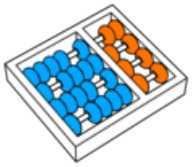

Universidade Estadual de Campinas
Instituto de Computação

Prof. Dr. Bruno Barbieri de Pontes Cafeo
cafeo@ic.unicamp.br
https://ic.unicamp.br/~cafeo/

UNICAMP

INF1900 - Recursos Avançados de C++ (2023)
Exercício 5 - Explorando o Desenvolvimento de Aplicações com ATL e MFC

Contexto

Neste trabalho, você estará mergulhando no mundo do ATL (Active Template Library) e do MFC (Microsoft Foundation Classes).

O exercício é formado por duas partes, ou seja, dois projetos distintos. Na parte 1, o objetivo principal é desenvolver uma aplicação que utiliza a ATL para programar um aproximador de π. Na parte 23 o objetivo é utilizar o mesmo enunciado do exercício 4, mas agora utilizando o MFC.

Parte 1 - ATL: Desenvolvendo um aproximador de π

Você será responsável por desenvolver uma classe chamada Calculator utilizando ATL. Esta classe será responsável por calcular uma aproximação do número π.

1. **Definição da IDL - Calculator.idl**: No arquivo Calculator.idl, defina a interface para a sua classe. Essa interface COM, que pode ser nomeada como ICalculator, deve expor um método chamado **ApproximatePi**. Este retornará um double representando a aproximação de π. Utilize as diretivas apropriadas do IDL para garantir que a interface seja registrada corretamente e que possa ser acessada por outros componentes COM.

2. **Implementação do Header - Calculator.h**: Comece incluindo as diretivas e bibliotecas necessárias para a operação com ATL. A classe Calculator deve ser derivada de uma classe base adequada do ATL, como **CComObjectRootEx** e **CComCoClass**. Implemente as macros necessárias para registrar a classe como um componente COM, como **DECLARE_REGISTRY_RESOURCEID(IDR_CALCULATOR)**. Declare o método **ApproximatePi** e qualquer outro método necessário para a operação COM.

3. **Implementação da Classe - Calculator.cpp** : No arquivo Calculator.cpp, você precisará implementar a lógica para calcular uma aproximação do número π. Uma maneira comum de fazer isso é utilizando o método de Monte Carlo, que se baseia em simulações aleatórias. Para este método:

   (a) Imagine um círculo inscrito em um quadrado. O lado do quadrado tem comprimento 2 e o círculo tem raio 1.

   (b) Gere pontos aleatórios dentro do quadrado.

   (c) Determine a proporção de pontos que caem dentro do círculo.

   (d) Usando essa proporção, você pode estimar o valor de π como:

$$\pi \approx 4 \times \frac{\text{número de pontos dentro do círculo}}{\text{número total de pontos}}$$

4. **Cliente - Cliente.cpp**: Finalmente, em Cliente.cpp, você criará um cliente simples para testar sua classe Calculator

MFC: Refazendo o Exercício 4 com a utilização de MFC

Agora, você irá refazer o exercício feito com a Windows API, mas utilizando o MFC.

1. Comece pela criação da janela principal. Utilize a classe **CFrameWndo** ou uma derivada dela. Dentro desta janela, adicione um controle **CEdit** para que os usuários possam inserir uma string, bem como um botão usando a classe **CButton**

2. Aproveite a arquitetura de manipulação de mensagens do MFC, evitando a necessidade de implementar manualmente a função WndProc. Para estabelecer mapeamentos entre mensagens e funções de manipulação, utilize as macros **BEGIN_MESSAGE_MAP** e **END_MESSAGE_MAP**.

3. Configure os manipuladores de eventos para os controles. Isso pode ser feito com o auxílio do Class Wizard ou adicionando manualmente. Associe, por exemplo, o evento de clique do botão a um método específico, como **OnButtonClicked**.

4. Na ação do botão, recupere o texto do controle **CEdit** e use **AfxMessageBox** para exibir o conteúdo em uma MessageBox.