Instituto Superior de Engenharia de Coimbra
Departamento de Engenharia Informática e de Sistemas

Programação Orientada a Objetos - 2023/2024

Trabalho Prático

O trabalho prático de POO é constituído por um programa em C++. Este programa deve:

- Seguir os princípios e práticas de orientação a objetos;
- Ser feito em C++, usando corretamente a semântica desta linguagem;
- Usar as classes/bibliotecas usadas nas aulas, onde apropriado, ou outras da biblioteca standard C++;
- Não devem ser usadas outras bibliotecas sem o consentimento prévio dos docentes;
- Deve concretizar as funcionalidades do tema referidas no enunciado;
- Não é permitida uma abordagem baseada na mera colagem de excertos de outros programas ou de exemplos. Todo o código apresentado terá que ser demonstradamente entendido por quem o apresenta e explicado em defesa, caso contrário não será contabilizado.

Visão geral

Tema e conceitos gerais.

Pretende-se construir em C++ um simulador de uma habitação controlada por componentes de domótica interligados entre si. O simulador inclui zonas da habitação, que têm propriedades tais como temperatura, luz, etc., que são inspecionadas por sensores, os quais fornecem os valores lidos a regras que são geridas por processadores de regras que determinam o que fazer em função das leituras dos sensores. Nas zonas existem também aparelhos, os quais afetam as propriedades da zona onde se encontram e reagem a comandos emitidos pelos processadores de regras. O simulador é controlado por comandos escritos pelo utilizador. Existem uns quantos comandos, dando a falsa ideia de dimensão/complexidade, mas na verdade o simulador pouco mais é que um conjunto de objetos interligados entre si e uma interface de utilizador.

O simulador inclui a noção de passagem de tempo. Esta característica é importante para o funcionamento dos componentes. Por exemplo, um sensor detetor de movimento pode causar a ativação de um aparelho lâmpada, que permanecerá acesa por X instantes. A passagem do tempo no simulador é totalmente independente do tempo real medido pelo computador, sendo descrita em unidades abstratas de “instantes” e avança-se para o instante seguinte por comando do utilizador.

Interação com o utilizador

O simulador apresentará no ecrã uma visão esquemática da habitação incluindo as zonas e os vários componentes nelas existentes, e permitirá ao utilizador interagir com os componentes e perceber as propriedades de cada uma das zonas. O utilizador poderá também modificar a configuração da habitação e dos componentes e respetivas interligações, assim como efetuar operações de alto nível tais como: criar nova habitação, salvaguardar processadores de regras, recuperá-los posteriormente, etc. A interface do utilizador deve ser acima de tudo funcional e informativa, sendo aspetos de embelezamento gráfico secundários. Deverá existir uma área do ecrã dedicada à visualização da habitação, uma segunda área dedicada à introdução de comandos do utilizador, e uma terceira área dedicada à apresentação de informações textuais sobre o que está a acontecer no simulador e também respostas dos comandos introduzidos pelo utilizador. Será necessário o controlo do posicionamento do cursor para a impressão de caracteres em linha/coluna específicas e será disponibilizada uma biblioteca que auxilia nessa tarefa e na gestão de cores.

Conceitos principais envolvidos

Habitação. A habitação consiste num conjunto de **zonas** de habitação. Cada zona é colocada numa posição de uma grelha (uma zona por posição), podendo acontecer algumas posições ficarem vazias (sem zona de habitação nenhuma). A grelha que suporta as zonas (de habitação) tem um tamanho dinâmico decidido em *runtime*, com dimensões mínimas de 2x2 posições e máximo de 4x4 posições. A habitação será simplesmente o conjunto de zonas existentes na grelha. As zonas da habitação não necessitam de estar em posições contíguas. As zonas nunca se sobrepõem: cada uma está na sua posição da grelha, Visualmente (*user interface*), a informação de uma zona não deve ocultar informação de outra zona.

Zonas de habitação. Trata-se de um elemento de relevo, sendo que é a uma zona que estão associadas **propriedades** do ambiente de interesse para os **sensores** e **aparelhos**. A zona contém tudo o que são os componentes da domótica: os **sensores**, os **processadores** de regras, e os **aparelhos** (explicados mais adiante).

map

Propriedades do ambiente. As propriedades do ambiente estão associadas a uma **zona** e manifestam-se como um par chave-valor, em que a chave é uma string, e o valor é uma quantidade numérica. Por exemplo: (“temperatura”, graus célsius), ou (“luz”, quantidade de lumens). Os sensores e aparelhos, que interagem com as propriedades consultam a propriedade pela chave, obtendo/modificando o valor a ela associado.

Cada propriedade tem associadas regras de valor mínimo e máximo. Por exemplo, o valor mínimo que a temperatura pode ter é o valor absoluto de -273 C (arredondado), não existindo um valor máximo. A luz tem o valor mínimo de 0 lumens, não existindo máximo. Outras propriedades poderão não ter valor mínimo, mas ter máximo, não ter nem mínimo nem máximo, ou ter ambos. Cada propriedade saberá gerir os seus valores admissíveis.

O simulador suportará o seguinte conjunto de propriedades, mas o código deve ser compatível com a adição de novas propriedades, e essa adição deve ser trivial (não obrigar a mudar o código todo).

Propriedades	Unidade	Mínimo	Máximo
Temperatura	graus celsius	-273	-
Luz	Lúmens	0	-
Radiação	Becquerel	0	-
Vibração (p/ movimento)	Hertz (na atmosfera)	0	-
Humidade (Rel.)	%	0	100
Fumo	Obscuração (%)	0	100
Som	Decibéis (na gama audível)	0	-

Sensores. Os sensores são pequenos aparelhos que medem uma determinada propriedade do ambiente da zona na qual se encontram inseridos, proporcionando essa leitura a uma ou mais regras que a ele se encontrem ligadas. Cada sensor lida apenas com uma propriedade, sendo capaz de aferir o valor dessa propriedade. O sensor é completamente passivo e não “sente” nem reage à passagem do tempo.

O simulador permitirá, à partida, os seguintes sensores, podendo ser aumentado com novos sensores (e nesse caso sem ser necessário modificar significativamente o código).

Sensor	Letra (p/ visualização)	Propriedade observada
temperatura	t	Temperatura
mo**v**imento	v	Vibração
lu**m**inosidade	m	Luz
ra**d**iação	d	Radiação
humidade	h	Humidade
s**o**m	o	Som
fumo	f	Fumo

Processadores de regras. Os processadores de regras (apenas "**processador**" daqui em diante) são dispositivos que possuem regras. Não existe ligação direta entre processador e sensores, mas sim entre regras e sensores. Os processadores têm como output um comando (configurável). Esse comando é comunicado a um ou mais aparelhos que estejam ligados à saída do processador quando todas as regras nele existentes são cumpridas. Cada regra do processador pode estar ligada a um sensor diferente, pelo que a ativação do comando do processador pode estar dependente de mais do que um sensor.

Regras do processador. Uma regra está ligada a um sensor (duas regras diferentes podem estar ligadas ao mesmo sensor). Cada regra concretiza uma determinada condição de comparação entre o valor do sensor ao qual está ligada e um ou mais valores configuráveis. O resultado das regras será simplesmente um valor lógico. Isto faz com que o funcionamento do processador seja extremamente simples: tem um conjunto de regras, a cada instante avalia o resultado de cada uma delas. Se forem todas verdadeiro, o processador ativa o comando que está configurado.

O processador de regras é configurável da seguinte forma: as regras podem ser adicionadas ou removidas, e o comando a acionar é definido na construção do processador de regras, podendo mais tarde ser alterado.

Existem várias regras possíveis, sendo que o simulador deve já vir preparado para as seguintes:

Regra	Funcionamento: "verdadeiro se o valor da leitura…"	Parâmetros
igual_a	for igual a ***x***	x
menor_que	for menor que ***x***	x
maior_que	for maior que ***x***	x
entre	estiver entre ***x*** e **y**	x, y
fora	não estiver entre ***x*** e ***y***	x, y

Por omissão, um processador de regras não tem nenhuma regra associada, e portanto nunca aciona o seu comando. Depois de adicionar um processador de regras a uma zona, terá que se configurar, adicionando-lhe determinadas regras. As regras de um processador não são partilhadas com mais nenhum processador.

O processador conseguirá concretizar condições arbitrariamente complexas se for esse o desejo do utilizador (não significa que venham a ser exigidas). Para perceber o potencial disto, e a título meramente de exemplo, considere-se um processador com estas duas regras: *entre* 10 e 40 de um dado sensor de humidade, e *fora* de 20 a 30 do mesmo sensor. Basicamente o comando do processador com estas duas regras é acionado se o valor observado estiver entre 10 e 20 ou entre 30 e 40 (ou seja, conseguiu-se o efeito de um "ou" lógico). Esta complexidade emerge do conjunto das regras e não do código do programa, sendo um assunto do utilizador que configura o simulador, e o programador apenas tem que se preocupar em fornecer a funcionalidade das regras simples e elementares que estão listadas acima e garantir que o programa é expansível com novas regras e novas condições caso surgisse essa necessidade.

Aparelhos. Os aparelhos podem ser comandados quer pelo utilizador, quer pelo comando de um processador de regras ao qual estejam ligados. Os aparelhos podem modificar uma ou mais propriedades da zona em que se encontram.

Existem os seguintes aparelhos

Aparelho (nome)	Letra	comandos reconhecidos	Efeito: "se ligado…"	Efeito: "ao desligar"
aquecedor	a	**liga** - liga o aparelho **desliga** - desliga o aparelho	Adiciona um grau celsius à temperatura da zona por cada 3 instantes até ao máximo de 50º, e adiciona 5 db de ruído uma única vez.	remove 5 db de ruído
aspersor	s	**liga**: liga o aparelho, permanecendo ligado até 5 instantes após o comando desliga **desliga**: desliga o aparelho, mas continua a fazer efeito tal como se estivesse ligado por mais 5 instantes (restos de água que ainda está no cano)	No primeiro instante de ligado (ou seja, uma única vez por período em que está ligado): - Adiciona 50% de humidade relativa, até ao máximo de 75% de humidade. - Adiciona vibração de 100 Hz. Coloca o fumo a 0 uma única vez no segundo instante	remove 100 Hz de vibração
refrigerador	r	**liga** - liga o aparelho **desliga** - desliga o aparelho	Remove um grau celsius à temperatura da zona a cada 3 instantes ligado e adiciona 20 db de ruído no primeiro instante de ligado.	remove 20 db de ruído
lampada	l	**liga** - liga o aparelho **desliga** - desliga o aparelho	Adiciona 900 lúmens uma única vez por período em que está ligado.	remove 900 lúmens

Parece estar a funcionar

Parece estar a funcionar

Nao testado

Nao testado

Notas:

- Por coincidência, todos os aparelhos do enunciado reconhecem apenas os comandos "liga" e "desliga". No entanto, deve preparar o seu programa de forma a que os aparelhos (ou eventuais outros aparelhos) aceitam quaisquer comandos, que serão sempre uma string.
- Reparar que "ao desligar" significa que o efeito acontece apenas no primeiro instante em que está desligado: para voltar a fazer o efeito de desligar terá novamente que ser ligado e depois desligado.
- Alguns aparelhos apercebem-se da passagem do tempo e podem precisar de um contador de instantes interno e outras flags para saber quando fazem os seus efeitos.

Cada elemento do simulador deve ser facilmente identificado, existindo, para tal, um código único, incremental, e diferente para cada um, que permita especificar a criação, eliminação e associação dos componentes do simulador. Este identificador é composto por uma letra identificadora do tipo de componente ('a' - aparelho, 's' - sensor, 'p' - processador, 'r' - regra, armazenados como minúsculas) e um valor inteiro sempre crescente, único a nível da habitação. Este ID deve ser gerado tão automaticamente quanto possível. Para efeitos de apresentação, no caso dos aparelhos, a letra será apresentada como minúscula se o aparelho estiver desligado, e maiúscula se estiver ligado. Cada zona da casa, deve possuir também o seu identificador. Estes ID serão fundamentais para os comandos escritos pelo utilizador.

Para certos casos, poderá ser necessário identificar a que zona pertence cada equipamento, sendo necessário definir um mecanismo de identificação de equipamentos, por zona.

Funcionamento do simulador

Toda a simulação decorre num tempo simulado em "instantes". Na passagem de um instante para o seguinte o simulador faz os processadores de regras analisarem as suas regras, que consultam os sensores, para determinar que comando deve ser emitido. A cada instante também faz os aparelhos desempenharem as suas ações.

Apenas se avança para o instante seguinte por ordem do utilizador.

O utilizador é livre de mandar executar as ações que entender antes de avançar para o instante seguinte. As consequências visíveis dos seus comandos devem ser imediatamente representadas.

Interface com o utilizador

Em cada novo instante o simulador disponibiliza a informação relativa à generalidade do que se passa: comandos emitidos pelos processadores de regras, aparelhos que foram ligados ou desligados, propriedades que foram alteradas (qual, de que zona e qual o novo valor), o número do instante em que se vai, etc.

As ordens do jogador são dadas textualmente, segundo o paradigma dos comandos escritos. Cada ordem é escrita como uma frase em que a primeira palavra é um comando e as palavras seguintes são os parâmetros ou informações adicionais. A linha de texto correspondente à ordem é escrita na totalidade, só então sendo interpretada e executada pelo simulador. O programa deve validar a sintaxe e coerência do comando (toda a informação/parâmetros necessários foram escritos? Os valores que era suposto serem inteiros são mesmo inteiros? Estão pela ordem certa?). Pode ser assumido que será sempre tudo escrito em minúsculas.

O ecrã deve estar organizado em três áreas lógicas. Cada uma é dedicada a:

- apresentação de uma visão esquemática da habitação (um seja, a grelha com as zonas, cada uma com os seus componentes).
- introdução de comandos escritos pelo utilizador.
- apresentação de resultados dos comandos escritos e informações acerca do que se está a passar no simulador. Inclui-se nesta última área dados tais como: propriedade que tenham o seu valor alterado, e aparelhos que são ligados ou desligados.

Os pormenores de posicionamento exato dos elementos visuais ficam em aberto desde que sejam respeitadas a capacidade informativa e a qualidade de utilização. Deve ser dada atenção às seguintes restrições ou requisitos:

- Não haverá espaço no ecrã para apresentar menus e o seu uso é vedado.
- Em cada zona existem propriedades, sensores, processadores de regras e aparelhos. Os sensores e aparelhos devem ter uma representação visual que é o caracter indicado nas tabelas atrás, onde foram descritos. O posicionamento desses caracteres no espaço de ecrã reservado para a zona não é importante desde que sejam visíveis. A zona só ocupa "espaço" no ecrã, e a sua representação em memória não tem nada a ver com grelhas de caracteres (novamente, de forma a que fique claro: a implementação interna da zona não tem nada a ver com grelhas de caracteres).
- A representação visual de cada zona indica também o ID da zona.
- Pretende-se que toda a grelha com as zonas seja apresentada no ecrã de forma a evitar ter que fazer deslizar a informação (*scroll*). É necessário planear quantos caracteres de altura e largura se reservam para cada zona, de forma a não ocupar demasiado espaço, mas ao mesmo tempo garantir que se consegue visualizar os componentes todos em cada zona.

Comandos do utilizador

O utilizador pode interagir com o simulador e com os componentes através de comandos escritos na área do ecrã reservada para esse efeito. Abaixo descrevem-se os comandos.

Nesta descrição:

- <xxx> refere-se a um valor (número ou texto), e os caracteres < e > não fazem propriamente parte do que deve ser escrito.
- < x | y | z> indica que deve ser indicado apenas um dos x, y, z (novamente, sem os <>).
- [v] indica que o parâmetro v é opcional, e os caracteres [e] não devem ser escritos.

Os resultados dos comandos devem ser descritos na zona do ecrã reservada para esse efeito.

>> Comandos para o tempo simulado

prox

Avança 1 instante, despoletando os efeitos dos componentes que reagem à passagem do tempo.

avanca <n>

Avança **n** instantes, um de cada vez, em cada 1 é despoletado o efeito dos componentes que reagem à passagem do tempo.

>> Comandos para gerir habitação e zonas

hnova <num linhas> <num colunas>

Cria uma habitação (grelha para zonas) nova com o tamanho indicado. Não existe nenhuma habitação "default", pelo que este comando deverá ser o primeiro a ser executado numa simulação. Ao criar a nova habitação a existente será destruída assim como tudo o que lhe estiver associado.

hrem

Remove a habitação: apaga todas as zonas e seu conteúdo e a própria grelha (habitação). Basicamente, apaga todo o conteúdo do simulador.

znova <linha> <coluna>

Cria uma nova zona na grelha da habitação na posição linha,coluna indicada. A posição indicada terá que estar vazia.

zrem <ID zona>

Apaga a zona com o ID indicado e todo o seu conteúdo.

zlista

Lista todas as zonas da habitação. Para cada zona apresenta o ID e o número de sensores, o número de processadores, e o número de aparelhos nela existentes.

>> Comandos para gerir zonas e seu conteúdo

zcomp <ID zona>

Lista os componentes existentes na zona com o ID indicado. Cada componente deve incluir: o tipo (sensor, processador ou aparelho), através de uma letra "s, p ou a", seguido do seu ID numérico, seguido do seu nome (aquecedor, lâmpada, processador, etc), e do seu estado (se for um sensor o valor atual da leitura, se for um aparelho, do último comando recebido, um processador indica apenas o número de regras que tem).

zprops <ID zona>

Lista as propriedades reconhecidas pelo simulador na zona com o ID indicado. Cada propriedade é descrita pelo seu nome e o seu valor atual.

pmod <ID zona> <nome> <valor>

Modifica o valor da propriedade cujo nome é indicado (se existir tal propriedade) na zona com o ID indicado (se existir) para o valor especificado (se for aceitável). O comando informa se a operação correu bem (tal como todos os outros comandos).

cnovo <ID zona> <s | p | a> <tipo | comando>

Adiciona um sensor (s) / processador (p) / aparelho (a) do tipo indicado à zona que tem o ID indicado. "tipo" refere-se ao tipo de sensor (sensor de quê) ou aparelho (que aparelho). No caso de processador, só há um tipo de processador e a palavra que é escrita no lugar do tipo significa o comando a enviar quando as regras são todas avaliadas como verdadeiro (o comando é qualquer coisa que os aparelhos que estão ligado à saída do processador possam entender, mas geralmente será ou "liga" ou "desliga" (poderão ser outros se forem implementadas classes de aparelhos que os reconheçam, por ex. aumentar o volume de uma televisão). O ID do componente é determinado pelo simulador e é indicado no resultado deste comando. O utilizador deve apontar esse ID pois vai ser necessário para interagir com esse componente mais tarde. Os "tipos" de sensor/aparelho devem ser os usados nas respetivas tabelas indicadas acima e devem ser reconhecidos quando escritos em minúsculas. Deve usar nomes constituídos por uma única palavra.

crem <ID zona> <s | p | a> <ID>

Remove o componente com ID indicado da zona especificada. Dado que os ID dos componentes são únicos, bastaria indicar o ID do componente e o programa procurava em todas as zonas. Para simplificar o programa, indica-se também o ID da zona.
Importante: no caso do componente estar a ser referido por outro componente (por exemplo, um sensor que é usado por uma ou mais regras), deve ser garantida a coerência da ação: ou recusa a remover o componente, ou apaga "em cascata" todos os componentes que o usam (e depois os que usam estes últimos e por aí em diante), ou de alguma forma o componente que usa fica a saber que já não pode contar com o componente que foi apagado. Seja como for, o simulador tem que permanecer coerente.

>> Comandos para processadores de regras

Nota: nestes comandos, o ID da zona é sempre especificado para que o programa possa ser simplificado, procurando os componentes envolvidos diretamente na zona indicada.

rnova <ID zona> <ID proc. regras> <regra> <ID sensor> [param1] [param2] [...]

Cria uma nova regra do tipo "regra" (ver tabela de regras) e adiciona-se logo ao processador cujo ID é indicado. A regra fica associada ao sensor ID_sensor. Consoante a regra que foi criada, podem ser precisos um ou mais parâmetros que são indicados no fim do comando. Apresenta o ID que foi gerado para a regra. As regras têm um ID próprio, independente de todos os outros elementos do simulador e trata-se de um número crescente gerido automaticamente.

pmuda <ID zona> <ID proc. regras> <novo comando>

Muda o comando do processador indicado, na zona especificada**.** O comando é uma palavra.

rlista <ID zona> <ID proc. regras>

Lista as regras do processador de regras, uma por linha, indicando o seu nome, o seu ID e o nome e ID do sensor associado.

rrem <ID zona> <ID proc. regras> <ID regra>

Remove regra indicada do processador de regras especificado.
Nota: não existem comandos para modificar regras. Essa funcionalidade é conseguida removendo uma regra e voltando a acrescentá-la com a diferença pretendida.

asoc <ID zona> <ID proc. regras> <ID aparelho>

Estabelece a associação entre a saída do processador indicado e o aparelho especificado.

ades <ID zona> <ID proc. regras> <ID aparelho>

Remove a associação entre a saída do processador indicado e o aparelho especificado.

>> Comandos para interagir com aparelhos

acom <ID zona> <ID aparelho> <comando>

Envia "manualmente" o comando indicado ao aparelho cujo ID foi especificado.

>> Comandos para cópia/recuperação de processadores de regras

psalva <ID zona> <ID proc. regras> <nome>

Salva o estado de um processador de regras e de tudo o que lhe estiver associado para memória. Fica armazenada nessa memória cópia de tudo aquilo que pertence ao processador, e não fica armazenado aquilo que não pertence (exemplo: as regras pertencem, os sensores não). A informação armazenada inclui o ID da zona para que se saiba mais tarde de onde veio o processador. A cópia do processador fica associada a um nome que mais tarde é usado para repor o processador. O nome deve ser único.

prepoe <nome>

Repõe o processador previamente gravado em memória com o nome indicado. O processador é reposto na zona em que estava inicialmente, exceto se essa zona tiver sido, entretanto, apagada. O processador atualmente existente com esse ID, se ainda existir, é substituído pela cópia restaurada (se já não existir, funciona como uma espécie de acrescentar processador). Poderá ser necessário verificar e ajustar o conteúdo do processador restaurado à nova realidade da zona dado que esta pode ter mudado desde que o processador foi salvaguardado.

prem <nome>

Apaga uma cópia do processador de regras armazenado em memória.

plista

Apresenta uma lista de cópias de processadores de regras salvaguardados em memória. Na lista apresentada deve fazer parte o nome, o ID do processador e o ID da zona de onde veio.

Nota: a funcionalidade de salvaguarda/recuperação de cópias de processadores deve ser cuidadosamente pensada. Pode haver várias alternativas: mantenha-se nas aulas e em contacto com os professores.

>> Comandos adicionais de carácter geral

exec <nome de ficheiro>

Carrega um ficheiro de texto com comandos e executa-os. O ficheiro tem um comando por linha. Os comandos devem ser executados tal como se fossem introduzidos pelo teclado pelo utilizador, ou seja, têm o mesmo formato e sintaxe dos comandos indicados nesta seção do enunciado (inclusivamente, um comando num ficheiro até poderá carregar comandos de um outro ficheiro). Este comando é muito importante pois permite ter vários comandos previamente gravados em ficheiro, acelerando muito o processo de teste do programa.

sair

Encerrar o programa. Isto implica uma saída "limpa", ou seja, libertar todos os recursos alocados.

Restrições quanto à implementação

- O programa deve compilar sem nenhum *warning* ou erro. O programa deve executar sem nenhum erro ou exceção. O código deve ser robusto e completo.
- É esperado um programa orientado a objetos em C++. Uma solução não usando os conceitos de objetos (por exemplo, na lógica C) terá 0 ou muito próximo disso.

Há mais do que uma estratégia de implementação possível.

Acerca do enunciado

- É inevitavelmente longo dado que a) é necessário descrever características dos elementos do simulador, b) tenta responder de antemão a questões que normalmente surgem. Extensão de texto não se traduz necessariamente em complexidade ou trabalho adicional.
- Deduzem-se do texto algumas classes necessárias para as entidades mais salientes, mas poderão ser necessárias outras classes.
- São omitidos intencionalmente alguns assuntos. O enunciado age como uma situação representativa de cenários de indústria em que existem pormenores que devem ser deduzidos e completados por quem executa o trabalho. Estes aspetos devem ser abordados segundo o bom senso e sem remover dimensão, matéria ou complexidade ao enunciado (em caso de dúvida convém perguntar ao professor atempadamente). Não será aceitável não concretizar algo cuja necessidade é óbvia apenas porque não foi explicitamente pedido no enunciado. Estes aspetos em aberto incluem:
    - Funcionalidades cuja necessidade se torna óbvia por outras que são mencionadas;
    - Aspetos de implementação que envolvam o uso correto dos mecanismos e semântica de C++;
    - Aspetos de organização segundo as boas práticas de programação e do paradigma de orientação a objetos.

Regras gerais do trabalho

As que estão descritas na FUC e nos *slides* da primeira aula teórica:

- Grupos de dois alunos no máximo. Grupos de 3 não são aceites. Grupos de 1 devem ser previamente justificados junto do professor.
- Em ambas as metas entregar: projeto CLion e relatório num arquivo zip com o nome indicado mais adiante.
- Defesa obrigatória em ambas as metas. Ambos os alunos devem comparecer ao mesmo tempo e quem faltar fica sem nota.
- A defesa afeta bastante a nota, tem carácter individual e os alunos do mesmo grupo podem ter notas diferentes.

Metas e entregas

Meta 1 – 25 de novembro

Requisitos - funcionalidades para a meta 1:

- Leitura dos comandos e respetiva validação (mesmo que não façam ainda nada, devem ser validados quanto à sintaxe e parâmetros).
- Leitura do ficheiro de comandos.
- Construção parcial da interface com o utilizador. Pretende-se com isto garantir que os alunos trabalham atempadamente com a biblioteca (fornecida) para lidar com a consola.
- Construção de classes (ainda esvaziadas) para cada um dos tipos genéricos de componentes e de uma propriedade genérica. Podem ser especificadas apenas pelo .h.
- Construção da classe para as zonas.
- Construção de classes para o Processador e para uma regra genérica (mais tarde é melhorada).
- O projeto já deverá estar devidamente organizado em .h e .cpp.

Relatório – Nesta meta o relatório é simplificado, mas deve incluir a descrição das opções tomadas e a descrição das estruturas usadas. Deve também dar uma indicação da estruturação do trabalho em termos de classes (quais, para que servem / o que representam e qual a relação entre elas).

-> **Arquivo** a entregar: arquivo **zip** (não é rar nem arj - **é zip**) com o **nome obrigatório** seguinte:

poo_2324_**m1**_*nome1_numero1_nome2_numero2*.**zip**

Meta 2 – 30 de dezembro

Requisitos: programa completo, com relatório detalhado e completo. No relatório deve incluir uma lista com os requisitos implementados, parcialmente implementados e não implementados (com indicação da razão dos não implementados).
-> **Arquivo** zip com o nome: poo_2324_**m2**_*nome1_numero1_nome2_numero2*.**zip**

Em ambas as entregas:

- **Apenas um dos alunos** do grupo faz a submissão e **associa obrigatoriamente a submissão ao colega de grupo.**
- **Os alunos têm de estar obrigatoriamente inscritos numa turma (podem ser turmas diferentes no mesmo grupo).**