



Instituto Politécnico do Cávado e do Ave

Flexible Job Shop

Parte II

Diogo Alexandre Mendes Simões

Projecto de Licenciatura EIM

em Estruturas de Dados Avançadas

Orientador:

Docente João Carlos Silva, IPCA- Instituto Politécnico do Cávado e do Ave

Abril 2022



Flexible Job Shop, C

Diogo Alexandre Mendes Simões

Resumo

Este trabalho, teve como objetivo o desenvolvimento de aplicação em linguagem C, utilizando estruturas de dados dinâmicas.

O trabalho foi dividido em duas partes, que correspondentes a diferentes etapas: Definição de estrutura de um job e operações, Definição de um conjunto finito de jobs e inserção dos mesmos.

Na aplicação desenvolvida na segunda parte, o objetivo é definir um Txt Process Plan, Operações relacioandas a um determinado Job e a representação de uma proposta de rescalonamento FJSSP e diferentes Process Plan (Tabelas HTML), tendo que aplicar listas ligadas multidimensionais desenvolvidas em linguagem de programação C.

Palavras-Chave: Estruturas de dados; Linguagem C; Txt; Listas ligadas

Abstract

This work aimed to develop an application in C language, using dynamic data structures.

The work was divided into two parts, corresponding to different stages: Definition of a job's structure and operations, Definition of a finite set of jobs and their insertion.

In the application developed in the second part, the objective is to define a Txt Process Plan, Operations related to a certain Job and the representation of a FJSSP rescheduling proposal and different Process Plan (Table HTML), having to apply multidimensional linked lists developed in a programming language C.

Keywords: Data structures; C language; txt; linked lists

Índice

Resumo	3
Abstract.....	4
Introdução	7
Objetivos.....	8
Justificativa / Motivação.....	8
Contexto	9
Problema.....	10
Descrição da implementação	11
Bibliotecas	11
Descrição Técnica.....	11
Desenvolvimento	13
Armazenamento do ficheiro de dados	14
Funções	15
Inicialização do Programa	15
Inserir um novo Job	16
Remover um Job	16
Visualização de Jobs.....	17
Adicionar Operações em Determinados Jobs	18
Determinar a quantidade mínima de unidades de tempo necessárias para completar o job e listagem das respetivas operações	20
Determinação da quantidade máxima de unidades de tempo necessárias para completar o job e listagem das respetivas operações;	21
Determinação da quantidade média de unidades de tempo necessárias para completar uma operação, considerando todas as alternativas possíveis;	22
Conclusão	23
Bibliografia.....	24

Índice de Figuras

Figura 1-Tabela FJSSP	10
Figura 2-Estruturas	12
Figura 3-Diagrama Simplificação	13
Figura 4-Ficheiro Txt	14
Figura 5-Menu Principal	15
Figura 6-Adicionar Operação	16
Figura 7-Apagar Job Parte 1	16
Figura 8-Listar Job Parte 1	17
Figura 22-Tempo Mínimo Job	20
Figura 23-Tempo Máximo cada Job	21
Figura 24-Tempo Máximo cada Operação.....	22

Introdução

No âmbito da unidade curricular de Estruturas de dados avançadas, do segundo semestre do primeiro ano do curso de Engenharia Informática Médica do Instituto Politécnico do Cavado e do Ave, no ano letivo 2021/2022 foi-nos proposto a elaboração de um trabalho prático que está dividido em duas partes.

Este trabalho surge no contexto da avaliação contínua composta por um trabalho prático. O objetivo principal deste trabalho é levar ao melhor entendimento dos conhecimentos adquiridos ao longo do semestre e a melhor aplicação dos mesmos.

Assim, com este trabalho pretendemos sedimentar mais os nossos conhecimentos sobre os temas abordados nas respetivas aulas.

O trabalho realizado individual tem como objetivo a elaboração de um programa em C para o problema de escalonamento denominado Flexible Job Shop Problem (FJSSP).

Objetivos

Espera-se, com a execução desse projeto, sedimentar os conhecimentos relativos à definição e manipulação de estruturas de dados dinâmicos em linguagem de programação C no desenvolvimento de uma solução digital para o problema de escalonamento denominado por Flexible Job Shop Problem, permitindo gerar uma proposta de escalonamento para a produção de varios produto envolvendo várias operações e a utilização de várias máquinas.

Justificativa / Motivação

A intensa disputa por novos mercados e o apogeu das novas tecnologias tem obrigado as empresas a desenvolverem constantes estratégias que possam dinamizar seus processos internos e inovar os seus sistemas.

O Software a ser desenvolvida visa prestar suporte as organizações focadas no processo são oficinas de artesanato, restaurantes, empresas de mecânica, oficinas de pintura, gráficas comerciais e outras indústrias que fabricam produtos personalizados em pequenos lotes.

Contexto

Neste projeto que nos foi proposto, tendo origem num trabalho prático a avaliar na disciplina do 1º ano do curso LEIM, Estruturas de dados avançadas, que está dividido em duas partes, iremos abordar neste relatório a segunda parte do projeto.

Nesta parte do projeto será necessário realizar os seguintes requisitos:

- Definição de uma estrutura de dados dinâmica para representação de um conjunto finito de m jobs associando a cada job um determinado conjunto finito de operações.
- Armazenamento/leitura de ficheiro de texto com representação de um process plan (considerar obrigatoriamente para efeito de teste o process plan da Tabela 1);
- Inserção de um novo job;
- Remoção de um job;
- Inserção de uma nova operação num job;
- Remoção de uma determinada operação de um job;
- Edição das operações associadas a um job;
- Cálculo de uma proposta de escalonamento para o problema FJSSP (obrigatoriamente limitado a um tempo máximo de processamento configurável), apresentando a distribuição das operações pelas várias máquinas, minimizando o makespan (unidades de tempo necessárias para a realização de todos os jobs). A proposta de escalonamento deverá ser exportada para um ficheiro de texto possibilitando uma interpretação intuitiva (utilizar por exemplo um formato tabular ou representação gráfica html, ou outra);
- Representação de diferentes process plan (variando a quantidade de máquinas disponíveis, quantidade de job, e sequência de operações, etc) associando as respetivas propostas de escalonamento.

Problema

Foi fornecido um projeto com um problema do tipo FJSSP, que este por sua vez representa tarefas de planos de processos variáveis, também conhecidos como Empregos.

Cada plano de processo, possui um conjunto de operações que devem ser feitas por ordem, não tendo a opção de saltar operações.

Cada máquina pode realizar apenas 1 operação por vez.

Process Plan	Operation						
	0 1	0 2	0 3	0 4	0 5	0 6	0 7
pr _{1,2}	(1,3) [4,5]	(2,4) [4,5]	(3,5) [5,6]	(4,5,6,7,8) [5,5,4,5,9]			
pr _{2,2}	(1,3,5) [1,5,7]	(4,8) [5,4]	(4,6) [1,6]	(4,7,8) [4,4,7]	(4,6) [1,2]	(1,6,8) [5,6,4]	(4) [4]
pr _{3,3}	(2,3,8) [7,6,8]	(4,8) [7,7]	(3,5,7) [7,8,7]	(4,6) [7,8]	(1,2) [1,4]		
pr _{4,2}	(1,3,5) [4,3,7]	(2,8) [4,4]	(3,4,6,7) [4,5,6,7]	(5,6,8) [3,5,5]			
pr _{5,1}	(1) [3]	(2,4) [4,5]	(3,8) [4,4]	(5,6,8) [3,3,3]	(4,6) [5,4]		
pr _{6,2}	(1,2,3)	(4,5)	(3,6)				

Figura 1-Tabela FJSSP

Descrição da implementação

Para desenvolver o programa e para que este funcione corretamente, foi necessário que todos os procedimentos e funções criados interagissem de forma eficaz, sem erros e evitando situações do tipo “beco sem saída” que ocorrem quando as aplicações bloqueiam ou ficam presas em estados sem retorno. O resultado final é o (código) fonte que se encontra anexado a este relatório, sendo que nesta sessão serão explicadas algumas técnicas implementadas.

Bibliotecas

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <locale.h>
#include <ctype.h>
#include <string.h>
```

Estas instruções permitem incluir bibliotecas ao programa. Além das necessidades normais de input/output ([stdio.h](#)), alocação de memória ([stdlib.h](#)), manipulação de strings ([string.h](#)) e operações com diferentes tipos de variáveis ([ctype.h](#)), foi necessário recorrer à biblioteca [locale.h](#) para poder incluir caracteres especiais em língua portuguesa e à biblioteca ([stdbool.h](#)) recurso true e false, que serão largamente utilizados ao longo do programa.

Descrição Técnica

Para a realização deste trabalho foi necessário recorrer à criação de diversas estruturas. Uma estrutura é um conjunto de variáveis de tipos distintos ou não, agrupadas sob um único nome. As variáveis que compõem a estrutura são chamadas membros, campos ou elementos. Com a palavra-chave **struct** declara-se um novo tipo de dado. A palavra seguinte será o seu identificador ou tipo de estrutura.

Quando se faz a declaração **struct data**:

- **struct** é um tipo distinto de dado como int ou float.
- **data** é referência a um tipo específico de estrutura.

Os membros são declarados entre chaves. A declaração termina com um “ponto e vírgula” (;).

Foram criadas as seguintes estruturas:

- Job
- Maquina
- Operação

```
//-----INICIO ESTRUTURAS -----
//-----ESTRUTURA MAQUINA -----
typedef struct maquina
{
    int maquinaNum; //ID de uma determinada Maquina
    int maquinaTempo; //Tempo de uma determinada Maquina
    struct maquina *Seguinte; //Maquina Seguinte-> Seguinte
    struct maquina *Anterior; ///Maquina Anterior
} maquina; //
//-----ESTRUTURA OPERACAO -----

typedef struct operacao
{
    int operacaoNum; //Valor Calculo de uma Operacao
    int jobNum; //ID Sturct de uma Operacao
    struct maquina *MaquinaHead; //Topo do cabecalho de uma maquina
    struct operacao *Seguinte; //Operação Seguinte -> Seguinte
    struct operacao *Anterior; //Operação Anterior
} operacao; //
//-----ESTRUTURA JOB -----
typedef struct job
{
    int jobis; // Valor Calculo de uma Operacao
    int jobNum; //ID Sturct de um Job
    struct operacao *OperacaoHead; // Topo do cabecalho de uma Operacao
    struct job *Seguinte; //Proximo Job -> Seguinte
} job;
//-----FIM ESTRUTURAS-----
```

Figura 2-Estruturas

Desenvolvimento

Para a inicialização de este projeto, o primeiro a ser feito seria entender o problema em si e o que solicitava, sendo uma das partes mais complexas e demoradas.

Determinamos que um determinado Job tem varias operações que estas mesmas tem vários processos, nomeadamente dividido em Tempo e a Maquina que o executa.

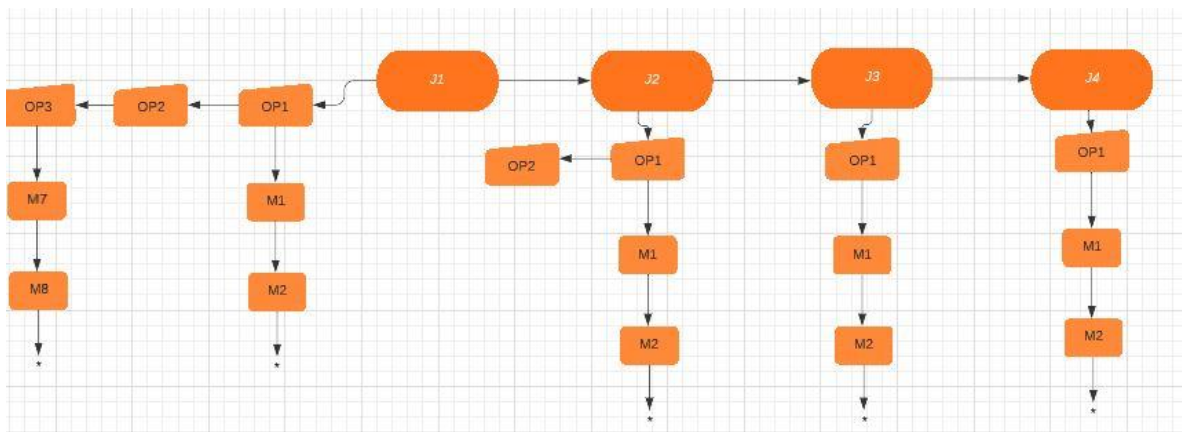


Figura 3-Diagrama Simplificação

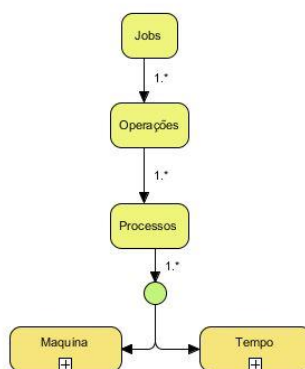


Figura 4-Diagrama Simplificação

Armazenamento do ficheiro de dados

O armazenamento de dados em ficheiro Txt, que a ideia principal seria implementar em ficheiro Json, o formato do registo será um job adicionado quando deteta anteriormente nada (vazio ou caracter) e o primeiro caracter seguido seja (“>”), a primeira linha corresponde as maquinas da primeira operação e a segunda linha corresponde ao tempo de cada maquina associado anteriormente.

Uma operação será associada quando a primeira linha e segunda forem preenchidas, assim sucessivamente.

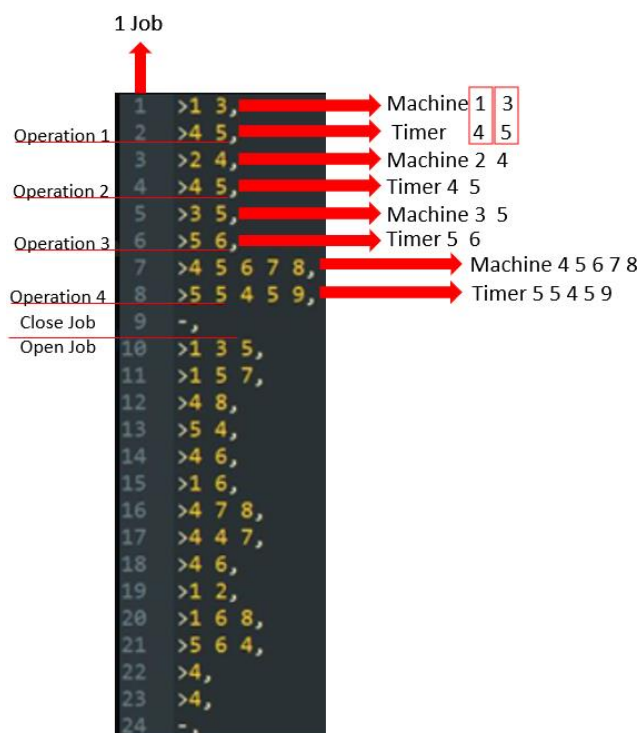


Figura 4-Ficheiro Txt

Ao processamento do arquivo, utilizamos especificadores do tipo fscanf, %d, este é o espaço reservado mais usado na língua C. É usado para ler o valor inteiro.

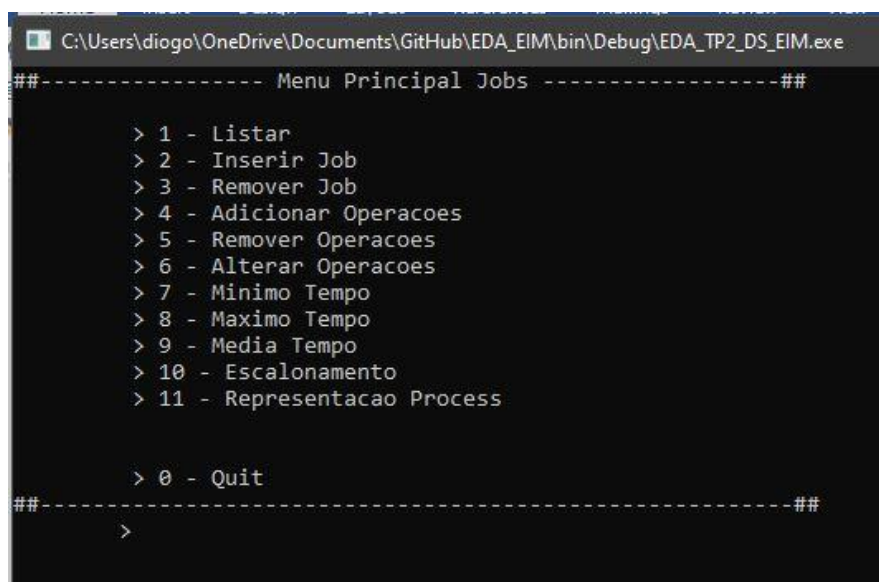
- A função scanf() é usada para ler entrada formatada de stdin na linguagem C. Ele retorna todo o número de caracteres escritos nele de outra forma, retorna um valor negativo.
- A função fscanf() é usada para ler a entrada formatada a partir do fluxo dado na língua C. Ele retorna zero, se não tiver sucesso. Caso contrário, ele retorna a sequência de entrada, se for bem-sucedido.

Funções

Na linguagem C, a função é um conjunto de comandos que realiza uma tarefa específica num módulo dependente de código. A função é referenciada pelo programa principal através do nome atribuído a ela. A utilização de funções visa modelizar um programa em várias partes, no qual é muito comum em programação estruturada. Desta forma podemos dividir um programa em várias partes, no qual cada função realiza uma tarefa bem definida. No nosso programa está implementadas as seguintes funções:

Inicialização do Programa

Para ajudar o utilizador, realizou-se um “Main-Menu”, para que o utilizador possa navegar de uma maneira simplificada para poder manipular os dados.



```
C:\Users\diogo\OneDrive\Documents\GitHub\EDA_EIM\bin\Debug\EDA_TP2_DS_EIM.exe
##----- Menu Principal Jobs -----##
> 1 - Listar
> 2 - Inserir Job
> 3 - Remover Job
> 4 - Adicionar Operacoes
> 5 - Remover Operacoes
> 6 - Alterar Operacoes
> 7 - Minimo Tempo
> 8 - Maximo Tempo
> 9 - Media Tempo
> 10 - Escalonamento
> 11 - Representacao Process
> 0 - Quit
##-----##
>
```

Figura 5-Menu Principal

Inserir um novo Job

Selecionando a opção “Adicionar Operação” o utilizador poderá escolher o Id que irá atribuir à operação.

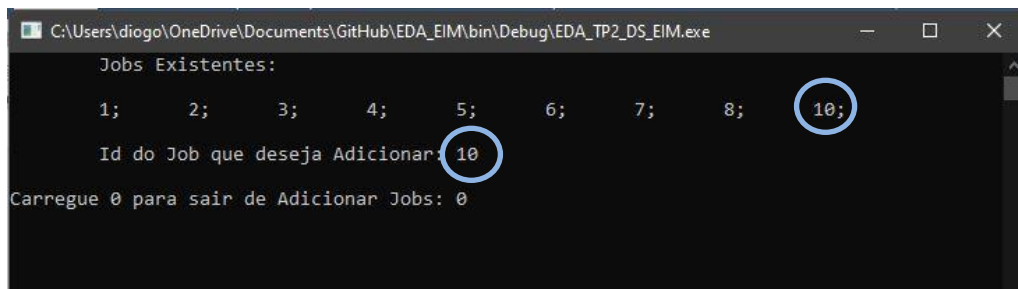


Figura 6-Adicionar Operação

Remover um Job

Selecionando a opção “Remover Job” o utilizador poderá apagar o job existente ou criado.

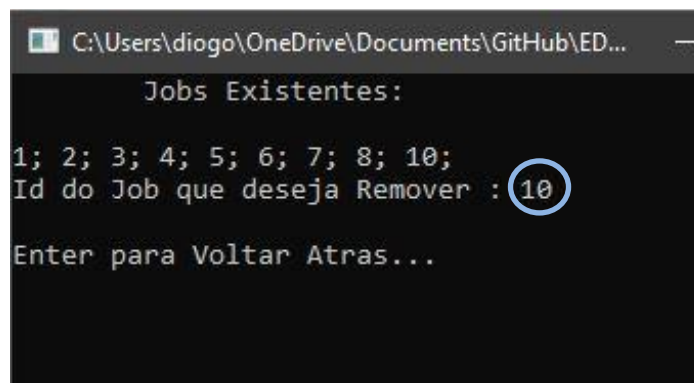


Figura 7-Apagar Job Parte 1

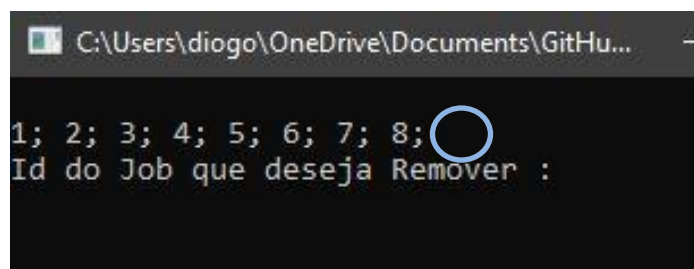
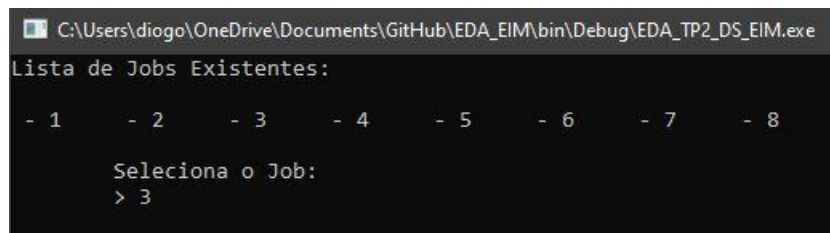


Figura 8-Apagar Job Parte 2

Visualização de Jobs

Introduzindo o Id do Job, ele possibilitará a visualização das operações, máquinas e tempos associados.



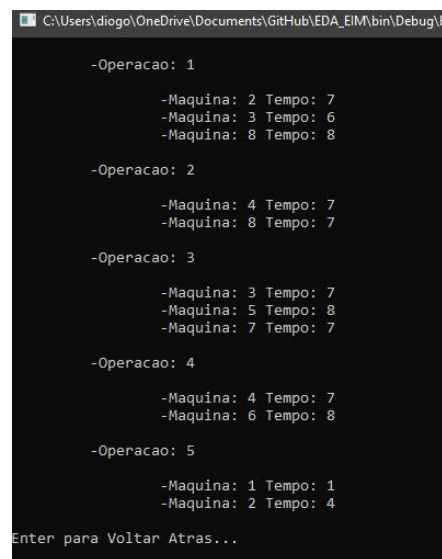
```

C:\Users\diogo\OneDrive\Documents\GitHub\EDA_EIM\bin\Debug\EDA_TP2_DS_EIM.exe
Lista de Jobs Existentes:

- 1      - 2      - 3      - 4      - 5      - 6      - 7      - 8

Selecione o Job:
> 3
  
```

Figura 8-Listar Job Parte 1



```

C:\Users\diogo\OneDrive\Documents\GitHub\EDA_EIM\bin\Debug\EDA_TP2_DS_EIM.exe

-Operacao: 1
    -Maquina: 2 Tempo: 7
    -Maquina: 3 Tempo: 6
    -Maquina: 8 Tempo: 8

-Operacao: 2
    -Maquina: 4 Tempo: 7
    -Maquina: 8 Tempo: 7

-Operacao: 3
    -Maquina: 3 Tempo: 7
    -Maquina: 5 Tempo: 8
    -Maquina: 7 Tempo: 7

-Operacao: 4
    -Maquina: 4 Tempo: 7
    -Maquina: 6 Tempo: 8

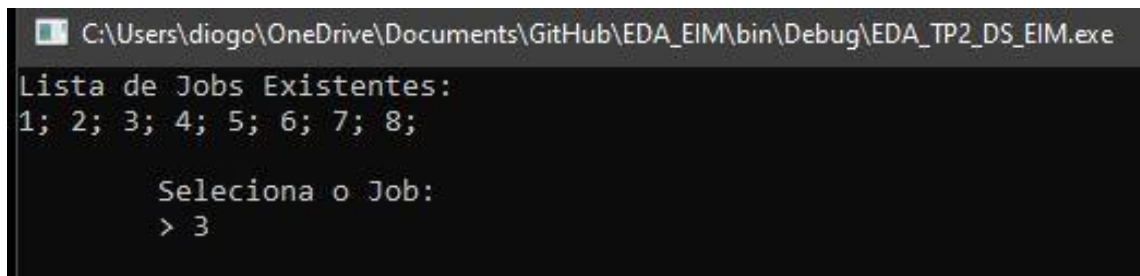
-Operacao: 5
    -Maquina: 1 Tempo: 1
    -Maquina: 2 Tempo: 4

Enter para Voltar Atras...
  
```

Figura 9-Listar Job Parte 2

Adicionar Operações em Determinados Jobs

Selecione a opção “Adicionar Operação” o utilizador poderá escolher o Job onde deseja adicionar a operação.

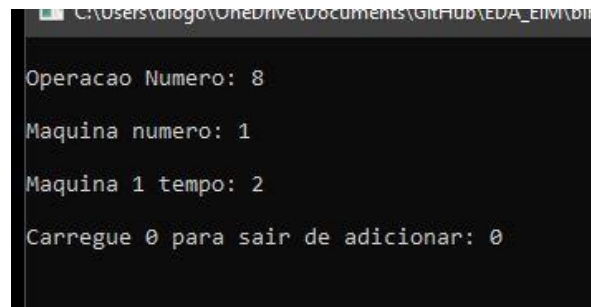


```
C:\Users\diogo\OneDrive\Documents\GitHub\EDA_EIM\bin\Debug\EDA_TP2_DS_EIM.exe
Lista de Jobs Existentes:
1; 2; 3; 4; 5; 6; 7; 8;

    Selecciona o Job:
    > 3
```

Figura 10 -Adicionar Parte 1

Caso o utilizador pretenda adicionar mais maquinas, é só necessario fazer um Skip(Enter) do aviso (“Carregue 0 para sair de adicionar: “).

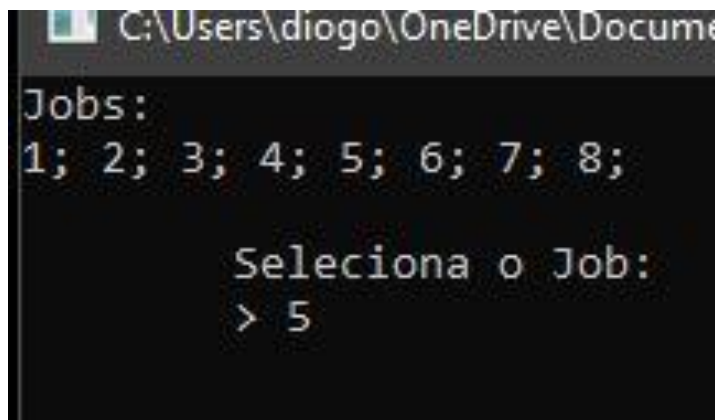


```
C:\Users\diogo\OneDrive\Documents\GitHub\EDA_EIM\bin
Operacao Numero: 8
Maquina numero: 1
Maquina 1 tempo: 2
Carregue 0 para sair de adicionar: 0
```

Figura 11-Listar Jobs Parte 2

Remover Operações em Determinados Jobs

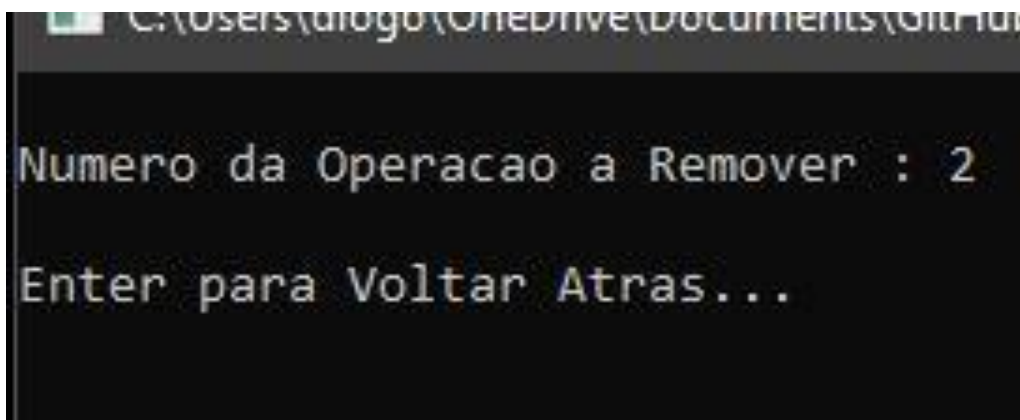
Selecione a opção “Remover Operação” o utilizador poderá escolher o Job onde deseja remover a determinada operação desejada.



```
C:\Users\diogo\OneDrive\Documents>
Jobs:
1; 2; 3; 4; 5; 6; 7; 8;

      Selecciona o Job:
      > 5
```

Figura 12-Remover Job Parte 1

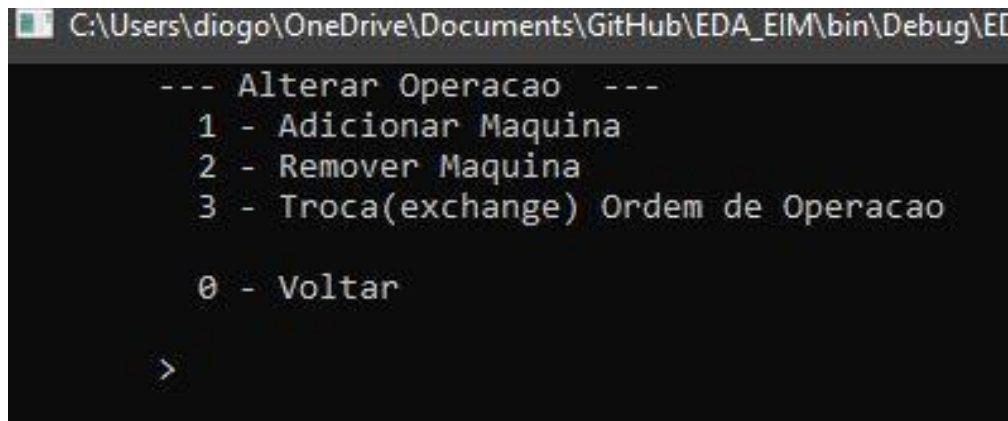


```
C:\Users\diogo\OneDrive\Documents\GitHub>
Numero da Operacao a Remover : 2
Enter para Voltar Atras...
```

Figura 13-Remover Job Parte 2

Alterar Dados de uma determinada Operação de um determinado Job

O utilizador terá acesso à área de alterar uma operação em um determinado job à escolha, como adicionar uma maquina nova em uma determinada operação como realizar uma troca que será uma troca de posicoes dos Id das operações indicadas.



```
C:\Users\diogo\OneDrive\Documents\GitHub\EDA_EIM\bin\Debug\EI

--- Alterar Operacao ---
1 - Adicionar Maquina
2 - Remover Maquina
3 - Troca(exchange) Ordem de Operacao

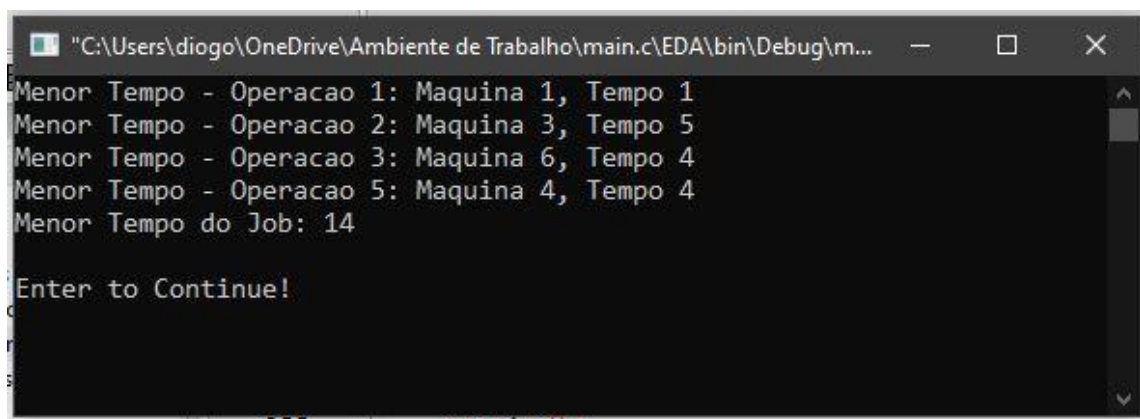
0 - Voltar

>
```

Figura 14-Alterar Operação em um Job

Determinar a quantidade mínima de unidades de tempo necessárias para completar o job e listagem das respetivas operações

O utilizador terá acesso à área de visualizar o menor tempo para completar um determinado Job, para ajudar o utilizador, será apresentado a lista de operações existentes.



```
"C:\Users\diogo\OneDrive\Ambiente de Trabalho\main.c\EDA\bin\Debug\m...

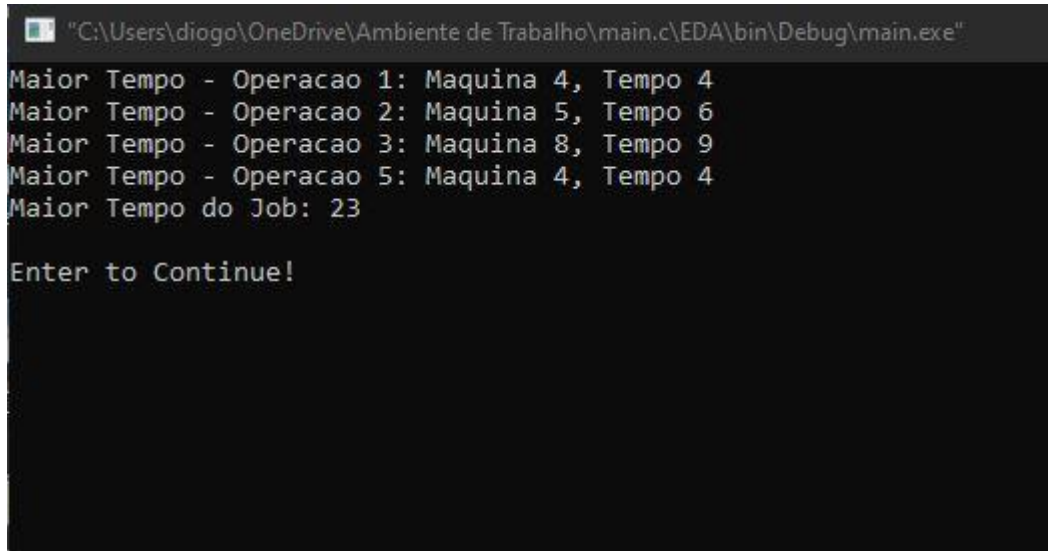
Menor Tempo - Operacao 1: Maquina 1, Tempo 1
Menor Tempo - Operacao 2: Maquina 3, Tempo 5
Menor Tempo - Operacao 3: Maquina 6, Tempo 4
Menor Tempo - Operacao 5: Maquina 4, Tempo 4
Menor Tempo do Job: 14

Enter to Continue!
```

Figura 15-Tempo Minimo Job

Determinação da quantidade máxima de unidades de tempo necessárias para completar o job e listagem das respectivas operações;

O utilizador terá acesso à área de visualizar o maior tempo para completar um determinado Job, para ajudar o utilizador, será apresentado a lista de operações existentes.



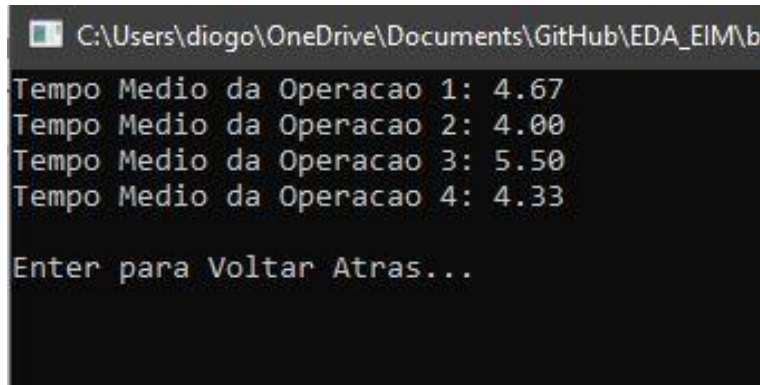
```
"C:\Users\diogo\OneDrive\Ambiente de Trabalho\main.c\EDA\bin\Debug\main.exe"
Maior Tempo - Operacao 1: Maquina 4, Tempo 4
Maior Tempo - Operacao 2: Maquina 5, Tempo 6
Maior Tempo - Operacao 3: Maquina 8, Tempo 9
Maior Tempo - Operacao 5: Maquina 4, Tempo 4
Maior Tempo do Job: 23
Enter to Continue!
```

Figura 16-Tempo Máximo cada Job

Determinação da quantidade média de unidades de tempo necessárias para completar uma operação, considerando todas as alternativas possíveis;

O utilizador terá acesso à área de consultar a média de um determinado Job, para ajudar o utilizador, será apresentado a lista de operações existentes no Job que o proprio introduziu.

Apresentando assim todas as operações do Job com a respetiva média.



```
C:\Users\diogo\OneDrive\Documents\GitHub\EDA_EIM\bi
Tempo Medio da Operacao 1: 4.67
Tempo Medio da Operacao 2: 4.00
Tempo Medio da Operacao 3: 5.50
Tempo Medio da Operacao 4: 4.33

Enter para Voltar Atras...
```

Figura 9-Tempo Máximo cada Operação

Conclusão

Após terminarmos a realização do trabalho, conseguimos pôr em prática os conhecimentos dados nas aulas, embora nos tenhamos deparado com bastantes dificuldades.

Apesar de todas as dificuldades encontradas neste trabalho, como a implementação dos últimos dois pontos do projeto, o escalonamento e a representação Process Plan devido à sua complexidade, mas mesmo assim conseguimos superar algumas com sucesso.

Este trabalho consegue-nos demonstrar um pouco o nível de dificuldade que podemos ter no exterior a nível profissional.

Contudo, este trabalho vem nos trazer mais experiência e conhecimento da disciplina, preparando-nos assim para a realização de mais trabalhos no futuro com outros programadores/empresas.

Para consultar a documentação e código, será disponibilizado este link GitHub para poder acompanhar o processo e os feitos do mesmo.

Link : https://github.com/DiogoAlex960/EDA_EIM

Bibliografia

Na realização deste trabalho, fomos encontrando várias dificuldades ao longo do projeto, e quase todos ultrapassados com sucesso, como foi dito anteriormente.

Usamos assim livros de programação, os conteúdos e códigos usados nas aulas da Unidade Curricular Estrutura de dados Avançadas.

O que é FJSP - <https://www.localsolver.com/docs/last/exampletour/flexiblejobshop.html>

<https://www.youtube.com/watch?v=SNE36xuGfV8>

<https://www.youtube.com/watch?v=B36tjfUJNBM>

Bibliotecas C - <https://www.ime.usp.br/~pf/algoritmos/apend/interfaces.html>

https://www.tutorialspoint.com/c_standard_library/index.htm

Corrigir Código C - <https://docs.microsoft.com/pt-br/cpp/code-quality/walkthrough-analyzing-c-cpp-code-for-defects?view=msvc-170>

Duvidas sobre Funções - [Stack Overflow - Where Developers Learn, Share, & Build Careers](#)

DoxyGen - <https://www.youtube.com/watch?v=R150qI6e7HU>

