Relatório Técnico: Gerenciador de Tarefas (Todolist)

Autor: Marcos Paulo Santos Bento da Silva

Curso: Introdução a Técnicas de Programação (2025.2)

Professor: Fernando Figueira

1. Introdução

Este relatório apresenta o desenvolvimento da versão inicial de um Gerenciador de Tarefas, uma aplicação de linha de comando (CLI) concebida para a disciplina de Introdução a Técnicas de Programação. O projeto, de caráter acadêmico, teve como propósito aplicar os conceitos teóricos abordados na Unidade 1 do curso.

O objetivo primordial foi criar um software funcional em Linguagem C, demonstrando o domínio sobre variáveis, operadores, vetores, estruturas condicionais, laços de repetição e o uso de funções para modularização. Adicionalmente, buscou-se desenvolver um código organizado, legível e compilável com ferramentas padrão como GCC e Make.

2. Metodologia

2.1 Ferramentas e Tecnologias

- Linguagem de Programação: O projeto foi integralmente implementado em Linguagem C, selecionada por ser o foco da disciplina e por sua eficiência e controle de recursos.
- Compilador: O GCC (GNU Compiler Collection) foi utilizado para compilar o código-fonte. A compilação incluiu as flags -Wall (para alertas de erros potenciais) e -g (para informações de depuração).
- Automação de Build: O processo de compilação foi otimizado com o utilitário make, utilizando um arquivo Makefile para seguir as boas práticas de desenvolvimento em C.
- Ambiente de Desenvolvimento (IDE): O Visual Studio Code (VS Code) foi o editor de código e ambiente de depuração escolhido, configurado com as extensões C/C++ da Microsoft.
- Controle de Versão: Para gerenciamento do código, utilizou-se o Git, com o projeto hospedado em um repositório público conforme as diretrizes do curso.

2.2 Abordagem de Desenvolvimento

Adotou-se uma abordagem de programação modular, segmentando o código em arquivos distintos para otimizar a separação de responsabilidades, conforme a estrutura de repositório proposta:

- tarefas.h: Arquivo de cabeçalho que define as declarações (protótipos) das funções e constantes, funcionando como a "interface" do módulo de tarefas.
- tarefas.c: Arquivo de implementação, contendo a lógica de cada função (e.g., adicionar, listar).
- main.c: Arquivo principal, responsável pelo controle do fluxo do programa, exibição do menu e interação com o usuário.

Essa organização visa aprimorar a manutenção e a legibilidade do código.3. Análise do Código

Esta seção detalha a aplicação dos conceitos da Unidade 1 no projeto.

- Conceitos Aplicados:
    - Variáveis e Vetores: A principal estrutura de dados é um vetor bidimensional de caracteres (char tarefas[MAX_TAREFAS][TAM_DESCRICAO]), que armazena as tarefas. Variáveis inteiras como numTarefas e opcao gerenciam o estado do programa (quantidade de tarefas e escolha do usuário).
    - Estruturas de Repetição: O laço do-while em main garante a exibição do menu ao menos uma vez antes da condição de saída ser verificada. A função listarTarefas emprega um laço for para iterar sobre o vetor de tarefas, ideal para conjuntos com número conhecido de elementos.
    - Estruturas Condicionais: A estrutura if/else if/else em main direciona o fluxo do programa conforme a opção selecionada. Adicionalmente, adicionarTarefa usa um if para validar o limite de tarefas, e listarTarefas emprega um if/else para lidar com a lista vazia, evitando a execução desnecessária do laço for.
    - Funções: A modularização foi obtida através da criação de funções com responsabilidades específicas: exibirMenu, adicionarTarefa e listarTarefas.
- Manutenção Facilitada pela Organização: A divisão do código em funções e arquivos distintos simplifica drasticamente a manutenção. Se um erro na listagem de tarefas for identificado, o desenvolvedor pode focar exclusivamente em tarefas.c e na função listarTarefas, sem necessidade de revisar o código do menu ou de outras funcionalidades em main.c. O uso do arquivo de cabeçalho tarefas.h desvincula a

implementação de sua utilização, permitindo futuras modificações na lógica interna de uma função sem exigir alterações nos arquivos que a chamam, desde que a assinatura (parâmetros e retorno) permaneça inalterada.

4. Dificuldades e Soluções

- Desafios Técnicos Enfrentados:
    1. Configuração do Ambiente de Desenvolvimento: O principal desafio inicial foi configurar o ambiente no Windows, onde o comando make não era reconhecido. A solução foi instalar o MinGW-w64 via MSYS2 e, crucialmente, adicionar o diretório bin do compilador à variável de ambiente PATH do sistema, permitindo que o terminal localizasse os executáveis gcc.exe e make.exe.
    2. Manuseio do Buffer de Entrada (stdin): Após a leitura de um número com scanf(), o caractere de nova linha (\n) permanecia no buffer de entrada, causando o ignorar de chamadas subsequentes a fgets() para leitura de strings. A solução foi "limpar" o buffer, consumindo os caracteres restantes com um laço while(getchar()!='\n'); imediatamente após o scanf().
    3. Formatação da String de Entrada: A função fgets() incluía o caractere de nova linha (\n) no final da string lida, resultando em quebras de linha indesejadas na exibição das tarefas. O problema foi resolvido utilizando strcspn() da biblioteca <string.h> para localizar o \n e substituí-lo por um terminador nulo (\0), "limpando" a string.

5. Conclusão

O desenvolvimento da versão inicial do Gerenciador de Tarefas possibilitou a aplicação prática e a consolidação dos conceitos fundamentais da linguagem C abordados na Unidade 1. Os desafios enfrentados, principalmente na configuração do ambiente e no tratamento de entrada e saída, proporcionaram um aprendizado significativo sobre as particularidades da programação em C e a importância de uma depuração sistemática.

Para futuras entregas, o projeto será expandido com novos conceitos das próximas unidades:

- Unidade 2: Substituição do vetor de char por um vetor de structs, permitindo que

cada tarefa contenha múltiplos atributos (e.g., descrição, prioridade, status).

- Unidade 3: Implementação de persistência de dados, salvando as tarefas em um arquivo de texto para que não sejam perdidas ao encerrar o programa, o que envolverá o aprendizado de manipulação de arquivos.