

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

PROJETO ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE DE PERFURAÇÃO DE POÇOS DE
PETRÓLEO
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

FILLIPE RODRIGUES SIAS
LUCAS ARMANDO DE CARVALHO
Prof. André Duarte Bueno

MACAÉ - RJ
DEZEMBRO - 2015

Sumário

1	Introdução	2
1.1	Escopo do Problema	2
1.2	Objetivos	4
2	Especificação	5
2.1	Especificação do software - descrição dos requisitos	5
2.1.1	Nome do sistema e componentes	5
2.1.2	Especificação	5
2.1.3	Requisitos funcionais	6
2.1.4	Requisitos não funcionais	6
2.2	Casos de uso do Programa	6
2.3	Diagrama de caso de uso geral do programa	7
2.4	Diagrama de caso de uso específico do programa	7
3	Elaboração	9
3.1	Análise de domínio	9
3.2	Formulação teórica	9
3.3	Identificação de pacotes – assuntos	14
3.4	Diagrama de pacotes – assuntos	14
4	AOO – Análise Orientada a Objeto	16
4.1	Diagramas de classes	16
4.1.1	Dicionário de classes	16
4.2	Diagrama de sequência – eventos e mensagens	16
4.3	Diagrama de comunicação – colaboração	17
4.4	Diagrama de máquina de estado	18
5	Projeto	20
5.1	Projeto do sistema	20
5.2	Projeto orientado a objeto – POO	21
6	Implementação	23
6.1	Código fonte	23

7	Teste	49
7.1	Teste 1: Entrando com valores comuns ao problema, observando os resultados em tela e salvando arquivo	49
7.2	Teste 2: Testando valores de profundidade errados	51
7.3	Teste 3: Testando valores de profundidade que não representam o problema real	54
8	Documentação	56
8.1	Manual do Usuário	56

Capítulo 1

Introdução

No presente trabalho desenvolve-se um software de perfuração de poço de petróleo que calcula os volumes de fluido de perfuração e cimento, a quantidade de tubos utilizados e o tempo necessário para realizar a operação. Os cálculos são baseados nos conhecimentos de volumes de cilindros e em taxas obtidas através da média histórica dessa operação.

1.1 Escopo do Problema

Após confirmar a existência de um campo petrolífero e localizar um reservatório, inicia-se um estudo para o desenvolvimento deste campo. Este estudo envolve diversas disciplinas, sendo uma delas a engenharia de poço. Seu foco de trabalho é desenvolver projetos robustos e economicamente viáveis, ou seja, projetos seguros e de alta confiabilidade e que estejam dentro da margem de lucros prevista para este campo.

Levando em consideração a expressão “tempo é dinheiro”, torna-se necessária a criação de ferramentas para agilizar e reduzir a margem de erro nos futuros projetos. Nos projetos de perfuração de poços impera uma grande quantidade de cálculos básicos de geometria, tempo e custos, que se repetem, estabelecendo uma rotina.

A perfuração de um poço de petróleo envolve a repetição de duas etapas:

- Etapa Perfuração
- Etapa Revestimento

A cada etapa perfura-se e reveste-se um cilindro, com diâmetro padronizado e profundidade que depende do projeto do poço. A Figura 1.1 esquematiza as etapas de perfuração de um poço, onde LDA e MR representam as profundidades da lâmina d’água e da mesa rotativa, respectivamente. No lado esquerdo da figura estão representados os diâmetros dos revestimentos, enquanto no lado direito estão representados os diâmetros perfurados em cada fase (poço aberto).

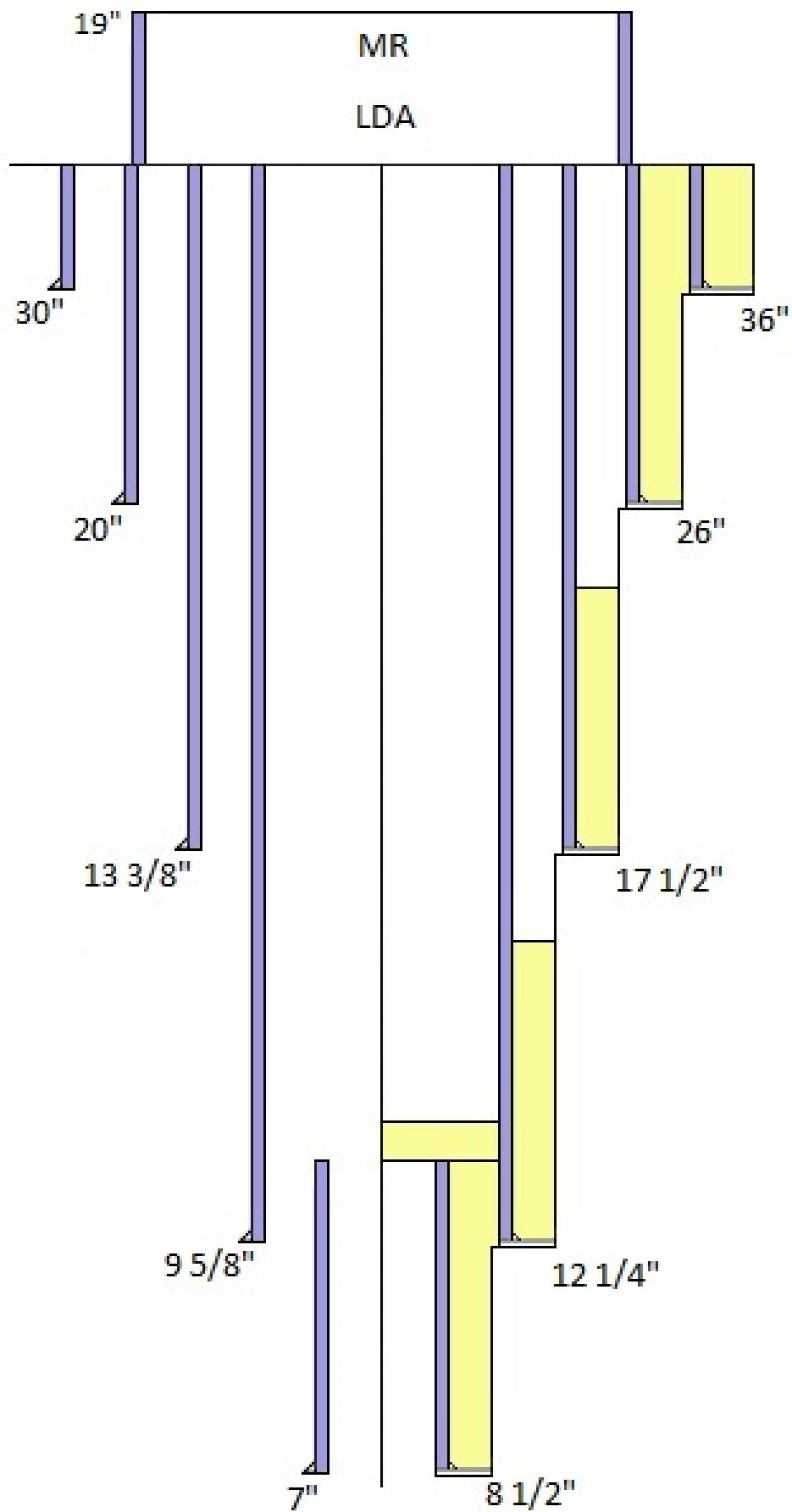


Figura 1.1: Sequencia de Perfuração de um poço de Petróleo.

Dada essa repetibilidade nos cálculos, observamos que o desenvolvimento de um código computacional otimizaria a construção de projetos de perfuração, agregando agilidade e confiabilidade a estes cálculos e auxiliando na avaliação econômica e na redução dos custos iniciais.

O software tem licença GPL 2.0, podendo ser livremente distribuído.

1.2 Objetivos

Os objetivos deste trabalho são:

- Objetivo geral:
 - Desenvolver um programa para cálculo dos volumes de fluido de perfuração e cimento, quantidade de tubos e o tempo gasto em uma operação de perfuração de um poço de petróleo.
- Objetivos específicos:
 - Permitir ao usuário entrar com as profundidades das fases da perfuração e visualizar os resultados dos cálculos gerados pelo programa.
 - Disponibilizar método de entrada de dados manual e a opção de salvar os resultados em um arquivo de texto, além da saída em tela.
 - Gerar gráfico de Tempo de Operação *versus* Profundidade Alcançada utilizando o Gnuplot.

Capítulo 2

Especificação

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

2.1 Especificação do software - descrição dos requisitos

Nesta seção será apresentada a especificação do software.

2.1.1 Nome do sistema e componentes

Apresenta-se a seguir, na Tabela 2.1, o nome do software, seus componentes principais e sua missão.

Nome	Software de Perfuração de Poços de Petróleo.
Componentes principais	Sistema para cálculo de dimensões físicas de uma operação de Perfuração de Poços de Petróleo.
Missão	Calcular volumes de fluido de perfuração e cimento, a quantidade de equipamentos utilizados e o tempo gasto na perfuração do poço.

Tabela 2.1: Nome do sistema e componentes.

2.1.2 Especificação

Apresenta-se a seguir a especificação do software.

O software a ser desenvolvido deverá realizar os cálculos de volume de fluido de perfuração, volume de cimento e quantidades de equipamentos (tubos) utilizados na perfuração de um poço de petróleo, além do tempo gasto em toda a operação. O programa será desenvolvido em linguagem C++, com orientação a objeto, e poderá ser usado nos sistemas operacionais GNU/Linux e Windows, sendo operado em modo texto e terá apenas uma janela. O usuário do software informará as profundidades da lâmina d'água e das sapatas

dos revestimentos manualmente. Após a realização dos cálculos e apresentação dos resultados na tela, será oferecida ao usuário a opção de salvar em um arquivo de texto onde serão apresentados os resultados para cada etapa e para toda a operação. Outra opção oferecida será gerar um gráfico de tempo da operação *versus* profundidade, optando por gerar uma imagem ou um arquivo de texto com os conjuntos de dados.

2.1.3 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

- RF-01: O usuário deve ser capaz de inserir os valores de profundidade.
- RF-02: O usuário deverá ter liberdade para escolher salvar os resultados.
- RF-03: O usuário poderá plotar seus resultados em um gráfico. O gráfico poderá ser salvo como imagem e ter seus dados exportados como texto.

2.1.4 Requisitos não funcionais

Apresenta-se a seguir os requisitos não funcionais.

- RNF-01: Os cálculos devem ser feitos utilizando-se de fórmulas geométricas.
- RNF-02: O programa deverá ser multi-plataforma, podendo ser executado em Windows, GNU/Linux ou Mac.

2.2 Casos de uso do Programa

A Tabela 2.2 mostra um cenário de funcionamento do programa.

Tabela 2.2: Caso de Uso	
Nome do caso de uso:	Salvando resultados.
Resumo/descrição:	Salvando arquivo de texto com os resultados obtidos.
Etapas:	<ol style="list-style-type: none">1. Inserir valores das profundidades.2. Salvar os resultados.3. Analisar os resultados.
Cenários alternativos:	Inserir valores negativos ou incompatíveis com a ordem de grandeza do problema real.

2.3 Diagrama de caso de uso geral do programa

O diagrama de caso de uso da Figura 2.1 mostra o usuário utilizando o sistema, onde ele insere as profundidades, salva os resultados e analisa os mesmos.

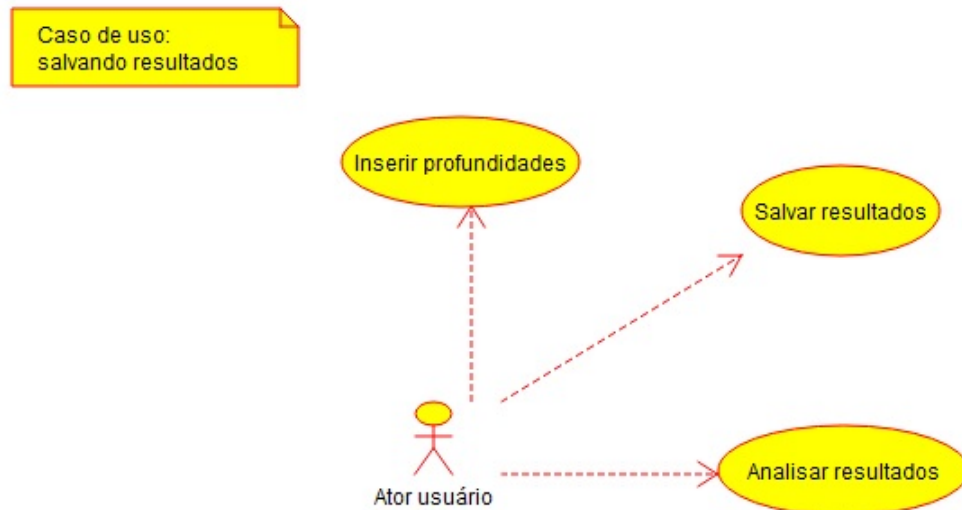


Figura 2.1: Diagrama de caso de uso geral

2.4 Diagrama de caso de uso específico do programa

A Figura 2.2 representa o caso de uso específico onde o usuário decide gerar o gráfico, salvando a imagem e o arquivo de texto contendo as informações do gráfico. Primeiramente as profundidades são inseridas manualmente, a opção de salvar os dados e o gráfico é escolhida e o usuário pode analisar os resultados.

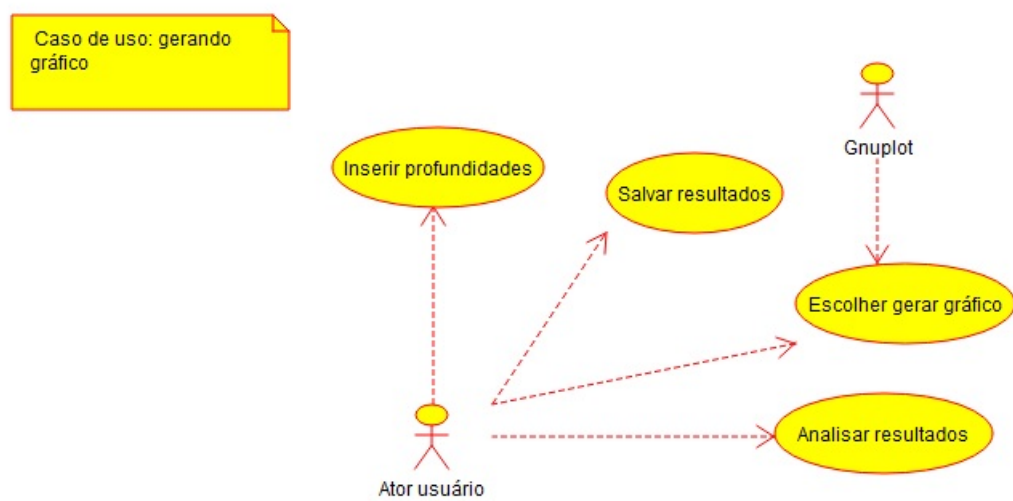


Figura 2.2: Diagrama de caso uso específico

Capítulo 3

Elaboração

Neste capítulo será apresentada a elaboração do projeto.

3.1 Análise de domínio

- O *software* envolverá disciplinas de Geometria, Matemática, Física e Engenharia de Poço.
- O programa será desenvolvido em ambiente universitário, contará com ajuda de professores e engenheiros da indústria.
- O *software* se baseará em métodos determinísticos para os cálculos e, na maioria das vezes, estará levemente otimista em relação aos valores reais por conta de acidentes e imprevistos decorrentes da operação no campo.

3.2 Formulação teórica

Aqui serão apresentadas as fórmulas e medidas utilizadas para a realização dos cálculos no *software*.

Um poço é dividido em fases, e cada fase é determinada pelo diâmetro da broca utilizada na perfuração. No *software* foi utilizada a configuração típica de um poço de petróleo, composto pelas fases com diâmetros de 36", 26", 17 1/2", 12 1/4" e 8 1/2". Após a finalização de cada fase, é descida uma coluna de revestimento para proteger as formações e permitir a utilização de um fluido de perfuração mais adequado à fase seguinte a ser perfurada. As medidas das tubulações e brocas utilizadas na operação são apresentadas na Tabela 3.1. [RA07]

	Diâmetro (polegadas)	Comprimento (metros)
<i>Riser</i>	19	15
Revestimento fase 1	30	12
Revestimento fase 2	20	12
Revestimento fase 3	13 3/8	12
Revestimento fase 4	9 5/8	12
Revestimento fase 5	7	12
Broca fase 1	36	-
Broca fase 2	26	-
Broca fase 3	17 1/2	-
Broca fase 4	12 1/4	-
Broca fase 5	8 1/2	-

Tabela 3.1: Medida padrão de tubulações e brocas.

Um dos cálculos realizados pelo *software* tem como resultado a quantidade de revestimentos necessários para cada fase e para o riser. Para o cálculo foi utilizado o comprimento dos tubos, conforme mostra a Equação 3.1 (revestimentos das fase 1, 2, 3 e 4), a Equação 3.2 (*liner*, revestimento da fase 5) e a Equação 3.3 (*riser*).

$$N^{\circ}tubos_{revestimento} = \frac{profundidade[n] - profundidade[0]}{12}, \quad (3.1)$$

$$N^{\circ}tubos_{liner} = \frac{profundidade[5] - profundidade[4] + 100}{12}, \quad (3.2)$$

$$N^{\circ}tubos_{riser} = \frac{profundidade[0]}{15}. \quad (3.3)$$

O valor entre *colchetes* representa a fase do poço. A profundidade[0] representa a lâmina d'água (LDA). Na fase 5, é instalado o liner de 7", sendo necessário exceder 100m acima da sapata da fase anterior.

Para o cálculo dos volumes de fluido de perfuração e cimento, foi utilizado o conceito de *Capacidade*, que é uma relação entre volume e comprimento. A Equação 3.4 mostra como são calculadas as *Capacidades* (*bbl/m*) da broca, revestimento e do tubo de perfuração, sendo D o diâmetro dos mesmos.

$$Capacidade = 0,0031871 \times D^2. \quad (3.4)$$

Com isso, o volume de fluido de perfuração que será utilizado é calculado a partir da Equação 3.5. Lembrando que nas fases 1 e 2, o fluido de perfuração utilizado é a água do mar, portanto, estes valores não foram considerados no cálculo do programa, visto que o programa visa estimar os volumes de fluidos com necessidade de preparação e estocagem na planta da sonda. Na fórmula é utilizado um fator de multiplicação 1,1. Esse fator é referente à margem de segurança de 10% utilizada pela indústria petrolífera.

$$\begin{aligned}
V_{fluido}[n] = & 1,1 \times [(prof[n] - prof[n-1] + 10) \times capacidade_{broca}[n] + \dots \\
& \dots + (prof[n-1] - LDA) \times capacidade_{revest}[n-1] + LDA \times capacidade_{riser}]. \quad (3.5)
\end{aligned}$$

O volume de cimento utilizado é calculado a partir da Equação 3.6, para as fases 1 e 2. É utilizado o fator de multiplicação 2 para garantir a qualidade do cimento no contato revestimento-formação. Esse cimento tem retorno ao meio marítimo.

$$V_{cimento}[n] = 2 \times [(prof[n] - LDA) \times (capacidade_{broca}[n] - capacidade_{revest}[n])], \quad (3.6)$$

e, para as fases 3 e 4, utiliza-se a Equação 3.7.

$$\begin{aligned}
V_{cimento}[n] = & (prof[n] - prof[n-1] - 100) \times \dots \\
& \dots \times (capacidade_{broca}[n] - capacidade_{revest}[n]), \quad (3.7)
\end{aligned}$$

e, na fase 5, utiliza-se a Equação 3.8.

$$\begin{aligned}
V_{cimento}[5] = & (prof[5] - prof[4]) \times (capacidade_{broca}[5] - capacidade_{revest}[5]) + \dots \\
& \dots + 100 \times (capacidade_{revest}[4] - capacidade_{revest}[5]) + 60 \times capacidade_{revest}[4]. \quad (3.8)
\end{aligned}$$

Além dos cálculos apresentados anteriormente, o *software* também estima o tempo de duração de toda a operação de perfuração, incluindo o tempo para perfurar a formação rochosa, revestir, cimentar, manobras com BOP (*blow out preventer*), entre outros. Na operação, alguns tempos são fixos, como o tempo para montar e desmontar o BHA (*Bottom Hole Assembly*), circular o fluido de perfuração para limpeza do poço e esperar a *pega* do cimento, enquanto que o restante dos tempos são calculados baseados em equações que envolvem algumas taxas como a velocidade de perfuração, descida e subida do BHA, montagem e descida de revestimento, bombeio do cimento, entre outras. Os valores dos tempos fixos são apresentados na Tabela 3.2 e as taxas utilizadas na Tabela 3.3.

Procedimento	Tempo (horas)
Montar BHA	8
Desmontar BHA	8
Circular fluido para limpeza	3
Esperar pega do cimento	12

Tabela 3.2: Tempos fixos de operações.

Procedimento	Taxa	Unidade
Descer/Subir BHA	166,667	metro/hora
Perfurar (fases 1 e 2)	10	metro/hora
Perfurar (fases 3 e 4)	5	metro/hora
Perfurar (fase 5)	3	metro/hora
Montar revestimento (fase 1)	2	junta/hora
Montar revestimento (fase 2)	6	junta/hora
Montar revestimento (fase 3)	8	junta/hora
Montar revestimento (fase 4)	11	junta/hora
Montar revestimento (fase 5)	13	junta/hora
Descer Revestimento (trecho de poço aberto)	240	metro/hora
Descer Revestimento (trecho de poço revestido)	360	metro/hora
Descer Revestimento (trecho com <i>Riser</i>)	720	metro/hora
Bombear cimento	600	barril/hora
Subir <i>drill pipe</i>	166,667	metro/hora
Instalação do BOP e <i>Riser</i>	20,833	metro/hora
Retirada do BOP e <i>Riser</i>	31,25	metro/hora

Tabela 3.3: Taxas utilizadas para cálculo dos tempos da operação.

Esses valores são utilizados junto com as profundidades das fases e da lâmina d'água para calcular o tempo gasto na operação. Os cálculos são divididos entre as etapas de perfuração, de revestimento e a descida/subida do BOP e *Riser*. A Equação 3.9 apresenta o cálculo do tempo gasto durante a perfuração do poço.

$$\begin{aligned}
Tempo_{Perfuracao}[n] = & montarBHA + \frac{prof[n-1]}{descerBHA} + \frac{prof[n] - prof[n-1]}{taxaPerfuracao[n]} + \dots \\
& \dots + \frac{prof[n]}{subirBHA} + circularFluido + desmontarBHA.
\end{aligned} \tag{3.9}$$

Para a etapa de instalação do revestimento, os cálculos são divididos em três grupos, fases 1 e 2, 3 e 4, e por último a fase 5. A Equação 3.10 apresenta o cálculo para a instalação do revestimento nas duas primeiras fases.

$$Tempo_{Revestimento}[n] = \frac{n^{\circ}tubos[n]}{montarRevest[n]} + \frac{LDA - (prof[n] - LDA)}{descerRevest_{(pocoAberto)}} + \dots$$

$$\begin{aligned}
& \dots + \frac{(prof[n] - 50) \times capacidade_{DrillPipe} + 50 \times capacidade_{Revest}[n] + volume_{Cimento}[n]}{taxaBombeioCimento} + \dots \\
& \dots + esperarPega + \frac{prof[n] - 50}{subirDrillPipe}. \tag{3.10}
\end{aligned}$$

A Equação 3.11 apresenta o cálculo para as fases 3 e 4.

$$\begin{aligned}
Tempo_{Revestimento}[n] &= \frac{n^o tubos[n]}{montarRevest[n]} + \frac{prof[n-1] - (prof[n] - LDA)}{descerRevest_{(pocoRevestido)}} + \dots \\
& \dots + \frac{prof[n] - prof[n-1]}{descerRevest_{(pocoAberto)}} + \frac{LDA \times capacidade_{Revest}[30]}{taxaBombeioCimento} + esperarPega + \dots \\
& \dots + \frac{(prof[n] - LDA) \times capacidade_{Revest}[n] + volume_{Cimento}[n]}{taxaBombeioCimento} + \frac{LDA}{subirDrillPipe}. \tag{3.11}
\end{aligned}$$

Para a fase 5, é utilizada a Equação 3.12.

$$\begin{aligned}
Tempo_{Revestimento}[5] &= \frac{n^o tubos[5]}{montarRevest[5]} + \frac{LDA - (prof[5] - prof[4] + 100)}{descerRiser} + \dots \\
& \dots + \frac{(prof[5] - prof[4] + 100) \times capacidade_{Revest}[4] + volume_{Cimento}[5]}{taxaBombeioCimento} + \dots \\
& \dots + \frac{(prof[4] - 100) \times capacidade_{DrillPipe}}{taxaBombeioCimento} + \frac{prof[4] - (prof[5] - prof[4] - 100) - LDA}{descerRevest_{(pocoRevestido)}} + \dots \\
& \dots + \frac{prof[5] - prof[4] - 100}{descerRevest_{(pocoAberto)}} + esperarPega + \frac{prof[4] - 100}{subirDrillPipe}. \tag{3.12}
\end{aligned}$$

Com os tempos calculados, o tempo total da operação pode ser obtido, somando-se o tempo de todas as etapas e os tempos de instalação e retirada do BOP e *Riser*. Estas são calculadas através das Equações 3.13 e 3.14.

$$Tempo_{instalacao_{BOP-Riser}} = \frac{prof[0]}{descida_{BOP}}, \tag{3.13}$$

$$Tempo_{retirada_{BOP-Riser}} = \frac{prof[0]}{subida_{BOP}}. \tag{3.14}$$

3.3 Identificação de pacotes – assuntos

- Poço: O poço é uma via física de ligação entre a superfície e um alvo geológico. Sua construção se dá a partir da perfuração de rochas em subsuperfície.
- Perfuração: A perfuração consiste no conjunto de várias operações e atividades necessárias para atravessar as formações geológicas que formam a porção superficial da crosta terrestre até atingir-se o alvo geológico. Nas atividades de perfuração de poços de petróleo utiliza-se sondas de perfuração, que consistem em conjuntos de equipamentos bastante complexos e robustos.
 - Coluna de Perfuração: Para realizar a perfuração utiliza-se um conjunto de ferramentas, dentre elas a coluna de perfuração. Essa coluna é formada por duas partes, o *BHA* e os *drill pipes*. O *BHA* (*Bottom Hole Assembly*) é um conjunto formado pela broca, estabilizadores, comandos e, dependendo da necessidade, alguns equipamentos mais complexos (ferramentas de perfilagem, medidores de temperatura e pressão, etc). Os *drill pipes* são tubos de 5 1/2” utilizados para conectar o *BHA* à sonda de perfuração.
- Geometria: A geometria está relacionada diretamente a todos os parâmetros do poço, que consiste em cilindros concêntricos de diferentes dimensões radiais e longitudinais. A partir das medidas geométricas são calculados os volumes.
- Gráfico: O gráfico é uma representação dos resultados obtidos pelo software, onde duas grandezas, o tempo da operação (horas) e a profundidade alcançada (metros), são plotadas, de maneira que os pontos críticos (de maior duração) fiquem destacados.

3.4 Diagrama de pacotes – assuntos

A Figura 3.1 representa o diagrama de pacotes do sistema.

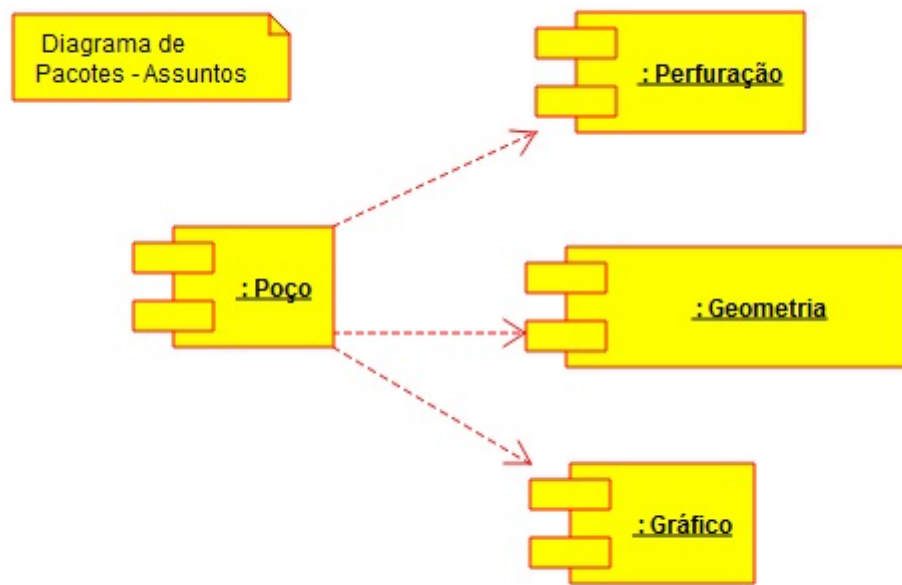


Figura 3.1: Diagrama de Pacotes.

Capítulo 4

AOO – Análise Orientada a Objeto

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

4.1.1 Dicionário de classes

- Classe COperacao: é a classe que gerencia as outras e troca informações com o usuário.
- Classe CPoco: esta classe é responsável pelo armazenamento de todos os atributos do poço, como profundidade das sapatas dos revestimentos, as capacidades, número de juntas necessárias para cada fase, e seus respectivos diâmetros. Ela possui os métodos Set para a profundidade e Get para todos os atributos. A classe possui também os métodos para realizar os cálculos de volume de fluido de perfuração e de cimento e o número de juntas utilizadas.
- Classe CPerfuracao: é a classe responsável pelos cálculos referentes ao tempo na etapa de perfuração do poço.
- Classe CRevestimento: esta classe possui métodos para realizar os cálculos referentes ao tempo na etapa de revestimento e cimenta o poço.
- Classe CGrafico: classe responsável por organizar os dados que serão utilizados para gerar o gráfico junto à Classe CGnuplot.

4.2 Diagrama de sequência – eventos e mensagens

Nesta seção é apresentado o diagrama de sequência (Figura 4.2), observe que o diagrama mostra a troca de eventos, respeitando sua ordem temporal, onde o usuário fornece os dados para a classe COperacao (evento 1), esta fornece informações para a classe CPoco

(evento 2). A classe CPoco passa informações para as classes CPerfuracao e CRevestimento eventos (3 e 5), que retornam informações para CPoco (eventos 4 e 6). No evento 7 a classe COperacao tem um retorno de informações de CPoco e, em seguida, envia informações para a classe CGrafico (evento 8). Esta troca informações com a classe CGnuplot (eventos 9 e 10) e retorna os dados para COperacao (evento 11). Por último, no evento 12, COperacao retorna as informações para o usuário.

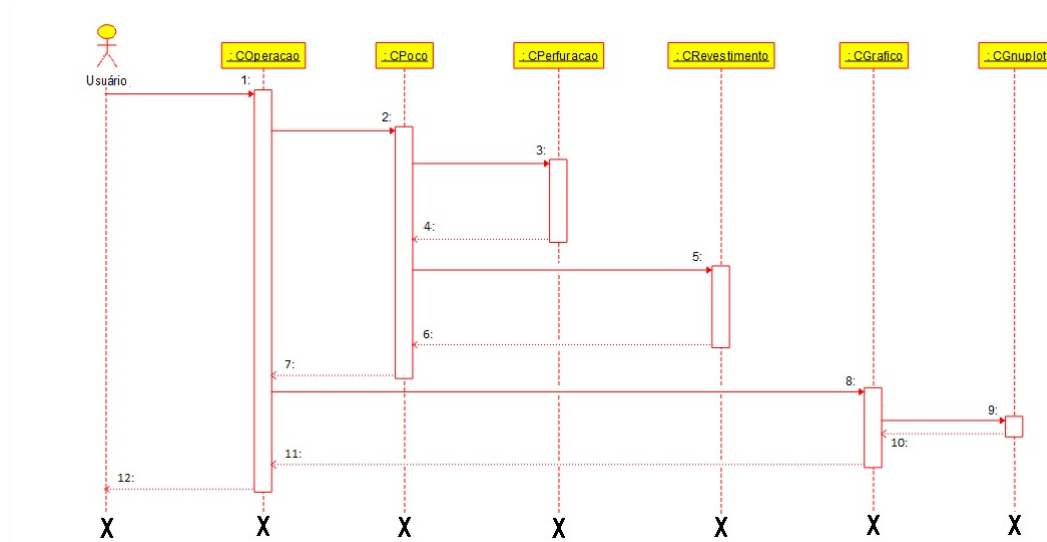


Figura 4.2: Diagrama de sequência.

4.3 Diagrama de comunicação – colaboração

Nesta seção é apresentado o diagrama de comunicação, conforme mostra a Figura 4.3. Observe que inicialmente o usuário entra com os dados, que são armazenados na classe COperacao. Esta classe, por sua vez, fornece informações para a classe CPoco, que então, realiza cálculos e troca informações com as classes CPerfuracao e CRevestimento, que realizam seus respectivos cálculos e retornam os resultados para a classe CPoco. Esta retorna informações para COperacao. A classe COperacao envia os dados para CGrafico, que troca informações com CGnuplot e retornam dados para COperacao. Por fim, COperacao apresenta os resultados ao usuário.

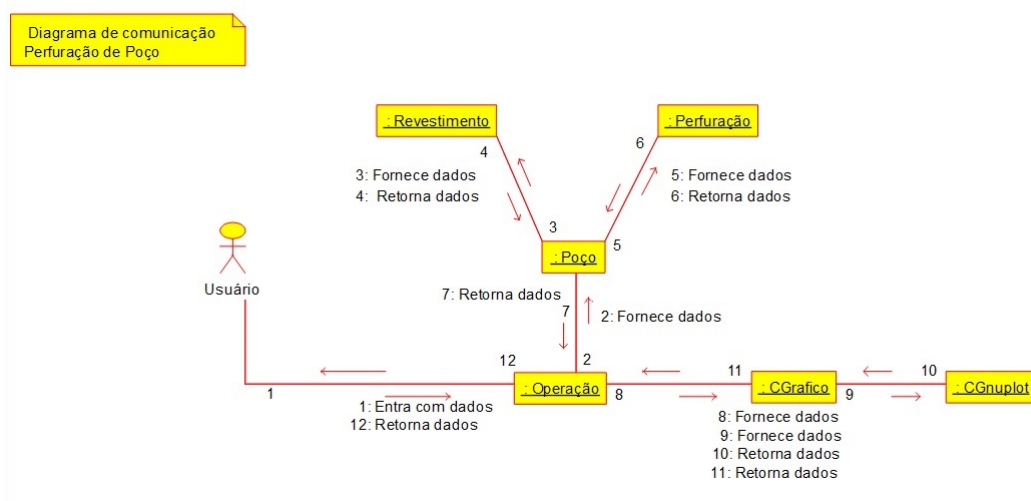


Figura 4.3: Diagrama de comunicação.

4.4 Diagrama de máquina de estado

Na Figura 4.4 é apresentado o diagrama de máquina de estado para a classe CPerfuracao.

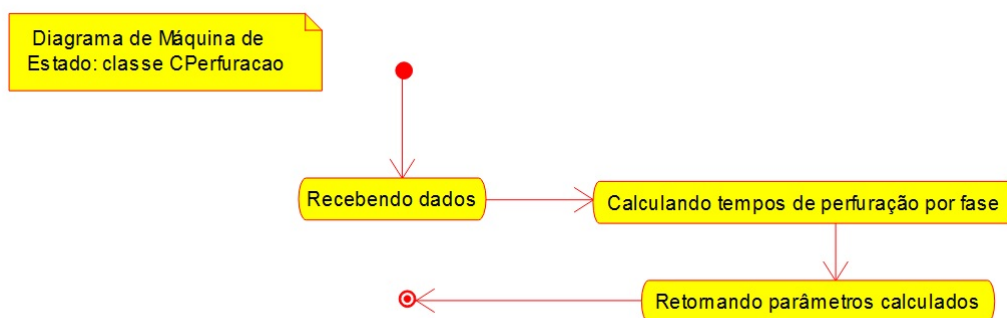


Figura 4.4: Diagrama de Máquina de Estado para a classe CPerfuracao.

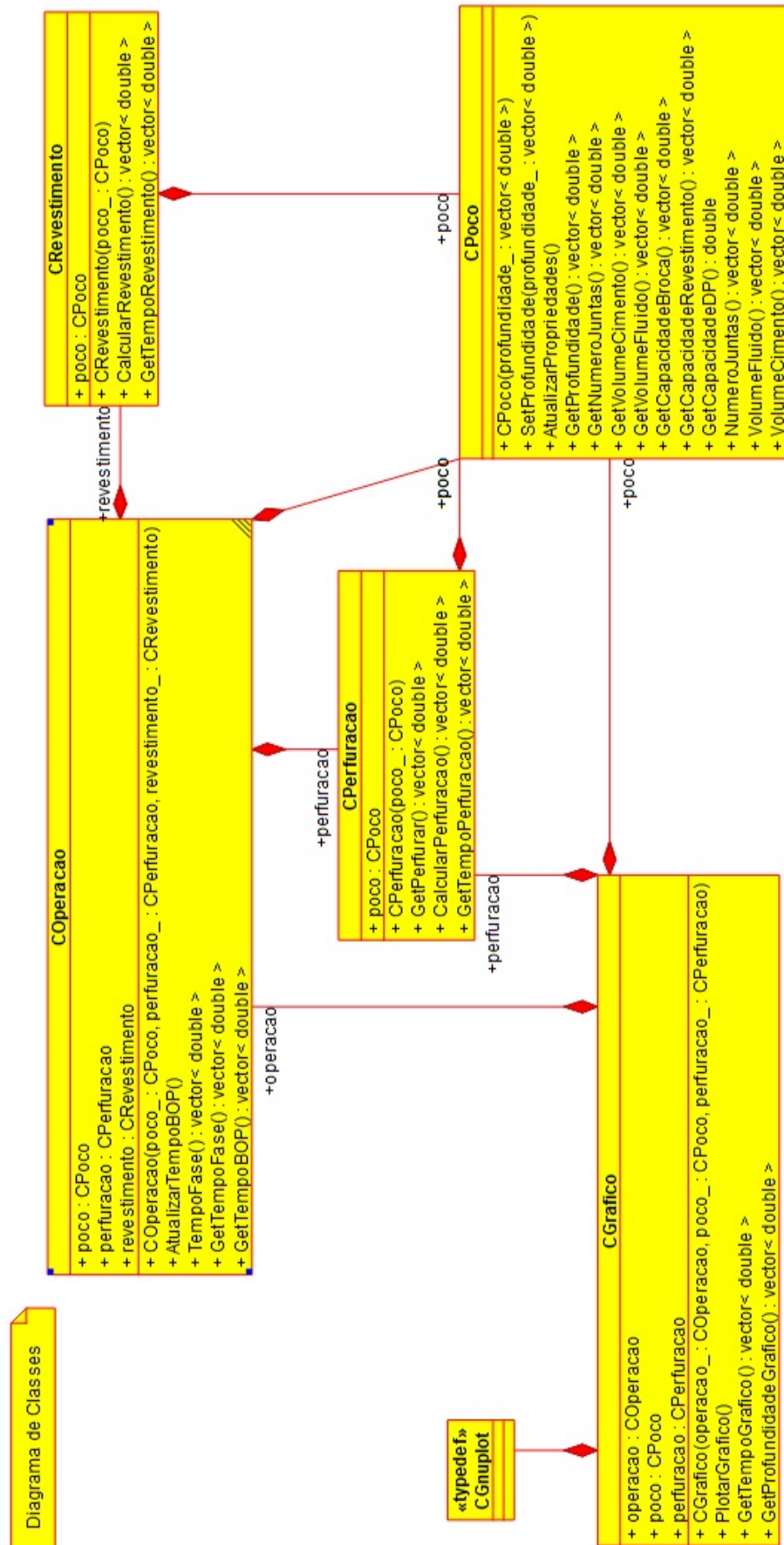


Figura 4.1: Diagrama de classes.

Capítulo 5

Projeto

Neste capítulo do projeto de engenharia veremos questões associadas ao projeto do sistema.

5.1 Projeto do sistema

Depois da análise orientada a objeto desenvolve-se o projeto do sistema, o qual envolve etapas como a definição dos protocolos, da interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas de controle, a seleção das plataformas do sistema, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto.

Deve-se definir padrões de documentação, padrões para o nome das classes, padrões de retorno e de parâmetros em métodos, características da interface do usuário e características de desempenho.

O projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar uma solução. Você deve se preocupar com itens como:

1. Plataformas

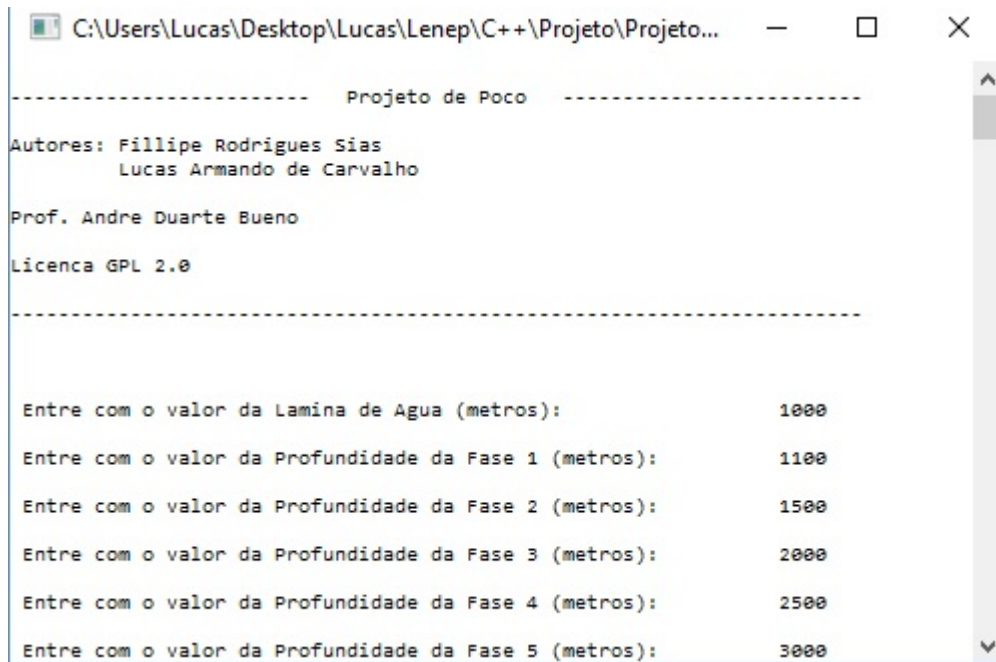
- O software irá funcionar nos sistemas operacionais Windows e GNU/Linux, sendo desenvolvido no Windows e testado no Windows e GNU/Linux.
- Não hávera necessidade de grandes mudanças para tornar o programa multiplataforma pois a linguagem utilizada, C++, tem suporte em todos estes sistemas operacionais.

2. Bibliotecas

- Será utilizada biblioteca padrão da linguagem C++.
- O programa será desenvolvido com a interface Dev-C++ 5.6.3.

3. Ambiente de desenvolvimento

- O programa será desenvolvido e testado em um notebook *Gateway* com Windows 10 64-bit, processador Intel core i3-3110M, 4GB de memória RAM e placa de video intel HD Graphics 4000.
- O software é executado em uma janela do terminal (linux ou windows). A Figura 5.1 mostra o programa sendo executado.



```
C:\Users\Lucas\Desktop\Lucas\Lenep\C++\Projeto\Projeto...
----- Projeto de Poco -----
Autores: Fillipe Rodrigues Sias
        Lucas Armando de Carvalho
Prof. Andre Duarte Bueno
Licenca GPL 2.0
-----

Entre com o valor da Lamina de Agua (metros):          1000
Entre com o valor da Profundidade da Fase 1 (metros):  1100
Entre com o valor da Profundidade da Fase 2 (metros):  1500
Entre com o valor da Profundidade da Fase 3 (metros):  2000
Entre com o valor da Profundidade da Fase 4 (metros):  2500
Entre com o valor da Profundidade da Fase 5 (metros):  3000
```

Figura 5.1: Janela de execução do Programa de Poço

5.2 Projeto orientado a objeto – POO

Efeitos do projeto no modelo estrutural

- Aqui são estabelecidos as dependencias e restrições do Programa de Poço.
 - O Software necessita das plataformas GNU/linux ou Windows para ser executado.
 - No Sistema Operacional Windows, existe a necessidade de instalação do *software* Gnuplot para o funcionamento do programa.

Efeitos do projeto nos métodos

- Inicialmente seria plotado um gráfico ilustrando o poço, respeitando as suas proporções, usando a classe CGnuplot, porém esta etapa foi revisada, devido às dimensões de um poço, onde existem diferenças consideráveis entre uma fase e outra, o que

deixaria a ilustração ruim. A ilustração foi então substituída por outra que representa um esquema pré-definido de um poço e o gráfico gerado pela classe CGnuplot passou a ser o de tempo *versus* profundidade. Este gráfico representa o avanço da profundidade atingida conforme passa o tempo da operação.

Efeitos do projeto nas heranças

- Não houve necessidade de rever o projeto nesta etapa.

Efeitos do projeto nas associações

- Não houve necessidade de rever o projeto nesta etapa.

Efeitos do projeto nas otimizações

- Foi incluído um novo tipo de gráfico representando os dados referentes aos dados calculados.

Capítulo 6

Implementação

Neste capítulo será apresentado o código fonte do Programa “Projeto de Poço”.

6.1 Código fonte

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

Apresenta-se na listagem 6.1 o arquivo com código da classe *COperacao.h*.

```
#ifndef COPERACAO_H
#define COPERACAO_H

#include "CPoco.h"
#include "CPerfuracao.h"
#include "CRevestimento.h"

#include <string>
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

/// Classe COperacao, que ira solicitar os resultados das classes
/// CPerfuracao e CRevestimento, e calcular o tempo total gasto em toda a
/// operacao para cada fase
class COperacao
{
    /// Atributos
    private:

        /// Variavel que representa a velocidade de descida do
        BOP
        double descidaBOP;
```

```

    /// Variavel que representa a velocidade de subida do
    BOP
    double subidaBOP;

    /// Vetor que armazena o tempo gasto com subida e
    descida do BOP
    vector < double > tempoBOP;

    /// Vetor que armazena o tempo total gasto em toda
    operacao para cada fase
    vector < double > tempoTotal;

// Metodos
public:

    /// Criando objeto do tipo CPoco com o nome poco
    CPoco poco;

    /// Criando objeto do tipo CPerfuracao com o nome
    perfuracao
    CPerfuracao perfuracao;

    /// Criando objeto do tipo CRevestimento com o nome
    revestimento
    CRevestimento revestimento;

    /// Construtor, define valores para os atributos e chama
    o metodo AtualizarTempoBOP
    COperacao (CPoco poco_, CPerfuracao perfuracao_,
    CRevestimento revestimento_) : poco(poco_),
    perfuracao(perfuracao_), revestimento(revestimento_)
    {
        descidaBOP = 500.0/24.0;

        subidaBOP = 750.0/24.0;

        AtualizarTempoBOP ();
    }

    /// Metodo AtualizarTempoBOP, que calcula o tempo gasto
    com a descida e subida do BOP e armazena no vetor
    tempoBOP
    void AtualizarTempoBOP ()
    {
        tempoBOP = vector< double >(2);

        vector< double > profundidade = poco.

```

```
        GetProfundidade ();

        tempoBOP[0] = profundidade[0]/descidaBOP;
        tempoBOP[1] = profundidade[0]/subidaBOP;
    }

    /// Metodo TempoFase, que calcula o tempo gasto em toda
    a operacao para cada fase e armazena no vetor
    tempoTotal
    /// O tempo total e a soma do tempo gasto na perfuracao,
    revestimento e cimentacao por fase
    vector< double > TempoFase ()
    {
        tempoTotal = vector< double >(6);

        vector< double > tempoPerfuracao = perfuracao.
            GetTempoPerfuracao ();
        vector< double > tempoRevestimento =
            revestimento.GetTempoRevestimento ();

        for ( int i = 1 ; i < 6 ; i++ )
        {
            tempoTotal[i] = tempoPerfuracao[i] +
                tempoRevestimento[i];
        }

        return tempoTotal;
    }

    /// Metodo que retorna o vetor tempoTotal
    vector< double > GetTempoFase ()
    {
        return tempoTotal;
    }

    /// Metodo que retorna o vetor tempoBOP
    vector< double > GetTempoBOP ()
    {
        return tempoBOP;
    }
};

#endif
```

Listing 6.1: Arquivo de cabeçalho da classe COperacao.h

Apresenta-se na listagem 6.2 o arquivo de implementação da classe *COperacao.cpp*.

```
#include "COperacao.h"
```

Listing 6.2: Arquivo de cabeçalho da classe COperacao.cpp

Apresenta-se na listagem 6.3 o arquivo com código da classe *CPoco.h*.

```
#ifndef CPOCO_H
#define CPOCO_H

#include <string>
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

/// Classe CPoco, que armazena as propriedades do poco, como
/// profundidades, diametros e capacidades das fases do poco
/// Tambem calcula o numero de tubos necessarias e os volumes de fluido
/// de perfuracao e de cimento
class CPoco
{
    /// Atributos
    private:

        /// vetor que armazena as profundidades das fases do
        /// poco
        vector< double > profundidade;

        /// vetor que armazena o numero de tubos utilizados em
        /// cada fase
        vector< double > numeroJuntas;

        /// vetor que armazena volume de fluido de perfuracao
        /// utilizado em cada fase
        vector< double > volumeFluido;

        /// vetor que armazena o volume de cimento utilizado em
        /// cada fase
        vector< double > volumeCimento;

        /// vetor que armazena o diametro da broca utilizada em
        /// cada fase
        vector< double > diametroBroca;

        /// vetor que armazena o diametro dos revestimentos
        /// utilizados em cada fase
        vector< double > diametroRevestimento;
```

```
/// vetor que armazena as capacidades das brocas
vector< double > capacidadeBroca;

/// vetor que armazena as capacidades dos revestimentos
vector< double > capacidadeRevestimento;

/// variavel que armazena o diametro do drill pipe
double diametroDP;

/// variavel que armazena a capacidade do drill pipe
double capacidadeDP;

// Metodos
public:

/// Construtor, define valores para os atributos e chama
    o metodo AtualizarPropriedades
CPoco (vector< double > profundidade_) : profundidade(
    profundidade_)
{
    diametroBroca = vector< double >(6);
    diametroBroca[0] = 0;
    diametroBroca[1] = 36;
    diametroBroca[2] = 26;
    diametroBroca[3] = 17.5;
    diametroBroca[4] = 12.25;
    diametroBroca[5] = 8.5;

    diametroRevestimento = vector< double >(6);
    diametroRevestimento[0] = 19;
    diametroRevestimento[1] = 30;
    diametroRevestimento[2] = 20;
    diametroRevestimento[3] = 13.375;
    diametroRevestimento[4] = 9.625;
    diametroRevestimento[5] = 7;

    capacidadeBroca = vector< double >(6);
    for (int i=0 ; i<6 ; i++)
    {
        capacidadeBroca[i] = 0.0031871*
            diametroBroca[i]*diametroBroca[i];
    }

    capacidadeRevestimento = vector< double >(6);
    for (int i=0 ; i<6 ; i++)
    {
        capacidadeRevestimento[i] = 0.0031871*
            diametroRevestimento[i]*
```

```
        diametroRevestimento[i];
    }

    diametroDP = 4.275;

    capacidadeDP = 0.0031871*diametroDP*diametroDP;

    AtualizarPropriedades();
}

/// Metodo para atribuir valores a profundidade
void SetProfundidade (vector< double > profundidade_)
{
    /// Salva a profundidade
    profundidade = profundidade_;

    /// Atualiza as outras propriedades do poço
    AtualizarPropriedades();
}

/// Metodo que atualiza as propriedades do poço
    dependentes da profundidade de cada fase
void AtualizarPropriedades()
{
    /// Atualiza o número de juntas
    NumeroJuntas ();

    /// Atualiza o volume de fluido
    VolumeFluido ();

    /// Atualiza o volume de cimento
    VolumeCimento ();
}

/// Metodo para retornar a profundidade
vector< double > GetProfundidade ()
{
    return profundidade;
}

/// Metodo para retornar o numero de tubos
vector< double > GetNumeroJuntas ()
{
    return numeroJuntas;
}

/// Metodo para retornar o volume de cimento
vector< double > GetVolumeCimento ()
```

```
{  
    return volumeCimento;  
}  
  
/// Metodo para retornar o volume de fluido de  
    perfuracao  
vector< double > GetVolumeFluido ()  
{  
    return volumeFluido;  
}  
  
/// Metodo para retornar a capacidade da broca  
vector< double > GetCapacidadeBroca ()  
{  
    return capacidadeBroca;  
}  
  
/// Metodo para retornar a capacidade do revestimento  
vector< double > GetCapacidadeRevestimento ()  
{  
    return capacidadeRevestimento;  
}  
  
/// Metodo para retornar a capacidade do drill pipe  
double GetCapacidadeDP ()  
{  
    return capacidadeDP;  
}  
  
/// Metodo que calcula o numero de tubos utilizados em  
    cada fase  
vector< double > NumeroJuntas ()  
{  
    numeroJuntas = vector< double >(6);  
    numeroJuntas[0] = profundidade[0] / 15.0;  
  
    for (int i = 1 ; i < 5 ; i++)  
    {  
        numeroJuntas[i] = (profundidade[i] -  
            profundidade[0]) / 12.0;  
    }  
  
    numeroJuntas[5] = (profundidade[5] -  
        profundidade[4] + 100.0) / 12.0;  
  
    return numeroJuntas;  
}
```

```

    /// Metodo que calcula o volume de fluido de perfuracao
    utilizado em cada fase; as fases 1 e 2 utilizam agua
    do mar e por isso nao sao calculadas
    vector< double > VolumeFluido ()
    {
        volumeFluido = vector< double >(6, 0);

        for (int i = 3 ; i < 6 ; i++)
        {
            volumeFluido[i] = 1.1*((profundidade[i]
                - profundidade[i-1] + 10)*
                capacidadeBroca[i] + (profundidade[i
                -1] - profundidade[0])*
                capacidadeRevestimento[i-1] +
                profundidade[0]*
                capacidadeRevestimento[0]);
        }

        return volumeFluido;
    }

    /// Metodo que calcula os volumes de cimento utilizados
    nas fases
    vector< double > VolumeCimento ()
    {
        volumeCimento = vector< double >(6);

        for (int i = 1 ; i < 3 ; i++)
        {
            volumeCimento[i] = 2.0*((profundidade[i]
                ]-profundidade[0])*(capacidadeBroca[i
                ]-capacidadeRevestimento[i]));
        }

        for (int i = 3 ; i < 5 ; i++)
        {
            volumeCimento[i] = (profundidade[i]-
                profundidade[i-1]-100.0)*(
                capacidadeBroca[i]-
                capacidadeRevestimento[i]);
        }

        volumeCimento[5] = (profundidade[5]-profundidade
            [4])*(capacidadeBroca[5] -
            capacidadeRevestimento[5])+100.0*(
            capacidadeRevestimento[4]-
            capacidadeRevestimento[5])+60.0*
            capacidadeRevestimento[4];
    }

```



```
        return volumeCimento;
    }

};

#endif
```

Listing 6.3: Arquivo de cabeçalho da classe CPoco.h

Apresenta-se na listagem 6.4 o arquivo com código da classe *CPoco.cpp*.

```
#include "CPoco.h"
```

Listing 6.4: Arquivo de cabeçalho da classe CPoco.cpp

Apresenta-se na listagem 6.5 o arquivo com código da classe *CPerfuracao.h*.

```
#ifndef CPERFURACAO_H
#define CPERFURACAO_H

#include "CPoco.h"

#include <string>
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

/// Classe CPerfuracao, responsavel pelos calculos da etapa de
/// perfuracao de um poco
class CPerfuracao
{
    /// Atributos
    private:

        /// Vetor para armazenar o tempo gasto na perfuracao,
        /// para cada fase do poco
        vector< double > tempoPerfuracao;

        /// Variavel que representa o tempo gasto para montar e
        /// desmontar o BHA, eh um valor fixo
        double montarBHA;

        /// Variavel que representa a velocidade de descida e
        /// subida do BHA
        double descerBHA;

        /// Vetor que representa a taxa de penetracao para cada
        /// fase do poco
```

```
vector< double > perfurar;

/// Variavel que representa o tempo de duracao da
    circulacao do fluido dentro do poço
double circular;

// Metodos
public:

    /// Criando objeto do tipo CPoco com o nome poco
    CPoco poco;

    /// Construtor, define valores para os atributos e chama
        o metodo CalcularPerfuracao
    CPerfuracao (CPoco poco_) : poco(poco_)
    {
        montarBHA = 8.0;

        descerBHA = 1.0/0.006;

perfurar = vector< double >(6);
        perfurar[0] = 0.0;
        perfurar[1] = 10.0;
        perfurar[2] = 10.0;
        perfurar[3] = 5.0;
        perfurar[4] = 5.0;
        perfurar[5] = 3.0;

        circular = 3.0;

        CalcularPerfuracao ();
    }

    /// Metodo para retornar os valores de taxa de
        perfuracao
    vector< double > GetPerfurar ()
    {
        return perfurar;
    }

    /// Metodo que calcula o tempo gasto em toda a etapa de
        perfuracao
    vector< double > CalcularPerfuracao ()
    {
        tempoPerfuracao = vector< double >(6);

        /// Cria um vetor profundidade para receber os
            valores do vetor de profundidades da classe
```

```

        CPoco
        vector< double > profundidade = poco.
            GetProfundidade ();

        /// Um "for" para realizar a conta para cada fase e
        preencher o vetor com os tempos gastos em cada fase
        for ( int i = 1 ; i < 6 ; i++ )
        {
            tempoPerfuracao[i] = 2.0*montarBHA +
                profundidade[i-1]/descerBHA + (
                    profundidade[i]-profundidade[i-1])/
                    perfurar[i] + profundidade[i]/
                    descerBHA + circular;
        }

        return tempoPerfuracao;
    }

    /// Metodo para retornar o vetor com os tempos gastos na
    perfuracao
    vector< double > GetTempoPerfuracao ()
    {
        return tempoPerfuracao;
    }
};

#endif

```

Listing 6.5: Arquivo de cabeçalho da classe CPerfuracao.h

Apresenta-se na listagem 6.6 o arquivo de implementação da classe *CPerfuracao.cpp*.

```
#include "CPerfuracao.h"
```

Listing 6.6: Arquivo de cabeçalho da classe CPerfuracao.cpp

Apresenta-se na listagem 6.7 o arquivo com código da classe *CRevestimento.h*.

```

#ifndef CREVESTIMENTO_H
#define CREVESTIMENTO_H

#include "CPoco.h"

#include <string>
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

```

```
/// Classe CRevestimento, responsavel pelos calculos da etapa de
    revestimento e cimentacao
class CRevestimento
{
    /// Atributos
    private:

        /// Vetor para armazenar o tempo gasto nesta etapa, para
            cada fase
        vector< double > tempoRevestimento;

        /// Vetor com os valores que representam a taxa de
            montagem do revestimento em cada fase (cada fase
            possui uma taxa diferente)
        vector< double > montarRevestimento;

        /// Variavel que representa a velocidade de descida do
            revestimento a poco aberto
        double descerAberto;

        /// Variavel que representa a velocidade de descida do
            revestimento a poco revestido
        double descerRevestido;

        /// Variavel que representa a velocidade de descida do
            revestimento por dentro do riser
        double descerRiser;

        /// Variavel que representa a vazao de boeiro de cimento
        double bombearCimento;

        /// Variavel que representa o tempo necessario para que
            o cimento de pega
        double esperarCimentoSecar;

        /// Variavel que representa a taxa de subida do drill
            pipe
        double subirDP;

    /// Metodos
    public:

        /// Criando objeto do tipo CPoco com o nome poco
        CPoco poco;

        /// Construtor, define valores para os atributos e chama
            o metodo CalcularRevestimento
}
```

```

CRevestimento (CPoco poco_) : poco(poco_)
{
    montarRevestimento = vector< double >(6);
    montarRevestimento[0] = 0.0;
    montarRevestimento[1] = 2.0;
    montarRevestimento[2] = 6.0;
    montarRevestimento[3] = 8.0;
    montarRevestimento[4] = 11.0;
    montarRevestimento[5] = 13.0;

    descerAberto = 240.0;

    descerRevestido = 360.0;

    descerRiser = 720.0;

    bombearCimento = 600.0;

    esperarCimentoSecar = 12.0;

    subirDP = 1.0/0.006;

    CalcularRevestimento ();
}

/// Metodo CalcularRevestimento, que calcula o tempo
/// gasto para a etapa de instalacao do revestimento e
/// cimentacao do poco
vector< double > CalcularRevestimento ()
{
    tempoRevestimento = vector< double >(6);

    /// Recebe da classe CPoco os valores de
    /// profundidade, numero de tubos, volumes de
    /// cimento e capacidades das brocas, dos
    /// revestimentos e do drill pipe
    vector< double > profundidade = poco.
        GetProfundidade ();
    vector< double > numeroJuntas = poco.
        GetNumeroJuntas ();
    vector< double > volumeCimento = poco.
        GetVolumeCimento ();
    vector< double > capacidadeBroca = poco.
        GetCapacidadeBroca ();
    vector< double > capacidadeRevestimento = poco.
        GetCapacidadeRevestimento ();
    double capacidadeDP = poco.GetCapacidadeDP ();
}

```

```

    /// Primeiro "for" para calcular o tempo para as
    /// duas primeiras fases
    for ( int i = 1 ; i < 3 ; i++ )
    {
        tempoRevestimento[i] = numeroJuntas[i]/
            montarRevestimento[i] + (profundidade
            [0]-(profundidade[i]-profundidade[0]))/
            descerAberto + ((profundidade[i]
            ]-50.0)*capacidadeDP+50.0*
            capacidadeRevestimento[i]+
            volumeCimento[i])/bombearCimento +
            esperarCimentoSecar + (profundidade[i]
            ]-50.0)/subirDP;
    }

    /// Segundo "for" para calcular o tempo para a
    /// terceira e quarta fases
    for ( int i = 3 ; i < 5 ; i++ )
    {
        tempoRevestimento[i] = numeroJuntas[i]/
            montarRevestimento[i] + (profundidade
            [i-1]-(profundidade[i]-profundidade
            [0]))/descerRevestido + (profundidade
            [i]-profundidade[i-1])/descerAberto +
            (profundidade[0]*
            capacidadeRevestimento[0]+(
            profundidade[i]-profundidade[0])*
            capacidadeRevestimento[i]+
            volumeCimento[i])/bombearCimento +
            esperarCimentoSecar + (profundidade
            [0])/subirDP;
    }

    /// Calculo do tempo para a quinta fase
    tempoRevestimento[5] = numeroJuntas[5]/
        montarRevestimento[5] + (profundidade[0]-(
        profundidade[5]-profundidade[4]+100.0))/
        descerRiser + (profundidade[4]-(profundidade
        [5]-profundidade[4]-100.0)-profundidade[0])/
        descerRevestido + (profundidade[5]-
        profundidade[4]-100.0)/descerAberto + ((
        profundidade[4]-100.0)*capacidadeDP+(
        profundidade[5]-profundidade[4]+100.0)*
        capacidadeRevestimento[4]+volumeCimento[5])/
        bombearCimento + esperarCimentoSecar + (
        profundidade[4]-100.0)/subirDP;

    return tempoRevestimento;

```

```

        }

        /// Metodo para retornar o tempo gasto na operacao, em
        /// cada fase
        vector< double > GetTempoRevestimento ()
        {
            return tempoRevestimento;
        }
    };

#endif

```

Listing 6.7: Arquivo de cabeçalho da classe CRevestimento.h

Apresenta-se na listagem 6.8 o arquivo de implementação da classe *CRevestimento.cpp*.

```
#include "CRevestimento.h"
```

Listing 6.8: Arquivo de cabeçalho da classe CRevestimento.cpp

Apresenta-se na listagem 6.9 o arquivo de implementação da classe *CGrafico.h*.

```

#ifndef CGRAFICO_H
#define CGRAFICO_H

#include "CPoco.h"
#include "CPerfuracao.h"
#include "CRevestimento.h"
#include "COperacao.h"
#include "CGnuplot.h"

#include <string>
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

/// Classe CGrafico, que ira solicitar os resultados das classe
/// COperacao, e calcular os pares ordenados do grafico a ser gerado
class CGrafico
{
    /// Atributos
    private:

        /// Vetor que armazena os valores de profundidade para
        /// gerar o grafico
        vector< double > profundidadeGrafico;

```

```
    /// Vetor que armazena os valores de profundidade para
    /// gerar o grafico
    vector< double > tempoGrafico;

    /// Vetor que armazena os valores de profundidade para
    /// gerar o grafico
    vector< double > tempoDescidaDP;

    /// Vetor que armazena os valores de profundidade para
    /// gerar o grafico
    vector< double > tempoTaxaPerfuracao;

    /// Vetor que armazena os valores de profundidade para
    /// gerar o grafico
    vector< double > tempoPatamar;

// Metodos
public:

    /// Criando objeto do tipo COperacao com o nome operacao
    COperacao operacao;

    /// Criando objeto do tipo CPoco com o nome poco
    CPoco poco;

    /// Criando objeto do tipo CPerfuracao com o nome
    /// perfuracao
    CPerfuracao perfuracao;

    /// Construtor, define valores para os atributos e chama
    /// o metodo AtualizarTempoBOP da classe COperacao
    CGrafico (COperacao operacao_, CPoco poco_, CPerfuracao
    perfuracao_) : operacao(operacao_), poco(poco_),
    perfuracao(perfuracao_)
    {

        /// Cria um vetor profundidade para receber os
        /// valores do vetor de profundidades da classe
        /// CPoco
        vector< double > profundidade = poco.
        GetProfundidade ();

        /// Carrega o vetor profundidadeGrafico com os
        /// seus valores de profundidade respectivos
        profundidadeGrafico = vector< double >(18);
        profundidadeGrafico[0] = 0.0;
        profundidadeGrafico[1] = profundidade[0];
        profundidadeGrafico[2] = profundidade[1];
        profundidadeGrafico[3] = profundidade[1];
    }
}
```



```
profundidadeGrafico[4] = profundidade[1];
profundidadeGrafico[5] = profundidade[2];
profundidadeGrafico[6] = profundidade[2];
profundidadeGrafico[7] = profundidade[2];
profundidadeGrafico[8] = profundidade[2];
profundidadeGrafico[9] = profundidade[3];
profundidadeGrafico[10] = profundidade[3];
profundidadeGrafico[11] = profundidade[3];
profundidadeGrafico[12] = profundidade[4];
profundidadeGrafico[13] = profundidade[4];
profundidadeGrafico[14] = profundidade[4];
profundidadeGrafico[15] = profundidade[5];
profundidadeGrafico[16] = profundidade[5];
profundidadeGrafico[17] = profundidade[5];

/// Variavel que representa o tempo gasto para
    montar e desmontar o BHA
double montarBHA = 8.0;

/// Variavel que representa a velocidade de
    descida e subida do BHA
double descerBHA = 1.0/0.006;

/// Vetor que armazena os tempos de descida dos
    DP de cada uma das fases
tempoDescidaDP = vector< double >(6);

for (int i=1 ; i<6 ; i++)
{
    tempoDescidaDP[i] = montarBHA +
        profundidade[i-1]/descerBHA;
}

/// Vetor que armazena as taxas de perfuracao de
    cada uma das fases
vector< double > perfurar = perfuracao.
    GetPerfurar ();

/// Vetor que armazena os tempos de perfuracao
    de cada uma das fases
tempoTaxaPerfuracao = vector< double >(6);

for (int i=1 ; i<6 ; i++)
{
    tempoTaxaPerfuracao[i] = (profundidade[i]
        ]-profundidade[i-1])/perfurar[i];
}
```

```
/// Vetor que armazena os tempos calculados pela
    classe COperacao
vector< double > TempoFase = operacao.TempoFase
    ();

///Vetor que armazena os tempos de subida de DP,
    revestiir e cimentar cada uma das fases
tempoPatamar = vector< double >(6);

for (int i=1 ; i<6 ; i++)
{
    tempoPatamar[i] = TempoFase[i]-
        tempoDescidaDP[i]-tempoTaxaPerfuracao
        [i];
}

/// Vetor que armazena os tempos de descida e
    subida do BOP, calculados pela classe
    COperacao
vector< double > TempoBOP = operacao.GetTempoBOP
    ();

/// Vetor que armazena os tempos para plotar o
    grafico
tempoGrafico = vector< double >(18);
tempoGrafico[0] = 0.0;
tempoGrafico[1] = tempoDescidaDP[1] +
    tempoGrafico[0];
tempoGrafico[2] = tempoTaxaPerfuracao[1] +
    tempoGrafico[1];
tempoGrafico[3] = tempoPatamar[1] + tempoGrafico
    [2];
tempoGrafico[4] = tempoDescidaDP[2] +
    tempoGrafico[3];
tempoGrafico[5] = tempoTaxaPerfuracao[2] +
    tempoGrafico[4];
tempoGrafico[6] = tempoPatamar[2] + tempoGrafico
    [5];
tempoGrafico[7] = TempoBOP[0] + tempoGrafico[6];
tempoGrafico[8] = tempoDescidaDP[3] +
    tempoGrafico[7];
tempoGrafico[9] = tempoTaxaPerfuracao[3] +
    tempoGrafico[8];
tempoGrafico[10] = tempoPatamar[3] +
    tempoGrafico[9];
tempoGrafico[11] = tempoDescidaDP[4] +
    tempoGrafico[10];
tempoGrafico[12] = tempoTaxaPerfuracao[4] +
```

```

        tempoGrafico[11];
        tempoGrafico[13] = tempoPatamar[4] +
            tempoGrafico[12];
        tempoGrafico[14] = tempoDescidaDP[5] +
            tempoGrafico[13];
        tempoGrafico[15] = tempoTaxaPerfuracao[5] +
            tempoGrafico[14];
        tempoGrafico[16] = tempoPatamar[5] +
            tempoGrafico[15];
        tempoGrafico[17] = TempoBOP[1] + tempoGrafico
            [16];
    }

    /// Metodo que gera o grafico
    void PlotarGrafico ()
    {
        CGnuplot gplot("lines");
        gplot.set_xlabel("Tempo");
        gplot.set_ylabel("Profundidade");
        gplot << "set yrange [*:~] reverse";
        gplot.plot_xy(tempoGrafico, profundidadeGrafico);
        gplot << "set terminal png size 1200,900";
        gplot << "set output 'Grafico.png'";
        gplot.plot_xy(tempoGrafico, profundidadeGrafico);
        cout << "Pressione <enter> para continuar" <<
            endl;
        cin.get();
    }
};

#endif

```

Listing 6.9: Arquivo de cabeçalho da classe CGrafico.h

Apresenta-se na listagem 6.10 o arquivo de implementação da classe *CGrafico.cpp*.

```
#include "CGrafico.h"
```

Listing 6.10: Arquivo de cabeçalho da classe CGrafico.cpp

Apresenta-se na listagem ?? o arquivo de implementação da função *main.cpp*.

```

#include "CPoco.h"
#include "CPerfuracao.h"
#include "CRevestimento.h"
#include "COperacao.h"
#include "CGrafico.h"
#include "CGnuplot.h"

```

```

#include <string>
#include <vector>
#include <iostream>
#include <fstream>

using namespace std;

int main ()
{
    cout << "\n----- Projeto de Poco
    -----" ;
    cout << "\n\nAutores: Fillipe Rodrigues Sias\n\t Lucas Armando
    de Carvalho" << endl;
    cout << "\nProf. Andre Duarte Bueno" << endl;
    cout << "\nLicenca GPL 2.0" << endl;
    cout << "\n
    -----
    n\n" << endl;

    /// Vetor que armazena as profundidades das fases, para
    posteriormente passar para o objeto poco
    vector< double > profundidade (6);

    cout << "\n Entre com o valor da Lamina de Agua (metros):\t\t\t"
    ;
    cin >> profundidade [0];
    cin.get();

    for (int i=1; i<6 ; i++)
    {
        cout << "\n Entre com o valor da Profundidade da Fase "
        << i << " (metros):\t\t";
        cin >> profundidade [i];
        cin.get();
    }

    /// Cria objeto poco do tipo CPoco e utiliza o construtor da classe
    para passar os valores de profundidade inseridos pelo usuario
    CPoco poco = CPoco (profundidade);

    /// Passa os dados do objeto poco para o objeto perfuracao, do
    tipo CPerfuracao
    CPerfuracao perfuracao = CPerfuracao (poco);

    /// Passa os dados do poco para o objeto revestimento, do tipo
    CRevestimento
    CRevestimento revestimento = CRevestimento (poco);

```

```

    /// Passa os dados do poço, perfuração e revestimento para o
    /// objeto operacao, do tipo COperacao
    COperacao operacao = COperacao (poco, perfuracao, revestimento);

    /// Vetor que armazena o numero de tubos de riser e revestimento
    /// por fase, este vetor Ã© de numeros inteiros por se tratar de
    /// uma contagem de pecas, sendo que suas fracoes sao
    /// descartadas
    vector< double > numeroJuntas = poco.GetNumeroJuntas ();

    /// Vetor que armazena os tempos calculados pela classe
    /// COperacao
    vector< double > TempoFase = operacao.TempoFase ();

    /// Vetor que armazena os tempos de descida e subida do BOP,
    /// calculados pela classe COperacao
    vector< double > TempoBOP = operacao.GetTempoBOP ();

    /// Retorna vetor com volume de fluido de perfuracao calculado na
    /// Classe CPoco
    vector< double > volumeFluido = poco.GetVolumeFluido();

    /// Variavel para calcular volume total de fluido de perfuracao
    /// utilizado
    double fluidoTotal = 0;

    for (int i = 3 ; i < 6 ; i++)
    {
        fluidoTotal = fluidoTotal + volumeFluido[i];
    }

    /// Retorna vetor com volume de cimento calculado na Classe
    /// CPoco
    vector< double > volumeCimento = poco.GetVolumeCimento();

    /// Variavel para calcular volume total de cimento utilizado
    double cimentoTotal = 0;

    for (int i = 1 ; i < 6 ; i++)
    {
        cimentoTotal = cimentoTotal + volumeCimento[i];
    }

    /// Saida dos resultados na tela

    cout << "\n\n\n----- Tubos Necessarios
    -----" << endl;

```

```

cout << "\n\nTubos de Riser na Lamina de Agua: \t\t\t" <<
    numeroJuntas [0] << " tubos" << endl;
cout << "\nTubos de Revestimento de 30 polegadas na Fase 1: \t"
    << numeroJuntas [1] << " tubos" << endl;
cout << "\nTubos de Revestimento de 20 polegadas na Fase 2: \t"
    << numeroJuntas [2] << " tubos" << endl;
cout << "\nTubos de Revestimento de 13 3/8 polegadas na Fase 3:
    \t" << numeroJuntas [3] << " tubos" << endl;
cout << "\nTubos de Revestimento de 9 5/8 polegadas na Fase 4: \t"
    << numeroJuntas [4] << " tubos" << endl;
cout << "\nTubos de Liner de 7 polegadas na Fase 5: \t\t" <<
    numeroJuntas [5] << " tubos" << endl;

cout << "\n\n\n----- Duracao da Operacao
    -----" << endl;

cout << "\n\nDuracao da Fase 1: \t" << TempoFase [1] << " horas"
    << endl;
cout << "\nDuracao da Fase 2: \t" << TempoFase [2] << " horas"
    << endl;
cout << "\nDescida do BOP: \t" << TempoBOP[0] << " horas" <<
    endl;
cout << "\nDuracao da Fase 3: \t" << TempoFase [3] << " horas"
    << endl;
cout << "\nDuracao da Fase 4: \t" << TempoFase [4] << " horas"
    << endl;
cout << "\nDuracao da Fase 5: \t" << TempoFase [5] << " horas"
    << endl;
cout << "\nSubida do BOP: \t\t" << TempoBOP[1] << " horas" <<
    endl;

cout << "\nTempo total: \t\t" << TempoFase [1] + TempoFase [2] +
    TempoFase [3] + TempoFase [4] + TempoFase [5] + TempoBOP[0] +
    TempoBOP[1] << " horas" << endl;
    cout << "\nTempo total: \t\t" << (TempoFase [1] + TempoFase [2]
        + TempoFase [3] + TempoFase [4] + TempoFase [5] + TempoBOP[0]
        + TempoBOP[1]) / 24.0 << " dias" << endl;

cout << "\n\n\n----- Volumes
    -----" << endl;

cout << "\n\nFase 1 e 2 utilizam agua do mar como fluido de
    perfuracao" << endl;
cout << "\nVolume de fluido de perfuracao usado na Fase 3: \t"
    << volumeFluido[3] << " barris" << endl;
cout << "\nVolume de fluido de perfuracao usado na Fase 4: \t" <<
    volumeFluido[4] << " barris" << endl;
cout << "\nVolume de fluido de perfuracao usado na Fase 5: \t" <<

```

```

        volumeFluido[5] << " barris" << endl;
    cout << "\nVolume Total de fluido de perfuracao usado: \t\t" <<
        fluidoTotal << " barris" << endl;

    cout << "\n\nVolume de cimento usado na Fase 1: \t\t\t" <<
        volumeCimento[1] << " barris" << endl;
    cout << "\nVolume de cimento usado na Fase 2: \t\t\t" <<
        volumeCimento[2] << " barris" << endl;
    cout << "\nVolume de cimento usado na Fase 3: \t\t\t" <<
        volumeCimento[3] << " barris" << endl;
    cout << "\nVolume de cimento usado na Fase 4: \t\t\t" <<
        volumeCimento[4] << " barris" << endl;
    cout << "\nVolume de cimento usado na Fase 5: \t\t\t" <<
        volumeCimento[5] << " barris" << endl;
        cout << "\nVolume Total de cimento usado: \t\t\t\t" <<
            cimentoTotal << " barris" << endl;

    /// Variavel para receber a resposta do usuario quanto a salvar em
    disco
    string resposta;

    cout << "\n\nDeseja salvar os resultados em disco? (s/n): ";

    cin >> resposta;
    cin.get();

    if ( resposta == "s")
    {
        /// Saida dos resultados em disco
        string nomeArquivo;

        cout << "\n\nEntre com o nome do arquivo em que os
            resultados serao salvos: " << endl;

        cin >> nomeArquivo;
        cin.get();

        nomeArquivo = nomeArquivo + ".txt";

        ofstream fout (nomeArquivo.c_str());

        fout << "\n----- Resultados
            -----" << endl;

        fout << "\n\nLamina de Agua: \t";
        fout << profundidade[0] << " metros" << endl;

        for(int i = 1 ; i < 6 ; i++)

```

```

{
    fout << "\nProfundida da Fase "<< i <<": \t";
    fout << profundidade[i] << " metros" << endl;
}

fout << "\n\n\n----- Tubos
Necessarios -----" << endl;

fout << "\n\nTubos de Riser na Lamina de Agua: \t\t\t"
<< numeroJuntas [0] << " tubos" << endl;
fout << "\nTubos de Revestimento de 30 polegadas na Fase
1: \t" << numeroJuntas [1] << " tubos" << endl;
fout << "\nTubos de Revestimento de 20 polegadas na Fase
2: \t" << numeroJuntas [2] << " tubos" << endl;
fout << "\nTubos de Revestimento de 13 3/8 polegadas na
Fase 3: \t" << numeroJuntas [3] << " tubos" << endl;
fout << "\nTubos de Revestimento de 9 5/8 polegadas na
Fase 4: \t" << numeroJuntas [4] << " tubos" << endl;
fout << "\nTubos de Liner de 7 polegadas na Fase 5: \t\t"
" << numeroJuntas [5] << " tubos" << endl;

fout << "\n\n\n----- Duracao da
Operacao -----" << endl;

fout << "\n\nDuracao da Fase 1: \t" << TempoFase [1] <<
" horas" << endl;
fout << "\nDuracao da Fase 2: \t" << TempoFase [2] << "
horas" << endl;
fout << "\nDescida do BOP: \t" << TempoBOP[0] << " horas
" << endl;
fout << "\nDuracao da Fase 3: \t" << TempoFase [3] << "
horas" << endl;
fout << "\nDuracao da Fase 4: \t" << TempoFase [4] << "
horas" << endl;
fout << "\nDuracao da Fase 5: \t" << TempoFase [5] << "
horas" << endl;
fout << "\nSubida do BOP: \t\t" << TempoBOP[1] << "
horas" << endl;
fout << "\nTempo total: \t\t" << TempoFase [1] + TempoFase
[2] + TempoFase [3] + TempoFase [4] + TempoFase [5] +
TempoBOP[0] + TempoBOP[1] << " horas" << endl;
fout << "\nTempo total: \t\t" << (TempoFase [1] +
TempoFase [2] + TempoFase [3] + TempoFase [4] +
TempoFase [5] + TempoBOP[0] + TempoBOP[1]) / 24 << "
dias" << endl;

fout << "\n\n\n----- Volumes
-----" << endl;

```



```

        fout << "\n\nFase 1 e 2 utilizam agua do mar como fluido
        de perfuracao" << endl;
        fout << "\nVolume de fluido de perfuracao usado na Fase
        3: \t" << volumeFluido[3] << " barris" << endl;
        fout << "\nVolume de fluido de perfuracao usado na Fase 4: \
        t" << volumeFluido[4] << " barris" << endl;
        fout << "\nVolume de fluido de perfuracao usado na Fase 5: \
        t" << volumeFluido[5] << " barris" << endl;
        fout << "\nVolume Total de fluido de perfuracao usado: \t\t"
        << fluidoTotal << " barris" << endl;

        fout << "\n\nVolume de cimento usado na Fase 1: \t\t\t" <<
        volumeCimento[1] << " barris" << endl;
        fout << "\nVolume de cimento usado na Fase 2: \t\t\t" <<
        volumeCimento[2] << " barris" << endl;
        fout << "\nVolume de cimento usado na Fase 3: \t\t\t" <<
        volumeCimento[3] << " barris" << endl;
        fout << "\nVolume de cimento usado na Fase 4: \t\t\t" <<
        volumeCimento[4] << " barris" << endl;
        fout << "\nVolume de cimento usado na Fase 5: \t\t\t" <<
        volumeCimento[5] << " barris" << endl;
        fout << "\nVolume Total de cimento usado: \t\t\t\t" <<
        cimentoTotal << " barris" << endl;

        fout.close ();

    }

    else
    {
        cout << "\n";
    }

    /// Cria objeto do tipo CGrafico
    CGrafico grafico(operacao, poco, perfuracao);

    string resposta_grafico;

    cout << "\n\nDeseja gerar grafico? (s/n): ";
    cin >> resposta_grafico;
    cin.get();

    if (resposta_grafico == "s")
    {
        grafico.PlotarGrafico();
    }

```

```
        cout << "\n\n----- FIM DO PROGRAMA\n-----";  
    }  
    else  
    {  
        cout << "\n\n----- FIM DO PROGRAMA\n-----";  
    }  
  
    cin.get();  
  
    return 0;  
}
```

Listing 6.11: Arquivo de implementação da função main.cpp

Capítulo 7

Teste

Neste capítulo serão apresentados alguns testes do programa Projeto de Poço, a fim de orientar o usuário na utilização do software em questão.

7.1 Teste 1: Entrando com valores comuns ao problema, observando os resultados em tela e salvando arquivo

Neste teste estamos apresentando como utilizar o programa e apresentaremos também as opções que ele oferece de saída de dados.

Primeiramente, será apresentado na Figura 7.1, como inserir as profundidades das fases do poço.

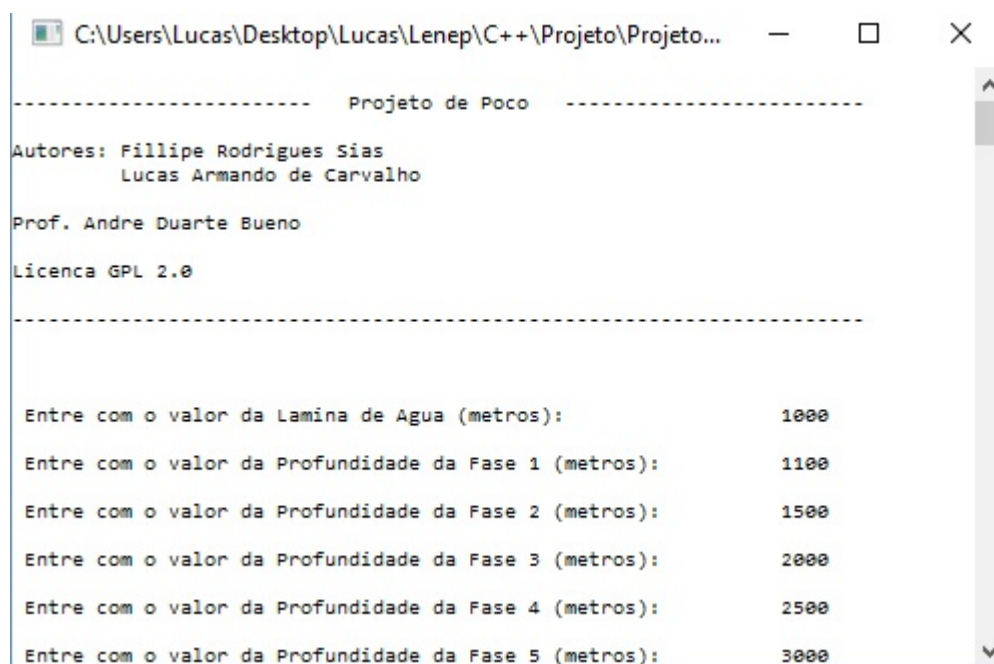
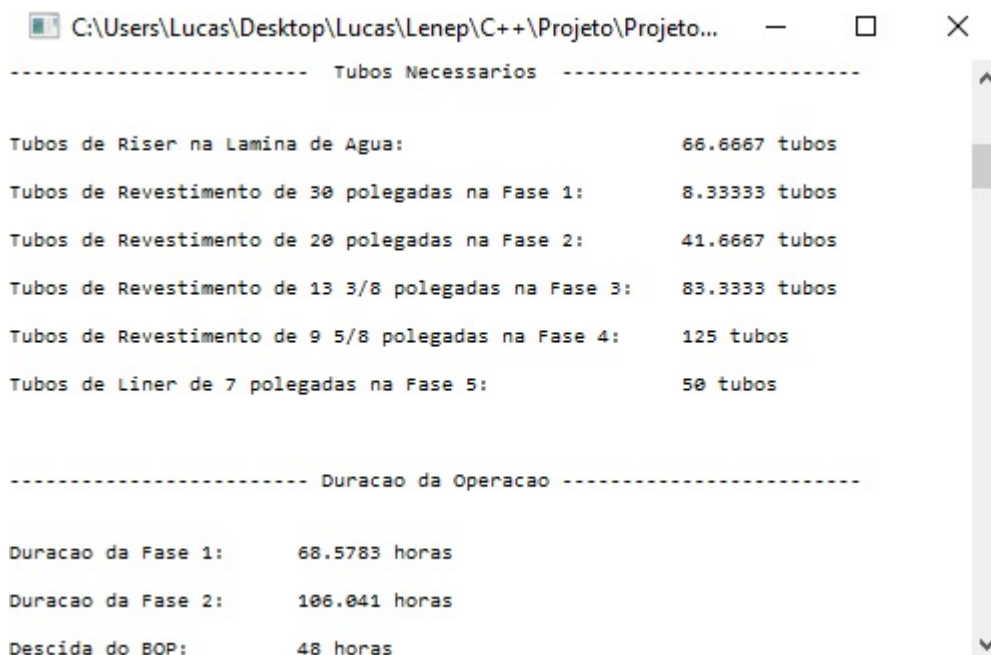


Figura 7.1: Entrada de dados.

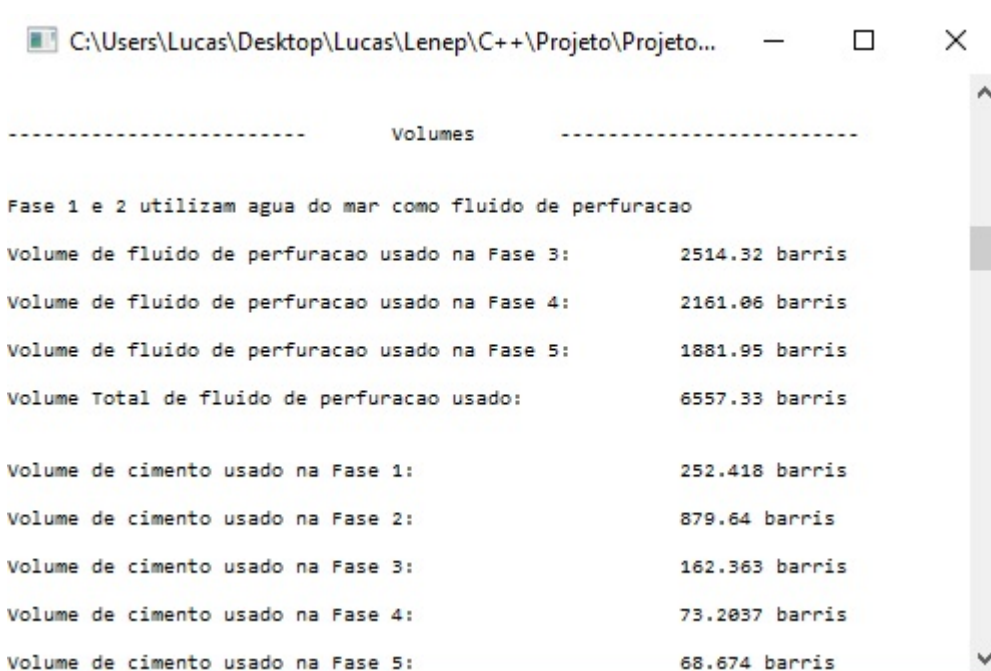
Nas Figuras 7.2 e 7.3 pode ser visto a apresentação dos resultados em tela, onde o usuário pode visualizar os resultados sem precisar salvá-los.



Tubos Necessarios	
Tubos de Riser na Lamina de Agua:	66.6667 tubos
Tubos de Revestimento de 30 polegadas na Fase 1:	8.33333 tubos
Tubos de Revestimento de 20 polegadas na Fase 2:	41.6667 tubos
Tubos de Revestimento de 13 3/8 polegadas na Fase 3:	83.3333 tubos
Tubos de Revestimento de 9 5/8 polegadas na Fase 4:	125 tubos
Tubos de Liner de 7 polegadas na Fase 5:	50 tubos

Duracao da Operacao	
Duracao da Fase 1:	68.5783 horas
Duracao da Fase 2:	106.041 horas
Descida do BOP:	48 horas

Figura 7.2: Resultados em tela.



Volumes	
Fase 1 e 2 utilizam agua do mar como fluido de perfuracao	
Volume de fluido de perfuracao usado na Fase 3:	2514.32 barris
Volume de fluido de perfuracao usado na Fase 4:	2161.06 barris
Volume de fluido de perfuracao usado na Fase 5:	1881.95 barris
Volume Total de fluido de perfuracao usado:	6557.33 barris
Volume de cimento usado na Fase 1:	252.418 barris
Volume de cimento usado na Fase 2:	879.64 barris
Volume de cimento usado na Fase 3:	162.363 barris
Volume de cimento usado na Fase 4:	73.2037 barris
Volume de cimento usado na Fase 5:	68.674 barris

Figura 7.3: Resultados em tela.

A seguir, é apresentado na Figura 7.4 um exemplo de salvamento dos resultados em arquivo de disco. Primeiramente escolhe-se a opção de salvar resultados em disco e em seguida define-se um nome para o arquivo.

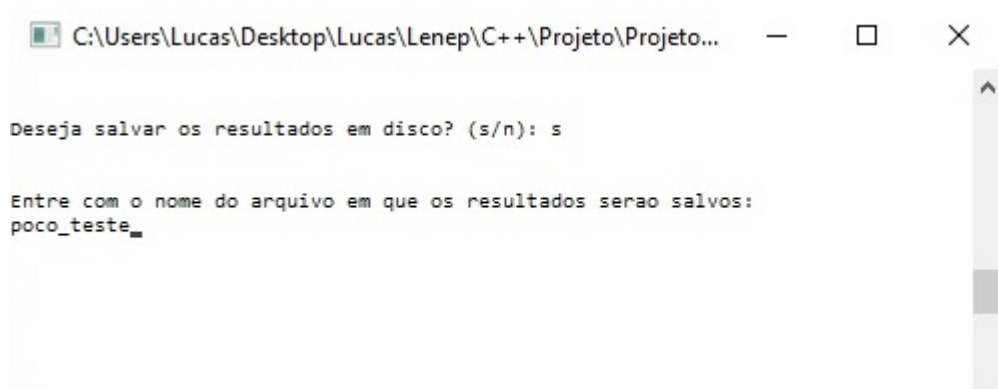


Figura 7.4: Salvando arquivo de disco.

Em seguida, é oferecida ao usuário a opção de gerar um gráfico de tempo *versus* profundidade, conforme mostra a Figura 7.5.

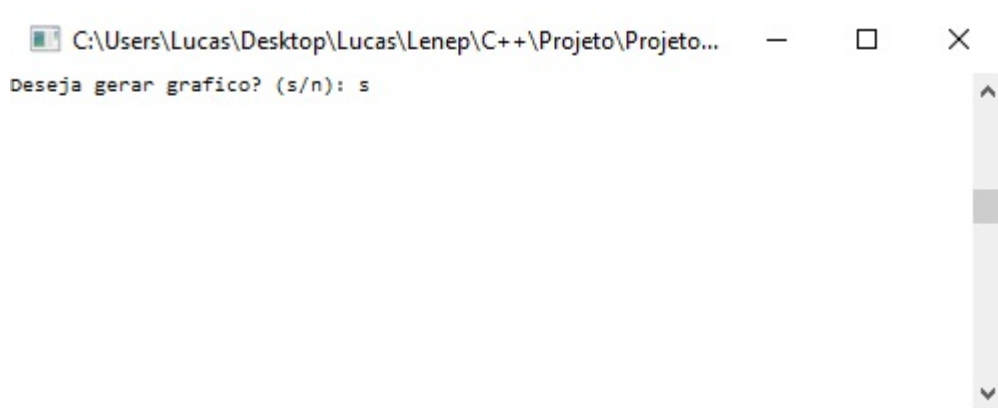


Figura 7.5: Escolhendo gerar gráfico.

A Figura 7.6 mostra um exemplo de gráfico gerado.

7.2 Teste 2: Testando valores de profundidade errados

Primeiramente será apresentado na Figura 7.7 a entrada com os dados errados. O programa trabalha com valores de profundidade medida e não com as diferenças entre fases, e por isso a profundidade da fase seguinte deve sempre ser maior que a da fase anterior.

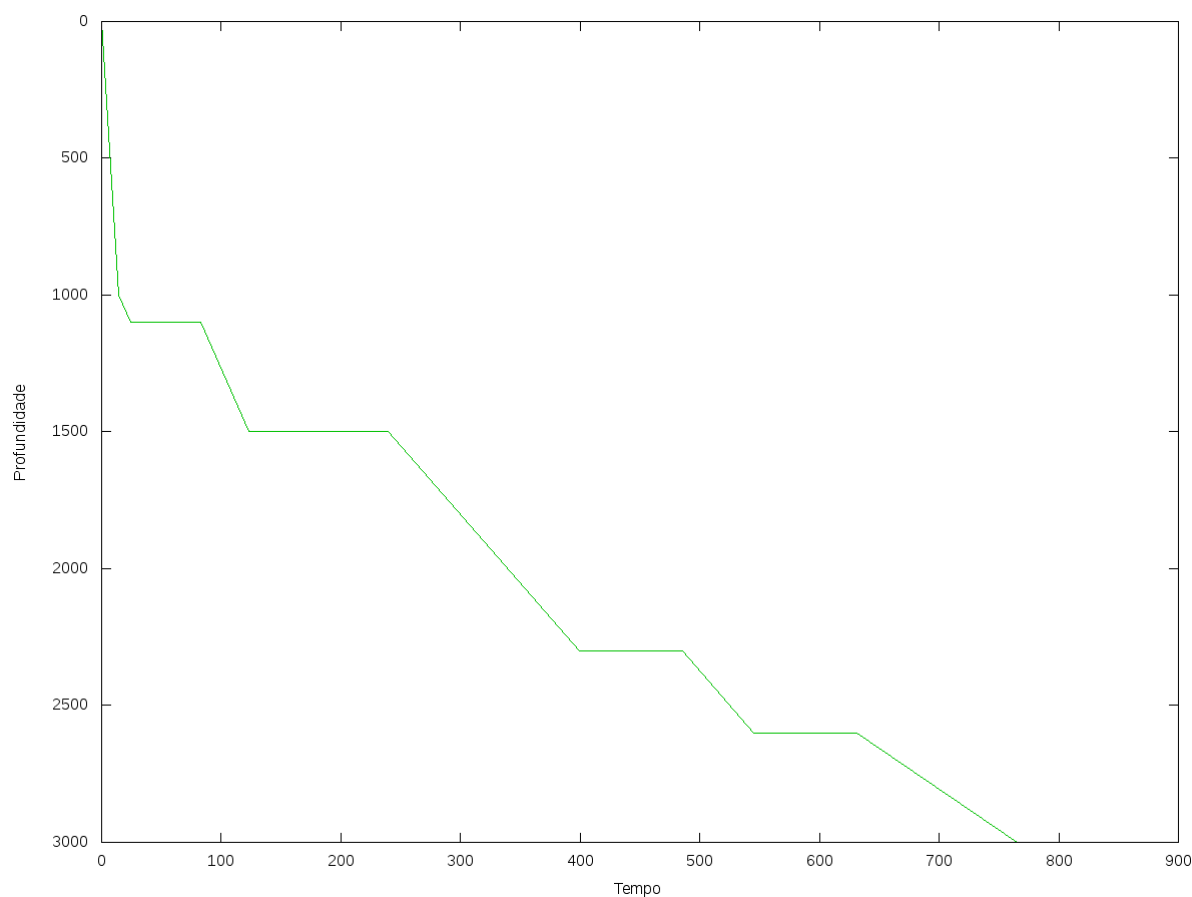


Figura 7.6: Gráfico do tempo *versus* profundidade gerado.

C:\Users\Lucas\Desktop\Lucas\Lenep\C++\Projeto\Projeto... — □ ×

----- Projeto de POCO -----

Autores: Fillipe Rodrigues Sias
Lucas Armando de Carvalho

Prof. Andre Duarte Bueno

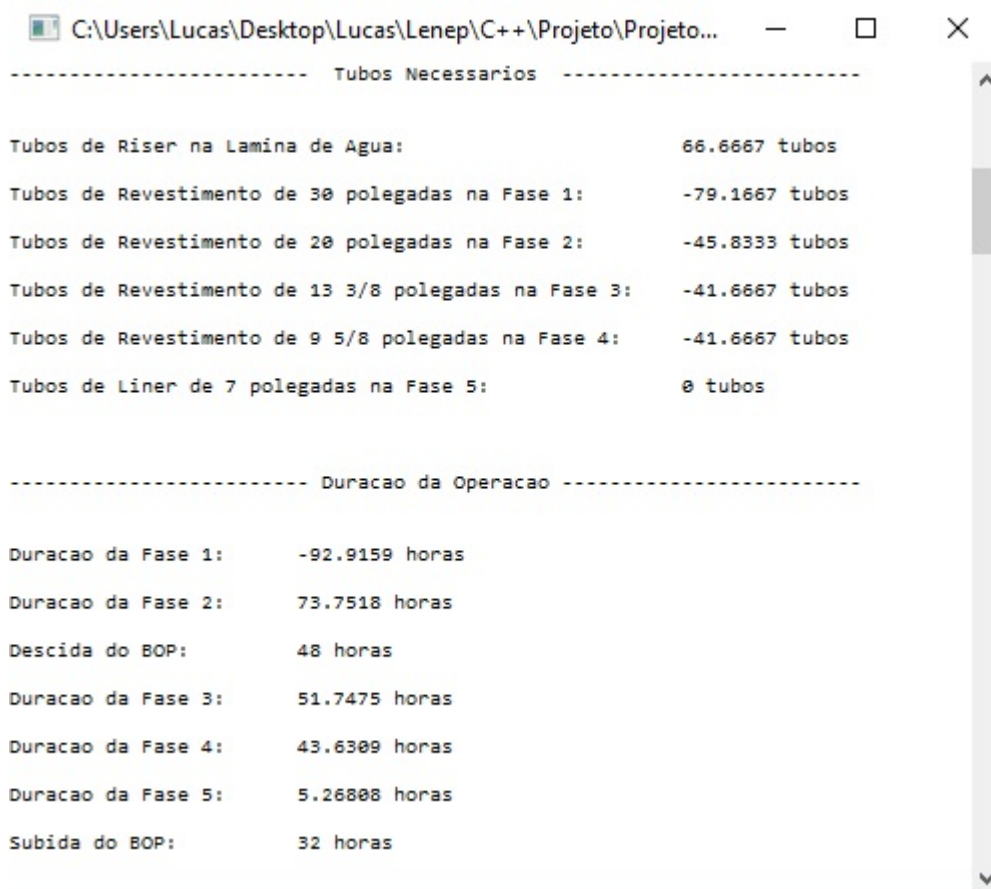
Licença GPL 2.0

Entre com o valor da Lamina de Agua (metros):	1000
Entre com o valor da Profundidade da Fase 1 (metros):	50
Entre com o valor da Profundidade da Fase 2 (metros):	450
Entre com o valor da Profundidade da Fase 3 (metros):	500
Entre com o valor da Profundidade da Fase 4 (metros):	500
Entre com o valor da Profundidade da Fase 5 (metros):	400

Figura 7.7: Entrada de dados errados.

Os resultados são apresentados nas Figuras 7.8 e 7.9. Como pode ser observado, quando o usuário fornece profundidades irreais das fases do poço, como uma fase com

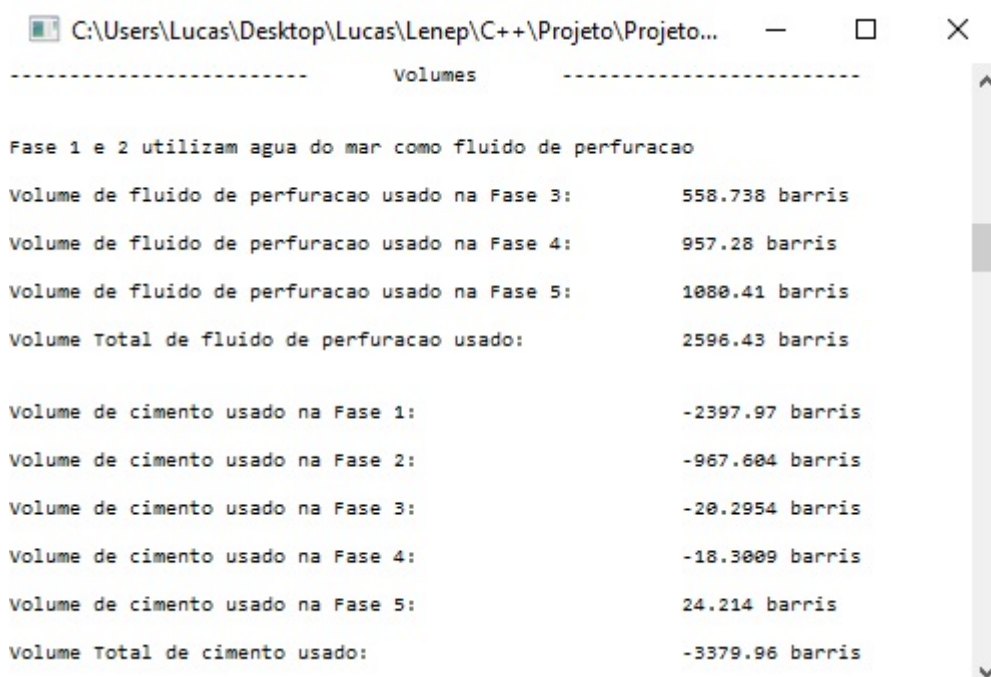
profundidade menor ou igual à fase anterior, as respostas obtidas também não são realizadas, apresentando valores negativos ou nulos.



Tubos Necessarios	
Tubos de Riser na Lamina de Agua:	66.6667 tubos
Tubos de Revestimento de 30 polegadas na Fase 1:	-79.1667 tubos
Tubos de Revestimento de 20 polegadas na Fase 2:	-45.8333 tubos
Tubos de Revestimento de 13 3/8 polegadas na Fase 3:	-41.6667 tubos
Tubos de Revestimento de 9 5/8 polegadas na Fase 4:	-41.6667 tubos
Tubos de Liner de 7 polegadas na Fase 5:	0 tubos

Duracao da Operacao	
Duracao da Fase 1:	-92.9159 horas
Duracao da Fase 2:	73.7518 horas
Descida do BOP:	48 horas
Duracao da Fase 3:	51.7475 horas
Duracao da Fase 4:	43.6309 horas
Duracao da Fase 5:	5.26808 horas
Subida do BOP:	32 horas

Figura 7.8: Resultados da entrada errada.



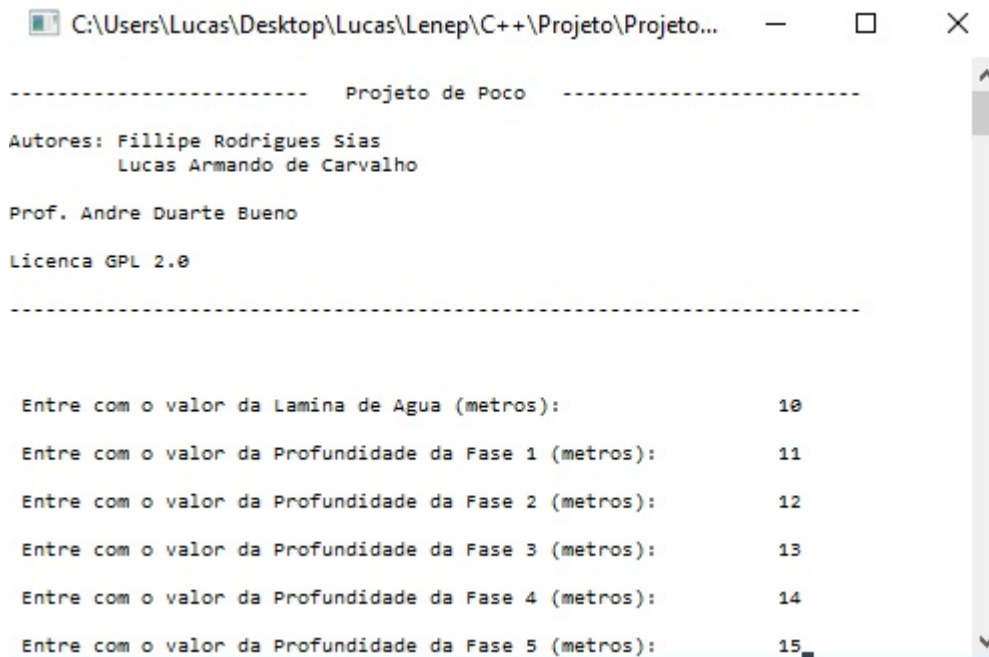
Fase 1 e 2 utilizam agua do mar como fluido de perfuracao	
Volume de fluido de perfuracao usado na Fase 3:	558.738 barris
Volume de fluido de perfuracao usado na Fase 4:	957.28 barris
Volume de fluido de perfuracao usado na Fase 5:	1080.41 barris
Volume Total de fluido de perfuracao usado:	2596.43 barris

Volumes	
Volume de cimento usado na Fase 1:	-2397.97 barris
Volume de cimento usado na Fase 2:	-967.604 barris
Volume de cimento usado na Fase 3:	-20.2954 barris
Volume de cimento usado na Fase 4:	-18.3009 barris
Volume de cimento usado na Fase 5:	24.214 barris
Volume Total de cimento usado:	-3379.96 barris

Figura 7.9: Resultados da entrada errada.

7.3 Teste 3: Testando valores de profundidade que não representam o problema real

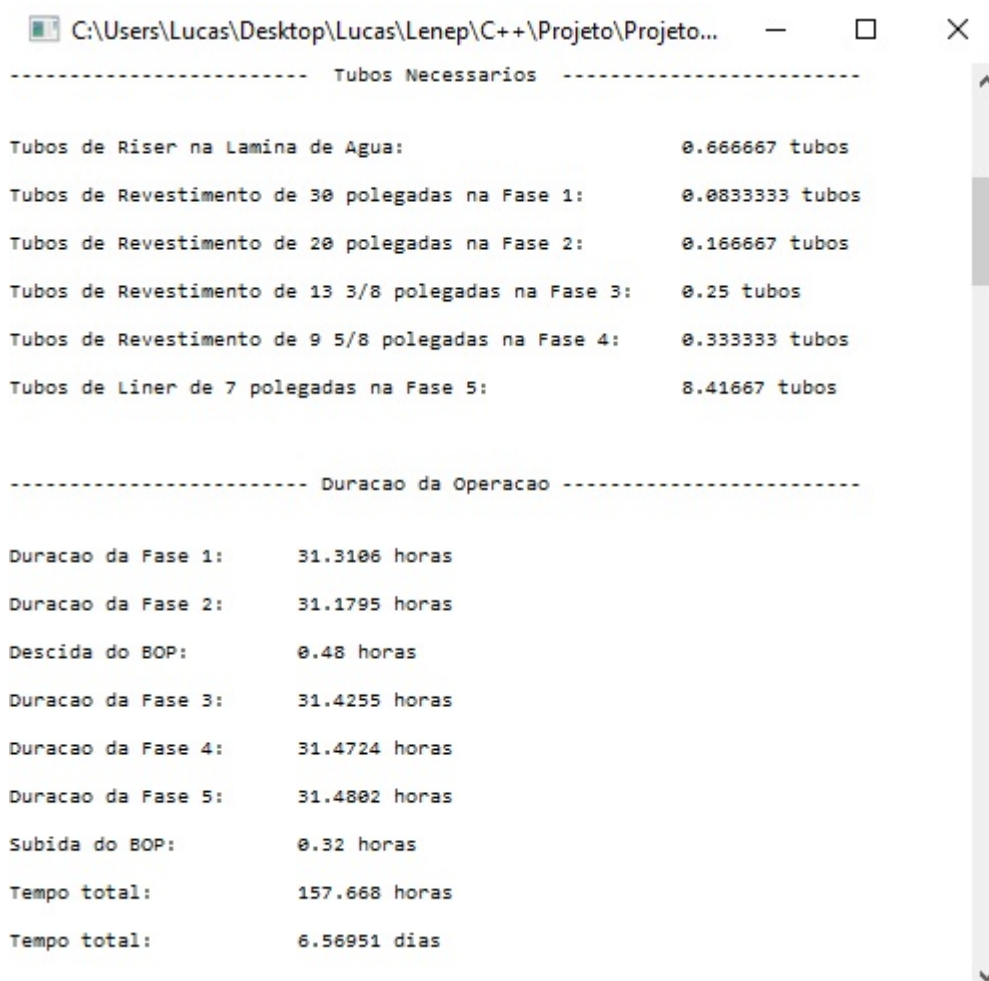
Na Figura 7.10 temos os dados de entrada que não representam as dimensões de um poço real.



```
----- Projeto de POCO -----  
Autores: Fillipe Rodrigues Sias  
        Lucas Armando de Carvalho  
  
Prof. Andre Duarte Bueno  
  
Licenca GPL 2.0  
  
-----  
  
Entre com o valor da Lamina de Agua (metros):      10  
Entre com o valor da Profundidade da Fase 1 (metros):  11  
Entre com o valor da Profundidade da Fase 2 (metros):  12  
Entre com o valor da Profundidade da Fase 3 (metros):  13  
Entre com o valor da Profundidade da Fase 4 (metros):  14  
Entre com o valor da Profundidade da Fase 5 (metros):  15
```

Figura 7.10: Entrada de dados que não correspondem ao problema real.

Nos resultados apresentados nas Figuras 7.11 e 7.12 podemos ver que os cálculos realizados pelo programa não são consistentes.



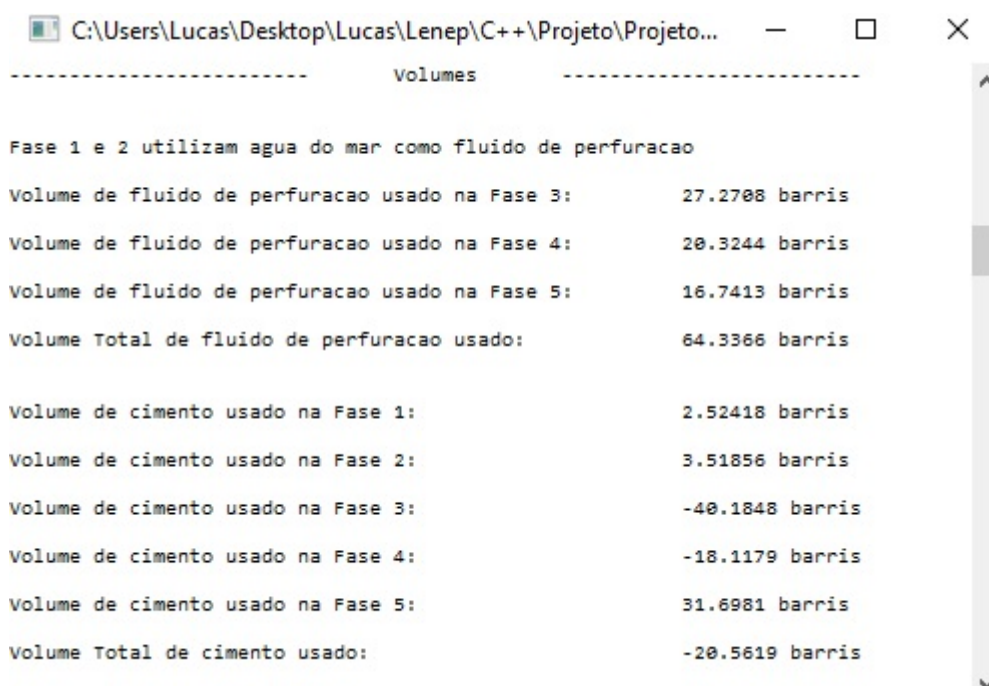
----- Tubos Necessarios -----

Tubos de Riser na Lamina de Agua:	0.666667 tubos
Tubos de Revestimento de 30 polegadas na Fase 1:	0.0833333 tubos
Tubos de Revestimento de 20 polegadas na Fase 2:	0.166667 tubos
Tubos de Revestimento de 13 3/8 polegadas na Fase 3:	0.25 tubos
Tubos de Revestimento de 9 5/8 polegadas na Fase 4:	0.333333 tubos
Tubos de Liner de 7 polegadas na Fase 5:	8.41667 tubos

----- Duracao da Operacao -----

Duracao da Fase 1:	31.3106 horas
Duracao da Fase 2:	31.1795 horas
Descida do BOP:	0.48 horas
Duracao da Fase 3:	31.4255 horas
Duracao da Fase 4:	31.4724 horas
Duracao da Fase 5:	31.4802 horas
Subida do BOP:	0.32 horas
Tempo total:	157.668 horas
Tempo total:	6.56951 dias

Figura 7.11: Resultados da entrada não real.



----- Volumes -----

Fase 1 e 2 utilizam agua do mar como fluido de perfuracao

Volume de fluido de perfuracao usado na Fase 3:	27.2708 barris
Volume de fluido de perfuracao usado na Fase 4:	20.3244 barris
Volume de fluido de perfuracao usado na Fase 5:	16.7413 barris
Volume Total de fluido de perfuracao usado:	64.3366 barris
Volume de cimento usado na Fase 1:	2.52418 barris
Volume de cimento usado na Fase 2:	3.51856 barris
Volume de cimento usado na Fase 3:	-40.1848 barris
Volume de cimento usado na Fase 4:	-18.1179 barris
Volume de cimento usado na Fase 5:	31.6981 barris
Volume Total de cimento usado:	-20.5619 barris

Figura 7.12: Resultados da entrada não real.

Capítulo 8

Documentação

A presente documentação refere-se ao uso do Programa "Projeto de Poço". Esta documentação tem o formato de uma apostila que explica passo a passo como usar o programa.

8.1 Manual do Usuário

Neste manual serão abordados os pontos principais do uso do programa, mostrando a instalação, execução e a opção de salvar os dados em disco.

Para usar o programa, primeiramente é preciso executá-lo. Na plataforma GNU/Linux, o procedimento deve ser feito usando o seguinte comando no terminal, estando no diretório do arquivo executável do programa.

```
./a.out
```

Após ser executado, a tela inicial do programa se mostrará como na Figura 8.1.

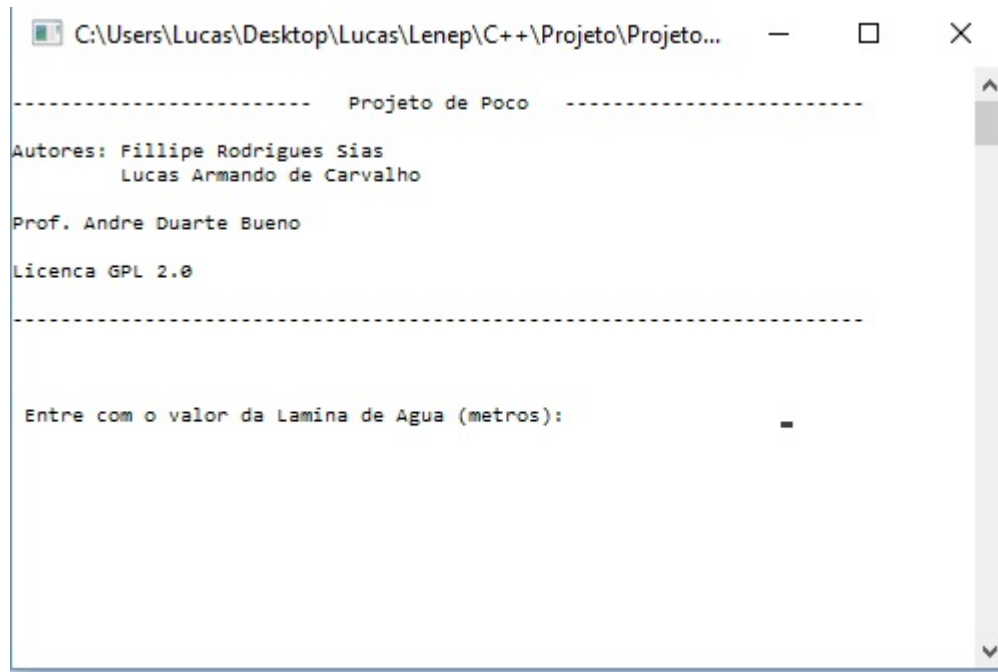


Figura 8.1: Tela inicial do Programa.

Como podemos ver na Figura 8.1, o usuário deverá inserir as profundidades para cada fase, iniciando pela profundidade da lâmina de água. Após inserir as profundidades, serão apresentados na tela os resultados e uma mensagem, na qual o usuário poderá optar entre salvar os resultados em um arquivo de texto ou não, conforme mostrado na Figura 8.2.

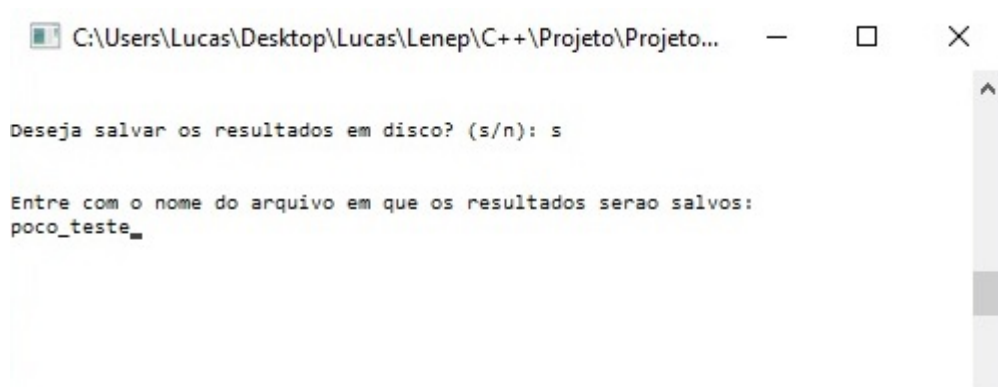


Figura 8.2: Escolhendo salvar resultados em arquivo de texto.

Caso o usuário opte por salvar os resultados, uma nova mensagem será apresentada, onde o usuário deverá definir um nome para o arquivo a ser salvo. Após a finalização do programa, o arquivo estará armazenado na mesma pasta do executável.

No caso de o usuário optar por não salvar os resultados, o programa passa para a etapa seguinte, onde é apresentada a opção de gerar um gráfico do tempo da operação *versus* profundidade atingida. Essa opção é apresentada na Figura 8.3.

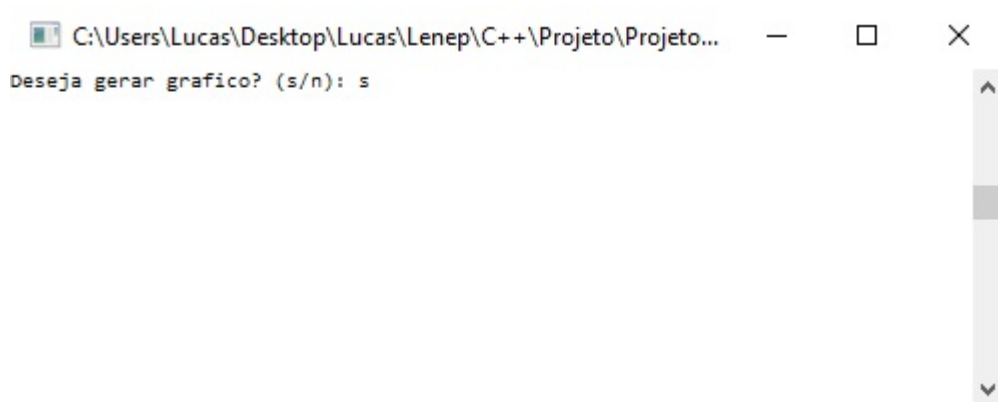


Figura 8.3: Opção de gerar um gráfico de tempo *versus* profundidade.

A Figura 8.4 mostra um exemplo do arquivo de texto com os resultado.

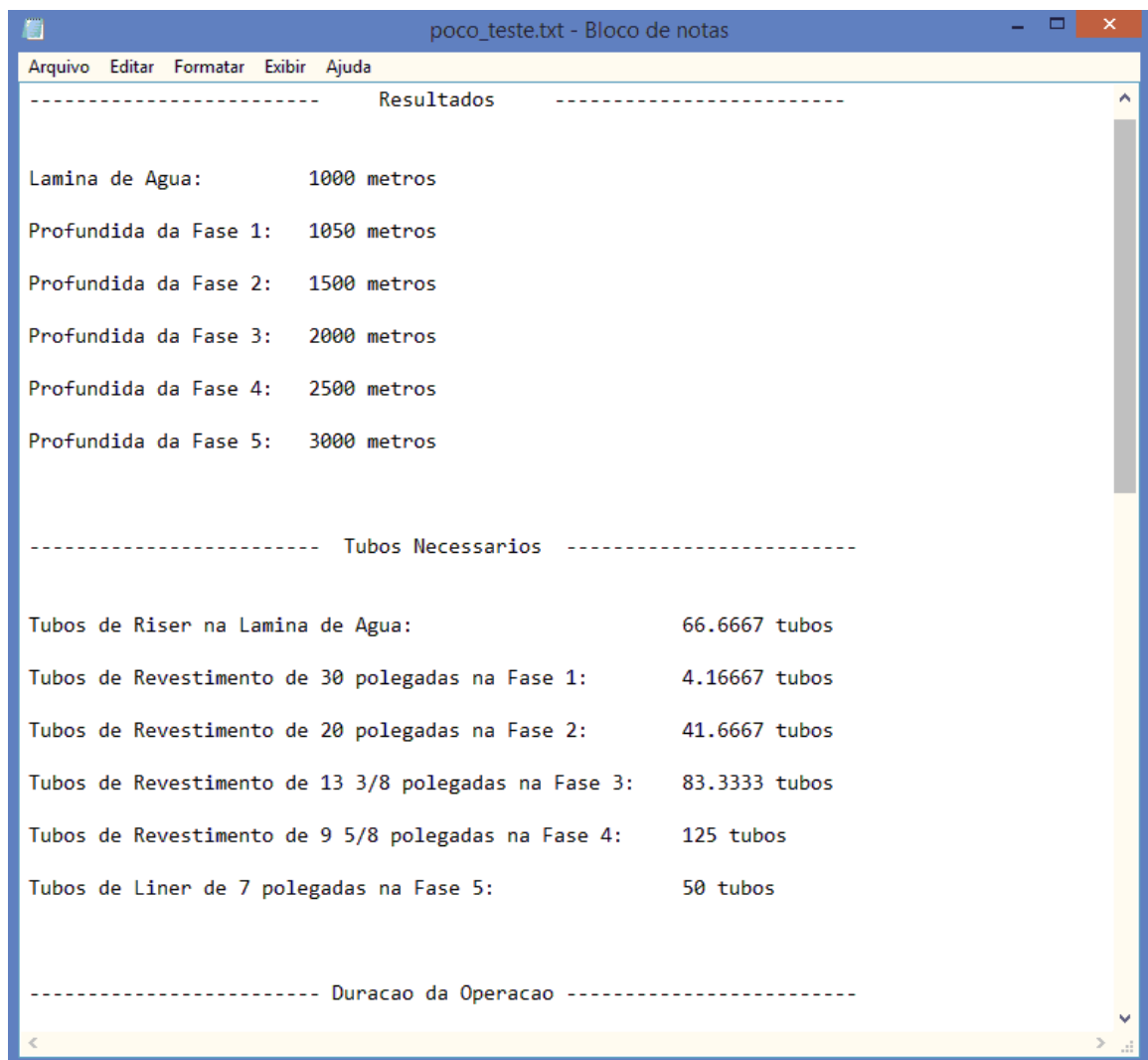


Figura 8.4: Arquivo gerado pelo programa.

Referências Bibliográficas

- [RA07] Luiz Alberto Santos ROCHA and Cecília Toledo de Azevedo, *Projetos de poços de petróleo—geopressões e assentamento de colunas de revestimentos*, Editora Interciência: PETROBRAS, Rio de Janeiro (2007). 9

Índice Remissivo

Ambiente de desenvolvimento, 21

Análise orientada a objeto, 16

AOO, 16

Associações, 22

Bibliotecas, 20

Casos de uso, 6

colaboração, 17

comunicação, 17

Diagrama de colaboração, 17

Diagrama de sequência, 16

Efeitos do projeto nas associações, 22

Efeitos do projeto nas heranças, 22

Efeitos do projeto nos métodos, 21

Elaboração, 9

Eventos, 16

Heranças, 22

heranças, 22

Implementação, 23

máquina de estado, 18

métodos, 21

Mensagens, 16

modelo, 21

otimizações, 22

Plataformas, 20

POO, 21

Projeto do sistema, 20

Projeto orientado a objeto, 21