Relatorio de Sistema de Biblioteca

Natalie Menato
NUSP: 10295051
October 13, 2017

1 DESCRIÇÃO DO TRABALHO

O programa desenvolvido para este trabalho é um sistema de controle para uma biblioteca que tem funcionalidade para cadastrar alunos, cadastrar livros, retirar livros, retornar livros, remover alunos, remover livros e imprimir os emails de notificação para um aluno. O sistema foi criado considerando um biblioteca com 1,000 livros e 100 alunos. O sistema precisa ter uma funcionalidade para uma lista de espera quando um aluno quer retirar um livro que não está disponível. Está estimado que o número total de estudantes em todas as listas de espera de todos os livros em qualquer momento vai ser no máximo 1,000 estudantes. Estes limites foram usado para o desenvolvimento do sistema. O sistema é implementado em duas versoões, usando duas estruturas diferentes: estática e dinâmica. A implementação das funcionalidades do menu, input e o loop do main é a mesma para as duas versões. Para mudar a versão que está sendo executado, basta mudar o include no `input.h`. Para a versão dinâmica usa: `#include "dynamic/library_dynamic.h"` e para a versão estática usa: `#include "static/library_static.h"`. O projeto tem um repositório no GitHub que pode ser acessado: https://github.com/nmeusling/scc0223_biblioteca. Também é possível ver a documentação do código gerado usando Doxygen no seguinte link: https://nmeusling.github.io/scc0223_biblioteca/files.html.

1.1 IMPLEMENTAÇÃO

O menu principal mostra todas as ações disponíveis para o usuário permitindo-o selecionar a ação desejada. Depois de selecionar a ação principal, submenus e outras opções são exibidos, permitindo o usuário executar a ação desejada. Por exemplo, quando o usuário seleciona a

ação para retirar um livro, um submenu aparece para dar a opção para buscar para o aluno que vai retirar o livro usando o nome ou número USP, e depois outro submenu aparece para escolher como buscar o livro: pelo título ou pelo ISBN. Cada campo de input tem verificaçôes para guardar contra input errado. Por exemplo, na hora de selecionar a ação do menu, se o usuário digite mais de um caráter, só o primeiro caráter vai ser considerado é o resto do input vai ser descartado. Todos os inputs do menu também tem validação para verificar que a seleção é uma opção válida no menu.

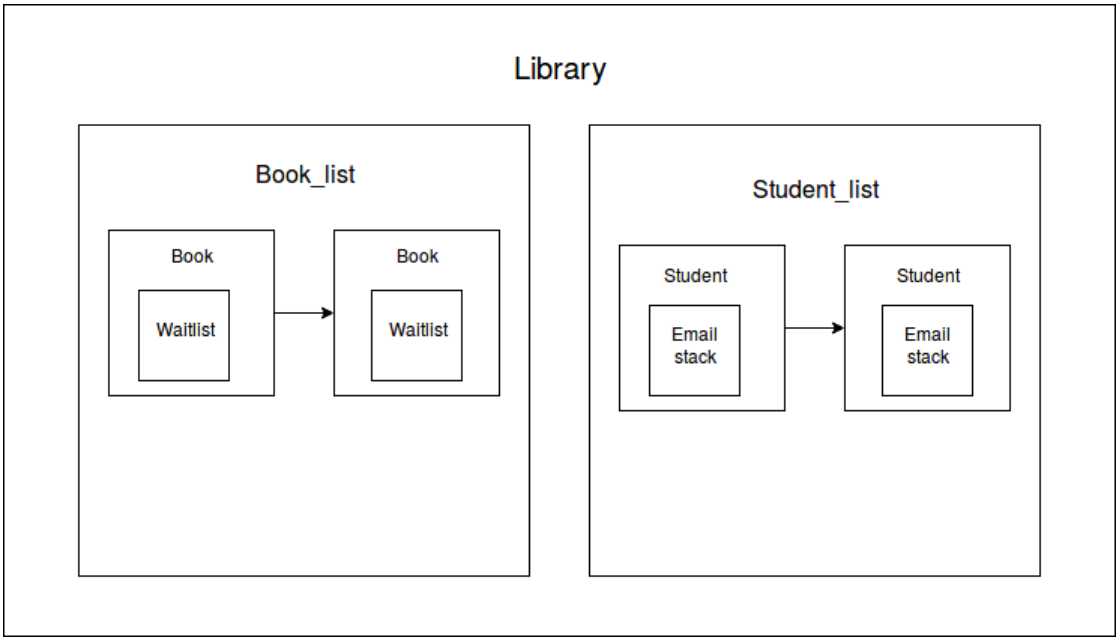
Para deixar o programa mais modularizado e organizado, está separado em vários arquivos (main, menu, input, library, book, e student). Os arquivos main, menu, e input têm uma versão só, mas os arquivos library, book e student tem duas versões: uma versão dinâmica e uma versão estática. O arquivo main contém só o loop principal do programa. O arquivo input contém todos as funções relacionadas ao input do usuário, por exemplo para pegar a seleção do usuário do menu principal. O arquivo menu tem todas as funções relacionados ao menu e controla o fluxo do programa, por exemplo se o usuário seleciona a ação 'remover aluno' o menu vai chamar a função certa para remover o aluno. O arquivo student contém o struct para o aluno, a estrutura para a lista de alunos, e a pilha para os emails do aluno. Este arquivo também tem todas as funções relacionadas, por exemplo as funções para criar uma nova lista, ou inserir um novo aluno na lista. O arquivo livro contém o struct para o livro, a estrutura para a lista de livros e a estrutura para a lista de espera além das funções relacionadas, por exemplo, criar um novo livro ou buscar um livro na lista de livros. O arquivo library contém a estrutura básica da biblioteca inteira: um struct library que contém uma lista de estudantes e uma lista de livros. Também tem as funções mais gerais e inter relacionadas que são necessárias para o funcionamento do sistema . Por exemplo, as funções para retirar ou retornar um livro fica no library.

File	**Description**
main	loop principal do programa
menu	menu e controle do programa
input	input do usuário
library	estrutura principal da biblioteca e as operações gerais
book	estruturas e operações relacionadas aos livros e lista de livros, incluindo a lista de espera
student	estruturas e operações relacionadas aos alunos e lista de alunos, incluindo a pilha de emails

2 Estruturas de Dados Utilizados

2.1 Implementação Dinâmica

Diagrama Veja o diagrama para ver a estrutura do programa.



BIBLIOTECA A estrutura principal do sistema é o struct “library”. Este struct contém a lista de estudantes e a lista de alunos que compor a biblioteca. Também contém todas a funções gerais necessárias para o sistema de biblioteca. Em geral, o programa principal vai usar as funções do "library" para controlar e manipular a biblioteca sem precisar usar as funções do aluno ou livro diretamente. Por exemplo, todas as ações principais tem uma função dentro do “library”. Isso permite que o “menu” chama as funções do “library” e as funções do “library” chamam as funções específicas de “student” ou “livro” para realizar a ação desejada. Esta estrutura ajuda a programa fica mais modularizada.

ALUNOS Cada aluno é um struct que contém todas as informações relacionados ao aluno (nome, nusp, email e telefone) e também contém um ponteiro para armazenar o próximo aluno na lista. Além do mais, cada estudante tem uma pilha de mensagens de email. A pilha na versão dinâmica é encadeada, e portanto só o espaço necessário para armazenar as mensagens vai ser usado. O espaço usado pela pilha vai diminuir e aumentar com necessidade.

Na versão dinâmica, os alunos são armazenados em uma lista dinâmica encadeada. Como a lista é dinâmica, é possível para o tamanho da lista expandir de forma que os alunos são adicionados no sistema.

A lista de alunos inclui um ponteiro para o primeiro e último aluno da lista. Como é esperado que o usuário vai precisar adicionar novos estudantes várias vezes, o número de computações vai ser reduzidos usando o ponteiro para fim. Sem este ponteiro seria necessário navegar pela lista inteira para achar o último elemento cada vez que um novo estudante foi adicionado. O espaço usado por um ponteiro adicional é relativamente pequeno e o número de operações vai ser reduzido significativamente.

Também a remoção de estudantes é facilitado porque a lista é encadeada e então, um aluno pode ser removido sem precisar relocar todos os outros elementos da lista. Estudantes sempre

estão adicionados no fim da lista e pode ser retirada de qualquer posição na lista, usando um busca para achar o aluno certo para remover. É possível buscar um aluno usando o nome ou o número USP do aluno. Uma limitação do sistema é que a busca tem que ser exato. Ou seja, não é possível buscar usando um nome ou número USP parcial, tem que ser o nome o número inteiro e exato.

Decidi não criar uma função para ordenar a lista, pois como é possível buscar usando dois campos distintos, o benefício em termos de eficiência na hora de buscar seria relativamente pequeno. A lista só pode ser ordenada usando um dos campos, então a maior eficiência só aplicará quando a busca é feito usando o mesmo campo pelo que a lista foi ordenada.

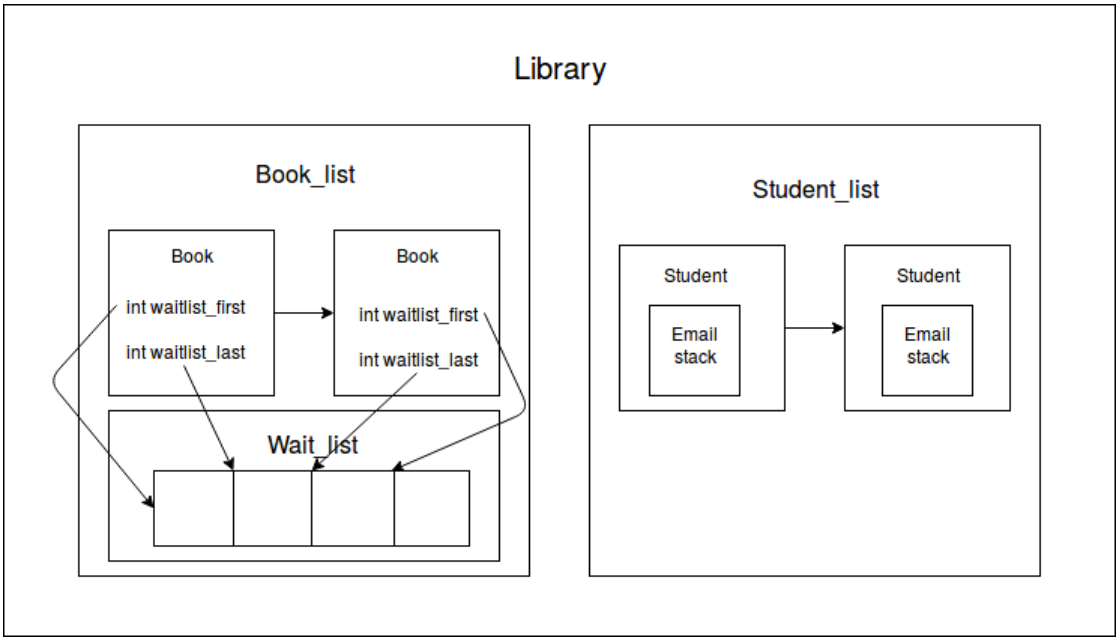
LIVROS Cada livro é um struct que contém todos os dados relacionados ao livro: (título, autor, editora, ISBN, ano, e edição). Também cada livro tem dois inteiros para armazenar o número total de cópias do livro que existe na biblioteca e o número total de cópias que estão disponíveis atualmente. Quando um novo livro é adicionado ao sistema, se o isbn é duplicado, o livro vai ser adicionado como uma cópia adicional do livro que já existe com o isbn. Neste caso, o número total e numero disponivel vão aumentar. Se um aluno retira o livro, o número total fica o mesmo, mas o número disponível diminui, porque tem uma copa menos que agora está disponível para outro estudante retirar. Cada livro também tem uma lista de espera. Veja a seção **Lista de espera** para mais detalhes.

A lista de livros é parecida à lista de alunos. A lista de livros também usa uma estrutura dinâmica e encadeada, permitindo a lista ficar o tamanho que é necessário, pois pode aumentar e diminuir quando livros são adicionados ou removidos do sistema. A busca de livros também pode ser feito usando dois campos: título ou isbn. Esta busca tem a mesma limitação que a busca de alunos - para achar o livro, o título ou isbn tem que ser exato e não é possível buscar usando um título ou ISBN parcial.

LISTA DE ESPERA A lista de espera é dinâmica e encadeada e é usado para armazenar os alunos que estão esperando para retirar um livro particular. Todo livro tem uma lista de espera, que pode ser vazia. Como a lista é dinâmica, cada livro pode ter uma lista separada sem ocupar muito memória, pois o tamanho da lista pode variar com necessidade. Quando um aluno tenta retirar um livro que não está disponível, o aluno é adicionado no fim da lista de espera para esse livro. Não é possível para um aluno entrar na lista de espera mais de uma vez e o sistema sempre verifica se um aluno já está na lista antes de adicionar. Não tem um limite no tamanho da lista de espera para a versão dinâmica porque pode crescer e diminuir com demanda. Quando um livro que tem uma lista de espera é retornado na biblioteca, um email é adicionado na pilha de emails do primeiro estudante na lista de espera para notificá-lo que o livro está disponível para o aluno retirar.

2.2 IMPLEMENTAÇÃO ESTÁTICA

DIAGRAMA Veja o diagrama para ver a estrutura do programa.



DETALHES A implementação da versão estática é bem parecida a versão dinâmica. A lista de livros e a lista de estudantes usam uma estrutura estática encadeada. Esta estrutura facilita a remoção do livros e alunos, porque um elemento pode ser retirado no meio de lista sem necessidade de reposicionar todos os elementos depois dele na lista. Os dois listas tem um tamanho máximo, baseados no requisitos do projeto e definido nos header files como um Macro para facilitar a mudança destes limites em caso de necessidade.

A pilha de emails ainda está contido dentro de struct para o aluno, mas na versão estática, a pilha é estática e sequencial, com um tamanho máximo de emails por aluno definido como um Macro. Isso significa que o espaço para a pilha com tamanho máximo sempre vai ser alocado para cada aluno, mesmo se não vai ser usado. Isso não é a implementação mais eficiente em termos de memória, mas como, não é esperado que o aluno vai ter muitas notificações, o tamanho máximo pode ser relativamente pequeno para não ocupar tanta memória. Sería possível criar um banco de memória para armazenar as mensagens para todos os alunos e só armazena o índice de início e fim para cada aluno, mas como o número de alunos e número de mensagens por aluno não é muito alto, decidi usar uma pilha simples para cada aluno. Esta implementação usa mais espaço mas também é mais simples e fácil de usar do que um banco de memória comum.

Uma outra diferença entre as duas versões é a implementação da lista de espera. Na versão dinâmica, cada livro tem uma associada lista de espera. Mas, na versão estática, tem um banco de memória para conter todos as listas de espera para todos os livros. Todo livro tem so o índice do primeiro e último elemento da lista de espera relacionado ao banco de memória. Esta implementação permite que o mesmo espaço de memória para listas de espera é compartilhada por todos os livros. Esta implementação faz sentido porque tem um número máximo estimado para quantos estudantes totais vão estar nas listas de espera em qualquer momento, mas não é possível saber quantos estudantes vão estar na lista de espera de um

livro particular. O tamanho da lista de espera pode variar bastante dependendo do livro, então não faz sentido ter uma lista de espera para cada livro porque o tamanho máximo teria que ser grande e o mesmo para cada livro, mesmo para os livros que não tem uma lista de espera.

3 AVALIAÇÃO

Uma das operações mais pesados computacionalmente é a operação para retirar um estudante de todos os wait lists, o que é feito quando um estudante é removido. A função atualmente itera sobre cada livro, e se o livro tem uma lista de espera remove todos os elementos da lista de espera, verifica se o elemento removido da lista é o aluno que vai ser removido, e se não, coloca de novo na lista. Cada vez que um estudante é removido, a função varre por todos os livros e todos os elementos da lista de espera de cada livro. Para melhorar isso, poderia tem uma estrutura relacionado a cada estudante que aponta para todos os livros que esse estudante pertence a lista de espera. Assim, não seria necessário varrer todos os livros, mas só os livros que o aluno realmente está na lista de espera. Este método poderia usar menos operações mas iria requerer mais espaço para cada nó de estudante e também a estrutura ficaria mais complexa e seria mais difícil para usar e manter.

Uma outra operação que é pesada computacionalmente é a operação para buscar um aluno ou um livro. A operação usado é de ordem n, pois a busca varre a lista até o aluno o livro desejado é achado. Para melhorar esta operação, seria necessário criar uma função para ordenar a lista e depois fazer a busca. Discuti melhor a decisão para não ordenar na seção **Alunos**.

Uma área do programa que também poderia melhorar e para adicionar uma verificação para duplicação de estudantes. Atualmente não tem, e por causa disso seria possível criar um aluno com dados duplicados de um que já existe. Neste caso os funções para retirar livro ou remover aluno vai pegar só o primeiro estudante com o campo desejado. Sería melhor tem uma validação para prevenir a criação de alunos duplicados, parecido à validação que já existe para livros.