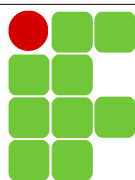


<div></div> <div><div>INSTITUTO FEDERAL DO NORTE DE MINAS GERAIS</div><div>Campus Montes Claros</div><div>Bacharelado em Ciência da Computação- 5º Período</div></div>				
Disciplina: OSA	Atividade: Lab01	Professor: Wagner Ferreira de Barros	Valor: 2pts	Nota:
Data: 04/09/2025	Aluno(a):			

INSTRUÇÕES	
<ul style="list-style-type: none"> Esta prática de laboratório é uma revisão de conceitos fundamentais e introdução a padrões de organização de projetos. Deverá ser feito em C/C++. A entrega deve ser feita em um arquivo ÚNICO compactado (contendo APENAS o <code>Makefile</code> e os arquivos fonte <code>.cpp</code> e <code>.h</code> necessários para compilação) e nomeado com o seu nome e sobrenome conforme descrito abaixo: <code>NomeSobrenome.zip</code> 	

Makefiles e Manipulação de Arquivos

Objetivo

O objetivo desta prática é duplo: primeiramente, revisar e aplicar o uso de `Makefiles` para a automação da compilação de projetos em C++. Em segundo lugar, exercitar a leitura e escrita de dados em arquivos de texto com diferentes formatos, reforçando a importância da modularização do código e da utilização de estruturas de dados adequadas para representar a informação.

Requisitos Gerais

Os seguintes requisitos são aplicáveis a todas as tarefas descritas abaixo.

- **Uso de Makefile:** Todo o processo de compilação deve ser automatizado por um `Makefile`. O arquivo deve conter, no mínimo, os alvos (targets) `all` (para compilar o projeto), `run` (para executar) e `clean` (para remover os arquivos gerados).
- **Modularização do Código:** O código deve ser organizado de forma modular. Separe a lógica em diferentes funções e, quando apropriado, em múltiplos arquivos de código-fonte (`.cpp`) e cabeçalho (`.h`). O arquivo `main.cpp` deve conter apenas a lógica principal do programa (chamada de funções).
- **Padrões de Codificação:** Mantenha o código limpo, bem comentado e utilize nomes de variáveis e funções que sejam significativos e claros.

Tarefa 1: Leitura de Arquivo com Registros Simples

Nesta primeira tarefa, você irá implementar um programa que lê um arquivo de texto contendo um único registro por linha (o nome de uma pessoa).

Descrição

O programa deverá ler todos os nomes de um arquivo de entrada e armazená-los em um vetor ou outra estrutura de dados em memória. Em seguida, deverá imprimir na tela e em um novo arquivo a lista de nomes que foi lida.

Formato do Arquivo de Entrada (`nomes.txt`)

Crie um arquivo `nomes.txt` com o seguinte formato:

`nomes.txt`

```
1 Maria
2 Joao
3 Ana
4 Pedro
5 Sofia
```

Requisitos

1. Implemente uma função para ler o arquivo e popular um `std::vector<std::string>`.
2. Implemente uma função para exibir na tela, em saída formatada, os nomes armazenados no vetor. Use funções da biblioteca `io manip` como o `setw` e o `setfill`, por exemplo, para formatar corretamente sua saída.
3. Implemente uma função que receba o vetor com os dados lidos do arquivo de entrada e um nome para o arquivo de saída, e salve os dados lidos no mesmo formato do arquivo de entrada, ou seja, um nome por linha.
4. O programa principal (`main`) deve orquestrar a chamada a essas funções.
5. Compile o projeto utilizando o `Makefile` criado.

Tarefa 2: Leitura de Arquivo CSV com Registros Estruturados

A segunda tarefa aumenta um pouco a complexidade. O programa deverá ler um arquivo no formato CSV (valores separados por vírgula), onde cada linha representa um registro com apenas dois campos: nome e idade.

Descrição

Para essa tarefa, será necessário criar uma `struct` ou `class` para representar cada registro (Pessoa). O programa deverá ler o arquivo, “quebrar” cada linha no caractere de vírgula para extrair os dois campos, e armazenar uma coleção de registros em memória. Ao final, deverá exibir os dados de forma organizada.

Estrutura de Dados

É obrigatório o uso de uma estrutura para agrupar os dados de cada pessoa. Por exemplo:

Exemplo de struct em C++

```
1 struct Pessoa {  
2     std::string nome;  
3     int idade;  
4 };
```

Formato do Arquivo de Entrada (dados.csv)

Serão disponibilizados arquivos com 100 registros em formato CSV, como no exemplo a seguir:

dados.csv

```
1 name, age  
2 Maria,34  
3 Joao,25  
4 Ana,41  
5 Pedro,19  
6 Sofia,28
```

Requisitos

Prossiga de forma semelhante à Tarefa 1:

1. Defina uma `struct` ou `class` chamada `Pessoa` para armazenar o nome e a idade.
2. Implemente uma função que leia o arquivo `dados.csv`, faça o *parsing* (análise e separação) de cada linha e retorne um vetor de `Pessoa` (`std::vector<Pessoa>`).
3. Implemente uma função que receba o vetor de pessoas e exiba seus dados de forma formatada no console.
4. Implemente uma função que receba o vetor de `Pessoas` e um nome para o arquivo de saída, e salve todos os dados lidos no mesmo formato do arquivo de entrada.
5. O programa principal (`main`) deve gerenciar a execução das funções.
6. Adapte seu `Makefile` para compilar todos os arquivos necessários para esta tarefa.

Observação sobre Arquivos CSV

Arquivos em formato CSV (*Comma-Separated Values*) são um padrão de arquivo de texto simples para troca de dados tabulares. Eles podem ser facilmente importados para visualização em processadores de planilhas (como Microsoft Excel, Google Sheets, etc.) ou manipulados em qualquer editor de texto.