ESTE EXEMPLAR CORRESPONDE À REDAÇÃO DO TRABALHO FINAL DE MESTRADO PROFISSIONAL DEFENDIDO POR NÚBIA CÉLIA BERGÊ CUTRIM E APROVADO PELA COMISSÃO JULGADORA EM 20/02/2004.

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Experimentos virtuais didáticos para ensino de Mecânica dos Fluidos.

Autor: Núbia Célia Bergê Cutrim
Orientador: Kamal Abdel Radi Ismail

Campinas - São Paulo

2004

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DEPARTAMENTO DE ENGENHARIA TÉRMICA E FLUIDOS

Experimentos virtuais didáticos para ensino de Mecânica dos Fluidos.

Autor: Núbia Célia Bergê Cutrim
Orientador: Kamal Abdel Radi Ismail

Curso: Engenharia Mecânica – Mestrado Profissional
Área de Concentração: Instrumentação e Controle Industrial

Trabalho Final de Mestrado Profissional apresentado à Comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Mestre Profissional em Engenharia Mecânica / Instrumentação e Controle Industrial

Campinas – São Paulo
2004

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C978e
9

Cutrim, Núbia Célia Bergê
Experimentos virtuais didáticos para ensino de mecânica dos fluidos / Núbia Célia Bergê Cutrim.-- Campinas, SP: [s.n.], 2004.

Orientador: Kama Abdel Radi Ismail.
Dissertação (mestrado profissional) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Didático. 2. Mecânica dos fluidos. 3. Ensino. I. Ismail, Kamal Abdel Radi. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DEPARTAMENTO DE ENGENHARIA TÉRMICA E FLUIDOS

Trabalho Final de Mestrado Profissional

Experimentos virtuais didáticos para ensino de Mecânica dos Fluidos.

Autor: Núbia Célia Bergê Cutrim
Orientador: Kamal Abdel Radi Ismail

Profº Dr. Kamal Abdel Radi Ismail (Orientador)
Universidade Estadual de Campinas

Profº Dr. Waldemir Silva de Lima
Universidade Estadual do Maranhão

Profº Dr. Valdemar Silva Leal
Universidade Estadual do Maranhão

Campinas (SP), fevereiro de 2004.

Dedicatória:

Dedico este trabalho aos meus filhos: Dannúbia, Bianca e João Igor.

Agradecimentos

A Deus, pai todo-poderoso, pela sua infinita generosidade ao me conceder essa oportunidade, dando força e coragem em todos os momentos da minha vida.

À Universidade Estadual do Maranhão – UEMA pela oportunidade de realizar este mestrado.

Ao meu orientador, que me mostrou os caminhos a serem seguidos.

Ao meu co-orientador, pela segura orientação.

A todos os professores e demais colegas, que direta e indiretamente contribuíram na conclusão deste trabalho.

"De agora em diante, meu grande instrumento de trabalho não mais será o alfabeto escrito. Utilizarei a fotografia, o rádio, a televisão, o computador etc. Com esses instrumentos, ensinarei meus alunos a se comunicarem, para que a comunicação não atinja apenas o pequeno número dos que sabem ler e escrever. Usando a imagem, eles poderão atingir a todos. Alfabetizados e analfabetos poderão receber suas mensagens, sem distinção".

Luís Carlos Lopes Manhães

Resumo

CUTRIM, Núbia Célia Bergê, *Experimentos virtuais didáticos para ensino de Mecânica dos Fluidos*, Campinas:Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 93 p. Trabalho Final de Mestrado Profissional.

Neste trabalho procurou-se desenvolver experimentos virtuais didáticos para ensino de Mecânicas dos Fluidos com ênfase na utilização da equação de Bernoulli, a experiência de Reynolds e funcionamento de um rotâmetro, permitindo a realização de experimentos computacionais, com base em experimentos teóricos já existentes. Utilizou-se softwares adequados e de comprovada credibilidade para o desenvolvimento dos experimentos virtuais. Os resultados obtidos foram analisados, testados e comparados com os resultados numericamente testados. Com a conclusão deste trabalho pretende-se uma melhor compreensão dos fundamentos relacionados às temáticas da disciplina mecânica dos fluidos e suas aplicações no ensino mediado pelas novas tecnologias.

Palavra Chave

Experimentos Virtuais, didáticos.

Abstract

CUTRIM, Núbia Célia Bergê, *Didatics virtual experiments to the Fluids Mecanics*, São Luís: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2003. 93 p. Trabalho Final de Mestrado Profissional.

In this work it looked for develop didatics virtual experiments to the Fluids Mecanics with emphasis on the utilization Bernoulli's equation, Reynolds' experience and the function of the rotameter, permiting the accomplishment of computers experiment, allude to teorics experiments that already exist. It used adequate softwares and it has confirmed credit for the delopment of the virtual experiments. The results abtained were analized and comproved with the results numerical tested. With the conclusion of this work it wish the best compreension of the basis related by the tematic of the discipline fluids mecanics and their inclusion on the teaching mediated by the news tecnologies.

Key Words

Didatics virtual experiments

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

1. INTRODUÇÃO

1.1 Visão do Problema

Em mecânica dos fluidos assim como na maioria das disciplinas de engenharia, existem vários casos em que o estudante fica em dúvida do que lhe é apresentado, pois na maioria das vezes é difícil a visualização do problema.

Existem vários tipos de experimentos já realizados que facilitam o entendimento do aluno, porém em muitas universidades estes experimentos só são vistos em livros e explicados exaustivamente pelos professores, mas não existem uma bancada de teste para a comprovação dos fatos.

Hoje, com o avanço das tecnologias, a maioria das Universidades possui computadores. E seu uso como equipamento de ensino tem-se revelado muito adequado, uma vez que auxiliam os estudantes a assimilar os conceitos associados a diversas disciplinas.

Muitas pesquisas buscam a integração do uso de computadores no meio educacional. Vários educadores defendem a utilização de novas tecnologias, entre elas o computador, e sua importância para o desenvolvimento e a aplicação de metodologias voltadas para a melhoria de qualidade do ensino de engenharia.

Com este trabalho queremos montar bancadas de testes virtuais em mecânica dos fluidos com ênfase na equação de Bernoulli, o número de Reynolds e o funcionamento de um rotâmetro, utilizando experiências já existentes que serão realizadas com softwares de fácil utilização, com o objetivo do aluno ficar incentivado a realizar suas próprias experiências.

1.2 Objetivos

Desenvolvimento de aplicativos para uma Bancada de teste Virtual em Mecânicas dos Fluidos, permitindo a realização de experimentos computacionais, utilizando experimentos já existentes.

Compreender melhor os fundamentos relacionados às temáticas da disciplina mecânica dos fluidos e suas aplicações no ensino mediado pelas novas tecnologias.

Desenvolvimento e aplicação de metodologias voltadas para a melhoria da qualidade do ensino de engenharia.

2. REVISÃO DA LITERATURA

Os recentes avanços no uso dos recursos de Tecnologia da Informação na sociedade têm promovido intensas alterações em quase todos os seus aspectos, especialmente aqueles que interagem fortemente com a comunicação, por exemplo, nas artes, nas relações comerciais e

pessoais e certamente na educação.

Têm-se desenvolvido diversos aplicativos na área de mecânica dos fluidos, transmissão de calor. Entre outros, os aplicativos aqui propostos pretende trabalhar a equação de Bernoulli, equação de Reynolds e funcionamento de um rotâmetro, simulando um laboratório.

2.1 Conceitos Fundamentais

A mecânica dos Fluidos se preocupa com o comportamento dos fluidos, ao quais são substância que se deformam continuamente sob a aplicação de uma tensão de cisalhamento, por menor que ela seja.

Os fluidos compreendem as fases líquida e gasosa ou vapor. A distinção de um fluido para um estado sólido é notada quando comparamos os seus comportamentos. Assim um sólido também se deforma quando uma tensão de cisalhamento lhe é aplicada, porém não continuamente.

De acordo com Streeter, o termo **fluido** é usado para descrever um objeto ou substância que deve estar em movimento para resistir forças aplicadas externamente. Um fluido sempre escorre quando forças deformantes lhe são aplicadas. Embora a tendência seja imaginar os fluidos principalmente como líquidos, os fluidos também descrevem o comportamento dos gases.

2.2 Conceitos Ligados ao Escoamento de Fluidos e Equações Fundamentais

Os escoamentos podem ser classificados de diversas formas como: turbulento ou laminar, real ou ideal, reversível ou irreversível, permanente ou variado, uniforme ou não uniforme e rotacional ou irrotacional (Streeter, 1982).

a) Escoamento Turbulento:

As partículas de fluido movem-se em trajetórias irregulares, causando uma transferência de quantidade de movimento de uma porção do fluido para outra. No escoamento turbulento as perdas variam com uma potência de 1,7 a 2 da velocidade, no escoamento laminar elas variam.

b) Escoamento Laminar:

As partículas movem-se ao longo de trajetórias suaves, em lâminas ou camadas, com cada uma destas deslizando suavemente sobre outra adjacente. No escoamento laminar a ação da viscosidade amortece a tendência de aparecimento de turbulência.

c) Escoamento Permanente:

Quando as condições em qualquer ponto do fluido não variam com o tempo, $\partial v / \partial t = 0$.

d) Escoamento Variado:

O escoamento é variado quando as condições variam em qualquer ponto com o tempo, $\partial v / \partial t \neq 0$

e) Escoamento Uniforme:

É quando o vetor velocidade é igual em todos os pontos, em módulo, sentido e direção, para qualquer instante, $\partial v / \partial s = 0$, onde o tempo é mantido constante e ∂s é um deslocamento em qualquer direção.

f) Escoamento Não Uniforme:

É o escoamento no qual o vetor da velocidade varia de local para local, num instante qualquer, $\partial v / \partial s \neq 0$

g) Escoamento Rotacional:

Quando as partículas de um fluido numa certa região possuírem rotação em relação a qualquer eixo.

h) Escoamento Irrotacional:

Quando as partículas de um fluido numa região não possuírem rotação.

2.2.1 Equação da Continuidade

A *equação da continuidade* estabelece que: o volume total de um fluido incompressível, isto é, fluido que mantém constante a densidade apesar das variações na pressão e na temperatura, entrando no tubo será igual àquele que está saindo do tubo.

O fluxo medido num ponto ao longo do tubo será igual ao fluxo num outro ponto ao longo do tubo, apesar da área da seção transversal do tubo em cada ponto ser diferente.

Isto pode ser expresso numa equação da forma: $Q = A_1 v_1 = A_2 v_2 = \text{constante}$

2.3 Equação de Bernoulli

A equação de Bernoulli é um corolário da lei de Newton. Nos líquidos sem atrito, uma força que age sobre uma superfície é sempre normal à dita superfície. Nos líquidos com atrito interno aparecem, durante o escoamento, tensões de cisalhamento, de modo que a força que age sobre uma superfície não lhe é mais perpendicular. A não-presença de tensões de cisalhamento pode ser utilizada como definição dos líquidos sem atrito.

Para deduzirmos a equação de Bernoulli, procederemos da maneira seguinte:

Dado um tubo de fluxo delgado, limitaremos entre duas secções transversais S_1 e S_2 um certo volume dV do líquido (Figura 2.1). A posição desse volume será dada pelo espaço s sobre a

trajetória, ao longo da linha de fluxo Q_oQ (pontilhada no desenho). A velocidade v e a pressão em Q são funções da posição s: $v = v_{(s)}$, $p = p_{(s)}$.

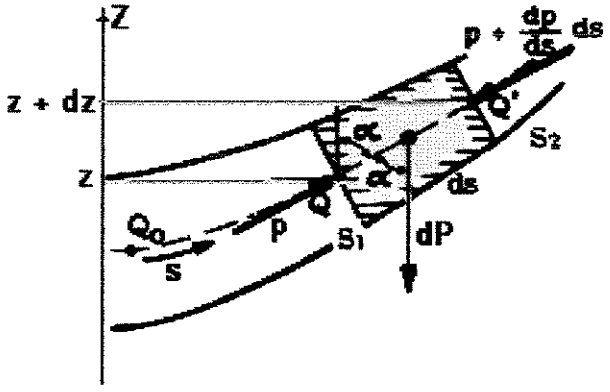


Figura 2.1 - Fluxo (Ferraz, 2002)

Suporemos, também que o líquido está num campo de gravidade uniforme **g**. Então atuará sobre o líquido contido no volume dV, o peso dP. Introduziremos, por isso, ainda um eixo vertical de coordenadas z. Como já está dada a linha de fluxo, será também z função de s: $z = z_{(s)}$.

Sobre o volume de líquido dV, entre as duas secções S_1 e S_2 e atuam, então, as seguintes forças:

1 — Aquela proveniente da pressão $p_{(s)}$ no ponto Q; que indicaremos por $p_{(s)}$;
2 — Aquela proveniente da pressão $p_{(s+ds)}$ no ponto Q'; que indicaremos como:

$$P(s+ds) = P(s) + \frac{\partial p}{\partial s} ds$$

3 — O peso dP. Interessa-nos somente o componente de dP na direção da tangente. Ela vale:

$$dP \cdot \cos \alpha = dP \frac{dz}{ds}$$

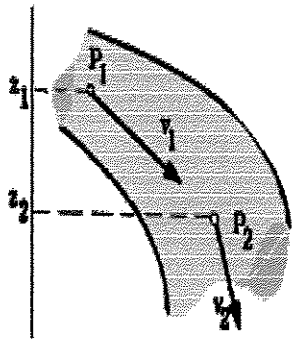
Valerá, então, para o componente da força total sobre dV, tomada na direção do movimento (tangente, portanto, à linha de fluxo Q_o Q):

$$dm \frac{dv}{dt} = dFs$$

que é a equação de Newton, aplicada ao volume líquido dV.

Designemos, ainda, por ρ a densidade absoluta do líquido e recordemos que:

dV = S.ds, de modo que a equação de Newton adquire, ao longo de uma linha de fluxo, o seguinte aspecto(figura 2.2):



$$\rho.S.ds\frac{dv}{dt} = S\left[p - \left(p + \frac{\partial p}{\partial s}ds\right)\right] - \rho.S.ds.g.\cos\alpha$$

$$\rho.S.ds\frac{dv}{dt} = -S\frac{\partial p}{\partial s}ds - \rho.S.g.\frac{dz}{ds}ds$$

$$\rho.ds\frac{dv}{dt} + dp + \rho.g.dz = 0 \quad \text{ou,} \quad \frac{\rho}{2}.d(v^2) + dp + \rho.g.dz = 0$$

$$d\left(p + \frac{\rho}{2}v^2 + \rho.g.z\right) = 0 \text{ a qual, por integração , fornece :}$$

$$\boxed{p + \frac{\rho}{2}v^2 + \rho gz = \text{const.}} \text{ (ao longo de uma linha de fluxo)}$$

Figura 2.2. Fluxo (Ferraz, 2002)

Essa equação é conhecida como **equação de Bernoulli**.

A grandeza p é denominada pressão estática, a grandeza $(\rho/2).v^2$ é a pressão dinâmica (ou cinética) e a grandeza $\rho.g.z$ é a pressão por gravidade, (ou de posição).

Para escoamento horizontal, z = constante. Então se reduz a equação de Bernoulli a:

$p + (\rho/2).v^2$ = constante ao longo de uma linha de fluxo.

2.3.1 Aplicações da equação de Bernoulli

1 — **Tubo de Venturi**

De acordo com a equação da continuidade, ter-se-á $\mathbf{v_1/v_2} = S_2/S_1$. De acordo com Bernoulli (figura 2.3), verifica-se também (no caso de z = constante), logo:

$$p_1 + (\rho/2).v_1^2 = p_2 + (\rho/2).v_2^2$$

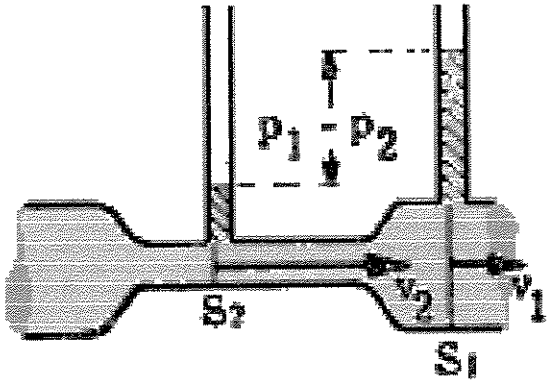


Figura 2.3 Tubo de Venturi (Ferraz, 2002)

Na região estreita, a velocidade **v** é maior, sendo, portanto, menor a pressão p.

Nisto se baseia a construção da bomba a jato d'água (trompa), como a ilustrada na figura 2.4:

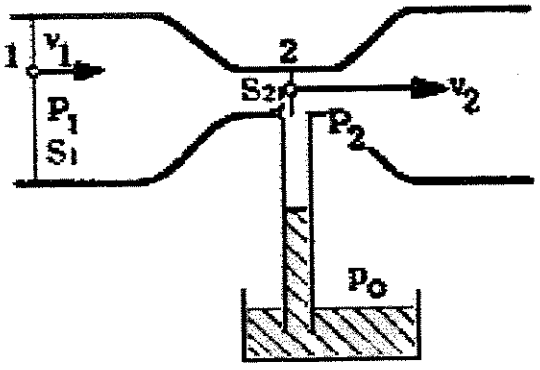


Figura 2.4 – Tubo de Venturi (Ferraz, 2002)

Seja, em particular, p_2 menor que a pressão barométrica p_o, de modo que o líquido é impulsionado para cima, mesmo se p_1 for maior que p_o. Pode-se, também, por a secção transversal

S_2 em comunicação com um recipiente, no qual se deseja obter o vácuo. O conteúdo fluido desse recipiente escoa, então, para o trecho onde reina a pequena pressão p_2, até que a pressão, no interior do recipiente considerado, se iguale a p_2. De acordo com Bernoulli, será:

$$p_2 = p_1 - \frac{\rho}{2}(v_2^2 - v_1^2) \quad e \quad v_2 = v_1 \frac{S_1}{S_2}$$

$$p_2 = p_1 - \frac{\rho}{2}\left(\frac{S_1^2}{S_2^2} - 1\right).v_1^2$$

$$\text{Sendo } \frac{S_1}{S_2} > 1 \text{, teremos } p_2 < p_1$$

Com o aumento de v_1, poder-se-ia conseguir que p_2 e, portanto, também a pressão no recipiente, se anulasse ou mesmo se tornasse negativa (sucção, $p_2 < p_o$). Por causa da vaporização da água, nunca se chega, entretanto, a uma pressão inferior à que corresponde à tensão de vapor d'água à temperatura ambiente. A 20°C essa tensão vale 17,5 mmHg.

Para obter-se um vácuo ainda melhor usa-se a bomba a jato de vapor de mercúrio ou a bomba de vapor de óleo, que operam segundo o mesmo princípio da bomba a jato d'água. Em lugar do jato d'água, usa-se neste caso, um jato de vapor de mercúrio ou de óleo. Como o Hg e o óleo possuem tensões de vapor menores que a da água (10^{-3} a 10^{-6} mmHg), obtém-se, com essas bombas, um vácuo efetivamente melhor.

2 — **Tubo de Pitot**

O tubo de Pitot (figura 2.5), serve para as medidas da velocidade.

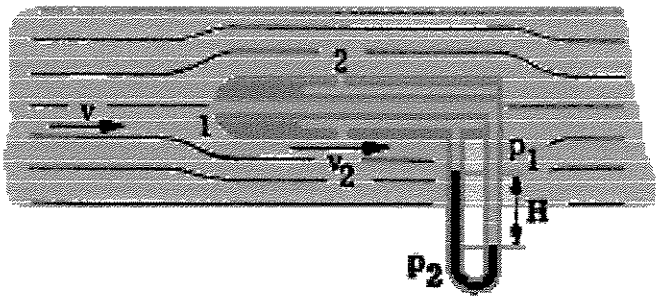


Figura 2.5 - Tubo de Pitot (Ferraz, 2002)

No ponto 1 temos a velocidade v = 0, ponto em que é retido o fluido. A este ponto corresponde uma pressão $p = p_1$. No ponto 2 a velocidade v_2, que é aproximadamente igual à velocidade do líquido (ou ar) no espaço exterior. A esta velocidade corresponde a pressão $p = p_2$. Segundo Bernoulli, a equação pode ser expressa como:

$$p_1 = p_2 + \frac{\rho_{gás}}{2}.v_2^2 \quad \text{ou} \quad \boxed{v_2 = \sqrt{\frac{2(p_1 - p_2)}{\rho_{gás}}}}$$

Neste caso a medida da velocidade pode, portanto, ser reduzida à medida de uma pressão. A diferença de pressão (p_1 - p_2) é medida no dispositivo manométrico, por meio da diferença de altura H das colunas líquidas. Designando-se por $\rho_{gás}$ a densidade do gás em movimento e por $\rho_{líq.}$ a densidade do líquido manométrico, a equação pode ser demonstrada como:

$$v_2 = \sqrt{2.g.H.\frac{\rho_{liq.}}{\rho_{gás}}}$$

3 — **Escoamento sob a influência de uma sobre pressão**

Para o dispositivo representado na figura 2.6, segundo Bernoulli, a equação será escrita na forma: $p_1 + (\rho/2).v_1^2 = p_2 + (\rho/2).v_2^2$

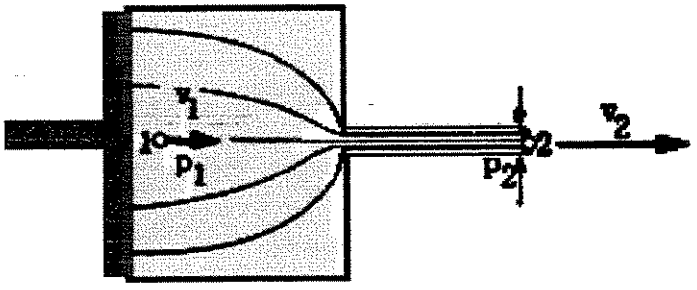


Figura 2.6 – Escoamento de um fluido com influência sobre pressão; Ferraz, 2002

Se a abertura do orifício for pequena, em relação à seção do conduto, será então, como se deduz da equação da continuidade, v_1 pequena, de maneira que poderemos em primeira aproximação considerá-la nula. Decorre, então, para a velocidade de saída será:

$$v_2 = \sqrt{\frac{2(p_1 - p_2)}{\rho_{liquido}}}$$

É característico neste resultado ser a velocidade v_2 diretamente proporcional à raiz quadrada da diferença de pressão e inversamente à raiz quadrada da densidade absoluta do líquido.

4 — **Escoamento de um fluido sob a influência da gravidade** (figura 2.7).

Para o ponto 1 vale: $z = z_1$; $v = v_1$; $p = p_1 = p_o$ = pressão barométrica.
Para o ponto 2 vale: $z = z_2 = z_1 - h$; $v = v_2$; $p = p_2 = p_o$ = pressão barométrica.

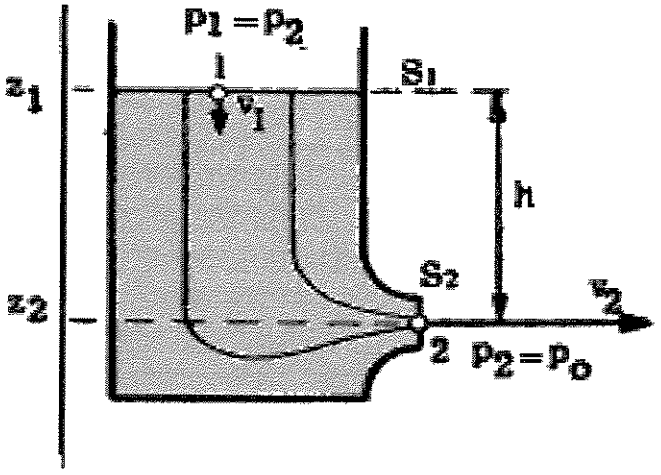


Figura 2.7 – Escoamento de um fluido sob influência da gravidade (Ferraz, 2002)

Segundo a equação da continuidade será: $v_1 / v_2 = S_2 / S_1$. Se, porém, S_1 é muito maior que S_2, podemos considerar nula v_1, sendo então p_1 igual a p_2. A equação de Bernoulli reduz-se agora a: $(\rho/2).v_2^2 = \rho.g.(z_1 - z_2)$

Decorre portanto, para a velocidade v_2:

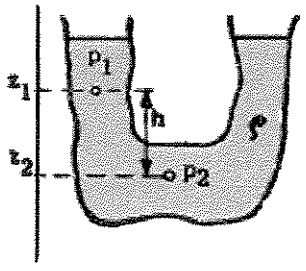
$$v_2 = \sqrt{2.g.(z_1 - z_2)} = \sqrt{2.g.h}$$

O líquido tem no escoamento a mesma velocidade que atingiria em queda livre da altura h (Princípio da Energia).

5 — **Empuxo** (figura 2.8).

Para o caso de velocidades muito pequenas, a equação de Bernoulli transforma-se na equação fundamental da hidrostática. Ela se reduz à seguinte expressão:

$p_2 - p_1 = \rho.g.(z_1 - z_2) = \rho.g.h$



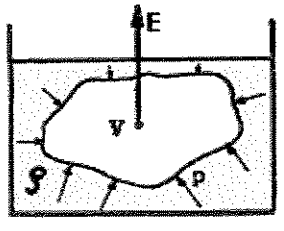


Figura 2.8 - Empuxo (Ferraz, 2002)

O empuxo **E** que sofre um corpo mergulhado num fluido de densidade ρ obtém-se como a resultante de todas as forças elementares. Acha-se, pois, para o empuxo a expressão: $\mathbf{E} = \rho.g.V_{liq.deslocado,}$

onde $V_{líq.desl.}$ representa o volume do fluido deslocado pelo corpo. O ponto de aplicação do empuxo é o centro de gravidade do volume anteriormente ocupado pelo fluido deslocado, e não o centro de gravidade do corpo mergulhado. Somente para corpos homogêneos coincidem esses dois pontos. O empuxo **E** pode ser utilizado para a avaliação cômoda da densidade ρ de gases, líquidos e sólidos.

2.4 Número de Reynolds

Em geral um fluido escoa laminarmente quando sua velocidade não é muito grande e o tubo é liso, sem saliência. Entretanto, se a velocidade de fluxo atingir valores acima de certo limite, isto é, depende de diversos fatores, como a natureza do fluido e sua temperatura, o fluido pode escoar de maneira irregular com formação de redemoinhos, resultado da mistura entre camadas adjacentes de fluido. A esse tipo de escoamento dá-se o nome de turbulento. Reynolds mostrou que, de modo geral, um escoamento por um tubo regular e retilíneo de diâmetro D deixa

de ser laminar quando o número de Reynolds, definido por:

$$\Re = \frac{\nu D \rho}{\mu}$$

laminar se $\Re_R < 2\,000$

turbulento se $\Re > 3\,000$

instável, mudando de um regime para outro se $2\,000 < \Re < 3\,000$

O número de Reynolds $VD\rho / \mu$ é a relação entre forças de inércia e forças viscosas. Um número de Reynolds crítico deferência os regimes de escoamento laminar e turbulento em condutos na camada limite ou ao redor de corpos submersos. Seu valor específico depende de cada situação.

A natureza de um escoamento, se ele é laminar ou turbulento (figura 2,9), e sua posição relativa numa escala de turbulência e indicada pelo número de Reynolds.

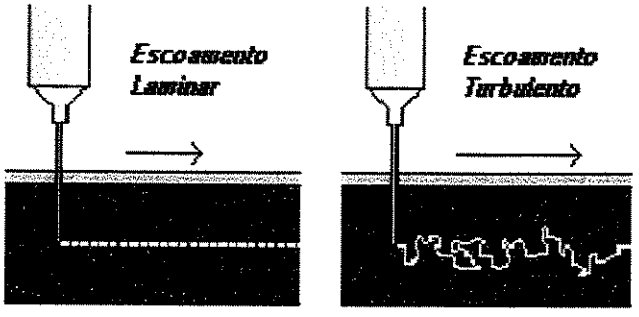


Figura 2.9 - Escoamentos laminar e turbulento

Reynolds estudou várias equações: uma equação do movimento com a hipótese de que o fluido era sem atrito, portanto viscosidade nula e outras equações mais gerais que levavam em conta a viscosidade com a inclusão as tensões de cisalhamento, tentando determinar quando dois escoamentos seriam semelhantes.

Reynolds considerou dois escoamentos geometricamente semelhantes e deduziu que os mesmos seriam dinamicamente semelhantes se as equações diferenciais gerais que os descreviam fossem iguais. Mudando as unidades e massa, comprimento e tempo em um conjunto de equações e determinando as condições que deveriam ser satisfeitas para torna-las idênticas às equações originais, Reynolds descobriu que o grupo adimensional $ul\rho / \mu$ deveria ser o mesmo para os dois casos. Neste grupo, u e uma velocidade característica, l um comprimento característico, ρ a massa específica e μ a viscosidade. Este grupo ou parâmetro é chamado de Número de Reynolds, representado pela letra **R.**

De acordo com Streeter, o número de Reynolds é encontrado pela fórmula:

$$\mathbf{R} = ul\rho / \mu$$

Para determinar o significado o grupo adimensional, Reynolds fez experiência com escoamento de água em tubos de vidro.

2.4.1 A Experiência de Reynolds (Figura 2.10)

Um tubo de vidro foi montado horizontalmente, ligado a um tanque e com ma válvula na outra extremidade. Uma entrada suave em forma de bocal foi colocada na extremidade de montante bem como um dispositivo para injetar um filete de tinta em qualquer ponto da seção de entrada do bocal. Reynolds usou a velocidade média V como velocidade característica e o diâmetro D do tubo como comprimento característico, de modo que $\mathbf{R} = VD\rho / \mu$

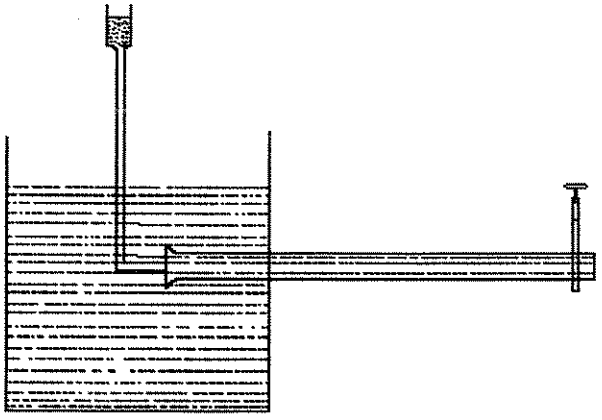


Figura 2.10 - Experiência de Reynolds, (Streeter, 1982)

2.5 Rotâmetro

2.5.1 Generalidades

A medição de fluxo é extremamente importante para qualquer processo, é feita com o fluido em movimento. Existem 3 tipos de medidores de fluxos: diretos, indiretos e especiais. O

rotâmetro é um medidor indireto. Este tipo de medidor utiliza fenômenos relacionados com a quantidade de fluido que passa. O rotâmetro é um tipo particular de medidores de fluxo baseado no principio da área variável.

Os medidores de fluxo são utilizados quer para gases quer para líquidos, e indicam a taxa de fluxo do fluido.

O rotâmetro é o mais conhecido medidor de fluxo de área variável. Foi desenvolvido há já vários anos, mas só nos últimos 30 é que se verificou um maior progresso, permitindo que hoje ele possa ter utilização nas mais diversas situações e condições.

Os rotâmetros são bastante utilizados na industria química, farmacêutica, petroquímica, alimentar, mecânica. São também bastante comuns em laboratórios e no tratamento de águas

2.5.2 Funcionamento do rotâmetro

O rotâmetro é constituído por um tubo cônico, com o diâmetro menor do lado de baixo, dentro do qual existe um flutuador ou bóia. É através da parte menor do tubo que o fluido entra. A bóia pode mover-se livremente na vertical, subindo ou descendo no tubo, conforme aumenta ou diminui o fluxo. O tubo possui uma escala de medida onde podemos ler diretamente o valor do fluxo através da borda de cima da bóia. Convém notar que a bóia terá que ter uma densidade superior à do fluido.

2.5.3 Princípio de Funcionamento do rotâmetro

O fluido - gás ou líquido - desloca-se no rotâmetro da base para o topo, resultando num movimento axial da bóia.

Ao longo do comprimento do tubo existe uma relação entre o diâmetro da bóia e o diâmetro interior do tubo. O diâmetro da bóia é fixo ao contrário do tubo interior do rotâmetro que vai aumentando da base até ao topo.

Em suma, o principio de funcionamento do rotâmetro baseia-se na força de arrastamento que o fluido exerce sobre a bóia, móvel, dentro de uma secção variável de escoamento.

3. Tecnologia e Experimento

Assim como a escrita estende a capacidade da memória, e é isto que explica sua eficácia como tecnologia intelectual, aliada ao desenvolvimento de um tipo de raciocínio lógico, estruturado, a simulação que podemos considerar uma imaginação auxiliada por computador é, ao mesmo tempo, uma ferramenta de ajuda ao raciocínio, mas introduz um modo diferente de raciocinar, de aprender, é uma nova forma de apresentação do saber, de produção intelectual que está transformando a maneira dos homens pensarem.

As simulações feitas por computadores propiciam atividades em que os alunos vejam de forma mais clara o funcionamento de determinadas situações, de outro modo, ilustram características importantes e relações funcionais dentro do sistema, não são réplicas de realidade,

mas sim construções pedagógicas designadas a fornecer material para a construção cognitiva dos alunos. Porém, a simulação permite que o aluno explore modelos mais complexos e em maior número do que sua imaginação seria capaz, retendo-os na tela para fazer comparações, e tudo isso em tempo real.

Para realizarmos tarefas no computador, teremos que utilizar conteúdos e estratégias. Por exemplo, para programar o computador usando uma linguagem de programação como a que será utilizada para realizarmos as experiências virtuais, é necessário realizarmos uma série de atividades que são de extrema importância na aquisição de novos conhecimentos: Primeiro, a interação com o computador através da programação requer a descrição de uma idéia em termos de uma linguagem formal e precisa; Segundo, o computador executa fielmente a descrição fornecida e o resultado obtido é fruto somente do que foi solicitado à máquina; Terceiro, o resultado obtido permite ao aluno refletir sobre o que foi solicitado ao computador e Finalmente, se o resultado não corresponde ao que era esperado, o aluno tem que depurar a idéia original através da aquisição de conteúdos ou de estratégias.

Em nossos dias faz-se necessário o questionamento dos paradigmas existentes, bem como estar habilitado para lidar com as mudanças na forma de produzir, armazenar e transmitir o conhecimento que dão origem a novas formas de pensar, aprender e fazer. Partindo desse pressuposto, este trabalho pretende apresentar alguns recursos computacionais voltados para a tecnologia educacional e apresentar experiências já conhecidas, mais que na maioria das vezes não compreendidas

A entrada na era da Informática não é uma opção para a Escola ou para o professor: é uma realidade tão inevitável como foi a da introdução da palavra escrita; a informática está nos bancos, nos hospitais, nos telefones, na cabine de votação, nos supermercados. Enfim, está na vida das

pessoas comuns, portanto, não pode ser ignorada pela escola.

4. Procedimento Experimental

Foram escolhidas três experiências bem conhecidas: a primeira é aplicação da equação de Bernoulli em um escoamento feito em um tubo de Venturi, a segunda é a experiência de Reynolds através de tubo de vidro em um reservatório de água e a terceira é o funcionamento de um rotâmetro.

4.1 Experimento 1: Equação de Bernoulli em um Tubo de Venturi

4.1.1. O Problema:

O tubo de venturi é um tubo horizontal, dotado de um estrangulamento, conforme indica a figura 4.1.

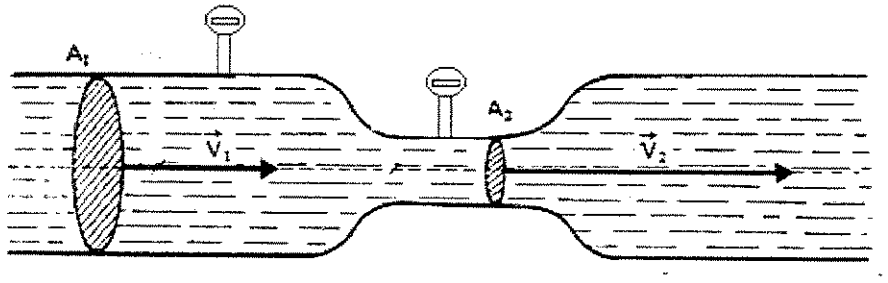


Figura 4.1 - Tubo de Venturi

Adaptando-se dois manômetros, observa-se que na parte de maior diâmetro, a pressão é maior que na parte de menor diâmetro. O contrário acontece com a velocidade.

De fato, pela equação da continuidade, tem-se:

$$A_1 . v_1 = A_2 . v_2$$

Como $A_1 > A_2$ temos $V_1 < V_2$

Pela equação de Bernoulli

$$p_1 + \frac{d.v_1^2}{2} = p_2 + \frac{d.v_2^2}{2}$$

conclui-se que: $p_1 > p_2$ pois $v_1 < v_2$

Em resumo, nos condutores de secção variável, nas regiões mais estreitas, a pressão é menor e a velocidade de escoamento é maior.

4.1.2. O Programa: Equação de Bernoulli em um Tubo de Venturi (Anexo I).

O objetivo do sistema é simular o comportamento de um fluído no Tubo de Venturi. O esquema montado consiste de dois manômetros nos lugar dos tubos verticais laterais. Um dos manômetros é acoplado na parte mais larga do tubo e outro na parte mais estreita.

O programa recebe as seguintes entradas:

- Densidade do fluído (em Kg/m^3)
- Pressão na parte de diâmetro maior (em $10^5 Pa$)
- Pressão na parte de diâmetro menor (em $10^5 Pa$)

- Diâmetro maior (em cm)

- Diâmetro menor (em cm)

Baseado nessas entradas o programa calcula as velocidades do fluido nas duas partes do tubo. Além disso, faz uma simulação gráfica do fluxo, de onde se pode observar as velocidades relativas entre as diferentes partes do tubo.

4.1.3 Estrutura do Programa

O programa foi desenvolvido no Borland Delphi 5, que utiliza o Object Pascal como linguagem de programação.

O paradigma da Orientação a Objetos foi utilizado e a estrutura do programa é composta de duas classes, descritas a seguir.

- **TfrmPrincipal**

Esta é uma classe de interface. Ela faz toda a comunicação entre as entradas do usuário e a classe TVenturi (descrita a seguir). Entre seus atributos ela contém, além dos controles para entrada de dados, um objeto da classe TVenturi. Possui ainda um Timer que é responsável pela chamada periódica da função que atualiza o desenho na tela.

Os principais métodos da classe TfrmPrincipal são: FormCreate, FluxoTimer e FormPaint, Atualiza:

- ✓ FormCreate: atribui valores iniciais para densidade, pressão na parte mais larga, pressão na parte mais estreita, diâmetro maior e diâmetro menor. Além disso, instancia um objeto da classe TVenturi e um objeto Bitmap onde será feito o desenho do tubo.

- ✓ FluxoTimer: é um método chamado automaticamente a cada milissegundo. Este método solicita ao objeto da classe TVenturi que atualize a posição do fluido e redesenhe a tela.

- ✓ FormPaint: sempre que a tela precisar ser redesenhada este método é chamado e então solicita ao objeto de TVenturi que refaça o desenho.

- ✓ Atualiza: lê as entradas presentes nos controles e as repassa ao objeto da classe TVenturi.

Além desses, cada controle visual possui um método que chama Atualizar sempre que seus valores forem mudados pelo usuário.

- **TVenturi**

É a classe responsável por todo o tratamento do tubo, desde o cálculo das velocidades até o desenho o mesmo. A função dela é atender as solicitações da classe TfrmPrincipal, que são basicamente três: atualizar valores, incrementar o progresso do fluido e desenhar o tubo. Seus atributos são compostos de dados sobre o problema (densidade, diâmetro maior, etc) e dados

sobre o desenho (posição do tubo, comprimento, etc).

A descrição dos métodos, bem como comentários sobre seu funcionamento, pode ser vista no código fonte (Anexo I I) do programa.

Resumindo a operação do programa (Figura 4.2):

1. O usuário interage com os controles da classe TfrmPrincipal
2. A classe TfrmPrincipal comunica as alterações das entradas à classe TVenturi
3. A classe TVenturi recalcula as velocidades e redesenha o tubo.

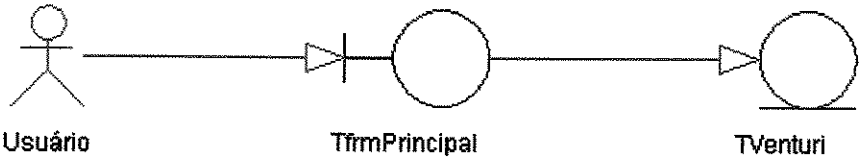
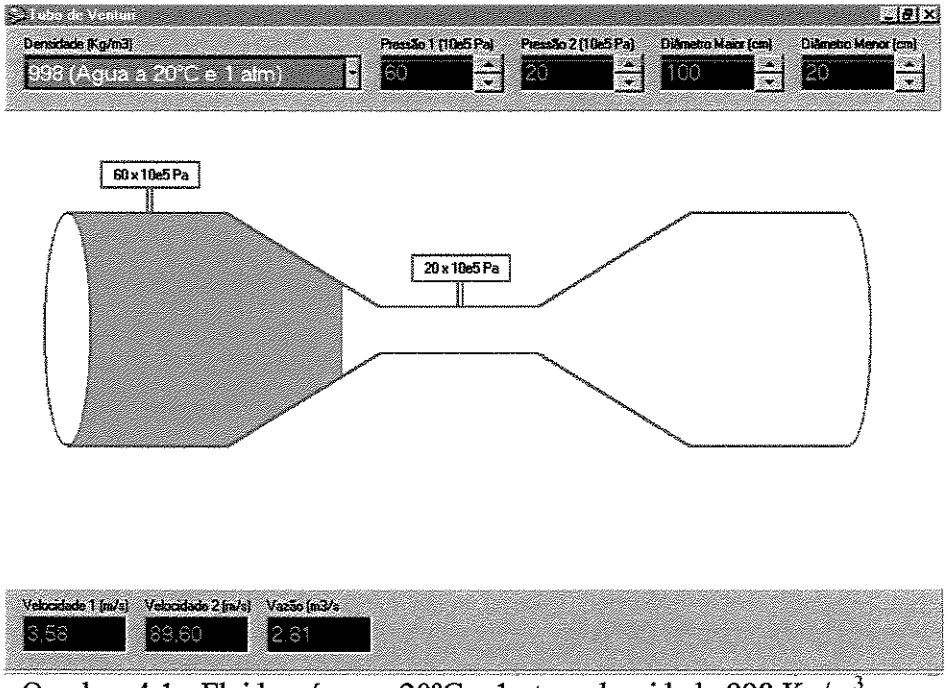


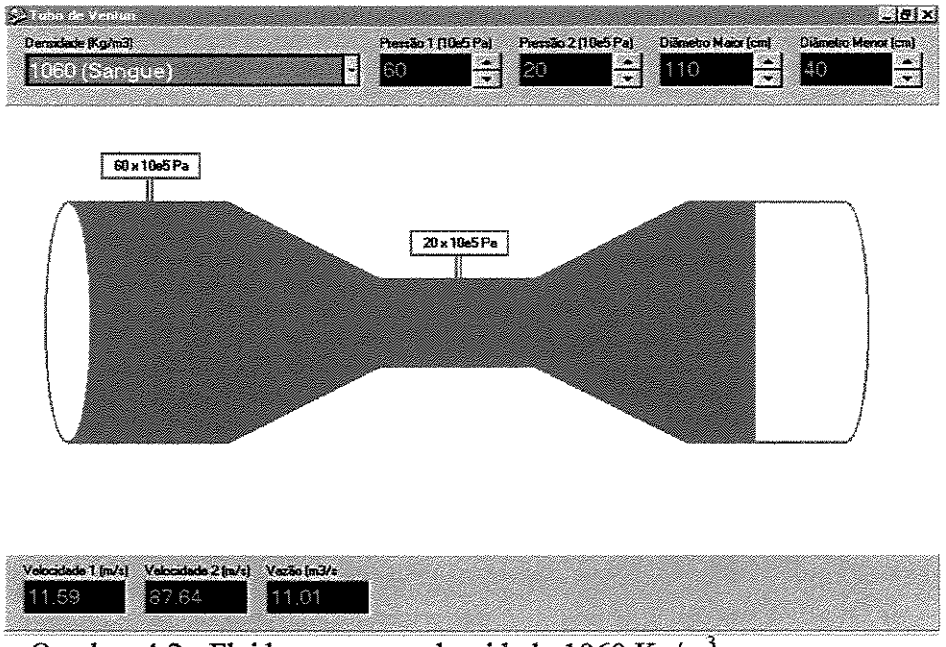
Figura 4.2 - Organograma do programa de Bernoulli

4.1.4 Quadros do Programa

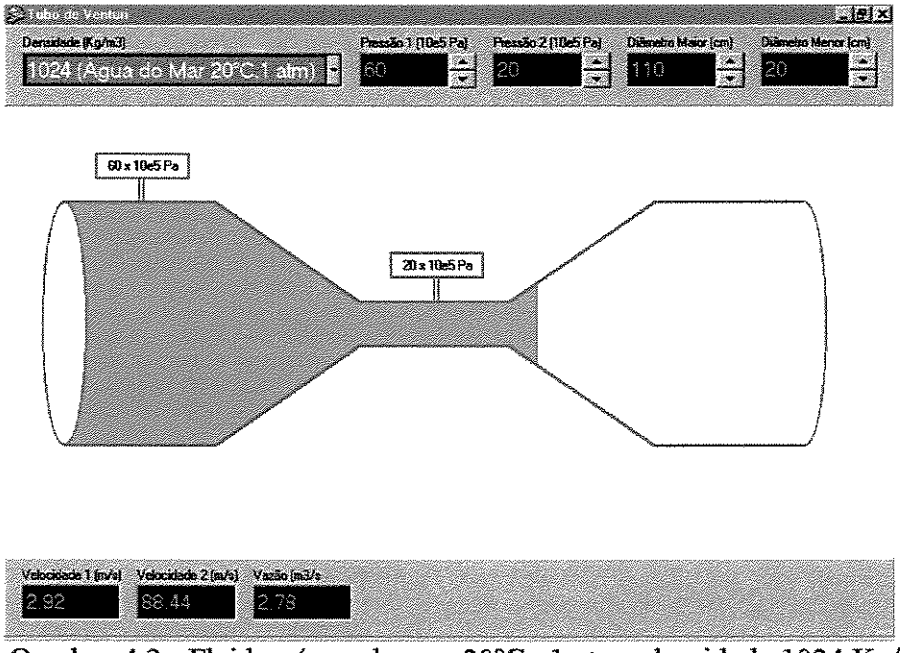
Os quadros indicam situações de vários fluidos com suas respectivas densidades, com indicações de pressões, diâmetros do tubo, velocidades e vazões encontradas em diversas situações como mostradas nos quadros: 4.1, 4.2, 4.3, 4.4 e quadro 4.5.



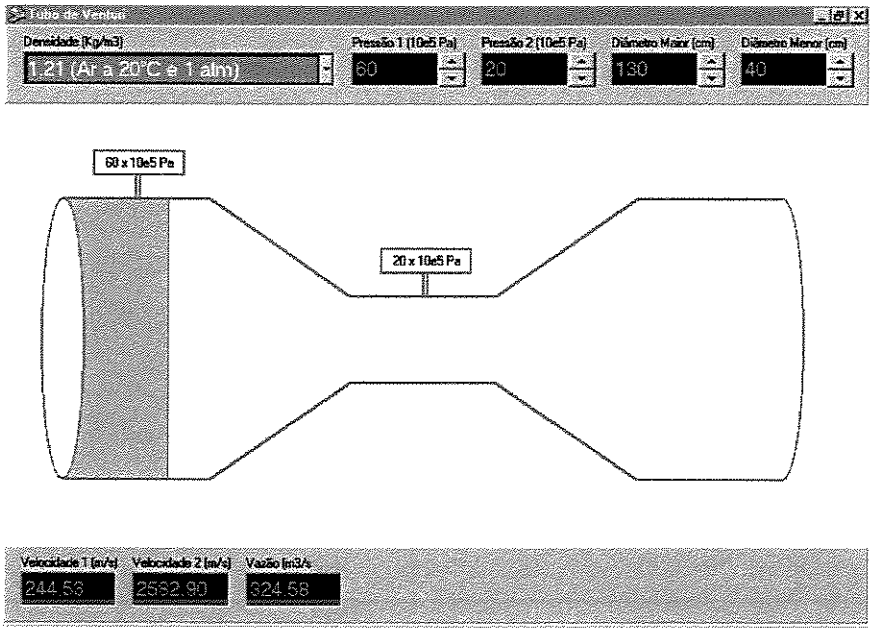
Quadro 4.1: Fluido: água a 20°C e 1 atm; densidade 998 Kg/m^3
Vazão: 2,81 m^3 / s



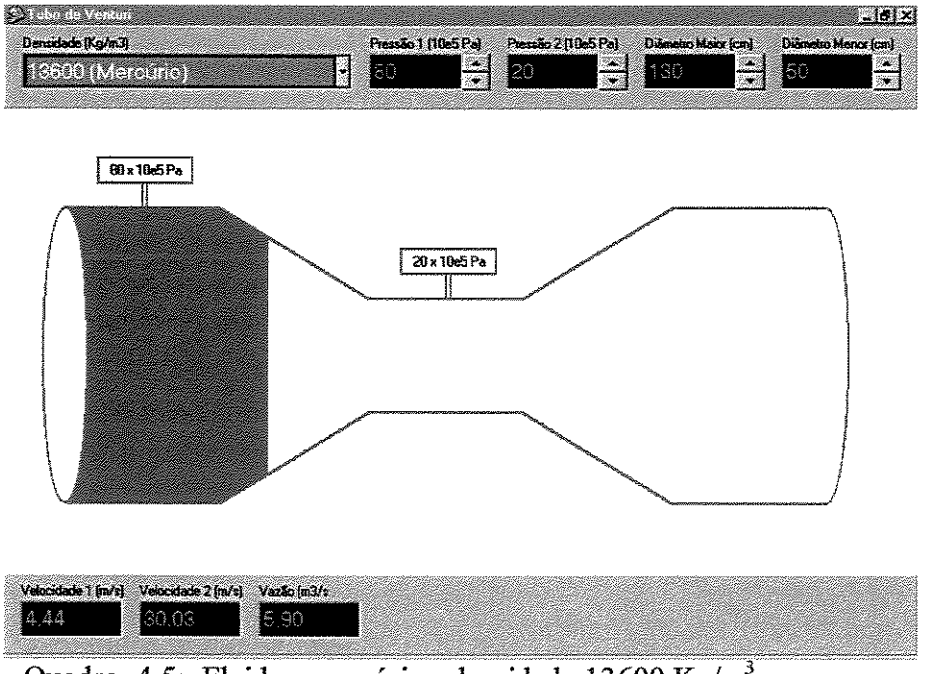
Quadro 4.2: Fluido: sangue; densidade 1060 Kg/m^3
Vazão: 11,01 m^3 / s



Quadro 4.3: Fluido: água do mar 20°Ce 1 atm; densidade 1024 Kg/m^3
Vazão: 2,78 m^3 / s



Quadro 4.4: Fluido: ar a 20°C e 1 atm; densidade 1,21 Kg/m^3
Vazão: 324,58 m^3 / s



Quadro 4.5: Fluido: mercúrio; densidade 13600 Kg/m^3
Vazão: 5,90 m^3 / s

4.2 Experimento 2: Experiência de Reynolds

4.2.1. O Problema:

O objetivo da experiência de Reynolds é visualizar e comparar os tipos de escoamento de fluídos. Para isso é montada a seguinte estrutura, representada na figura 4.3.

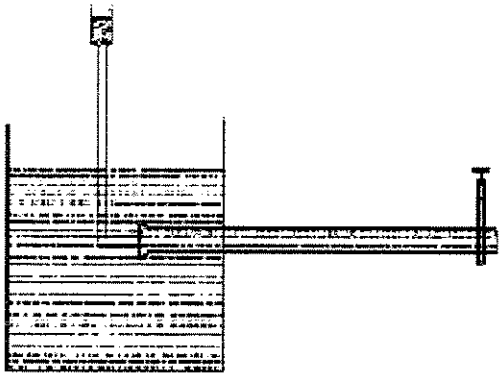


Figura 4.3 – Experiência de Reynolds (Streeter, 1982)

Pelo tubo superior lança-se um filete de tinta para que se possa visualizar os tipo de escoamento que irão ocorrer em função da vazão da água do reservatório através do tubo de vidro.

Reynolds observou que o fenômeno ensaiado dependia das seguintes variáveis:

- ρ – massa específica do fluido
- V – velocidade média de escoamento
- D – diâmetro interno da tubulação
- μ – viscosidade do fluido

Obteve ainda o chamado número de Reynolds (Re) e estabeleceu:

- Para Re <= 2000 – escoamento laminar
- Para 2000 < Re < 2400 – escoamento de transição
- Re >= 2400 – escoamento turbulento

O número de Reynolds é determinado pela fórmula:

$$R_e = \frac{\rho VD}{\mu} = \frac{VD}{\upsilon}$$

onde V é dado por:

$$V = \frac{4Q}{\pi D^2}$$ (Q é a vazão)

4.2.2 O Programa: **Experiência de Reynolds** (Anexo III).

O objetivo do sistema é simular a experiência e calcular o número de Reynolds associado a cada tipo de escoamento. O líquido em questão é a água e, portanto, a massa específica e a viscosidade do fluido serão constantes, não precisando ser informadas pelo usuário.

O programa recebe as seguintes entradas:

- Diâmetro do tubo de escoamento (em cm)
- Vazão da água pelo tubo (em ml/s)

Baseado nessas entradas o programa faz os cálculos necessários para determinar o tipo de escoamento e faz uma simulação gráfica do fluxo do permanganato de potássio através do tubo.

4.2.3 Estrutura do Programa

O programa foi desenvolvido no Borland Delphi 5, que utiliza o Object Pascal como linguagem de programação.

O paradigma da Orientação a Objetos foi utilizado e a estrutura do programa é composta de duas classes, descritas a seguir.

- **TfrmPrincipal**

Esta é uma classe de interface. Ela faz toda a comunicação entre as entradas do usuário e a classe TReynolds (descrita a seguir). Entre seus atributos ela contém, além dos controles para entrada de dados, um objeto da classe TReynolds. Possui ainda um Timer que é responsável pela chamada periódica da função que atualiza o desenho na tela.

Os principais métodos da classe TfrmPrincipal são: FormCreate, FluxoTimer e FormPaint, Atualiza:

✓ FormCreate: atribui valores iniciais para o diâmetro do tubo e a vazão do liquido. Além disso, instancia um objeto da classe TReynolds e um objeto Bitmap onde será feito o desenho da experiência.

✓ FluxoTimer: é um método chamado automaticamente de tempos em tempos. Este método solicita ao objeto da classe TReynolds que atualize a posição do fluido e redesenhe a tela.

A freqüência com que este método é chamado depende do tipo de escoamento observado.

✓ FormPaint: sempre que a tela precisar ser redesenhada este método é chamado e então solicita ao objeto de TReynolds que refaça o desenho.

✓ Atualiza: lê as entradas presentes nos controles e as repassa ao objeto da classe TReynolds.

Além desses, cada controle visual possui um método que chama Atualizar sempre que seus valores forem mudados pelo usuário.

- **TReynolds**

É a classe responsável por todo o tratamento da experiência, desde o cálculo do número de Reynolds até o desenho. A função dela é atender as solicitações da classe TfrmPrincipal, que são basicamente três: atualizar valores, incrementar o progresso do fluido e desenhar. Seus atributos são compostos de dados sobre o problema (vazão, diâmetro, tipo de escoamento, etc) e dados sobre o desenho (posição do tubo, comprimento, etc).

A descrição dos métodos, bem como comentários sobre seu funcionamento, pode ser vista no código fonte (Anexo IV) do programa.

Resumindo a operação do programa, representado na figura 4.4.

1. O usuário interage com os controles da classe TfrmPrincipal

2. A classe TfrmPrincipal comunica as alterações das entradas à classe TReynolds

3. A classe TReynolds recalcula o número de Reynolds e redesenha a tela.

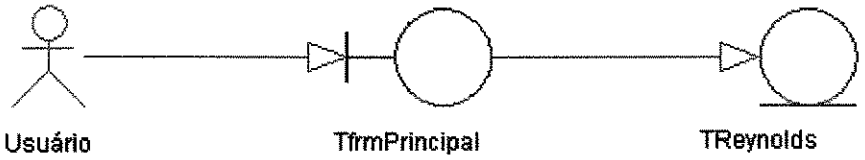
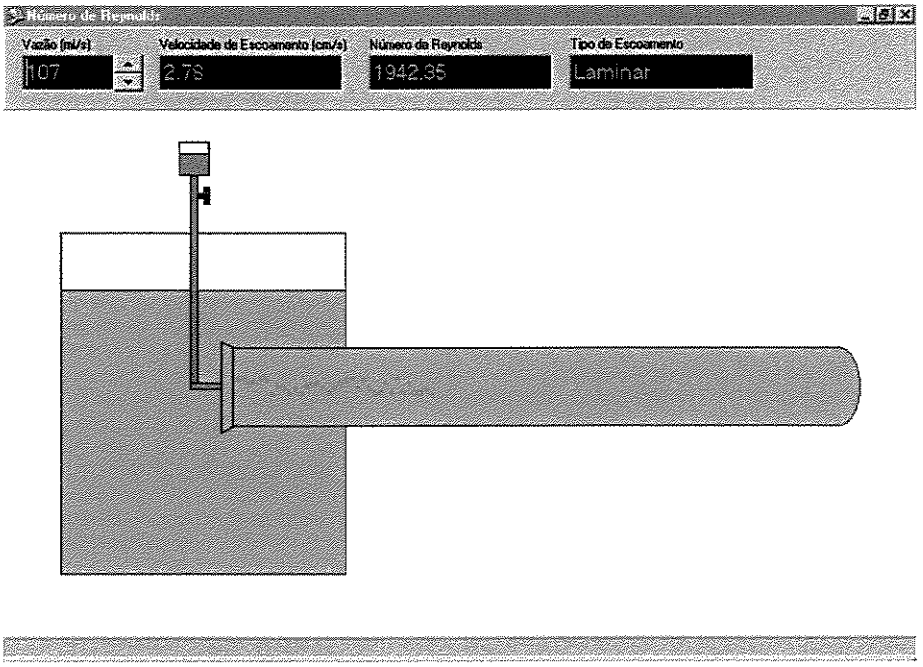


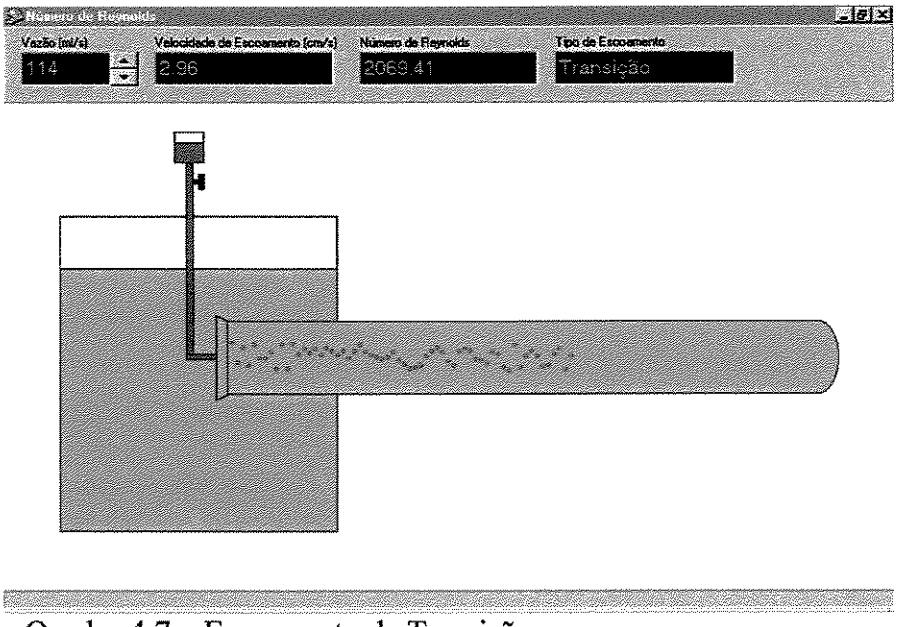
Figura 4.4 - Organograma do Programa de Reynolds

4.2.4 Quadros do Programa

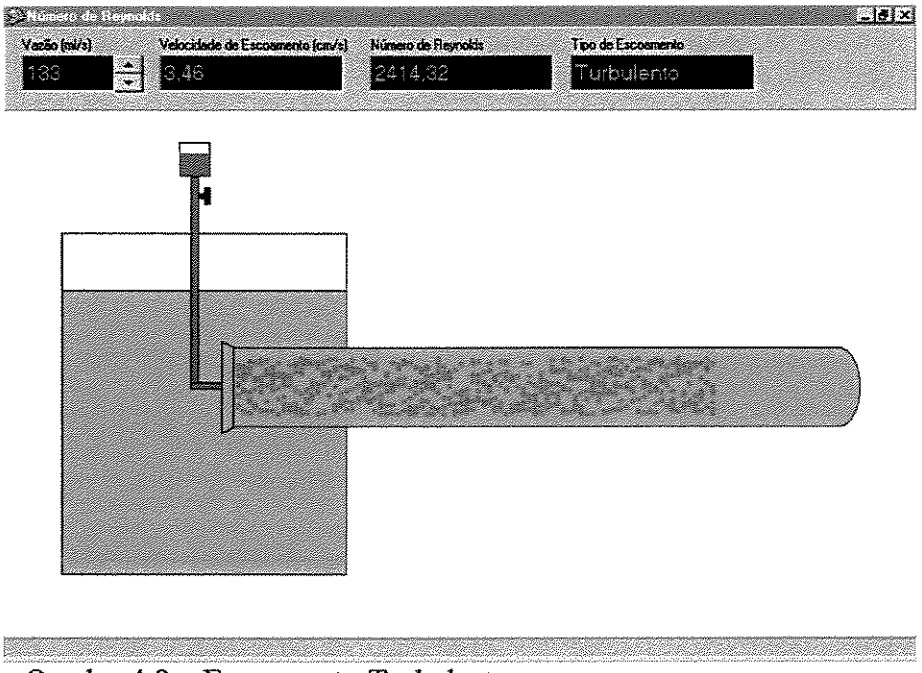
Os quadros 4.6, 4.7 e 4.8 representam os escoamentos: laminar, de transição e turbulento, encontrados por Reynolds em sua experiência.



Quadro 4.6 – Escoamento Laminar



Quadro 4.7 – Escoamento de Transição



Quadro 4.8 – Escoamento Turbulento

4.3 Experimento 3: Funcionamento de um rotâmetro (Flutuador)

4.3.1. O Problema:

O rotâmetro, forma mais amplamente usada na medição de vazão, é um tubo cônico provido de um medidor flutuante. O rotâmetro (figura 4.5) deve ser instalado na vertical de forma que fluido o atravesse de baixo para cima e o flutuador poderá ser de várias formas, dependendo da aplicação. Aqui usaremos um flutuador esférico.

As forças atuantes no flutuador serão, portanto, a força peso (para baixo), o empuxo de Arquimedes (para cima) e a força de arraste (para cima) que o fluido aplica na esfera. Uma vez em equilíbrio a altura da esfera indicará a vazão no tudo, desde que essa vazão esteja dentro da capacidade do instrumento. Essa observação é feita através de uma escala graduada em unidades de vazão na própria parede do tudo que, neste caso, deve ser transparente. Para medidores maiores e que devam suportar altas pressões o tubo geralmente é constituído de metal e um sensor

magnético é usado para indicar a posição do flutuador.

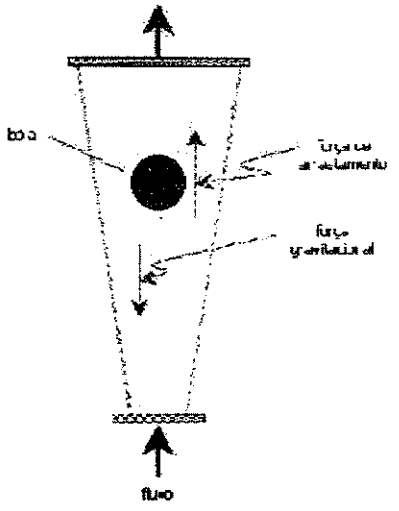


Figura 4.5 – Estrutura de um rotâmetro

A força resultante no flutuador será dada por:

$$F = M_f \cdot g \cdot \frac{(\rho_f - \rho)}{\rho_f}$$

Onde, M_f – massa do flutuador

g – aceleração da gravidade

ρ – densidade do fluido

ρ_f – densidade do flutuador

E o cálculo da vazão observada pela equação:

$$Q_t = (\alpha^2 - \alpha) \cdot \alpha \cdot \sqrt{\frac{\pi}{2}} \cdot D_f \cdot \sqrt{\frac{F}{\rho}}$$

Onde, $\alpha = D / D_f$

D – diâmetro do tubo na altura do flutuador

D_f – diâmetro do flutuador

4.3.2 O Programa: **Funcionamento de um Rotâmetro** (Anexo V)

O objetivo do sistema é simular o comportamento de um rotâmetro com flutuador esférico.

O programa recebe as seguintes entradas:

- Densidade do fluído (em Kg/m^3)
- Densidade do flutuador (em Kg/m^3)
- Diâmetro do flutuador (em cm)
- Altura do medidor (em cm)
- Velocidade com que o fluido entra no tubo (em m/s)

Baseado nessas entradas o programa calcula a altura do flutuador e a vazão indicada. Além disso mostra graficamente as diversas posições do flutuador e as graduações do medidor para as diversas combinações de entrada.

4.3.3 Estrutura do Programa

O programa foi desenvolvido no Borland Delphi 5, que utiliza o Object Pascal como linguagem de programação.

O paradigma da Orientação a Objetos foi utilizado e a estrutura do programa é composta de duas classes, descritas a seguir.

- **TfrmPrincipal**

Esta é uma classe de interface. Ela faz toda a comunicação entre as entradas do usuário e a classe TRotametro (descrita a seguir). Entre seus atributos ela contém, além dos controles para entrada de dados, um objeto da classe TRotametro.

Os principais métodos da classe TfrmPrincipal são: FormCreate, FormPaint e Atualiza:

✓ FormCreate: atribui valores iniciais para altura do tubo, densidade e velocidade fluido, densidade e diâmetro do flutuador. Além disso, instancia um objeto da classe TRotametro e um objeto Bitmap onde será feito o desenho do tubo.

✓ FormPaint: sempre que a tela precisar ser redesenhada este método é chamado e então solicita ao objeto de TRotametro que refaça o desenho.

✓ Atualiza: lê as entradas presentes nos controles e as repassa ao objeto da classe TRotametro.

Além desses, cada controle visual possui um método que chama Atualizar sempre que seus valores forem mudados pelo usuário.

- **TRotametro**

É a classe responsável por todo o tratamento do tubo, desde o cálculo das saídas até o desenho o mesmo. A função dela é atender as solicitações da classe TfrmPrincipal, que são basicamente três: atualizar valores, calcular as saídas e desenhar o tubo. Seus atributos são compostos de dados sobre o problema (densidade, diâmetro do flutuador, etc) e dados sobre o desenho (posição do tubo, comprimento em pixels, etc).

A descrição dos métodos, bem como comentários sobre seu funcionamento, pode ser vista no código fonte (Anexo VI) do programa.

Resumindo a operação do programa, representada na figura 4.6.

1. O usuário interage com os controles da classe TfrmPrincipal
2. A classe TfrmPrincipal comunica as alterações das entradas à classe TRotametro
3. A classe TRotametro recalcula as saídas e redesenha o tubo.

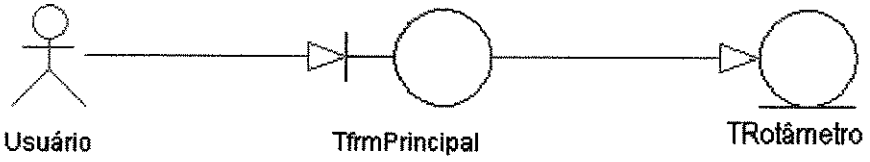
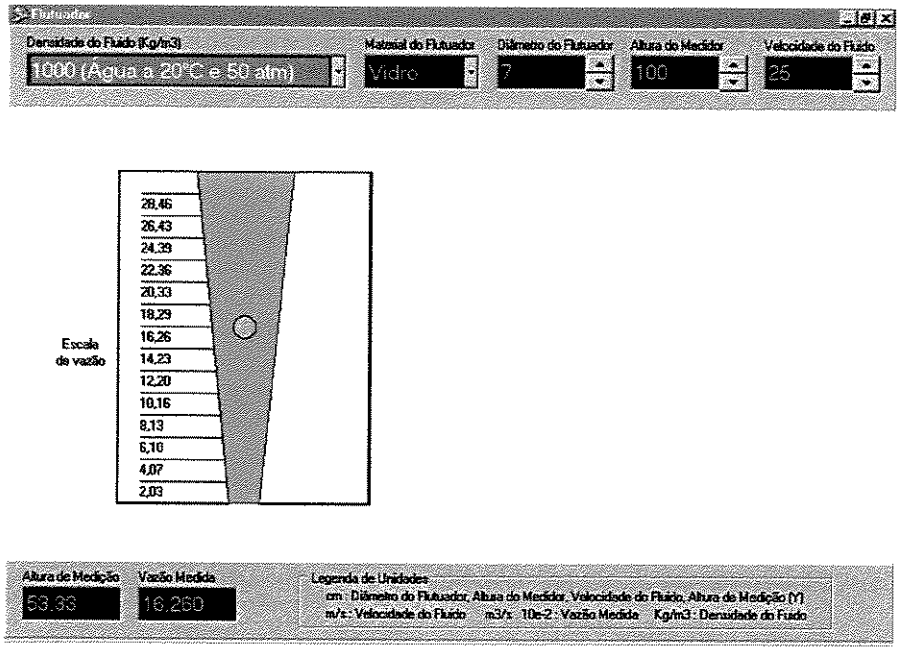


Figura 4.6 - Organograma do Programa do Rotâmetro

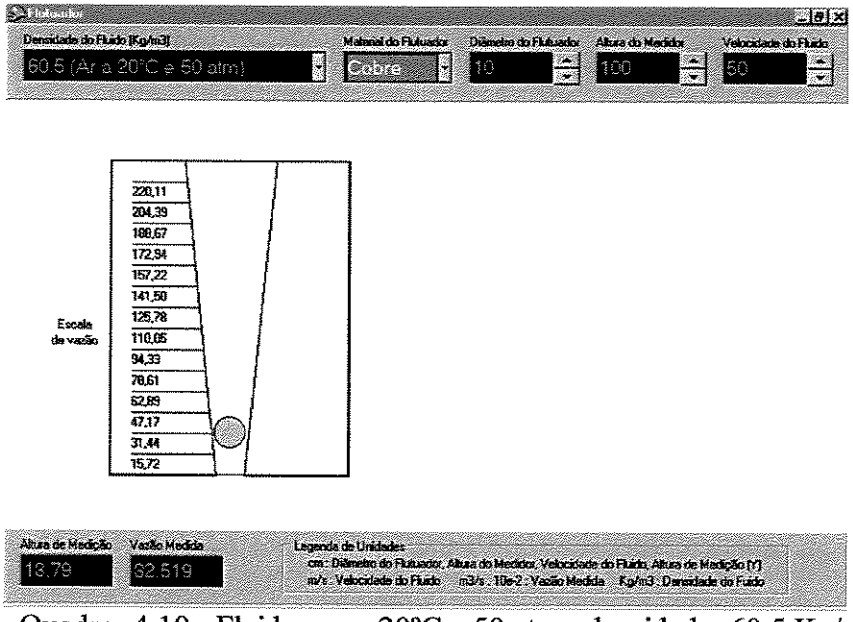
4.3.4 Quadros do Programa

Os quadros 4.9, 4.10, 4.11 e 4.12 demonstram situações onde pode-se variar o tamanho do rotâmetro, o tamanho e material do flutuador e tipo de fluido utilizado, para que seja possível a realização de vários testes em modelos diferentes de rotâmetros, dependendo da utilização que se queira desejar.

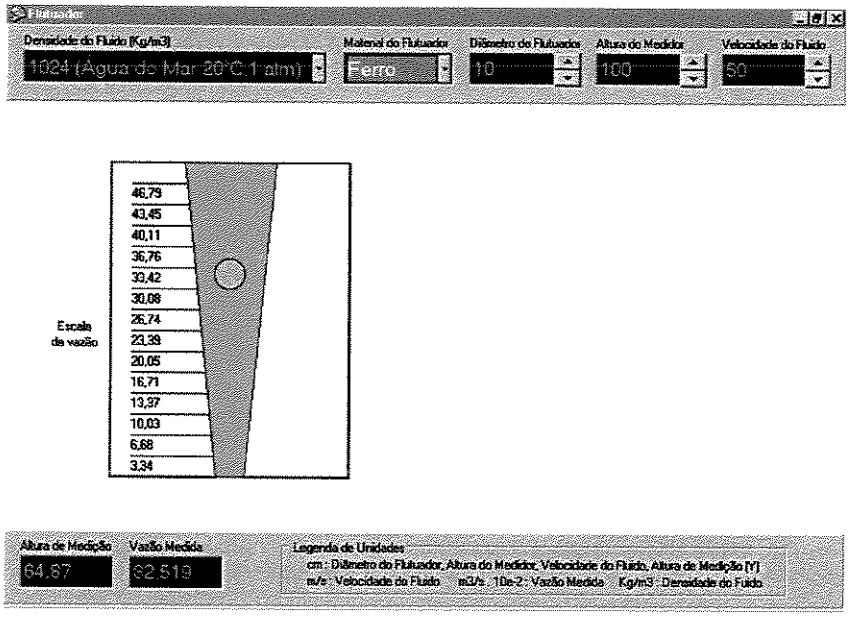


Quadro 4.9: Fluido: água a 20°C e 50 atm; densidade 1000 Kg/m^3

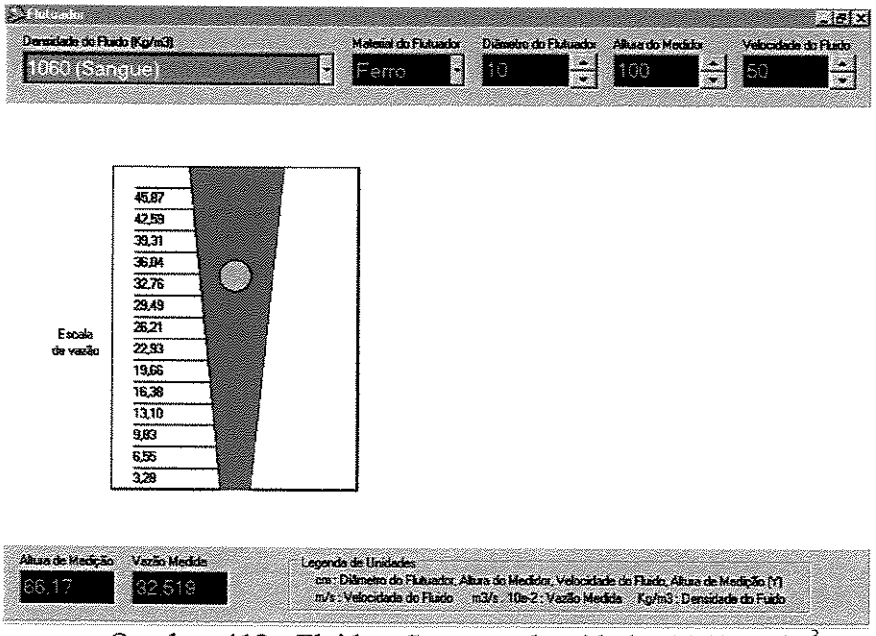
Material Flutuador: Vidro Vazão: 16,260 m^3 / s x 10^{-2}



Quadro 4.10: Fluido: ar a 20°C e 50 atm; densidade 60,5 Kg/m^3
Material Flutuador: Cobre Vazão: 32,519 m^3 / s x 10^{-2}



Quadro 4.11: Fluido: água do mar a 20°C e 1 atm; densidade 1024 Kg/m^3
Material Flutuador: Ferro Vazão: 32.519 m^3 / s x 10^{-2}



Quadro 412: Fluido: Sangue; densidade 1060 Kg/m^3
Material Flutuador: Ferro Vazão: 32,,519 m^3 / s x 10^{-2}

5. CONCLUSÃO

A evolução da tecnologia vem provocando uma revolução na legislação, administração do ensino, e conseqüentemente, no conhecimento. O acesso à Internet, a disseminação do uso do computador e os softwares estão possibilitando mudar a forma de produzir, armazenar e disseminar a informação. As fontes de pesquisa abertas aos alunos pela Internet, as bibliotecas digitais em substituição às publicações impressas e os cursos à distância vêm crescendo gradativamente. Diante disso, escolas e universidades estão iniciando o processo de repensar suas funções e metodologias de ensino-aprendizagem.

O Objetivo deste trabalho foi demonstrar que o planejamento e projeto de experiências virtuais podem auxiliar o ensino de mecânica dos fluidos nos curso superiores através de simulações de equações e funcionamento de equipamentos aqui demonstrados.

Esperamos contribuir para uma melhor compreensão dos fundamentos relacionados a mecânica dos fluidos e suas aplicações no ensino mediado pelas novas tecnologias.

6. BIBLIOGARFIA

ISMAIL, Kamal Adbel Radi. Técnicas Experimentais em Fenômenos de Transportes. Campinas, SP. Ed. do autor, 2000.488p.

STREETER, Victor L., **WYLIE**, Benjamin E. Mecânica dos Fluidos. São Paulo, SP, Ed. McGraw Hill do Brasil, 1982. 585p.

MUNSON, Bruce R,. **YUNG**, Donald F., **OKIISHI**, Theodore H. Fundamentos da Mecânica dos Fluidos . Volume 1.São Paulo, SP. Ed. Edgard Blucher Ltda. 1997. 404p.

SHAMES, Irving Herman. Mecânica dos Fluidos. Princípios Básicos.Volume 1. São Paulo, SP. Ed. Edgard Blucher Ltda. 1982. 404p. 192p.

LÉVY, Pierre. As tecnologias da inteligência - o futuro do pensamento na era da informática. Rio de Janeiro: ed. 34, 6ª edição, 1996.

VALENTE, José Armando. Informática na educação: uma questão técnica ou pedagógica. Revista Pátio, ano 3, nº 9, mai/jul 1999.

FERRAZ NETTO, Luiz. Equação de Bernoulli. http://www.feiradeciencias.com.br/sala07/07_RHD.asp. Acessado em 20/08/2002.

ANEXOS

ANEXO I

- **Programa Equação de Bernoulli em um Tubo de Venturi.**

D:\programas.zip\Venturi.exe

ANEXO II

Código Fonte Programa Bernoulli

Arquivo: Venturi.dpr

```
program Venturi;

uses
  Forms,
  untPrincipal in 'untPrincipal.pas' {frmPrincipal},
  untVenturi in 'untVenturi.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmPrincipal, frmPrincipal);
  Application.Run;
end.
```

Arquivo: untPrincipal.pas

```
untPrincipal;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, untVenturi, ComCtrls, TeeProcs, TeEngine, Chart,
  ImgList, math;

type
  array8 = array [0..7] of double;
  TfrmPrincipal = class(TForm)
    pnDados: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    edDiametro1: TEdit;
    Label3: TLabel;
    edDiametro2: TEdit;
    Label4: TLabel;
    edPressao1: TEdit;
    Label5: TLabel;
    edPressao2: TEdit;
    udD1: TUpDown;
    udP2: TUpDown;
    udD2: TUpDown;
    udP1: TUpDown;
    cmbDensidade: TComboBox;
    Timer: TTimer;
    TimerFluxo: TTimer;
    Panel1: TPanel;
    Label11: TLabel;
    Label14: TLabel;
    edVazao: TEdit;
    edVelocidade2: TEdit;
    edVelocidade1: TEdit;
    Label6: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure udD1Click(Sender: TObject; Button: TUDBtnType);
    procedure TimerFluxoTimer(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure cmbDensidadeChange(Sender: TObject);
  private
    { Private declarations }
    venturi : TVenturi;
  public
    { Public declarations }
    procedure Atualiza;
  end;
```

```
var
  frmPrincipal: TfrmPrincipal;
  densidades: array8;
  bitmap: Tbitmap;
  cor: TColor;

implementation

{$R *.DFM}

procedure TfrmPrincipal.Atualiza;
var d1,d2,den,p1,p2,vazao:double;

begin
   d1 := udD1.position;
   d2 := udD2.position;
   den := densidades[cmbDensidade.itemindex];
   p1 := udP1.position;
   p2 := udP2.position;

   if venturi.AtualizaValores(d1,d2,den,p1,p2,cor) = true then
   begin
      edVelocidade1.text := format('%.2f',[venturi.getVelocidade1]);
      edVelocidade2.text := format('%.2f',[venturi.getVelocidade2]);
      //calcular e exibir a vazao
      vazao := venturi.getVelocidade1 * PI * power((d1/100)/2,2);
      //lbVazao.Caption := 'Vazão = ' + format('%.2f',[vazao]) + ' m3/s';
      edVazao.Text := format('%.2f',[vazao]);
   end
   else begin
      //repõe os valores corretos
      udP1.position := trunc(p1);
      udP2.position := trunc(p2);
      udD1.position := trunc(d1);
      udD2.position := trunc(d2);
   end;
end;

procedure TfrmPrincipal.FormCreate(Sender: TObject);
var d1,d2,den,p1,p2:double;
begin
   cor := clAqua;

   bitmap := TBitmap.Create;
```

```
    bitmap.height:=480;
    bitmap.Width:=790;

    densidades[0]:=1.21;
    densidades[1]:=60.5;
    densidades[2]:=100.0;
    densidades[3]:=998.0;
    densidades[4]:=1000.0;
    densidades[5]:=1024.0;
    densidades[6]:=1060.0;
    densidades[7]:=13600.0;

    cmbDensidade.ItemIndex:=3;
    udD1.Position := 100;
    udD2.Position := 20;
    udP1.Position := 60;
    udP2.Position := 20;

    d1 := udD1.position;
    d2 := udD2.position;
    den := densidades[cmbDensidade.itemindex];
    p1 := udP1.position;
    p2 := udP2.position;

    venturi := TVenturi.Create(55,160,135,130,135,2,cor);

    venturi.AtualizaValores(d1,d2,den,p1,p2,cor);

    Atualiza;
end;

procedure TfrmPrincipal.udD1Click(Sender: TObject; Button: TUDBtnType);
begin
    Atualiza;
end;

procedure TfrmPrincipal.TimerFluxoTimer(Sender: TObject);
begin
    venturi.incProgresso(10);
    venturi.desenha(bitmap.canvas);
    canvas.draw(0,0,bitmap);
end;

procedure TfrmPrincipal.FormPaint(Sender: TObject);
begin
```

```
    venturi.desenha(bitmap.canvas);
    canvas.draw(0,0,bitmap);
end;

procedure TfrmPrincipal.FormDestroy(Sender: TObject);
begin
    bitmap.free;
end;

procedure TfrmPrincipal.cmbDensidadeChange(Sender: TObject);
begin
    case cmbDensidade.ItemIndex of
      0: cor := clSilver;  //ar
      1: cor := clSilver;  //ar
      2: cor := clOlive;   //espuma
      3: cor := clAqua;    //agua
      4: cor := clAqua;    //agua
      5: cor := clAqua;    //agua
      6: cor := clRed;     //sangue
      7: cor := clMaroon;  //mercurio
    end;

    Atualiza;
end;

end.
```

Arquivo: untVenturi.pas

```
unit untVenturi;

interface

uses math, graphics, windows, classes, sysutils;

type
  //definição dos tipos de vetores
  array4 = array [1..4] of integer;
  array6 = array [1..6] of integer;
  ArrayDePontos = array [1..12] of TPoint;

  //definição da classe TVenturi
  TVenturi = class
  private
    //atributos privados
    diametro1 : double;        //diâmetro maior em cm2
```

```
    diametro2 : double;       //diâmetro menor em cm2
    densidade : double;       //densidade do líquido em Kg/m3
    pressao1: double;         //pressão na parte maior em 10^5Pa
    pressao2: double;         //pressão na parte menor em 10^5Pa
    velocidade1 : double;     //velocidade na parte maior em m/s
    velocidade2 : double;     //velocidade na parte menor em m/s
    //atributos do desenho
    xi : integer;             //coluna do canto superior esquerdo
    yi : integer;             //linha do canto superior esquerdo
    m1 : integer;             //medida até a inclinação do gargalo
    m2 : integer;             //medida da inclinação do gargalo
    m3 : integer;             //medida do gargalo
    escala : integer;         //escala centimetros-pixels
    progresso : integer;      //posicao em que o fluxo de água se encontra
    cor : TColor;             //cor do liquido
    //métodos privados
    procedure CalculaVelocidades;
    function Interpola(p1,p2:TPoint;x:integer): integer;
    procedure DesenhaLiquido(canvas:TCanvas;pts:ArrayDePontos);
  public
    //métodos públicos
    constructor Create(xi,yi,m1,m2,m3,escala:integer;c:TColor);
    function AtualizaValores(var d1,d2,den,p1,p2:double;c:TColor): boolean;
    function getVelocidade1: double;
    function getVelocidade2: double;
    procedure Desenha(canvas:TCanvas);
    function getProgresso: integer;
    procedure incProgresso(step:integer);
  end;

implementation

//construtor da classe
constructor TVenturi.Create(xi,yi,m1,m2,m3,escala:integer;c:TColor);
begin
    Self.xi := xi;
    Self.yi := yi;
    Self.m1 := m1;
    Self.m2 := m2;
    Self.m3 := m3;
    Self.escala := escala;
    diametro1 := 1;
    diametro2 := 0;
    densidade := 0;
    pressao1 := 1;
```

```
   pressao2 := 0;
   progresso := xi+25;
   cor := c;
end;

//atualiza atributos da classe com os parâmetros passados
function TVenturi.AtualizaValores(var d1,d2,den,p1,p2:double;c:TColor): boolean;
begin
   cor:=c;
   if (d1>d2) and (p1>p2) then
   begin
      diametro1 := d1;
      diametro2 := d2;
      densidade := den;
      pressao1 := p1*power(10,5);
      pressao2 := p2*power(10,5);
      CalculaVelocidades;
      result := true;
   end
   else begin
      d1 := diametro1;
      d2 := diametro2;
      den := densidade;
      p1 := pressao1/power(10,5);
      p2 := pressao2/power(10,5);
      result := false;
   end;
end;

//faz o cálculo de velocidade1 e velocidade2
procedure TVenturi.CalculaVelocidades;
var area1,area2,auxiliar: double;   //variáveis temporárias
begin
   area1 := PI*power(diametro1/2,2);
   area2 := PI*power(diametro2/2,2);
   auxiliar := (densidade/2.00) * ((power(area1,2)/power(area2,2))-1.00);
   velocidade1 := sqrt((pressao1-pressao2)/auxiliar);
   velocidade2 := (area1/area2)*velocidade1;
end;

//retorna a velocidade1
function TVenturi.getVelocidade1: double;
begin
   result := velocidade1;
end;
```

```
//retorna a velocidade2
function TVenturi.getVelocidade2: double;
begin
   result := velocidade2;
end;

//desenha o Tubo de Venturi na superfície passada como parâmetro
procedure TVenturi.Desenha(canvas:TCanvas);
var x: array6;
   y: array4;
   ponto : ArrayDePontos;
   ul,ue: integer;
begin
   //calcula deslocamento da parte mais larga
   ul := trunc(diametro1) * escala;
   //calcula deslocamento da parte mais estreita
   ue := trunc(diametro2) * escala;

   //calcula abcissas necessárias
   x[1] := xi;
   x[2] := x[1] + m1;
   x[3] := x[2] + m2;
   x[4] := x[3] + m3;
   x[5] := x[4] + m2;
   x[6] := x[5] + m1;

   //calcula ordenadas necessárias
   y[1] := yi;
   y[2] := yi + (ul div 2) - (ue div 2);
   y[3] := yi + (ul div 2) + (ue div 2);
   y[4] := yi + ul;

   //monta os pontos dos cantos do tubo
   ponto[1] := Point(x[1],y[1]);
   ponto[2] := Point(x[2],y[1]);
   ponto[3] := Point(x[3],y[2]);
   ponto[4] := Point(x[4],y[2]);
   ponto[5] := Point(x[5],y[1]);
   ponto[6] := Point(x[6],y[1]);
   ponto[7] := Point(x[6],y[4]);
   ponto[8] := Point(x[5],y[4]);
   ponto[9] := Point(x[4],y[3]);
   ponto[10] := Point(x[3],y[3]);
   ponto[11] := Point(x[2],y[4]);
```

```
  ponto[12] := Point(x[1],y[4]);

  //limpa area de desenho
  canvas.pen.Color := clwhite;
  canvas.Rectangle(0,0,790,460);
  canvas.pen.Color := clBlue;

  //desenha o tubo
  canvas.Brush.Color := clWhite;
  canvas.Pen.Color := clBlue;
  canvas.Pen.Width := 2;
  canvas.Ellipse(x[6]-20,y[1],x[6]+20,y[4]);
  canvas.Polygon(ponto);
  canvas.Pen.Color := clWhite;
  canvas.Polyline([Point(x[6],y[1]+1),Point(x[6],y[4]-2)]);
  canvas.Pen.Color := clBlue;
  canvas.Ellipse(x[1]-20,y[1],x[1]+20,y[4]);

  //desenha o líquido
  canvas.Brush.Color := cor;
  desenhaLiquido(canvas,ponto);
  canvas.Brush.color := clWhite;

  //desenha hastes dos manômetros
  canvas.Rectangle((2*xi+m1) div 2,yi-22,((2*xi+m1) div 2)+5,yi+1);
  canvas.Rectangle((ponto[3].x+ponto[4].x) div 2,ponto[3].y-22,
    ((ponto[3].x+ponto[4].x) div 2)+5,ponto[3].y+1);

  //desenha os manômetros
  canvas.brush.Style := bsClear;
  canvas.Rectangle(((2*xi+m1) div 2)-39,yi-22-22,((2*xi+m1) div 2)+45,yi-21);
  canvas.TextOut(((2*xi+m1) div 2)-39+10,yi-22-18,
    IntToStr(trunc(pressao1/power(10,5)))+' x 10e5 Pa');
  canvas.Rectangle(((ponto[3].x+ponto[4].x) div 2)-39,ponto[3].y-22-22,
    ((ponto[3].x+ponto[4].x) div 2)+45,ponto[3].y-21);
  canvas.TextOut(((ponto[3].x+ponto[4].x) div 2)-39+10,ponto[3].y-22-18,
    IntToStr(trunc(pressao2/power(10,5)))+' x 10e5 Pa');
  canvas.brush.Style := bsSolid;

end;

function TVenturi.Interpola(p1,p2:TPoint;x:integer): integer;
var m:double;
begin
  m := (p2.y - p1.y) / (p2.x - p1.x);
```

```
  result := trunc( m * (x - p1.x) + p1.y );
end;

function TVenturi.getProgresso: integer;
begin
  result := progresso;
end;

procedure TVenturi.incProgresso(step:integer);
var velocidade: double;
begin

  //se está no primeiro trecho mais largo
  //ou se está no segundo trecho mais largo
  if (progresso<(xi+m1)) or
   ((progresso>=(xi+m1+2*m2+m3)) and (progresso<(xi+2*m1+2*m2+m3))) then
  begin
    velocidade := velocidade1;
  end
  //se está em outro trecho
  else begin
    velocidade := velocidade2;
  end;

  step := trunc((40 * velocidade)/8026);
  if (step < 1) and (velocidade>0) then
  begin
    step := 1;
  end;

  progresso := progresso + step;
  if progresso > xi+2*m1+2*m2+m3 then
  begin
    progresso := xi + 25;
  end;
end;

procedure TVenturi.DesenhaLiquido(canvas:TCanvas;pts:ArrayDePontos);
begin
  canvas.Pen.Width := 1;

  //se está no primeiro trecho mais largo
  if (progresso<(xi+m1)) then
  begin
    canvas.PenPos := Point(progresso,pts[1].y);
```

```
      canvas.LineTo(progresso,pts[12].y);
    end
    //se está no primeiro funil
    else if (progresso>=(xi+m1)) and (progresso<(xi+m1+m2)) then
    begin
      canvas.PenPos := Point(progresso,interpola(pts[2],pts[3],progresso));
      canvas.LineTo(progresso,interpola(pts[11],pts[10],progresso));
    end
    //se está no gargalo
    else if (progresso>=(xi+m1+m2)) and (progresso<(xi+m1+m2+m3)) then
    begin
      canvas.PenPos := Point(progresso,pts[3].y);
      canvas.LineTo(progresso,pts[10].y);
    end
    //se está no segundo funil
    else if (progresso>=(xi+m1+m2+m3)) and (progresso<(xi+m1+2*m2+m3)) then
    begin
      canvas.PenPos := Point(progresso,interpola(pts[4],pts[5],progresso));
      canvas.LineTo(progresso,interpola(pts[9],pts[8],progresso));
    end
    //se está no segundo trecho mais largo
    else if (progresso>=(xi+m1+2*m2+m3)) and (progresso<(xi+2*m1+2*m2+m3)) then
    begin
      canvas.PenPos := Point(progresso,pts[5].y);
      canvas.LineTo(progresso,pts[8].y);
    end;

    //desenha o líquido
    canvas.FloodFill(progresso-2,(pts[1].y+pts[12].y) div 2,clBlue,fsBorder);

    canvas.Pen.Width := 2;
  end;

  end.
```

ANEXO III

- **Programa Experiência de Reynolds**

D:\programas.zip|Reynolds.exe

ANEXO IV

Código Fonte do Programa Experiência de Reynolds

Arquivo: Reynolds.dpr

```
program Reynolds;

uses
  Forms,
  untPrincipal in 'untPrincipal.pas' {frmPrincipal},
  untReynolds in 'untReynolds.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmPrincipal, frmPrincipal);
  Application.Run;
end.
```

Arquivo: untPrincipal.pas

```
unit untPrincipal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
```

```
  StdCtrls, ExtCtrls, untReynolds, ComCtrls, TeeProcs, TeEngine, Chart,
  ImgList;

type
  array8 = array [0..7] of extended;
  TfrmPrincipal = class(TForm)
    pnDados: TPanel;
    Label3: TLabel;
    edVazao: TEdit;
    StatusBar: TStatusBar;
    udVazao: TUpDown;
    Timer: TTimer;
    TimerFluxo: TTimer;
    Label1: TLabel;
    edVelocidade: TEdit;
    Label2: TLabel;
    edReynolds: TEdit;
    Label4: TLabel;
    edTipo: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure udDiametroClick(Sender: TObject; Button: TUDBtnType);
    procedure TimerFluxoTimer(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure cmbDensidadeChange(Sender: TObject);
  private
    { Private declarations }
    reynolds: TReynolds;
  public
    { Public declarations }
    procedure Atualiza;
  end;

var
  frmPrincipal: TfrmPrincipal;
  bitmap: Tbitmap;

implementation

{$R *.DFM}

procedure TfrmPrincipal.Atualiza;
var diametro,vazao:extended;
begin
   //diametro := udDiametro.position/100.0;
```

```
    diametro := 7/100.0;
    vazao := udVazao.position/1000000.0;
    reynolds.AtualizaValores(diametro,vazao,998.0,0.001);
    reynolds.reset;
end;

procedure TfrmPrincipal.FormCreate(Sender: TObject);
var diametro, vazao:extended;
begin
    randomize;
    bitmap := TBitmap.Create;
    bitmap.height:=480;
    bitmap.Width:=790;

    diametro := 7/100.0;
    vazao := udVazao.position/1000000.0;

    reynolds := TReynolds.Create(200,280,530,10);

    reynolds.AtualizaValores(diametro,vazao,998,0.001);

    Atualiza;
end;

procedure TfrmPrincipal.udDiametroClick(Sender: TObject; Button: TUDBtnType);
begin
    Atualiza;
end;

procedure TfrmPrincipal.TimerFluxoTimer(Sender: TObject);
var interval:cardinal;
begin
    reynolds.incProgresso(interval);
    TimerFluxo.interval := interval;
    reynolds.desenha(bitmap.canvas);
    canvas.draw(0,0,bitmap);
end;

procedure TfrmPrincipal.FormPaint(Sender: TObject);
begin
    reynolds.desenha(bitmap.canvas);
    canvas.draw(0,0,bitmap);
end;

procedure TfrmPrincipal.FormDestroy(Sender: TObject);
```

```
begin
   bitmap.free;
end;

procedure TfrmPrincipal.cmbDensidadeChange(Sender: TObject);
begin
   Atualiza;
end;

end.
```

Arquivo: untReynolds.pas

```
unit untReynolds;

interface

uses math, graphics, windows, classes, sysutils;

type
 //definição dos tipos de vetores
 ArrayDePontos = array [1..500] of integer;

 //definição da classe TReynolds
 TReynolds = class
 private
  //atributos privados
  diametro : extended;        //diâmetro da tubulação
  densidade : extended;       //densidade do líquido em Kg/m3
  viscosidade : extended;     //viscosidade do líquido em Pa.s
  vazao: extended;            //vazão do líquido pela tubulação em m3/s
  velocidade : extended;      //velocidade de escoamento em m/s
  nreynolds : extended;       //número de reynolds associado ao escoamento
  tipo : string;
  //atributos do desenho
  xi : integer;               //coluna do canto superior esquerdo
  yi : integer;               //linha do canto superior esquerdo
  comprimento : integer;      //comprimento do tubo
  escala : integer;           //escala centimetros-pixels
  progresso : integer;        //posicao em que o fluxo de água se encontra
  indice : integer;           //indice do ultimo ponto do array de ordenadas
  pontos : ArrayDePontos;     //array de ordenadas
  pontos2 : ArrayDePontos;    //array de ordenadas
  pontos3 : ArrayDePontos;    //array de ordenadas
  pontos4 : ArrayDePontos;    //array de ordenadas
  pontos5 : ArrayDePontos;    //array de ordenadas
  pontos6 : ArrayDePontos;    //array de ordenadas
  pontos7 : ArrayDePontos;    //array de ordenadas
  //métodos privados
  procedure CalculaNumeroDeReynolds;
 public
  //métodos públicos
  constructor Create(xi,yi,comp,escala:integer);
  procedure incProgresso(var interval:cardinal);
  procedure AtualizaValores(d,vaz,den,vis:extended);
  procedure Desenha(canvas:TCanvas);
```

```
  procedure Reset;
 end;

implementation

uses untPrincipal;

//construtor da classe
constructor TReynolds.Create(xi,yi,comp,escala:integer);
begin
   Self.xi := xi;
   Self.yi := yi;
   comprimento := comp;
   Self.escala := escala;
   diametro := 0;
   densidade := 0;
   viscosidade := 0;
   velocidade := 0;
   vazao := 0;
   progresso := xi+5;
   indice := 0;
   tipo := '';
end;

//atualiza atributos da classe com os parâmetros passados
procedure TReynolds.AtualizaValores(d,vaz,den,vis:extended);
begin
   diametro := d;
   vazao := vaz;
   densidade := den;
   viscosidade := vis;
   CalculaNumeroDeReynolds;
end;

//faz o cálculo da velocidade de escoamento e do numero de reynolds
procedure TReynolds.CalculaNumeroDeReynolds;
begin
   velocidade := (4 * vazao) / (PI * power(diametro,2));
   nreynolds := (densidade * velocidade * diametro) / viscosidade;
end;

//desenha a tubulação na superfície passada como parâmetro
procedure TReynolds.Desenha(canvas:TCanvas);
var altura,i,x: integer;
   cor : TColor;
```

```
begin
   if tipo = 'Laminar' then
   begin
      cor := clTeal;
   end
   else if tipo = 'Transição' then
   begin
      cor := clPurple;
   end
   else begin
      cor := clRed;
   end;

   altura := trunc(diametro*100) * escala;

   canvas.Brush.Color := clWhite;
   canvas.pen.Color := clwhite;
   canvas.Rectangle(0,0,790,460);

   canvas.pen.Color := clBlue;
   //exibe propriedades
   {canvas.TextOut(xi+200,yi-90,'Velocidade Média de Escoamento = '
      + format('%.2f,[velocidade*100]) + ' cm/s      ');
   canvas.TextOut(xi+200,yi-90+25,'Número de Reynolds = '
      + format('%.2f,[nreynolds]) + '         ');
   canvas.TextOut(xi+200,yi-90+50,'Tipo de Escoamento = '
      + tipo + '         ');
   }
   with frmPrincipal do
   begin
      edVelocidade.text := format('%.2f,[velocidade*100]);
      edReynolds.text := format('%.2f,[nreynolds]);
      edTipo.text := tipo;
   end;

   //set pen e brush
   canvas.Brush.Color := clAqua;
   canvas.Pen.Color := clBlack;
   canvas.Pen.Width := 1;

   //desenha reservatorio
   canvas.rectangle(xi-150,yi-50,xi+100,yi+200);
   canvas.Brush.Color := clWhite;

   canvas.rectangle(xi-150,yi-100,xi+100,yi-49);
```

```
canvas.Brush.Color := cor;
canvas.rectangle(xi-37,yi-150,xi-30,yi+(altura div 2));
canvas.rectangle(xi-37,yi+(altura div 2)-4,xi-9,yi+(altura div 2)+2);
canvas.rectangle(xi-47,yi-170,xi-20,yi-150);

canvas.Brush.Color := clBlack;
canvas.rectangle(xi-30,yi-135,xi-25,yi-130);
canvas.rectangle(xi-25,yi-140,xi-20,yi-125);
canvas.Brush.Color := clWhite;
canvas.rectangle(xi-47,yi-180,xi-20,yi-169);

//desenha o tubo
canvas.Brush.Color := clAqua;

canvas.MoveTo(xi,yi);
canvas.LineTo(xi-10,yi-5);
canvas.MoveTo(xi,yi+altura);
canvas.LineTo(xi-10,yi+altura+5);
canvas.LineTo(xi-10,yi-6);

canvas.Ellipse(xi+comprimento-20,yi,xi+comprimento+20,yi+altura);
canvas.Rectangle(xi,yi,xi+comprimento,yi+altura);
canvas.Pen.Color := clAqua;
canvas.Polyline([Point(xi+comprimento-1,yi+1),Point(xi+comprimento-1,yi+altura-2)]);

//desenha liquido vermelho
canvas.Pen.Color := cor;
canvas.Pen.Width := 1;
x := xi + 5;

for i:=indice downto 1 do
begin
   canvas.Ellipse(x-2,pontos[i]-2,x+2,pontos[i]+2);
   if tipo = 'Turbulento' then
   begin
      canvas.Ellipse(x-2,pontos2[i]-2,x+2,pontos2[i]+2);
      canvas.Ellipse(x-2,pontos3[i]-2,x+2,pontos3[i]+2);
      canvas.Ellipse(x-2,pontos4[i]-2,x+2,pontos4[i]+2);
      canvas.Ellipse(x-2,pontos5[i]-2,x+2,pontos5[i]+2);
      canvas.Ellipse(x-2,pontos6[i]-2,x+2,pontos6[i]+2);
      canvas.Ellipse(x-2,pontos7[i]-2,x+2,pontos7[i]+2);
   end;
   x := x + 5;
end;
```

```
end;

procedure TReynolds.incProgresso(var interval:cardinal);
var step,altura,y,yeixo,area: integer;
begin
   step := 5;
   progresso := progresso + step;
   indice := indice + 1;
   if progresso >= (comprimento + xi) then
   begin
      reset;
   end;

   altura := trunc(diametro*100) * escala;
   if nreynolds < 2000 then
   begin
      //escoamento laminar
      area := trunc(altura*0.20);
      tipo := 'Laminar';
      interval := 150;
   end
   else if (nreynolds >= 2000) and (nreynolds < 2400) then
   begin
      //escoamento de transicao
      area := trunc(altura*0.40);
      tipo := 'Transição';
      interval := 70;
   end
   else begin
      //escoamento turbulento
      area := trunc(altura*0.70);
      tipo := 'Turbulento';
      interval := 20;
   end;

   yeixo := yi + (altura div 2);
   y := yeixo - (area div 2) + random(area);
   pontos[indice] := y;
   if tipo = 'Turbulento' then
   begin
      y := yeixo - (area div 2) + random(area);
      pontos2[indice] := y;
      y := yeixo - (area div 2) + random(area);
      pontos3[indice] := y;
      y := yeixo - (area div 2) + random(area);
```

```
      pontos4[indice] := y;
      y := yeixo - (area div 2) + random(area);
      pontos5[indice] := y;
      y := yeixo - (area div 2) + random(area);
      pontos6[indice] := y;
      y := yeixo - (area div 2) + random(area);
      pontos7[indice] := y;
   end;

end;

procedure Treynolds.Reset;
begin
   indice:=1;
   progresso := xi + 5;
end;

end.
```

ANEXO V

- **Programa Funcionamento do Rotâmetro (Flutuador)**

D:\programas.zip\Flutuador.exe

ANEXO VI

Código Fonte do Programa Rotâmetro (Flutuador)

Arquivo: Flutuador.dpr

```
program Flutuador;

uses
  Forms,
  untPrincipal in 'untPrincipal.pas' {frmPrincipal},
  untFlutuador in 'untFlutuador.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmPrincipal, frmPrincipal);
  Application.Run;
end.
```

Arquivo: untPrincipal.pas

```
unit untPrincipal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, untFlutuador, ComCtrls, TeeProcs, TeEngine, Chart,
  ImgList, math;

type
  array7 = array [0..6] of double;
  array4 = array [0..2] of double;
  TfrmPrincipal = class(TForm)
    pnDados: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    edAlturaMedidor: TEdit;
    Label4: TLabel;
```

```
    Label5: TLabel;
    edPressao2: TEdit;
    udAltura: TUpDown;
    udDf: TUpDown;
    cmbDensidade: TComboBox;
    Panel1: TPanel;
    Label11: TLabel;
    edVazao: TEdit;
    edAltura: TEdit;
    Label6: TLabel;
    cmbFlutuador: TComboBox;
    Label3: TLabel;
    edVelocidade: TEdit;
    udVelocidade: TUpDown;
    GroupBox1: TGroupBox;
    Label8: TLabel;
    Label10: TLabel;
    Label7: TLabel;
    Label9: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure cmbDensidadeChange(Sender: TObject);
    procedure udDfClick(Sender: TObject; Button: TUDBtnType);
  private
    { Private declarations }
    rotametro : TFlutuador;
  public
    { Public declarations }
    procedure Atualiza;
    function metros(centimetros:double) : double;
    function centimetros(metros:double) : double;
    procedure ajustaCmdDensidade(p:double);
    procedure ajustaCmbFlutuador(pf:double);
  end;

var
  frmPrincipal: TfrmPrincipal;
  densidade_fluido: array7;
  densidade_flutuador: array4;
  bitmap: Tbitmap;

implementation

{$R *.DFM}
```

```
procedure TfrmPrincipal.Atualiza;
var p,pf,df,h,ve,q,y,qmax,posicao:double;
   cor : Tcolor;
begin
   p := densidade_fluido[cmbDensidade.itemindex];

   case cmbDensidade.ItemIndex of
     0: cor := clWhite;   //ar
     1: cor := clWhite;   //ar
     2: cor := clAqua;    //agua
     3: cor := clAqua;    //agua
     4: cor := clAqua;    //agua
     5: cor := clRed;     //sangue
   else
     cor := clAqua;
   end;

   pf := densidade_flutuador[cmbFlutuador.itemindex];

   df := metros(udDf.position);
   h := metros(udAltura.position);
   //verificar consistencia
   ve := udVelocidade.position;

   if rotametro.verificaEntradas(p,pf,df,h,ve) = true then
   begin
      rotametro.AtualizaValores(p,pf,df,h,ve,cor);
      rotametro.getSaidas(q,y,qmax,posicao);
      edVazao.text := format('%.3f',[q*100]);
      edAltura.text := format('%.2f',[centimetros(y)]);
      FormPaint(Self);
   end
   else begin
      ShowMessage('Vazão máxima excedida. Selecione outra combinação de valores.');
      ajustaCmdDensidade(p);
      ajustaCmbFlutuador(pf);
      udVelocidade.position := trunc(ve);
      udAltura.position := trunc(centimetros(h));
      udDf.position := trunc(centimetros(df));
      Abort;
   end;
end;

procedure TfrmPrincipal.FormCreate(Sender: TObject);
```

```
begin
   bitmap := TBitmap.Create;
   bitmap.height:=550;
   bitmap.Width:=790;

   densidade_fluido[0]:=1.21;
   densidade_fluido[1]:=60.5;
   densidade_fluido[2]:=998.0;
   densidade_fluido[3]:=1000.0;
   densidade_fluido[4]:=1024.0;
   densidade_fluido[5]:=1060.0;

   densidade_flutuador[0]:=7800.0;  //ferro
   densidade_flutuador[1]:=8920.0;  //cobre
   densidade_flutuador[2]:=2500.0;  //vidro

   cmbDensidade.ItemIndex:=3;
   cmbFlutuador.ItemIndex:=2;
   udDf.Position := 7;
   udAltura.Position := 100;
   udVelocidade.position := 25;

   rotametro := TFlutuador.Create(100,430,100,70,3,metros(udDf.position));

   Atualiza;
end;

procedure TfrmPrincipal.FormPaint(Sender: TObject);
begin
   rotametro.desenha(bitmap.canvas);
   canvas.draw(0,0,bitmap);
   canvas.TextOut(50,280,'Escala');
   canvas.TextOut(44,295,'de vazão');
end;

procedure TfrmPrincipal.FormDestroy(Sender: TObject);
begin
   bitmap.free;
end;

procedure TfrmPrincipal.cmbDensidadeChange(Sender: TObject);
begin
   Atualiza;
end;
```

```
procedure TfrmPrincipal.udDfClick(Sender: TObject; Button: TUDBtnType);
begin
   Atualiza;
end;

function TfrmPrincipal.metros(centimetros: double): double;
begin
   result := centimetros / 100;
end;

function TfrmPrincipal.centimetros(metros: double): double;
begin
   result := metros * 100;
end;

procedure TfrmPrincipal.ajustaCmbFlutuador(pf: double);
var i : integer;
begin
   i := 2;
   while i>0 do
   begin
      if densidade_flutuador[i] = pf then
      begin
         break;
      end;
      i := i - 1;
   end;
   if i>=0 then
   begin
      cmbFlutuador.ItemIndex := i;
   end;
end;

procedure TfrmPrincipal.ajustaCmdDensidade(p: double);
var i : integer;
begin
   i := 6;
   while i>0 do
   begin
      if densidade_fluido[i] = p then
      begin
         break;
      end;
      i := i - 1;
   end;
```

```
  if i>=0 then
  begin
    cmbDensidade.ItemIndex := i;
  end;
end;

end.
```

Arquivo: untFlutuador.pas

```
unit untFlutuador;

interface

uses math, graphics, windows, classes, sysutils;

type
 //definição da classe TFlutuador
 TFlutuador = class

 private
  //atributos básicos do desenho (em pixels)
  xi    : integer;        //coluna do canto inferior esquerdo
  yi    : integer;        //linha do canto inferior esquerdo
  base  : integer;        //tamanho da base
  topo  : integer;        //tamanho do topo
  escala : integer;       //escala de desenho pixels/centimetro

  //atributos calculados do desenho (em pixels)
  espaco : integer;       //tamanho do espaco entre as hastes
  altura : integer;       //altura do medidor
  diametro : integer;     //diametro do medidor
  posicao : integer;      //altura de flutuacao

  //atributos de entrada do desenho
  cor : TColor;           //cor do fuido

  //atributos de entrada para cálculo
  p  : double;            //densidade do fluido (kg/m3)
  pf : double;            //densidade do flutuador (kg/m3)
  df : double;            //diametro do flutuador (m)
  h  : double;            //altura do medidor (m)
  ve : double;            //velocidade de injecao (m/s)
```

```
  //atributos de calculo intermediarios
  dtmin : double;          //diametro de entrada do medidor (m)
  dtmax : double;          //diametro de saida do medidor (m)
  vf    : double;          //volume do flutuador (m3)
  mf    : double;          //massa do flutuador (kg)
  f     : double;          //forca resultante no flutuador (Kg.m/s2)
  alfa  : double;          //razao entre dtmax e df (adimensional)
  qmax  : double;          //vazao maxima (m3/s)
  vemax : double;          //velocidade de injeção maxima

  //atributos de saida
  q : double;              //vazao medida (m3/s)
  y : double;              //altura de flutuacao (m)

  procedure RecalculaParametros;

 public
  //métodos públicos
  constructor Create(xi,yi,base,topo,escala:integer;dfi:double);
  procedure Desenha(canvas:TCanvas);
  function InterpolaX(p1,p2:TPoint;y:integer): integer;
  procedure AtualizaValores(p,pf,df,h,ve: double; cor:integer);
  function centimetros(metros:double) : double;
  function getVemax: double;
  procedure getSaidas(var q, y, qmax, posicao : double);
  procedure DesenhaFlutuador(canvas:TCanvas);
  function metros(centimetros:double) : double;
  function verificaEntradas(var p,pf,df,h,ve: double) : boolean;
 end;

implementation

//construtor da classe
constructor TFlutuador.Create(xi,yi,base,topo,escala:integer;dfi:double);
begin
   Self.xi := xi;
   Self.yi := yi;
   Self.base := base;
   Self.topo := topo;
   Self.escala := escala;
   df := dfi;
   dtmin := df + df*0.30;  //30% maior que df
end;

//atualiza dados
```

```
procedure TFlutuador.AtualizaValores(p,pf,df,h,ve: double; cor:integer);
begin
   //atualiza entradas
   Self.p := p;
   Self.pf := pf;
   Self.df := df;
   Self.h := h;
   Self.ve := ve;
   Self.cor := cor;

   RecalculaParametros;
end;

//recalcula veriaveis intermediarias e de saida
procedure TFlutuador.RecalculaParametros;
begin
   //dtmin := df + df*0.30;  //30% maior que df

   espaco := trunc( centimetros(dtmin) * escala );

   dtmax := metros((2*base + espaco - 2*topo) / escala);
   vf := 4/3 * PI * power(df/2,3);
   mf := pf * vf;
   f := mf * 10 * (pf-p)/pf;
   alfa := dtmax / df;
   qmax := (power(alfa,2)-1) * alfa * sqrt(PI/2) * df * sqrt(f/p);
   vemax := (qmax * 4) / (PI * power(dtmin,2));

   q := ve * PI * power(dtmin/2,2);
   y := (q * h)/qmax;

   altura := trunc( centimetros(h) * escala );
   diametro := trunc( centimetros(df) * escala );
   posicao := trunc( centimetros(y) * escala );
end;

//desenha o Rotametro na superfície passada como parâmetro
procedure TFlutuador.Desenha(canvas:TCanvas);
var x,y:integer;
   p1,p2:TPoint;
   marcacao : double;
begin
   //limpa area de desenho
   canvas.Brush.color := clWhite;
   canvas.pen.Color := clwhite;
```

```
canvas.Rectangle(0,0,790,550);

//desenha fundo (cor do liquido)
canvas.Pen.Color := clBlack;
canvas.Brush.Color := cor;
canvas.Rectangle(xi,yi-altura,xi+(2*base)+espaco,yi);

//desenha o tubo
canvas.Brush.Color := clWhite;
canvas.Pen.Color := clBlack;
canvas.Pen.Width := 1;

//parte esquerda
//(iniciando do canto inferior esquerdo do trapézio esquedo)
canvas.MoveTo(xi,yi);
canvas.LineTo(xi,yi-altura);
canvas.LineTo(xi+topo,yi-altura);
canvas.LineTo(xi+base,yi);
canvas.LineTo(xi,yi);
canvas.FloodFill(xi+3,yi-3,clBlack,fsBorder);

//parte direita
canvas.MoveTo(xi+base+espaco,yi);
canvas.LineTo(xi+base+espaco+base,yi);
canvas.LineTo(xi+base+espaco+base,yi-altura);
canvas.LineTo(xi+base+espaco+base-topo,yi-altura);
canvas.LineTo(xi+base+espaco,yi);
canvas.FloodFill(xi+base+espaco+3,yi-3,clBlack,fsBorder);
//(cursor do canto inferior esquerdo do trapézio direito)

//desenha escala de vazoes
p1.x := xi + topo;
p1.y := yi - altura;
p2.x := xi + base;
p2.y := yi;
y := p2.y - 20;
while (y>p1.y) do
begin
   x := interpolaX(p1,p2,y);
   canvas.MoveTo(x,y);
   canvas.LineTo(xi+20,y);
   //calulca marcacao
   marcacao := (((yi-y) * qmax) / h) / escala;
   canvas.TextOut(xi+20,y+3,format('%.2f',[marcacao]));
   y := y - 20;
```

```
    end;

    DesenhaFlutuador(canvas);
end;

function TFlutuador.InterpolaX(p1,p2:TPoint;y:integer): integer;
var m:double;
begin
    m := (p2.y - p1.y) / (p2.x - p1.x);
    result := trunc( ((y - p1.y) / m)  + p1.x);
end;

function TFlutuador.centimetros(metros: double): double;
begin
    result := metros * 100;
end;

function TFlutuador.getVemax: double;
begin
    result := vemax;
end;

procedure TFlutuador.getSaidas(var q, y, qmax, posicao: double);
begin
    q := Self.q;
    y := Self.y;
    qmax := Self.qmax;
    posicao := self.posicao;
end;

procedure TFlutuador.DesenhaFlutuador(canvas:TCanvas);
var xc,yc,x1,y1,x2,y2:double;
begin
    xc := xi + (2*base + espaco) / 2;
    yc := yi - posicao;
    x1 := xc - (diametro/2);
    y1 := yc - (diametro/2);
    x2 := xc + (diametro/2);
    y2 := yc + (diametro/2);

    canvas.Brush.color := clSilver;
    canvas.Ellipse(trunc(x1),trunc(y1),trunc(x2),trunc(y2));
end;

function TFlutuador.metros(centimetros: double): double;
```

```
begin
   result := centimetros / 100;
end;

function TFlutuador.verificaEntradas(var p, pf, df, h, ve: double): boolean;
var dtmin, espaco, dtmax, vf, mf, f, alfa, qmax, vemax, q : double;
begin
   result := false;

   dtmin := df + df*0.30;  //30% maior que df
   espaco := trunc( centimetros(dtmin) * escala );
   dtmax := metros((2*base + espaco - 2*topo) / escala);
   vf := 4/3 * PI * power(df/2,3);
   mf := pf * vf;
   f := mf * 10 * (pf-p)/pf;
   alfa := dtmax / df;
   qmax := (power(alfa,2)-1) * alfa * sqrt(PI/2) * df * sqrt(f/p);
   vemax := (qmax * 4) / (PI * power(dtmin,2));

   if ve > vemax then
   begin
      p := Self.p;
      pf := Self.pf;
      df := Self.df;
      h := Self.h;
      ve := Self.ve;
      Exit;
   end;

   q := ve * PI * power(dtmin/2,2);

   if q > qmax then
   begin
      p := Self.p;
      pf := Self.pf;
      df := Self.df;
      h := Self.h;
      ve := Self.ve;
      Exit;
   end;

   result := true;
end;
end.
```