Schedule Management System

Ano Letivo 2023/2024

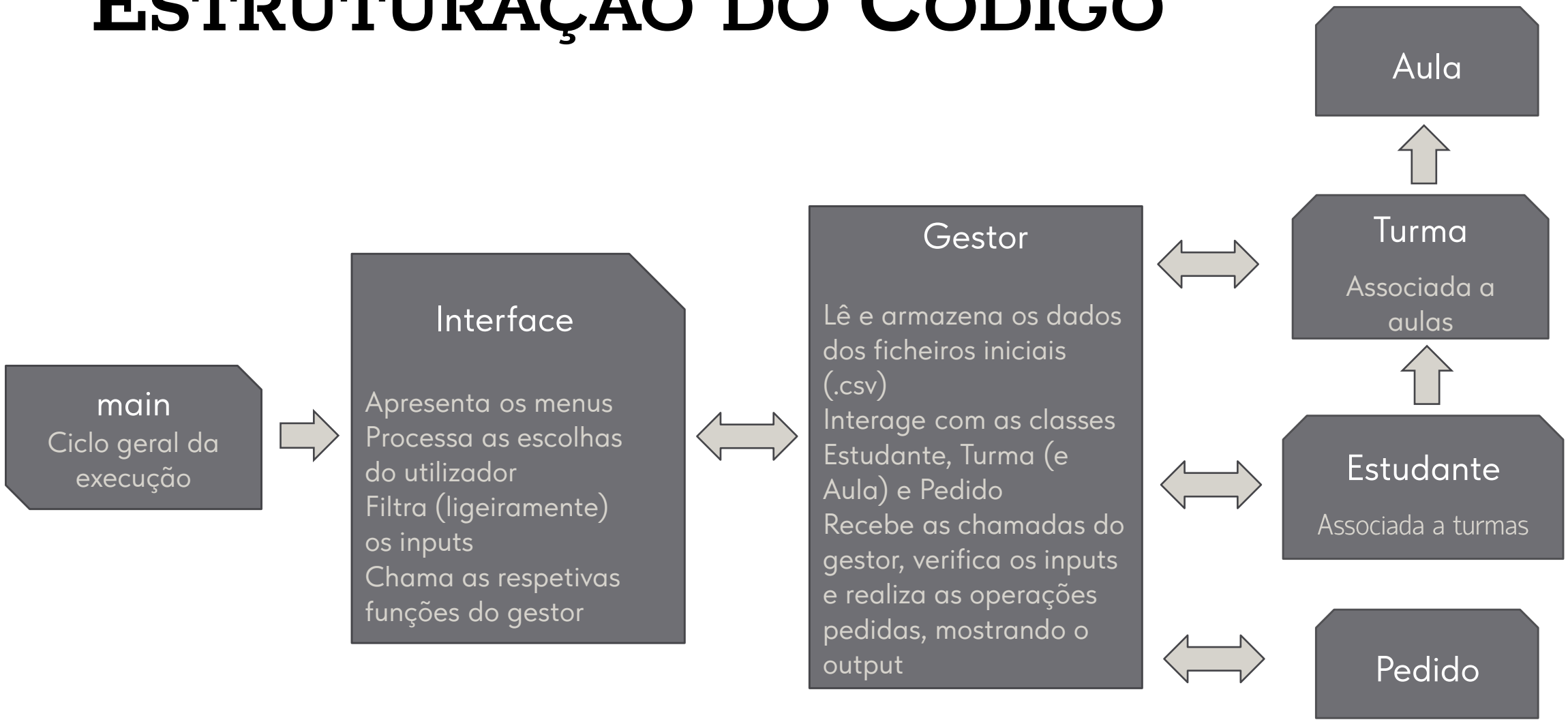
Algoritmos e Estruturas de Dados

(L.EIC011)

Turma 01, Grupo 15

Rafael Teixeira de Magalhães
Ricardo de Freitas Oliveira
Rodrigo Albergaria Coelho e Silva

Estruturação do Código



Funcionalidades Implementadas

- Leitura dos ficheiros iniciais
- Apresentação de Horários (aluno, turma, UC)
- Listagens (Estudantes (numa Turma, numa UC, num Ano), Turmas, UCs)
- Ocupações (Turmas, Ano, UC)
- Alterações/Pedidos (Remoção, Inserção, Troca, Desfazer último, Processar)
- Listagem Total (Estudantes, Turmas, Aulas)
- Guardar ficheiros alterados (novo ficheiro: updated_student_classes.csv)
- Guardar ficheiros com últimos pedidos (pedidos realizados; pedidos inválidos)
- Interface Intuitiva

Estruturas de Dados Utilizadas

Estrutura de Dados: stl::list

Ficheiro/Classes: Turma

Aplicação: Lista de Aulas

Justificação: as aulas associadas a uma determinada turma nunca vão mudar e como no máximo são 3 aulas diferentes, utilizamos uma estrutura linear fácil de aceder e percorrer/iterar em ordem

Estrutura de Dados: stl::list

Ficheiro/Classes: Estudante

Aplicação: Lista de Turmas

Justificação: um estudante está associado a um número variável de turmas [0-7]; como estas não necessitam de estar ordenadas, mas precisam de suportar fácil inserção e remoção (para realizar os pedidos), além de acesso e iteração (para apresentar output), utilizamos também uma lista

Estruturas de Dados Utilizadas

Estrutura de Dados: stl::vector

Ficheiro/Classes: Gestor

Aplicação: vetor de Turmas e Estudantes

Justificação: uma vez que vão ser guardadas todas as turmas e estudantes carregadas no sistema dentro do gestor, necessitamos de uma estrutura de dados mais versátil e que permita a realização de praticamente todo o tipo de operações (pesquisa, ordenação e acesso) e iterações (normal e invertida), assim sendo, optamos pelos vetores para estruturar o gestor

Estrutura de Dados: stl::map

Ficheiro/Classes: Gestor

Aplicação: mapas entre dias da semana e números

Justificação: para facilitar o armazenamento dos dias associados a cada aula, as aulas guardam apenas um id associado ao dia da semana ao invés de uma string; na abertura dos ficheiros há conversão de string→int e quando é para apresentar output há conversão de int→string

Estruturas de Dados Utilizadas

Estrutura de Dados: stl::queue

Ficheiro/Classes: Gestor

Aplicação: Fila de Pedidos e Pedidos Inválidos

Justificação: já que uma fila é baseada na ideia de FIFO (first-in first-out), utilizamo-la para armazenar os pedidos que são, por ordem da sua emissão, para serem processados cronologicamente; do mesmo modo os pedidos inválidos são guardados cronologicamente para no fim de execução serem guardados num ficheiro de forma sequencial

Estrutura de Dados: stl::stack

Ficheiro/Classes: Gestor

Aplicação: Pilha de Pedidos Realizados

Justificação: uma vez que uma das funcionalidades mencionadas é a de desfazer o último pedido realizado, os pedidos realizados com sucesso são colocados numa estrutura de LIFO (last-in first-out) para que seja possível aceder ao mais recente e, de seguida, desfazer a ação de acordo com o tipo de pedido em questão

Estruturas de Dados Utilizadas

Estrutura de Dados: stl::set

Ficheiro/Classes: Gestor

Aplicação: Set (BST) de Par(Aula, Turma) e Set de Estudantes

Justificação: Recorremos a árvores binárias para as funcionalidades que implicam a apresentação de dados filtrados e ordenados, uma vez que a construção destas estruturas de dados é baseada numa ordem específica, na inserção sucessiva de novos elementos e a sua iteração é apresentada por essa ordem, utilizamos esta estrutura nas funções de apresentação de horários e nas listagens de alunos

Algoritmos Usados

Em várias funções do Gestor, recorremos também ao uso de algoritmos de pesquisa e ordenação, tanto existentes na STL como implementados por nós no sentido do projeto.

Estes algoritmos foram usados para permitir uma maior otimização do código e formatação de output correta e ordenada.

Exemplos:

- Binary_search dentro do vetor de estudantes a fim de determinar o seu index, dado o código
- stl::sort aplicado ao vetor de turmas e estudantes no interior do gestor

Documentação do Código

No que toca à documentação, a mesma foi gerada usando o Doxygen, tal como era pedido na descrição do projeto. Desta faz parte:

- Descrição simples das classes usadas no projeto e da sua função
- Atributos e funções membro de todas as classes da função
- Descrição detalhada das funções e algoritmos mais importantes da classe Gestor, assim como a sua complexidade temporal associada

```
541  /**
542   * Imprime a lista de estudantes pertencentes a mais do que n UCs.
543   * Complexidade: O(n log n), sendo n o número de estudantes.
544   * @param n - Número de UCs.
545   * @param order - Ordem em qual deverão ser impressos os estudantes.
546   */
547  void Gestor::outputListaEstudanteMaisNUC(int n, int order){
```

◆outputListaEstudanteMaisNUC()

void Gestor::outputListaEstudanteMaisNUC (int n,
int order
)

Imprime a lista de estudantes pertencentes a mais do que n UCs. Complexidade: O(n log n), sendo n o número de estudantes.

Parameters

n - Número de UCs.

order - Ordem em qual deverão ser impressos os estudantes.

Conclusão

Em conclusão, podemos dizer que a realização deste projeto foi muito útil pois permitiu-nos consolidar os conhecimentos todos das aulas de Algoritmos e Estruturas de Dados, assim como trabalhar com as estruturas de dados e algoritmos já lecionados de uma maneira mais prática e aplicada à vida real.

Por ser um projeto de maior dimensão, a dificuldade é claramente um pouco maior do que os exercícios semanais, mas mesmo assim consideramo-nos bastante satisfeitos com o resultado final que apresentamos.

No final, a nossa solução tem todas as funcionalidades pedidas e ainda mais algumas implementadas e utiliza todas as estruturas de dados hierárquicas e lineares com que já trabalhamos, assim como alguns dos algoritmos de pesquisa e ordenação que já aprendemos.

Turma 01, Grupo 15

Rafael Teixeira de Magalhães
Ricardo de Freitas Oliveira
Rodrigo Albergaria Coelho e Silva