PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

ENGENHARIA DE COMPUTAÇÃO

GUILHERME ROELLI RA: 22899140

PEDRO RODOLFO RA: 22886287

TAYNARA ARAÚJO RA: 22904270

ESTRUTURA DE DADOS

Sistema de Gerenciamento de Tarefas

Relatório

CAMPINAS

2023

GUILHERME ROELLI

PEDRO RODOLFO

TAYNARA ARAÚJO

Sistema de Gerenciamento de Tarefas

Relatório

Trabalho de Graduação apresentado no curso de graduação Engenharia da Computação da Pontifícia Universidade Católica.

Orientadora: Prof. ª Lucia Filomena de Almeida Guimarães.
Disciplina: Estrutura de dados.

CAMPINAS

2023

SUMÁRIO

1. Introdução

Neste relatório, será mencionado como foi desenvolvido um Sistema de Gerenciamento de Tarefas para o controle as tarefas de forma simplificada. Este projeto foi desenvolvido utilizando a linguagem C e os conceitos de estrutura de dados aprendidos em aula.

1.1. Apresentação do Projeto

O projeto consiste em um sistema de gerenciamento de tarefas, com as seguintes funcionalidades:

- Criar uma tarefa;
- Listara tarefas;
- Modificar uma tarefa;
- Concluir uma tarefa;
- Adicionar tarefa pendente;
- Remover tarefa pendente;
- Atualizar seu status;
- Listar tarefas pendentes;
- Listar tarefas concluídas.

Cada funcionalidade teve seu grau de dificuldade, colocando em prática todos os conhecimentos obtidos de Fila e Lista mencionados nas aulas de Estrutura de Dados.

2. Desenvolvimento do Projeto

A biblioteca de fila e lista disponibilizada pela professora durante as aulas foram adaptadas para este projeto, fazendo alterações necessárias conforme a progressão do sistema. Em conjunto dessa biblioteca foi criado duas novas estruturas chamadas Data, que contém três campos do tipo int para armazenar Dia/Mês/Ano e a Tarefa, que armazena as seguintes informações:

- Código da tarefa do tipo int;
- Nome da tarefa do tipo char [30];

- Nome do projeto do tipo char [30];
- Data de início do tipo Data;
- Data de término do tipo Data;
- Status da tarefa do tipo int;
- Prioridade do tipo int.

O status da tarefa define se a tarefa está atrasada ou em dia. Nesse campo é considerado três tipos de valores inteiros, que são: 1 (atrasado), 0 (em dia) e -1 (pendente). A prioridade assume 3 valores diferentes 1, 2 e 3, sendo 1 – Alta prioridade, 2 – Prioridade normal, 3 – Baixa Prioridade.

2.1. Criar uma tarefa

Foi criado uma função chamada "novaTarefa". Ela é chamada no momento que é criado um nó na fila principal para guardar a informação que é uma struct do tipo tarefa. Em conformidade com essa função, é criado mais duas funções auxiliares, que são: "DataValida" para verificar se a data é válida, e "VerificarCod" para impossibilitar a criação de tarefas com mesmo código, visto que as lógicas de busca são baseadas neste código.

Lembrando que antes de chamar a função, a tarefa é verificada em qual fila ela está devido a sua prioridade.

A dificuldade na criação dessa funcionalidade foi de 1/6, pois foi necessário fazer algumas verificações para entrada de dados e a inclusão de filas com diferentes prioridades com base na biblioteca fila disponibilizada para o projeto.

2.2. Modificar uma tarefa

Na main, a função "editar" é chamada e dentro dela é realizada uma leitura do código da tarefa que se deseje modificar. Sendo assim, o programa vai percorrer a fila de tarefas e a lista de pendentes em busca deste código. Logo após disso, a função auxiliar "editar" é chamada e dentro dela o usuário terá quatro opções de informações para alterar a tarefa escolhida:

- Nome;
- Nome do Projeto;
- Data de início;

- Data de término;

E após isso ele poderá salvar as alterações feitas ou descartá-las caso queira.

A dificuldade na criação dessa funcionalidade foi de 1/6, pois foi necessário fazer algumas buscas através da lista e da fila.

2.3. Concluir Tarefa

A função concluir tarefa foi construída através da criação de três funções, que são "removerNoFila" e "inserirNoListaConcluida". Para concluir uma tarefa foi necessário que o usuário digite o código da tarefa desejada (não é possível concluir uma tarefa pendente) para verificar se na fila possui-a alguma tarefa com o mesmo código através da função auxiliar chamada "removerNoFila". O nó da tarefa encontrado nessa função é atualizado diretamente na variável "AuxN", e ao concluí-la é atualizado a data de término da tarefa para a data atual usando algumas funções da biblioteca <time.h>. Com isso, a tarefa é removida da fila e, posteriormente, adicionada na lista ordenada pela data de término.

A dificuldade na criação dessa funcionalidade foi de 3/6, devido à necessidade de criação de funções auxiliares e sua elaboração para a execução dessa função.

2.4. Atualizar seu status

A atualização manual do status feita pelo usuário, foi apenas para selecionar se a tarefa está pendente ou não. Nesse caso, para adicionar uma tarefa como pendente, foi utilizado a mesma lógica para concluir tarefas em uma fila. Para selecionar uma tarefa da fila como pendente, a função "removerNoFila", em conjunto com as funções auxiliares, o nó é removido da fila e, logo em seguida, o status é atualizado para –1 (pendentes). Depois disso, a função "inserirNoListaPendente" insere o nó na lista ordenada de acordo com a prioridade e data término.

Para atualizar o status da tarefa pendente, foi necessário percorrer a lista de pendentes através da verificação do código digitado pelo usuário, posteriormente o nó é achado e excluído da lista. Depois da remoção do nó escolhido, o status de pendente é alterado para não pendente e depois é verificado se "está em dia" ou "em atraso".

A dificuldade na criação dessa funcionalidade foi de 6/6, pois foi demandado muito tempo para execução da lógica em conjunto com ordenação dos nós de acordo com a prioridade e data término.

2.5. Listar tarefas

Para listar tarefas pendentes foi utilizado uma função que percorre toda a lista de pendentes e imprimindo os nós em sequência. Para a lista de concluídas, foi utilizado a mesma lógica, mas dividida em 3 funções, que tem como objetivo: imprimir a lista de concluídas completa, listar as concluídas com e sem atraso. Para isso, foi utilizado um "if" para verificar o status da tarefa antes de imprimi-la.

A dificuldade na criação dessa funcionalidade foi de 1/6, pois as funções eram mais simples, visto que a lógica para a manipulação da lista já estava desenvolvida.

3. Bibliografia

- Slides de aula;
- ESTR_PUC_RIO.pdf.