#endif
```

Apresenta-se na listagem 6.64 o arquivo com código da classe CPropPseudocriticas_ST.

Listing 6.64: Arquivo de implementação da classe CPropPseudocriticasST.

```
#include "CPropPseudocriticas_ST.h"
#include <iostream>
#include <fstream>

using namespace std;
```

```
void CPropPseudocriticas_ST::PropPseudocriticas_ST()
{
/*Propriedades pseudocriticas de gases naturais puros, utilizando o metodo
de Standing, M.B., Volumetric and Phase Behavior of Oil Field Hydrocarbon
Systems, 1977.

PropcST retorna os valores de ppc e Tpc para gases naturais puros
, utilizando as equacoes de STANDING (1977).*/

fstream fin;
fin.open("dados_prop_pseudocritica_ST.dat");


while(!fin.eof())
{
  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  int tipodegas;
  fin >> tipodegas;
  cout<<tipodegas<<endl;



//entrada de dados
/*cout<<"Entre com o valor da densidade do gas"<<endl;
cin>>gammag; cin.get();
*/


 while ( true )
 {
/* cout<<"Entre com o tipo de gas que deseja.Digite 1 para gas seco e 2 para gas umido
    ."<<endl;
 cin>>tipodegas; cin.get();
*/
 switch ( tipodegas )
 {
 case 1: tpc = 168 + 325*gammag - 12.5*gammag*gammag;  //gas seco
         ppc = 677 + 15*gammag - 37.5*gammag*gammag;  // gas seco
         break ;

 case 2:  tpc = 187 + 330 * gammag - 71.5 * gammag*gammag ; //gas umido
          ppc = 706 - 51.7*gammag - 11.1* (gammag*gammag) ; //gas umido
         break ;

 default :  tpc = 168 + 325*gammag - 12.5*gammag*gammag;  //gas seco
            ppc = 677 + 15*gammag - 37.5*gammag*gammag ; //gas seco
           break ;
 }

 }
 //mostra na tela os resultados das propriedades pseudocriticas
 cout<<"Os␣valores␣da␣pressao␣e␣temperatura␣pseudocriticas␣sao␣respectivamente:"<<ppc<<"␣
     ␣␣"<<tpc<<endl;

 }
}
```

Apresenta-se na listagem 6.65 o arquivo com código da classe CPropPseudocriticas_Piper.

Listing 6.65: Arquivo de cabeçalho da classe CPropPseudocriticasPiper.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CPropPseudocriticas_Piper_h
#define CPropPseudocriticas_Piper_h

#include "CPropPseudocriticas.h"

class CPropPseudocriticas_Piper : public CPropPseudocriticas

{

//atributos pertencentes somente a classe derivada CPropPseudocriticas_Piper
protected:

double tpcc; //temperatura pseudocritica
double ppcc; //pressao pseudocritica
double yN2; //porcentagem molar de N2
double yH2S; //porcentagem molar de H2S
double yCO2; //porcentagem molar de CO2
double PcN2; //pressao critica de N2
double PcH2S; //pressao critica de H2S
double PcCO2; //pressao critica de CO2
double TcN2; //temperatura critica de N2
double TcH2S; //temperatura critica de H2S
double TcCO2; //temperatura critica de CO2

public:

  //metodo calcula as propriedades pseudocriticas pelo metodo de Piper et al
void PropPseudocriticas_Piper();

};

#endif
```

Apresenta-se na listagem 6.66 o arquivo com código da classe CPropPseudocriticas_Piper.

Listing 6.66: Arquivo de implementação da classe CPropPseudocriticasPiper.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CPropPseudocriticas_Piper.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

void CPropPseudocriticas_Piper::PropPseudocriticas_Piper()
{
/*Correcao para calcular as propriedades pseudocriticas de gases com
contaminates utilizando metodo - Piper, L.D AND McCain, Jr. and Corredor,
J.H,, "Compressibility Factors for Naturally Occurring Petroleum Gases.
Gas Reservoir Engineering. Reprint Series", v.52 (1999), pp. p.186-200.
```

```
Este metodo correlaciona as Propriedade de pseudocriticas do gas com a
densidade do gas, porcentagem de contaminantes, e temperatura e pressao
critica dos contaminantes.


 gammag = densidade do gas (hidrocarbonetos gasosos e contaminates)
 yN2 =  porcentagem molar de N2 (ex: 35% entre com 0.35)
 yH2S = porcentagem molar de H2S
 yCO2 = pocentagem molar de CO2
 pcN2 = pressao critica do N2 (psia)
 pcH2S = pressao critica do H2S (psia)
 pcCO2 = pressao critica do CO2 (psia)
 TcN2 = Temperatura critica de N2 (R)
 TcH2S = Temperatura critica de H2S (R)
 TcCO2 = Temperatura critica de CO2 (R)

retorna:

Ppcc= pressao pseudo-critica com correcao de contaminates (psia)
Tpcc= temperatura pseudo-critica com correcao de contaminates (R) */

//inicializacao de variaveis
double a0= 1.1582*(0.1);
double a1= -4.5820*(0.1);
double a2= -9.0348*(0.1);
double a3= -6.6026*(0.1);
double a4= 7.0729*(0.1);
double a5= -9.9397*(0.01);

double b0= 3.8216;
double b1= -6.5340*(0.01);
double b2= -4.2113*(0.1);
double b3= -9.1249*(0.1);
double b4= 1.7438*(0.1);
double b5= -3.2191;

fstream fin;
fin.open("dados_prop_pseudocritica_Piper.dat");


while(!fin.eof())
{
  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double yH2S;
  fin >> yH2S;
  cout<<yH2S<<endl;

  double yCO2;
  fin >> yCO2;
  cout<<yCO2<<endl;

  double yN2;
  fin >> yN2;
  cout<<yN2<<endl;

  double PcH2S;
  fin >> PcH2S;
```

```
  cout<<PcH2S<<endl;

  double PcCO2;
  fin >> PcCO2;
  cout<<PcCO2<<endl;

  double PcN2;
  fin >> PcN2;
  cout<<PcN2<<endl;

  double TcH2S;
  fin >> TcH2S;
  cout<<TcH2S<<endl;

  double TcCO2;
  fin >> TcCO2;
  cout<<TcCO2<<endl;

  double TcN2;
  fin >> TcN2;
  cout<<TcN2<<endl;

//entrada de dados
/*cout<<"Entre com o valor da densidade do gas.";
cin>>gammag; cin.get();

cout<<"Entre com o valor da porcentagem molar do H2S.";
cin>>yH2S; cin.get();

cout<<"Entre com o valor da porcentagem molar do CO2.";
cin>>yCO2; cin.get();

cout<<"Entre com o valor da porcentagem molar do N2.";
cin>>yN2; cin.get();

cout<<"Entre com o valor da pressao critica do H2S.";
cin>>PcH2S; cin.get();

cout<<"Entre com o valor da pressao critica do CO2.";
cin>>PcCO2; cin.get();

cout<<"Entre com o valor da pressao critica do N2.";
cin>>PcN2; cin.get();

cout<<"Entre com o valor da temperatura critica do H2S.";
cin>>TcH2S; cin.get();

cout<<"Entre com o valor da temperatura critica do CO2.";
cin>>TcCO2; cin.get();

cout<<"Entre com o valor da temperatura critica do N2.";
cin>>TcN2; cin.get();
*/
//calculo de j
double J = a0 + a1*yH2S*(TcH2S/PcH2S)+ a2*yCO2*(TcCO2/PcCO2) + a3*yN2*(TcN2/PcN2) + a4*
    gammag + a5*gammag*gammag;

//calculo de k
double K = b0 + b1*yH2S*(TcH2S/(pow(PcH2S,(1/2))))+ b2*yCO2*(TcCO2/(pow(PcCO2,(1/2)))) +
    b3*yN2*(TcN2/(pow(PcN2,(1/2)))) + b4*gammag + b5*gammag*gammag;
```

```
//calculo da temperatura pseudocritica
double tpcc = (K*K)/J;

//calculo da pressao pseudocriticas
double ppcc = tpcc/J ;

//mostra na tela o resultado das propriedades pseudocriticas
cout<<"O valor da temperatura e pressao pseudocriticas com correcao de contaminantes sao
    respectivamente:"<<tpcc<<"    "<<ppcc<<endl;


}
}
```

Apresenta-se na listagem 6.67 o arquivo com código da classe CPropPseudocriticas_WA.

Listing 6.67: Arquivo de cabeçalho da classe CPropPseudocriticasWA.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CPropPseudocriticas_WA_h
#define CPropPseudocriticas_WA_h

#include "CPropPseudocriticas_Piper.h"

class CPropPseudocriticas_WA : public CPropPseudocriticas_Piper

{
 //a classe so possui atributos da classe base

public:

  //metodo calcula propriedades pseudocriticas pelo metodo de Wichert e Aziz
  void PropPseudocriticas_WA();

};

#endif
```

Apresenta-se na listagem 6.68 o arquivo com código da classe CPropPseudocriticas_WA.

Listing 6.68: Arquivo de implementação da classe CPropPseudocriticasWA.

```
//trabalho_programacao_pratica
//dupla_alex_aline

/*Correcao para calcular as propriedades pseudocriticas de gases com
contaminates utilizando metodo de  Wichert, E. AND Aziz, K, Calculation
of Z's for Sour Gases vol.51, 1972

 Este metodo correlaciona as Propriedade de pseudocriticas do gas com a
densidade do gas, porcentagem de contaminantes.

 yN2 =  porcentagem molar de N2
 yH2S = porcentagem molar de H2S
 yCO2 = pocentagem molar de CO2
 gammag = densidade da mistura de hidrocarbonetos gasosos e contaminates

retorna:
ppcc= pressao pseudo-critica com correcao de contaminates (psia)
```

```
tpcc= temperatura pseudo-critica com correcao de contaminates (R)*/

#include "CPropPseudocriticas_WA.h"
#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

void CPropPseudocriticas_WA :: PropPseudocriticas_WA()
{

fstream fin;
fin.open("dados_prop_pseudocritica_WA.dat");


while(!fin.eof())
{
  double gammag;
  fin >> gammag;
  cout<<gammag<<endl;

  double yH2S;
  fin >> yH2S;
  cout<<yH2S<<endl;

  double yCO2;
  fin >> yCO2;
  cout<<yCO2<<endl;

  double yN2;
  fin >> yN2;
  cout<<yN2<<endl;

   double ppc;
  fin >> ppc;
  cout<<ppc<<endl;

  double Tpc;
  fin >> Tpc;
  cout<<Tpc<<endl;


//entrada de dados
/*cout<<"Entre com o valor da densidade do gas.";
cin>>gammag; cin.get();

cout<<"Entre com o valor da porcentagem molar do H2S.";
cin>>yH2S; cin.get();

cout<<"Entre com o valor da porcentagem molar do CO2.";
cin>>yCO2; cin.get();

cout<<"Entre com o valor da porcentagem molar do N2.";
cin>>yN2; cin.get();
*/
//densidade de hidrocarbonetos

double dg = (gammag - 0.967 * yN2 - 1.52*yCO2-1.18*yH2S)/ (1.0- yN2 -yCO2 - yH2S);
```

```
//propriedades dos hidrocarbonetos
//entrada de dados
/*cout<<"Entre com o valor da pressÃfo pseudocritica.";
cin>>ppc; cin.get();
cout<<"Entre com o valor da temperatura pseudocritica.";
cin>>Tpc; cin.get();
*/
//propriedades da mistura

double ppcm = (1.0- yN2 -yCO2 - yH2S)*ppc + 493.0*yN2 + 1071.0*yCO2 + 1306.0* yH2S;

double Tpcm = (1.0- yN2 -yCO2 - yH2S)*Tpc + 227.0*yN2 + 548.0*yCO2 + 672.0* yH2S;

//ajustes

double e = 120.0* (((pow((yCO2+yH2S),0.9)) - (pow((yCO2 + yH2S),1.6))) + 15.0*((pow(yH2S
    ,0.5)) -(pow(yH2S,4.0))));

ppcc = (ppcm*( Tpcm - e))/(Tpcm + yH2S*(1.0-yH2S)* e);
tpcc = (Tpcm - e);

//mostra na tela o resultado das propriedades pseudocriticas
cout<<"A␣pressÃf˚o␣e␣a␣temperatura␣pseudocriticas␣com␣correcao␣de␣contaminante␣sao␣
    respectivamente:"
    <<ppcc<<"␣...␣␣"<<tpcc<<endl;

}
}
```

Apresenta-se na listagem 6.69 o arquivo com código da classe CPropPseudocriticas_CKB.

Listing 6.69: Arquivo de cabeçalho da classe CPropPseudocriticasCKB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CPropPseudocriticas_CKB_h
#define CPropPseudocriticas_CKB_h

#include "CPropPseudocriticas_Piper.h"

class CPropPseudocriticas_CKB : public CPropPseudocriticas_Piper

{
public:

 /*metodo calcula propriedades pseudocriticas pelo metodo
  de Carr, N. and Kobayashi, R. and Burrows,D.*/
void PropPseudocriticas_CKB();

};

#endif
```

Apresenta-se na listagem 6.70 o arquivo com código da classe CPropPseudocriticas_CKB.

Listing 6.70: Arquivo de implementação da classe CPropPseudocriticasCKB.

```
#include "CPropPseudocriticas_CKB.h"
#include <iostream>
#include <fstream>
```

```
using namespace std;

void CPropPseudocriticas_CKB :: PropPseudocriticas_CKB()

{
/*Correcao para caulcular as propriedades pseudocriticas de gases com
contaminates utilizando metodo de Carr, N. and Kobayashi, R. and Burrows,
D., Viscosity of hydrocarbon gases under pressure vol. 6 1954.

 Este metodo correlaciona as Propriedade de pseudocriticas do gas com a
densidade do gas, porcentagem de contaminantes.

 yN2 =  porcentagem de N2
 yH2S = porcentagem de H2S
 yCO2 = pocentagem de CO2
 gammag = densidade do gas (hidrocarbonetos gasosos e contaminates)

retorna:

ppcc= pressao pseudo-critica com correcao de contaminates (psia)
Tpcc= temperatura pseudo-critica com correcao de contaminates (R)*/

//propriedades dos hidrocarbonetos

fstream fin;
fin.open("dados_prop_pseudocritica_CKB.dat");


while(!fin.eof())
{
  double ppc;
  fin >> ppc;
  cout<<ppc<<endl;

  double Tpc;
  fin >> Tpc;
  cout<<Tpc<<endl;


//entrada de dados
/*cout<<"Entre com o valor da pressao pseudocritica.";
cin>>ppc; cin.get();
cout<<"Entre com o valor da temperatura pseudocritica.";
cin>>Tpc; cin.get();
*/
//calculo da pressao pseudocritica
double ppcc = ppc + 440*yCO2 + 600*yH2S - 170*yN2;

//calculo da temperatura pseudocritica
double Tpcc = Tpc -80*yCO2 + 130*yH2S - 250*yN2;

//mostra na tela o resultado das propriedades pseudocriticas
cout<<"O␣valor␣da␣pressao␣e␣temperatura␣pseudiocriticas␣com␣correcao␣de␣contaminantes␣sao
    ␣respectivamente:"
    <<"␣␣...␣␣"<<ppcc<<"␣␣...␣␣"<<Tpc<<endl;
}
}
```

Apresenta-se na listagem 6.71 o arquivo com código da classe CFatorCompZ.

Listing 6.71: Arquivo de cabeçalho da classe CFatorCompZ.

```
#ifndef CFatorCompZ_h
#define CFatorCompZ_h

#include <vector>

class CFatorCompZ

{

//atributos comuns a todas as classes derivadas da classe base CFatorCompZ

protected:

  double ppr; //pressao pseudo-reduzida
  double tpr; //temperatura pseudo-reduzida

};

#endif
```

Apresenta-se na listagem 6.72 o arquivo com código da classe CFatorCompZ.

Listing 6.72: Arquivo de implementação da classe CFatorCompZ.

```
#include "CFatorCompZ.h"
```

Apresenta-se na listagem 6.73 o arquivo com código da classe CFatorCompZ_BeB.

Listing 6.73: Arquivo de cabeçalho da classe CFatorCompZBeB.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorCompZ_BeB_H
#define CFatorCompZ_BeB_H

#include "CFatorCompZ.h"

class CFatorCompZ_BeB : public CFatorCompZ

{
//a classe so possui os atributos da classe base

public:

/*metodo calcula o fator de compressibilidade Z pelo metodo
de Brill e Beggs  */

void FatorCompZ_BeB();

};

#endif
```

Apresenta-se na listagem 6.74 o arquivo com código da classe CFatorCompZ_BeB.

Listing 6.74: Arquivo de implementação da classe CFatorCompZBeB.

```
#include "CFatorCompZ_BeB.h"

#include <iostream>
```

```
#include <cmath>
#include <fstream>

using namespace std;

void CFatorCompZ_BeB::FatorCompZ_BeB()

{
/* Programa estima o valor do fator de compressibilidade gas Z com ppr e tpr
utilizando o metodo apresentado po Brill e Beggs (1974)- Two-phase flow in pipes.

ENTRADA:

ppr = pressao pseudo-reduzida
tpr = temperatura pseudo-reduzida

SAIDA:

Z = fator de compressibilidade do gas */

//entrada de dados

fstream fin;
fin.open("dados_fator_comp_Z_BeB.dat");


while(!fin.eof())
{
  double ppr;
  fin >> ppr;
  cout<<ppr<<endl;

  double tpr;
  fin >> tpr;
  cout<<tpr<<endl;

/* cout<<"Entre com o valor da pressao pseudo-reduzida"<<endl;
cin>>ppr;  cin.get();

cout<<"Entre com o valor da temperatura pseudo-reduzida"<<endl;
cin>>tpr;  cin.get(); */

   double A = 1.39*pow(tpr - 0.92,2) - 0.36*tpr - 0.101;
   double  B = (0.62 - 0.23*tpr)*ppr + pow((0.066/(tpr - 0.86) - 0.037)*ppr,2)
        + pow(0.32*ppr,6)/pow(10,(9*(tpr - 1)));
   double  C = 0.132 - 0.32*tpr;
   double D = pow(10,(0.3106 - 0.49*tpr + pow(0.1824*tpr,2)));

   double Z = A + (1 - A)/exp(B) + pow(C*ppr,D);

//mostra na tela o valor do fator de compressibilidade Z

cout<<"O valor do fator de compressibilidade Z e: "<< Z <<endl;

}
}
```

Apresenta-se na listagem 6.75 o arquivo com código da classe CFatorCompZ_Lond.

Listing 6.75: Arquivo de cabeçalho da classe CFatorCompZLond.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorCompZ_Lond_h
#define CFatorCompZ_Lond_h

#include "CFatorCompZ.h"

class CFatorCompZ_Lond : public CFatorCompZ
{
 //a classe possui apenas os atributos da classe derivada CFatorCompZ

public:

/* metodo calcula fator de compressibilidade pelo metodo de Londono et al*/

void FatorCompZ_Lond();

};

#endif
```

Apresenta-se na listagem 6.76 o arquivo com código da classe CFatorCompZ_Lond.

Listing 6.76: Arquivo de implementação da classe CFatorCompZLond.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CFatorCompZ_Lond.h"

#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

void CFatorCompZ_Lond::FatorCompZ_Lond()

{
/*
londono extima o valor do fator de compressibilidade gas Z com ppr e tpr
utilizando o metodo apresentado por Londono et al (2002) - Simplified
Correlations for Hydrocarbon Gas Viscosity and Gas Density - Validation
and Correlation of Behavior Using a Large-Scale Database.

ENTRADA:

zlondono(ppr,Tpr)

ppr = pressao pseudo-reduzida
Tpr = temperatura pseudo-reduzida

SAIDA:

Z = fartor de compressibilidade do gas


*/
```

```
//inicilizacao de variavel

double A1  = +0.3024696;
double A2  = -1.046964;
double A3  = -0.1078916;
double A4  = -0.7694186;
double A5  = 0.1965439;
double A6  = +0.6527819;
double A7  = -1.118884;
double A8  = +0.3951957;
double A9  = +0.09313593;
double A10 = +0.8483081;
double A11 = +0.7880011;

//entrada de dados

fstream fin;
fin.open("dados_fator_comp_Z_Lond.dat");


while(!fin.eof())
{
  double ppr;
  fin >> ppr;
  cout<<ppr<<endl;

  double tpr;
  fin >> tpr;
  cout<<tpr<<endl;

/* cout<<"Entre com o valor da pressao pseudo-reduzida"<<endl;
cin>>ppr;  cin.get();

cout<<"Entre com o valor da temperatura pseudo-reduzida"<<endl;
cin>>tpr;  cin.get(); */

//calculo do fator de compressibilidade
double  F = (A1 + A2/tpr + A3 / pow(tpr,3) + A4/pow(tpr,4) + A5/pow(tpr,5))*(0.27*ppr/(
    tpr))
          + (A6 + A7/tpr + A8/pow(tpr,2))*pow((0.27*ppr/(tpr)),2)
          - A9*(A7/tpr + A8/pow(tpr,2))*pow((0.27*ppr/(tpr)),5)
          + A10*(1 + A11*pow((0.27*ppr/(tpr)),2))*(pow((0.27*ppr/(tpr)),2)/pow(tpr,3))
          * exp( -A11 * pow((0.27*ppr/(tpr)),2) );

        //mostra na tela o valor do fator de compressibilidade
 cout<<"O valor do fator de compressibilidade e"<< F <<endl;

}
}
```

Apresenta-se na listagem 6.77 o arquivo com código da classe CFatorCompZ_DeK.

Listing 6.77: Arquivo de cabeçalho da classe CFatorCompZDeK.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorCompZ_DeK_h
#define CFatorCompZ_DeK_h

#include "CFatorCompZ.h"
```

```
class CFatorCompZ_DeK : public CFatorCompZ
{
//a classe so possui os atributos da classe base CFatorCompZ

public:

/*metodo calcula o fator de compressibilidade Z pelo metodo de
Dranchuk e Kassem */

void FatorCompZ_DeK();

};

#endif
```

Apresenta-se na listagem 6.78 o arquivo com código da classe CFatorCompZ_DeK.

Listing 6.78: Arquivo de implementação da classe CFatorCompZDeK.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CFatorCompZ_DeK.h"

#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

void CFatorCompZ_DeK::FatorCompZ_DeK()
{
/*Programa extima o valor do fator de compressibilidade gas Z com ppr e tpr
utilizando o metodo apresentado por  Dranchuk and Abou-Kassem (1975)
- Calculation of Z factors for natural gases using equations of state.

ENTRADA:

ppr = pressao pseudo-reduzida
tpr = temperatura pseudo-reduzida

SAIDA:

Z = fator de compressibilidade do gas*/


//inicializacao de variaveis

double A1  = +0.32650;
double A2  = -1.07000;
double A3  = -0.53390;
double A4  = +0.01569;
double A5  = -0.05165;
double A6  = +0.54750;
double A7  = -0.73610;
double A8  = +0.18440;
double A9  = +0.10560;
double A10 = +0.61340;
double A11 = +0.72100;
```

```
//entrada de dados

fstream fin;
fin.open("dados_fator_comp_Z_Dek.dat");


while(!fin.eof())
{
  double ppr;
  fin >> ppr;
  cout<<ppr<<endl;

  double tpr;
  fin >> tpr;
  cout<<tpr<<endl;

/* cout<<"Entre com o valor da pressao pseudo-reduzida"<<endl;
cin>>ppr;  cin.get();

cout<<"Entre com o valor da temperatura pseudo-reduzida"<<endl;
cin>>tpr;  cin.get(); */

//calculo do fator de compressibilidade Z

      double  Z =  A1 + A2/tpr + A3/pow(tpr,3) + A4/pow(tpr,4) + A5/pow(tpr,5)*(0.27*ppr/
          tpr)
            + (A6 + A7/tpr) + A8/pow(tpr,2)*pow(0.27*ppr/tpr,2)
            - A9*(A7/tpr + A8/pow(tpr,2)*pow(0.27*ppr/tpr,5)
            + A10*(1 + A11*((0.27*ppr)/pow(tpr,2))*((pow(0.27*ppr/tpr,2)/pow(tpr,3))
            * exp( -A11 * pow(0.27*ppr/tpr,2)))));

    cout<<"O valor do fator de compressibilidade do gas Z e:"<<endl;
    cout<<Z<<endl;
}
}
```

Apresenta-se na listagem 6.79 o arquivo com código da classe CFatorCompZ_HeY.

Listing 6.79: Arquivo de cabeçalho da classe CFatorCompZHeY.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#ifndef CFatorCompZ_HeY_h
#define CFatorCompZ_HeY_h

#include "CFatorCompZ.h"

class CFatorCompZ_HeY : public CFatorCompZ
{
//a classe possui apenas os atributos da classe base CFatorCompZ_HeY

public:

/*metodo calcula fator de compressibilidade Z pelo metodo de
Hall e Yarbough  */

void FatorCompZ_HeY();

};
```

```
#endif
```

Apresenta-se na listagem 6.80 o arquivo com código da classe CFatorCompZ_HeY.

Listing 6.80: Arquivo de implementação da classe CFatorCompZHeY.

```
//trabalho_programacao_pratica
//dupla_alex_aline

#include "CFatorCompZ_HeY.h"

#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

void CFatorCompZ_HeY::FatorCompZ_HeY()

{

/*
Z_Hall extima o valor do fator de compressibilidade gas Z com ppr e tpr
utilizando o metodo apresentado por  Hall-Yarborough (1973)-A New
Equation of State for Z-factor Calculations

Z_Hall - Calculation of Z factors for natural gases using equations of state.

ENTRADA:
Z_Hall(ppr,Tpr)

Ppr = pressao pseudo-reduzida
Tpr = temperatura pseudo-reduzida

SAIDA
Z = fator de compressibilidade do gas
*/

//Hall & Yaborough


//entrada de dados

fstream fin;
fin.open("dados_fator_comp_Z_HeY.dat");


while(!fin.eof())
{
  double ppr;
  fin >> ppr;
  cout<<ppr<<endl;

  double tpr;
  fin >> tpr;
  cout<<tpr<<endl;

/* cout<<"Entre com o valor da pressao pseudo-reduzida"<<endl;
cin>>ppr;  cin.get();

cout<<"Entre com o valor da temperatura pseudo-reduzida"<<endl;
```

```
cin>>tpr;  cin.get(); */

//inicializacao de variavel
double y0 = 0.001;

//calculo temperatura atual
double t = 1./tpr;


//calculo fator de compressibilidade Z
double FZ = ((1+ y0 + pow(y0,2) + pow(y0,3))/pow((1-y0),3))- (14.7*t - 9.76*pow(t,2)
    + 4.58*pow(t,3))*(y0)+ (90.7*t - 242.2*pow(t,2)+
    42.4*pow(t,3))*pow(y0,(1.18 + 2.82*t));

//mostra na tela o valor do fator de compressibilidade Z
    cout<<"O valor do fator de compressibilidade Z e:" << FZ << endl;

}
}
```

Capítulo 7

Bibliografia

[Ron13] Fonseca Nunes da Silva, Rony. Biblioteca de funções para o cálculo de propriedades de mistura de hidrocarbonetos de composição desconhecida. Agosto, 2013.

[And08] Duarte Bueno, André. Programação Orientada a Objeto com C++, 2º edição. NOVATEC EDITORA LTDA. agosto, 2008.

Índice Remissivo