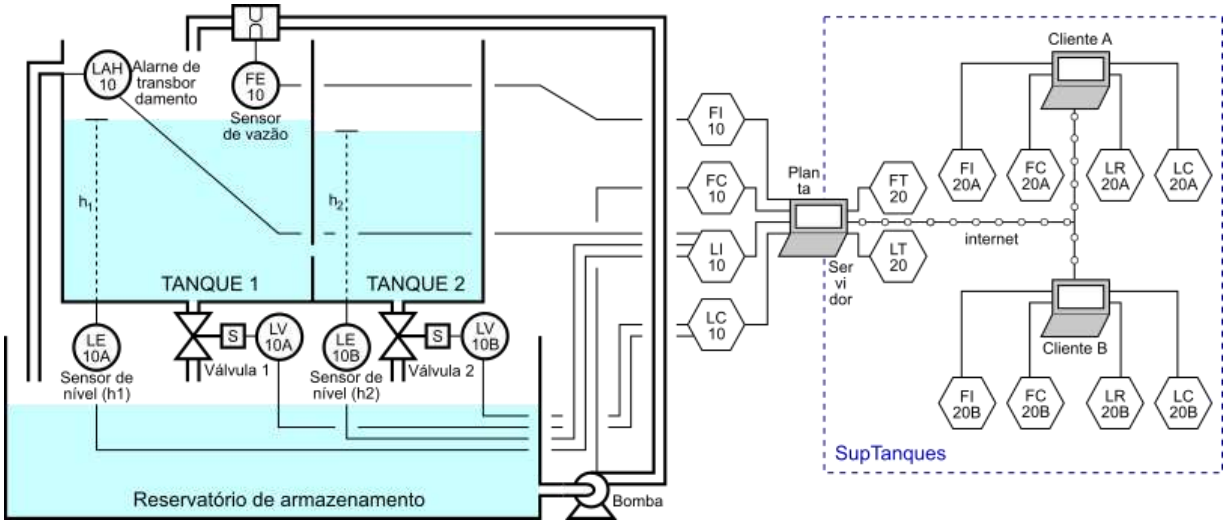
SUPTANQUES – SUPERVISÓRIO DE SISTEMA DE TANQUES
PROFESSOR: ADELARDO ADELINO DANTAS DE MEDEIROS

DESCRIÇÃO GERAL

O objetivo é desenvolver em C++ um sistema supervisório simplificado, denominado SupTanques, capaz de monitorar e gerenciar remotamente uma planta de controle de nível composto por 2 tanques, um reservatório de armazenamento, uma bomba, sensores e válvulas. Os diversos instrumentos e atuadores da planta estão interligados entre si e a um computador local por uma rede industrial, identificada pelo código 10, conforme diagrama de instrumentação a seguir.



Os dois tanques têm profundidade de 26 cm e altura de 28 cm. A largura do tanque 1 é de 22 cm e a do tanque 2, de 14 cm. Os tanques têm um orifício entre eles na altura de 6,5cm. O tanque 1 tem um orifício de transbordamento no nível de 25 cm. Em cada um dos tanques há um sensor (LE10A e LE10B) que mede o nível de líquido (h_1 e h_2) e uma válvula solenoide (LV10A e LV10B) que, quando aberta, permite escoamento do líquido para o reservatório de armazenamento. O tanque 1 tem um sensor de transbordamento (LAH10) que gera um sinal de alarme quando o nível h_1 igualar ou exceder a altura do orifício de transbordamento.

A bomba, quando ligada, envia líquido do reservatório de armazenamento para o tanque 1. A vazão da bomba é variável, de acordo com o sinal de entrada fixado pelo operador através do computador da planta. Um sensor de vazão (FE10) mede o fluxo de líquido bombeado para o tanque.

O computador local, conectado à planta, pode ler e exibir as medições dos sensores e do alarme de nível (LI10) e do sensor de vazão (FI10), abrir ou fechar as válvulas solenoides (LC10) e fixar o valor do sinal de entrada da bomba (FC10), além de ligar ou desligar todo o sistema.

O sistema SupTanques, a ser desenvolvido nesse projeto, deve permitir que a planta seja monitorada e gerenciada remotamente em outros computadores, e não apenas pelo computador local, utilizando uma arquitetura cliente-servidor. A rede de interligação entre o servidor e os diversos clientes, baseada em sockets TCP/IP na porta 23456 da internet, é identificada pelo código 20.

O programa servidor executará no mesmo computador local da planta, sendo capaz de interagir diretamente com os sensores e atuadores dos tanques. Ele se conecta com todos os programas clientes dos usuários, transmitindo dados e comandos de nível (LT20) e de vazão (FT20).

O programa cliente deve exibir o estado atual da planta, incluindo os níveis dos tanques (LR20) e a vazão da bomba (FI20) e permitir atuação do usuário sobre as válvulas (LC20) e sobre a entrada da bomba (FC20). A interface do programa cliente deve ser visual, representando graficamente os diversos componentes e informações da planta.

Todo usuário, antes de poder se conectar com o SupTanques através de um cliente, deve ter sido cadastrado no servidor com um login e senha. O login deve ser único e tanto login quanto senha devem ter entre 6 e 12 caracteres, diferenciando caracteres maiúsculos e minúsculos. Qualquer usuário cadastrado poderá visualizar todos os dados da planta, mas apenas os usuários cadastrados como administradores poderão atuar sobre a planta (abrir/fechar válvulas ou alterar a entrada da bomba). A categoria do usuário (administrador ou visualizador) será definida no cadastro.

A PLANTA

A planta é inspirada no sistema de controle de processos existente no LAUT, do DCA (ver figura a seguir), embora com alguns acréscimos (tais como a existência de um sensor de nível no tanque 2 e de um alarme de transbordamento no tanque 1), modificações (por exemplo, os tanques 1 e 2 estão em lados invertidos) e simplificações (tais como a redução na quantidade de válvulas existentes).

A planta será representada por um simulador, já disponibilizado no SIGAA (arquivos `tanques.h` e `tanques.cpp`). A simulação do sistema é feita pela classe `Tanks`, que tem funções públicas que correspondem às funcionalidades através das quais o computador da planta pode interagir com os sensores e atuadores do sistema de tanques:

- `bool tanksOn()` – Retorna se o sistema de tanques está ligado (`true`) ou desligado (`false`)
- `ValvState v1State()` – Retorna o estado da válvula 1: `OPEN,CLOSED`
- `ValvState v2State()` – Retorna o estado da válvula 2: `OPEN,CLOSED`
- `uint16_t hTank1()` – Retorna a medida do sensor de nível do tanque 1: 0 a 65535
- `uint16_t hTank2()` – Retorna a medida do sensor de nível do tanque 2: 0 a 65535
- `uint16_t pumpInput()` – Retorna o valor atual da entrada da bomba: 0 a 65535
- `uint16_t pumpFlow()` – Retorna a medida do sensor de vazão da bomba: 0 a 65535
- `bool isOverflowing()` – Retorna se o tanque 1 está transbordando (`true`) ou não (`false`)
- `void setTanksOn()` – Liga o sistema de tanques
- `void setTanksOff()` – Desliga o sistema de tanques
- `void setV1State(ValvState State)` – Fixa o estado da válvula 1: `OPEN,CLOSED`
- `void setV2State(ValvState State)` – Fixa o estado da válvula 2: `OPEN,CLOSED`
- `void setPumpInput(uint16_t Input)` – Fixa a entrada da bomba: 0 a 65535

As medições dos sensores são sempre retornadas em valores inteiros sem sinal de 2 bytes (de 0 a 65535). Para visualização dos valores em unidades físicas, o arquivo `suptanques.h` define constantes que correspondem aos valores máximos (fundo de escala) dos sensores:

- `MaxTankLevelMeasurement` – sensores de nível (0.265 m = 26.5 cm)
- `MaxPumpFlowMeasurement` – sensor de vazão (6.5 x 10^{-5} m^3/s = 3.9 litros/min)

COMUNICAÇÃO CLIENTE-SERVIDOR

As comunicações entre o cliente e o servidor, ou vice-versa, seguem um padrão:

1. Os primeiros 2 bytes sempre contêm um inteiro sem sinal (`uint16_t`) que indica um dos comandos da tabela abaixo. No arquivo `suptanques.h` já está definido um tipo enumerado, denominado `SupCommands`, que associa o valor numérico de cada comando com o seu nome.
2. Em seguida, seguem o(s) parâmetro(s) do comando, caso existam.
3. Os comandos e os parâmetros são enviados utilizando o padrão da biblioteca MySocket.

NOME	VALOR	PARÂMETROS	SIGNIFICADO (#)
`CMD_LOGIN`	1001	login e senha (string)	Conexão com o servidor (C→S)
`CMD_ADMIN_OK`	1002	-	Login válido de administrador (S→C)
`CMD_OK`	1003	-	Login válido de visualizador (S→C) Comando válido (S→C)
`CMD_ERROR`	1004	-	Comando inválido (S→C)
`CMD_GET_DATA`	1005	-	Solicitação de dados (C→S)
`CMD_DATA`	1006	dados (`SupState`)	Envio de dados (S→C)
`CMD_SET_V1` (§)	1007	estado (`uint16_t`)	Fixa estado da válvula 1 ou 2: 0 para fechada, ≠0 para aberta (C→S)
`CMD_SET_V2` (§)	1008	estado (`uint16_t`)	
`CMD_SET_PUMP` (§)	1009	entrada (`uint16_t`)	Fixa entrada da bomba (C→S)
`CMD_LOGOUT`	1010	-	Desconexão do cliente (C→S)

(#) Sentido de envio: cliente para servidor (C→S) ou servidor para cliente (S→C)
(§) Comandos disponíveis apenas para usuários administradores

Após `CMD_LOGIN` ser enviado pelo cliente, o que deve ser feito obrigatoriamente como o primeiro comando após o pedido de conexão ser aceito, o servidor responde com um dos 3 comados:

1. `CMD_ADMIN_OK` – Conexão aceita de um novo usuário administrador.
2. `CMD_OK` – Conexão aceita de um novo usuário não administrador (visualizador).
3. `CMD_ERROR` – Conexão rejeitada (usuário inexistente, senha incorreta, etc.).

O comando `CMD_DATA` é enviado pelo servidor após receber do cliente o comando `CMD_GET_DATA`, solicitando os dados atuais. O parâmetro único do comando `CMD_DATA` é do tipo `SupState`, uma struct definida em `suptanques.h` com 7 membros, todos do tipo `uint16_t`:

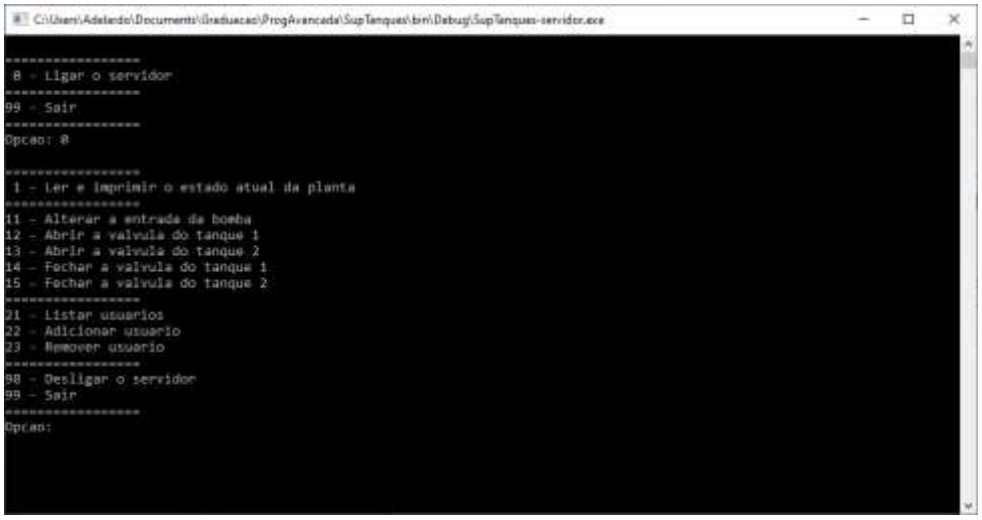
1. Estado atual da válvula 1 – 0 para fechada, ≠0 para aberta.
2. Estado atual da válvula 2 – 0 para fechada, ≠0 para aberta.
3. Medição do nível do tanque 1 – 0 (= 0.0) a 65535 (= `MaxTankLevelMeasurement`).
4. Medição do nível do tanque 2 – 0 (= 0.0) a 65535 (= `MaxTankLevelMeasurement`).
5. Valor atual da entrada da bomba – 0 (0%) a 65535 (100%).
6. Medição da vazão da bomba – 0 (= 0.0) a 65535 (= `MaxPumpFlowMeasurement`).
7. Valor atual do alarme – 0 para sem transbordamento, ≠0 para quando está transbordando.

Após envio pelo cliente de qualquer um dos comandos de atuação sobre a planta (`CMD_SET_V1`, `CMD_SET_V2` ou `CMD_SET_PUMP`), o servidor responde com `CMD_OK` ou `CMD_ERROR`, conforme tenha sido possível executar o comando ou não (por exemplo, o usuário não é administrador).

SERVIDOR

O programa servidor deve ter as seguintes funcionalidades:

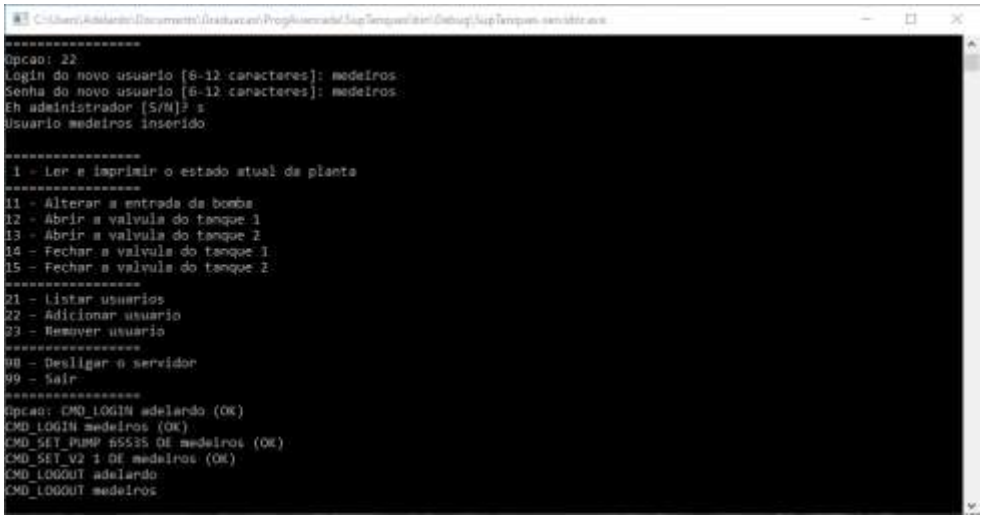
- Permite ao operador local:
  - Ligar e desligar o sistema como um todo.
  - Listar, adicionar e remover usuários cadastrados.
  - Visualizar (no console, em modo texto) todos os dados da planta.
  - Atuar sobre todos os componentes da planta.
- Monitora os sockets para reagir às comunicações dos clientes:
  - Em caso de pedido de dados (`CMD_GET_DATA`), envia os dados da planta (`CMD_DATA`).
  - Em caso de comandos de atuação (`CMD_SET_V1`, `CMD_SET_V2`, `CMD_SET_PUMP`), realiza a operação correspondente na planta, caso o usuário seja autorizado.
  - Em caso de pedido de conexão (`CMD_LOGIN`), estabelece e passa a monitorar o socket de comunicação com o cliente, caso o usuário esteja previamente cadastrado.



Para permitir o acompanhamento, o servidor deve imprimir uma mensagem sempre que:

- Um usuário se conectar ou se desconectar ou quando uma tentativa de conexão for rejeitada.
- Um comando de atuação (`CMD_SET_V1`, `CMD_SET_V2` ou `CMD_SET_PUMP`) for recebido.

Não deve ser impressa mensagem toda vez que receber um comando `CMD_GET_DATA`.

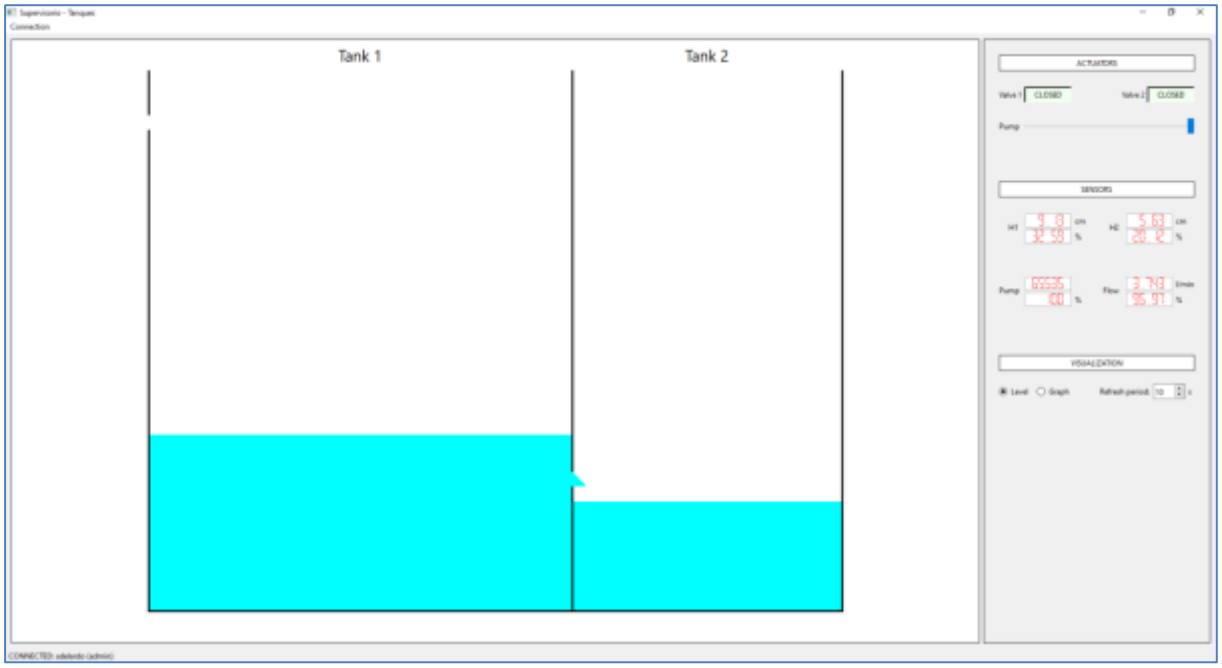


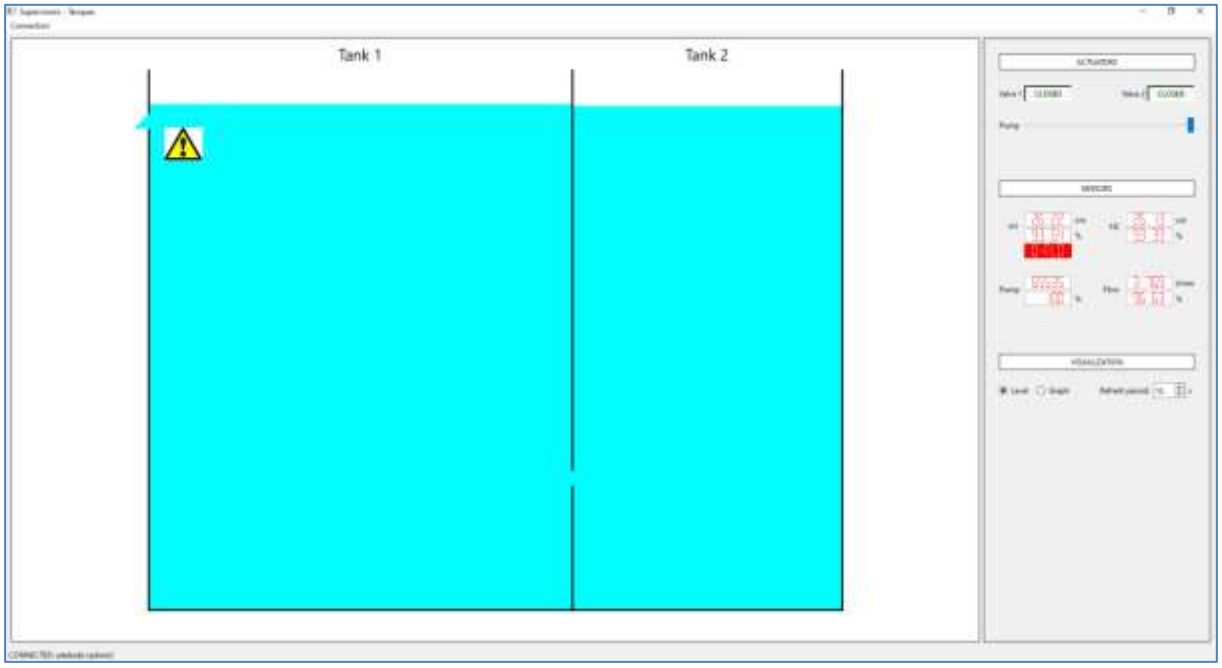
CLIENTE

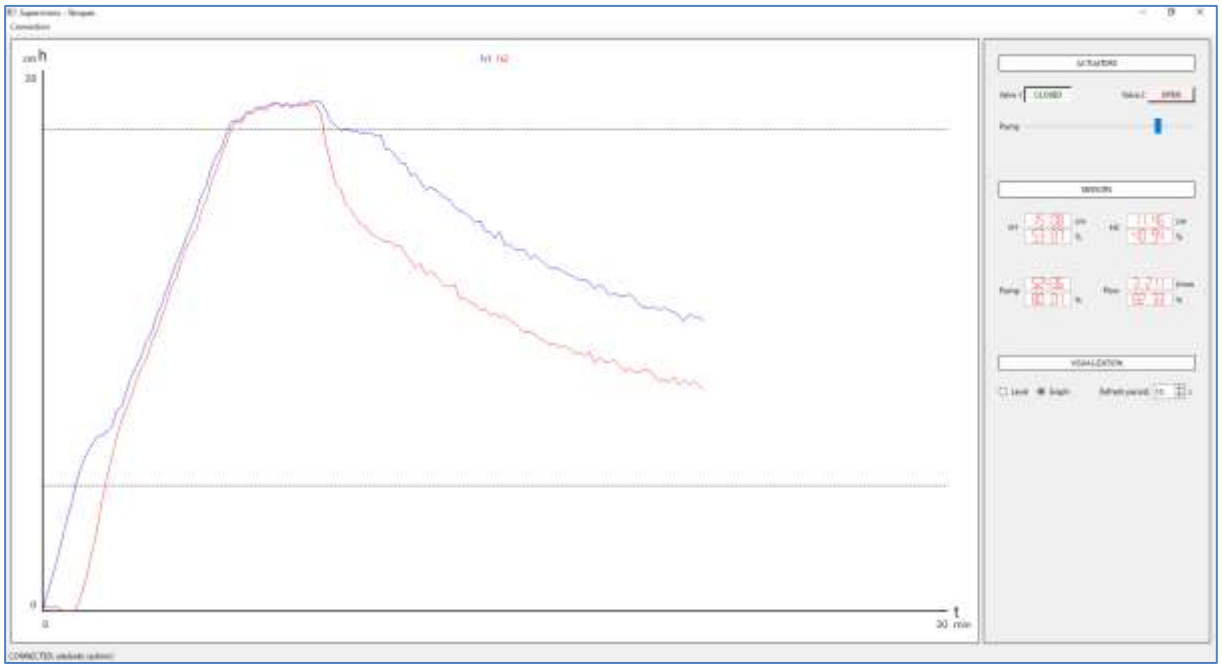
O programa cliente deve exibir o estado da planta e permitir atuação do usuário sobre ela através de uma interface visual com as seguintes funcionalidades:

- Exibir os níveis atuais dos tanques através do valor numérico e visualização em imagem, bem como os últimos valores em gráfico.
- Exibir o estado atual das válvulas e permitir atuar sobre elas.
- Exibir a vazão atual através do valor numérico e de visualização de percentual.
- Exibir através do valor numérico e de visualização de percentual o valor atual da entrada da bomba e permitir modificá-lo.

A visualização deverá poder ser feita em dois modos de apresentação: representação gráfica do nível atual dos tanques e da situação de escoamento nos orifícios ou representação do nível dos tanques ao longo do tempo em um gráfico. As figuras a seguir apresentam a interface a ser utilizada, nos dois modos de visualização.




Tank 1
Tank 2


h
t
30 min

IMPLEMENTAÇÃO

SERVIDOR:

O software do servidor do SupTanques é fornecido com uma implementação parcial. **NÃO** podem ser alterados (supressão, inclusão ou modificação) os arquivos que já são fornecidos completos:

- Biblioteca MySocket (`mysocket.h` e `mysocket.cpp`).
- Simulador dos tanques (`tanques-param.h`, `tanques.h` e `tanques.cpp`).
- Arquivo de definições gerais do projeto: `tanques-param.h`, `supdados.h` e `supdados.cpp`.
- O programa principal do servidor : `suptanques-servidor-main.cpp`.

A classe `SupServidor` está parcialmente declarada (`supservidor.h`) e implementada (`supservidor.cpp`). Ela herda da classe simuladora `Tanks`, de tal forma que as funções para interagir com os tanques estão disponíveis.

A classe `SupServidor` pode e deve receber acréscimos nas posições que estão comentadas no código. As partes que são fornecidas já implementadas **NÃO** podem ser suprimidas ou modificadas.

CLIENTE EM CONSOLE:

É fornecido um cliente em modo console parcialmente implementado. Os arquivos desse cliente que já são fornecidos completos **NÃO** podem ser alterados (supressão, inclusão ou modificação):

- Biblioteca MySocket (`mysocket.h` e `mysocket.cpp`).
- Arquivo de definições gerais do projeto: `supdados.h` e `supdados.cpp`.
- A classe do cliente em modo console (`supclient_term.h` e `supclient_term.cpp`)
- O programa principal: `supcliente_main_term.cpp`.

A classe `SupCliente` está parcialmente declarada (`supcliente.h`) e implementada (`supcliente.cpp`). Ela é uma classe base que deve implementar as partes comuns às duas versões do programa cliente: a versão em console e a versão visual (em Qt). As classes que implementam cada uma das interfaces específicas devem herdar dessa classe base.

A classe base `SupCliente` pode e deve receber acréscimos nas posições que estão comentadas no código. As partes que são fornecidas já implementadas **NÃO** podem ser suprimidas ou modificadas.

CLIENTE Qt:

A interface visual em Qt está parcialmente implementada, faltando ser feita a integração com a classe que implementa o cliente do sistema supervisório. **NÃO** podem ser alterados (supressão, inclusão ou modificação) os arquivos que já são fornecidos completos:

- Biblioteca MySocket (`mysocket.h` e `mysocket.cpp`).
- Arquivo de definições gerais do projeto: `tanques-param.h` e `supdados.h`.
- Classe da janela pop-up `SupLogin` (`.h`, `.cpp` e `.ui`).
- Classe de desenho `SupImg` (`.h`, `.cpp` e `.ui`).
- O programa principal do cliente Qt: `supcliente_main_qt.cpp`.

A classe que implementa o cliente em modo Qt `SupClienteQt` pode e deve receber acréscimos nas posições que estão comentadas no código. As partes que são fornecidas já implementadas **NÃO** podem ser suprimidas ou modificadas.

COMPILAÇÃO

Como se trata de um sistema distribuído, para testá-lo será necessário executar o cliente e o servidor simultaneamente, inclusive com várias instâncias do cliente em paralelo em algumas situações: um cliente visual e um ou mais clientes em console, todos supervisionando o sistema de tanques simultaneamente. Para isso, os executáveis na versão console deverão poder ser executados independentemente, fora da IDE do Code::Blocks. Para isso:

- No Code::Blocks, clique com o botão da direita no nome do projeto e escolha “Build options”.
- Nas opções do compilador (Compiler settings), assinale as opções de linkagem estática, ou seja, a parte necessária das bibliotecas do sistema passará a integrar seu código executável:
  - Static libgcc
  - Static libstdc++
  - Static linking

Lembrar que, para que a biblioteca MySocket funcione, é necessário linkar (no Windows) com a biblioteca Ws2_32. Para isso:

- Nas “Build options” do linkador (Linker settings), adicione a biblioteca “Ws2_32”