INF 213 - Roteiro da Aula Pratica 3

Arquivos fonte e diagramas utilizados nesta aula:
https://drive.google.com/open?id=1cdtIGVoY-m4ZuhpJE_EWEwnLF2b1VWTr

Etapa 1

Considere a pasta FigGeometricasSemPolimorfismo. Compile e execute o programa Teste1Aula3.cpp e analise o resultado (no final do arquivo Teste1Aula3.cpp). Verifique porque, nas 3 chamadas da funçãop->imprime(), a função executada sempre foi a função imprime() da classe FigBase embora p estivesse apontando para objetos de diferentes classes derivadas.

Agora, modifique as classes FigBase, Retangulo, Circulo e Segmento para que a função imprime passe a ter comportamento polimórfico. Teste as suas classes (novamente) com o MESMO programa Teste1Aula3.cpp e compare o resultado com o listado no arquivo Resultado-Teste1Aula3-com-Polimorfimo.pdf

Etapa 2

Modifique as classes FigBase, Retangulo, Circulo e Segmento para que as funções perímetro e area também passem a ter comportamento polimórfico. Teste as suas classes compilando e executando o programa ImpPerimArea.cpp – compare o resultado obtido com o resultado esperado listado no final do arquivo ImpPerimArea.cpp.

Etapa 3

Faça um programa (com nome Etapa3.cpp) que permita armazenar figuras (círculos, retângulos e segmentos) em um ÚNICO array. O programa deve criar dois círculos, um retângulo e dois segmentos (sendo que todos devem ter atributos com valores default) e estes objetos devem ser armazenados num ÚNICO vetor. Depois, o programa deve percorrer o array solicitando que o usuário forneça os valores para os atributos de cada figura (ou seja, o programa deve percorrer o array chamando o operador de entrada >> para cada item do array). Após fornecidas as figuras, o programa deve percorrer o array imprimindo os dados das figuras.

OBS: você deve alterar as implementações das classes para que os operadores de entrada >> e saída << passem a ter comportamento polimórfico. Dica: os operadores << e >> não pertencem às classes desenvolvidas nesta aula (eles pertencem às classes de “stream” do C++) e, dessa forma, uma maneira de implementar o comportamento polimórfico consiste em implementar funções auxiliares de entrada e saída (exemplo: “le”, “imprime”) que pertencam às classes representando as figuras geométricas e chamar tais funcoes a partir dos operadores.

Etapa 4

Como visto em sala, a linguagem C++ possui o comando *dynamic_cast,* que permite fazer a conversão (dinâmica) entre objetos numa hierarquia de classes e também há a biblioteca

typeinfo que inclui os comandos *typeid* e *typeinfo* para obter informações sobre o tipo dos objetos (veja o material sobre polimorfismo no link Aulas Teóricas do PVANet).
• dynamic_cast: Converte um ponteiro de classe básica em um ponteiro de classe derivada. Se um objeto subjacente for do tipo derivado, será executada a coerção. Do contrário, será atribuído 0.
• typeid: Retorna uma referência a um objeto da classe type_info que contém informações sobre o tipo de seu operando. A função-membro name retorna uma string baseada em ponteiro (char *) que contém o nome do tipo do argumento passado para typeid (porém, não há garantias sobre a qualidade do resultado dessa função!). Veja um exemplo nos slides 99 a 102.

Os arquivos ExemplosUsoTypeId.zip e ExemplosAdicionaisTypeid.zip incluem exemplos do uso da classe typeinfo e o comando dynamic_cast.

Nos exercícios das letras A e B você deve usar os operadores e funções da classe typeinfo e o comando dynamic_cast.

A. Faça um novo programa baseado no programa ImpPerimArea.cpp (do exercício 2) que, para cada objeto, imprima também o tipo do objeto. Isto é, a saída do programa deve ser similar à listada no arquivo ImpPerimAreaComTipo.pdf
B. Faça um novo programa estendendo o programa do exercício 3 que, após imprimir os dados das figuras, dobre de tamanho o raio de todos os círculos e imprima os novos dados desse círculo. OBS: Você não pode supor que você sabe em que posições do vetor os círculos estão; você deve ler os objetos do vetor e caso o objeto seja um círculo fazer a alteração solicitada e imprimir os dados do círculo alterado.

Submissao da aula pratica:

A solucao (todos arquivos) deve ser submetida ate as 18 horas da proxima Segunda-Feira utilizando o sistema submitty (submitty.dpi.ufv.br). Nesta aula não ha exigencias rigidas nem sobre o formato da entrada/saida nem sobre os nomes dos arquivos implementados em cada etapa. Atualmente a submissao so pode ser realizada dentro da rede da UFV.