Residência em Software

Persistência de Dados

Professores:

Álvaro Coelho, Edgar Alexander, Esbel Valero e Hélder Almeida

INSTITUIÇÃO EXECUTORA cepedi UESC

COORDENADORA MCTI FUTURO Softex

APOIO MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Residência em Software

Objetivo

O objetivo desta aula é capacitar os alunos a compreenderem e aplicarem os conceitos de persistência de dados em C++. Permitindo, a partir disso, desenvolvimento de soluções envolvendo maior volume de dados e complexidade.

Residência em Software

Relembrando métodos estáticos

São funções de uma classe que podem ser chamadas sem criar uma instância da classe. Eles são usados para executar ações que não dependem do estado interno de objetos e são comuns para todas as instâncias da classe. São úteis para operações utilitárias, como cálculos independentes de instância, acesso a configurações globais e implementação de funcionalidades que não precisam de contexto de objeto.

Residência em Software

Introdução

- O que é persistência de dados?
  - Refere-se à capacidade de armazenar informações de forma duradoura e acessível, permitindo que os dados sobrevivam além da execução temporária de um programa ou aplicativo.

- Qual a importância de armazenar e recuperar dados de forma eficiente?
  - É fundamental para a otimização de aplicativos, a satisfação do usuário, a economia de recursos e a confiabilidade do sistema. É um aspecto crítico do desenvolvimento de software que deve ser considerado desde o início de um projeto.

Residência em Software

Manipulação de arquivos em C++

A manipulação de arquivos em C++ é realizada por meio da biblioteca padrão **fstream** e suas classes e funções específicas.

A biblioteca padrão de C++ oferece um conjunto rico de classes e funções para manipulação de arquivos.

As operações de arquivo são suportadas através de objetos de fluxo (streams) que permitem a leitura e escrita de dados em arquivos.

Residência em Software

Tipos de Fluxos de Arquivo

- **ifstream**: Fluxo de entrada usado para leitura de arquivos.
- **ofstream**: Fluxo de saída usado para escrita em arquivos.
- **fstream**: Fluxo que pode ser usado tanto para leitura quanto para escrita.

```
// Abrindo um arquivo para escrita
ofstream arquivo_saida("exemplo.txt");
```

```
// Abrindo o arquivo para leitura
ifstream arquivo_entrada("exemplo.txt");
```

Residência em Software

Abertura e fechamento de arquivos

- A função **open()** é usada para abrir um arquivo em um modo específico (leitura, escrita, etc.).

```
std::ofstream arquivo_saida;
arquivo_saida.open(nomeArquivo);
```

- A função **close()** é usada para fechar um arquivo após a manipulação.

```
arquivo_saida.close();
```

Residência em Software

Leitura de arquivos

- **getline()**: lê uma linha inteira de um arquivo de texto;
- **>>**: é usado para ler dados em um formato mais amigável para humanos, como inteiros, ponto flutuante, caracteres e strings, a partir de um arquivo de texto. Ele lê dados token por token, separados por espaços em branco.
- **read()**: é usada para ler dados em formato binário. Ela lê uma quantidade específica de bytes diretamente do arquivo, independentemente de seu conteúdo. Geralmente é usada para ler tipos de dados personalizados ou estruturas de dados sem formatação específica.

Residência em Software

Escrita em arquivos

- **<<**: é usado para escrever dados em um formato mais legível para humanos, como inteiros, ponto flutuante, strings, etc. Ele converte os dados em uma representação de texto antes de escrevê-los no arquivo. É ideal para a escrita em arquivos de texto.
- **write()**: é usada para escrever dados em formato binário, sem fazer qualquer conversão de formato. Ela grava os dados exatamente como são, em sua representação de bytes. É adequada para a escrita de dados em formato binário ou quando você precisa controlar o formato exato dos dados no arquivo.

Outras funções

- **eof()**: verifica se o final do arquivo foi alcançado durante a leitura;
- **is_open()**: verifica se um arquivo foi aberto com sucesso;

```
// Abrindo o arquivo para leitura
ifstream arquivo_entrada("exemplo.txt");

// Verifica se o arquivo foi aberto com sucesso
if (arquivo_entrada.is_open()) {
    string linha;
   while (arquivo_entrada.eof() == false) {
        getline(arquivo_entrada, linha);
        cout << linha << endl;
    }
    arquivo_entrada.close();
} else {
    cout << "Erro ao abrir o arquivo para leitura." << endl;
}
```

- **seekg()**: Move o ponteiro de leitura para uma posição específica em um arquivo.
- **tellg()**: Retorna a posição atual do ponteiro de leitura.

```
std::ifstream arquivo("dados_binarios.dat", std::ios::binary);
arquivo.seekg(0, std::ios::end);
std::streamsize tamanho = arquivo.tellg();
arquivo.seekg(0, std::ios::beg);
```

Documentação é sua melhor amiga!!!!

https://devdocs.io/

https://cplusplus.com/doc/

Residência em Software

Primeiro exemplo…

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main(){
    // Abrindo um arquivo para escrita
    ofstream arquivo_saida;

    arquivo_saida.open("exemplo.txt",ios_base::out);

    // Verifica se o arquivo foi aberto com sucesso
    if (arquivo_saida.is_open()) {
        arquivo_saida << "Escrevendo no arquivo." << endl;
        arquivo_saida << 20+30 << endl;
        arquivo_saida.close();
    } else {
        cout << "Erro ao abrir o arquivo." << endl;
    }
```

```cpp
    // Abrindo o arquivo para leitura
    ifstream arquivo_entrada;
    arquivo_entrada.open("exemplo.txt",ios_base::in);

    // Verifica se o arquivo foi aberto com sucesso
    if (arquivo_entrada.is_open()) {
        string linha;
       while (arquivo_entrada.eof() == false) {
            getline(arquivo_entrada, linha);
            cout << linha << endl;
        }
        arquivo_entrada.close();
    } else {
        cout << "Erro ao abrir o arquivo." << endl;
    }

    return 0;
}
```

Residência em Software

Exercício

Agora modificar o exemplo anterior para encapsular o acesso ao arquivo através de uma classe chamada “BancoDeDados”.

Criar métodos estáticos para salvar e recuperar os dados do arquivo.

A assinatura do método para salvar os dados no arquivo deve ser:

```
static void salvarDados(vector<string> dados)
```

Já o método para recuperar os dados:

```
static vector<string> recuperarDados()
```

Residência em Software

Exercício extra: Lista de alunos e notas

Implemente uma solução em C++ para que os professores da Residencia gerenciem suas turmas.

Seu programa deve ser capaz de inserir o nome, email e duas notas (0 a 100) para cada discente.

Seu programa deve permitir inserir, alterar ou excluir os dados.

Seu programa deve salvar e carregar os dados em arquivo(s).

Seu programa deve implementar as classes: Aluno e BancoDeDados.