**Engenharia Eletrotécnica**

**e de Computadores**

**Estruturas de Dados Avançadas (EDA)**

Uma imagem com texto, ClipArt

Descrição gerada automaticamente

**Relatório Trabalho fase 2**

José Alberto Silva Rodrigues – 16372

Docente: [João Carlos Cardoso da Silva](https://elearning1.ipca.pt/2122/user/view.php?id=562&course=7574)

**1 de junho de 2022**

Conteúdo do trabalho

[1 Índice de Figuras 2](#_Toc105014321)

[2 Introdução 3](#_Toc105014322)

[3 Objetivos 4](#_Toc105014323)

[4 Estruturas 4](#_Toc105014324)

[4.1 Estruturas de dados utilizadas no programa 4](#_Toc105014325)

[4.2 Estrutura do programa 5](#_Toc105014326)

[5 Testes realizados 7](#_Toc105014327)

[6 Conclusão 15](#_Toc105014328)

[7 Bibliografia 15](#_Toc105014329)

# Índice de Figuras

[Figura 1 Menu principal 7](#_Toc105014142)

[Figura 2 leitura de ficheiro de texto com representação de um Process plan 7](#_Toc105014143)

[Figura 3 - Gravação no ficheiro de texto com representação de um Process plan 7](#_Toc105014144)

[Figura 4 Ficheiro gravado Saida.txt 8](#_Toc105014145)

[Figura 5 - Submenu 8](#_Toc105014146)

[Figura 6 - Em qual dos jobs deseja remover uma operação 9](#_Toc105014147)

[Figura 7 - Job selecionado para escolha de remoção operação 9](#_Toc105014148)

[Figura 8 - Operação removida do job 2 10](#_Toc105014149)

[Figura 9 - Em qual dos jobs vai inserir uma operação 10](#_Toc105014150)

[Figura 10 - inserida uma operação no job 2 com uma máquina 11](#_Toc105014151)

[Figura 11 - Remove uma máquina ou um registo referente a uma operação 11](#_Toc105014152)

[Figura 12 - Escolha do job e da operação que quer alterar 12](#_Toc105014153)

[Figura 13 - Resultado da alteração 13](#_Toc105014154)

[Figura 14 - Listagem total de máquinas utilizadas e o tempo menor para um job 13](#_Toc105014155)

[Figura 15 - Listagem total de máquinas utilizadas e o tempo maior para um job 14](#_Toc105014156)

[Figura 16 - Escalonamento 14](#_Toc105014157)

[Figura 17 - Listagem de todos os jobs 15](#_Toc105014158)

# Introdução

Com este trabalho prático pretende-se sedimentar os conhecimentos relativos a definição e manipulação de estruturas de dados dinâmicas na linguagem de programação C. A essência deste trabalho reside no desenvolvimento de uma solução digital para o problema de escalonamento denominado *Flexible Job ShopProblem* (FJSSP). A solução a implementar deverá permitir gerar uma proposta de escalonamento para a produção de um produto envolvendo várias operações e a utilização de várias máquinas, minimizando o tempo as unidades de tempo necessário na sua produção (*makespan*). Um FJSSP pode ser formulado da seguinte forma:

# Objetivos

O objetivo é elaborar um programa que obedeça as todas as condições que me foram pedidas para execução de um

processo fabril.

E também praticar a manipulação de lista ligadas simples, criação de estruturas, leitura e escrita em ficheiros e os comandos necessários para elaboração de programa em c, e tudo os programas.

Tendo como objetivo a execução do programa final que me foi proposto.

# Estruturas

## Estruturas de dados utilizadas no programa

//! Estrutura da lista ligada para guardar um job

typedef struct obj

{

int estado;

int process;

int operacoes;

int maquina;

int tempo;

struct obj\* seguinte;

}Jobs;

//! Estrutura da Lista ligada para extração dos dados dos processos do ficheiro

typedef struct pr

{

int linhaMaquinas;//!Linha onde se encontra o processo e maquinas

int linhaTempo;//!Linha onde se encontra os tempos

int proc;//!Numero do processo

struct pr\* seguinte;

}Prs;

//! Estrutura da Lista ligada dos endereços dos processos

typedef struct proc

{

struct Jobs\* job;

int operacoes;

int prs;

struct proc\* seguinte;

}Processo;

//! Estrutura da Lista ligada das máquinas para escalonamento

typedef struct maq

{

int tempo;

int maquina;

struct maq\* seguinte;

}Maquinas;

//! Estrutura da Lista ligada do escalonamento

typedef struct esc

{

struct EscalProcOpe\* eps;

int maquina;

struct esc\* seguinte;

}Escalonamento;

//! Estrutura da Lista ligada tempos processo operação para escalonamento

typedef struct eps

{

int processo;

int operacao;

struct eps\* seguinte;

}EscalProcOpe;

## Estrutura do programa

O programa é composto pelo seguinte:

Primeiro tem um menu principal e um submenu, que nos dá acesso a todas as operações necessárias á manipulação de um ficheiro tipo **Process plan,** bem como listagens e gravação de dados em ficheiro TXT.

**1 - Leitura de ficheiro de texto com representação de um Process plan:**

Neste menu é possível ler um ficheiro tipo Process plan representado em ficheiro TXT, e inserir todos os dados em duas listas ligadas. Uma tem os números dos jobs o endereço da lista onde estão as máquinas e o número de processos em cada job. Sendo possível ler qualquer tipo de Process plan pois é tudo dinâmico não estando restringido a dados fixos. É possível escolher o ficheiro que vamos importar para o programa.

**2 - Gravação de ficheiro de texto com representação de um** **Process plan**

Neste menu é possível gravar um ficheiro de saída TXT ou outro, pois podemos escolher o nome do ficheiro que vamos gravar. Fazendo uma apresentação com o formato do ficheiro de leitura. Os dados alterados no programa, também serão gravados também serão gravados pois é dinâmico.

**3 - Inserção e alteração de operações em todo o Process plan (Submenu)**

**3. 1-Remover uma operação de um job**

Em todo o Process plan podemos escolher em qual job vamos remover uma operação com ajuda de listagens que nos ajudam a escolher onde queremos remover.

Vai-nos dando informações de alerta e não deixando fazer operações não permitidas

**3. 2 - Inserir uma operação num job**

Em todo o Process plan podemos escolher em qual job vamos adicionar uma operação com ajuda de listagens que nos ajudam a escolher onde vamos adicionar.

Vai-nos dando informações de alerta e não deixando fazer operações não permitidas.

Quando adicionado uma operação onde pretendemos, vai alterar o número das operações seguintes, pois as operações têm que ser sequenciais.

**3. 3-Remoção de uma determinada operação de um job**

Aqui podemos eliminar um registo de uma operação de qualquer job

**3. 4-Altera uma das operações associadas a um job**

Aqui podemos alterar qualquer registo de qualquer job. Podendo alterar por operação ou máquina, tendo todas as operações de ajuda para não cometer erros. Todos os dados alterados são organizados para respeitar as sequencias e respeitar o Process plan.

**4 - Cálculo do tempo mínimo e listagem de um job**

Executa o cálculo de tempos mínimos de um job a diferença da primeira fase é que nesta fase é possível executar em qualquer job. Mostrando na consola o resultado.

**5 - Cálculo do tempo máximo e listagem de um job**

Executa o cálculo de tempos máximos de um job a diferença da primeira fase é que nesta fase é possível executar em qualquer job. Mostrando na consola o resultado

**6 - Inserção de um novo job**

Podemos inserir um novo job no Process plan. Todas as ajudas para executar a operação são visualizadas. Pode se escolher o job que queremos, se existir somos avisados dando-nos a opção no novo job

**7 - Remoção de um job**

Eliminação de um job em todo Process plan. Ao ser removido todos os endereços na lista são apagados e colocado na lista dos processos o registo que chamou o job a NULL. As operações são postas a 0, mas o registo na lista dos processos não é apagado.

**8 – Escalonamento**

**9 - Listagem de todos os jobs**

Listagem geral de todos os jobs e visualização de todos os registos

**10 - Sair**

Sai do programa

**Obs.:** Todo o programa é dinâmico. Tem avisos de introdução de dados repetidos ou impossíveis e avisos de operações executadas.

# Testes realizados

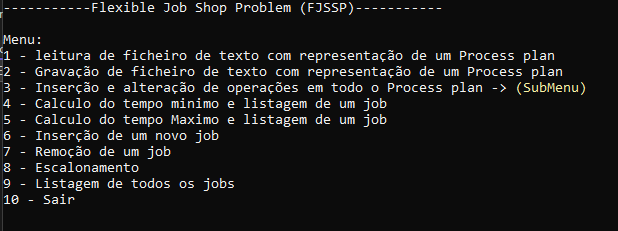


Figura 1 Menu principal





Figura 2 leitura de ficheiro de texto com representação de um Process plan

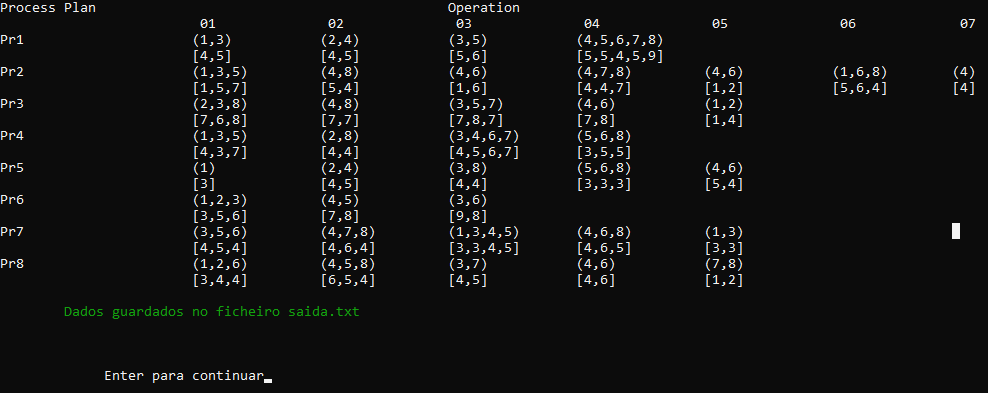


Figura 3 - Gravação no ficheiro de texto com representação de um Process plan

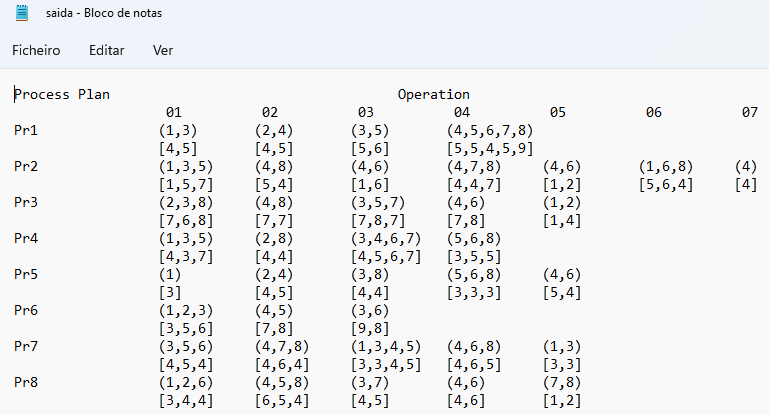


Figura 4 Ficheiro gravado Saida.txt

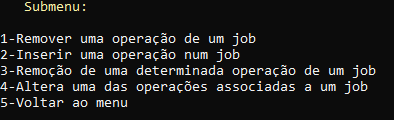


Figura 5 - Submenu

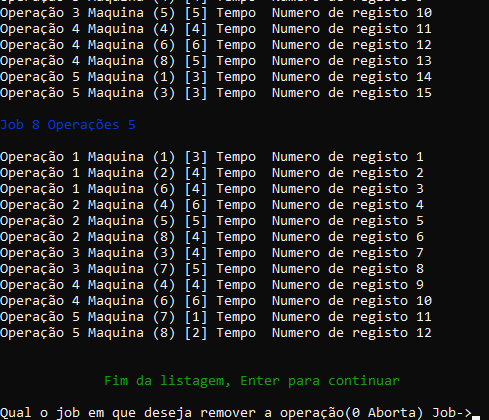


Figura 6 - Em qual dos jobs deseja remover uma operação

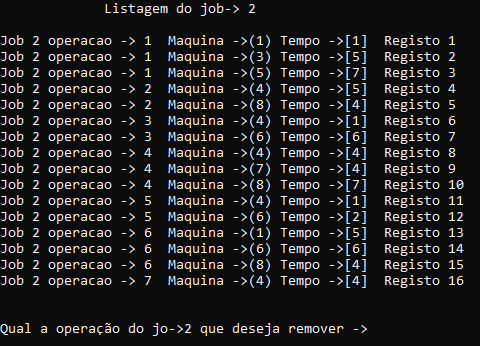


Figura 7 - Job selecionado para escolha de remoção operação

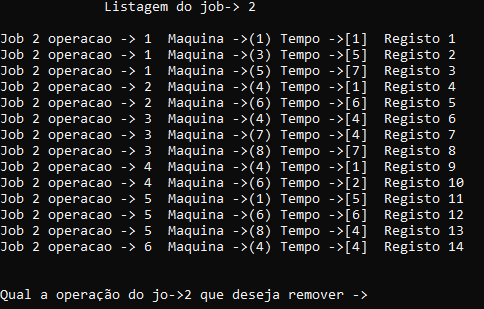


Figura 8 - Operação removida do job 2

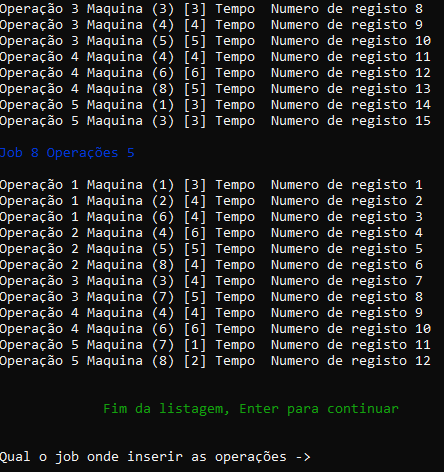
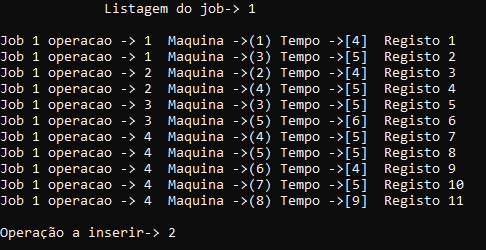
2

Figura - Em qual dos jobs vai inserir uma operação



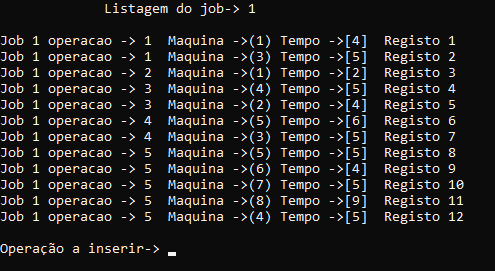


Figura 10 - inserida uma operação no job 2 com uma máquina

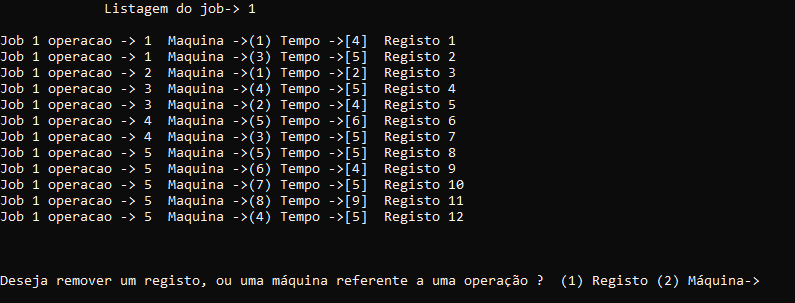
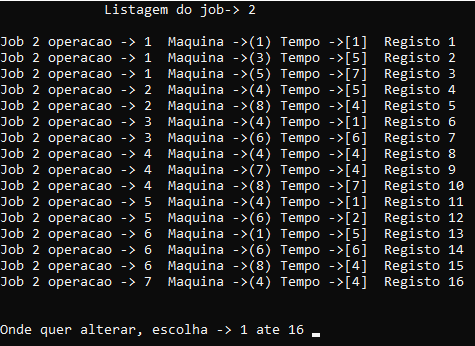


Figura 11 - Remove uma máquina ou um registo referente a uma operação



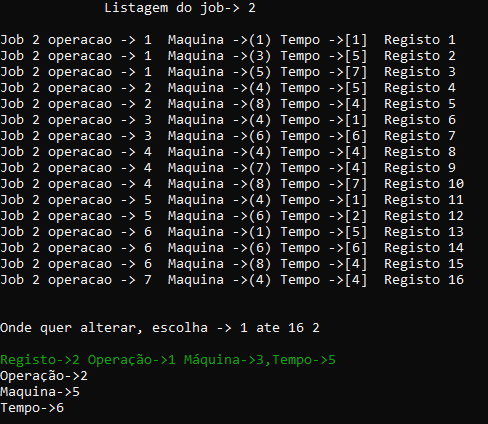


Figura 12 - Escolha do job e da operação que quer alterar

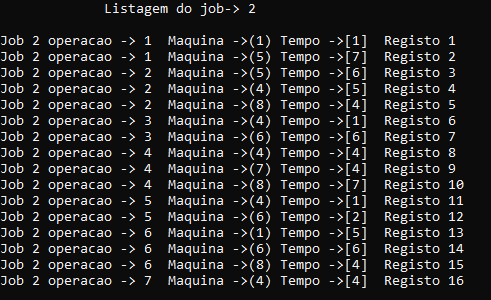


Figura 13 - Resultado da alteração

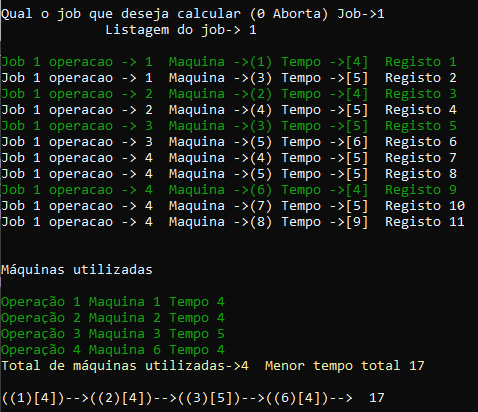


Figura 14 - Listagem total de máquinas utilizadas e o tempo menor para um job

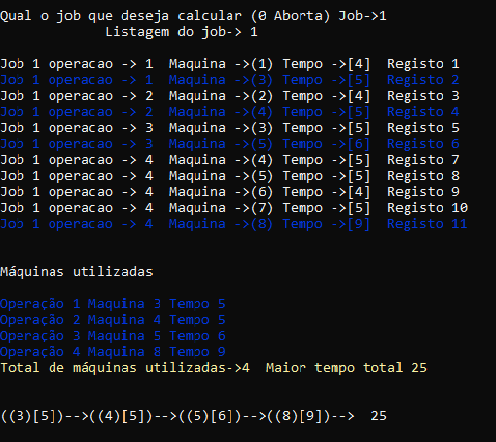


Figura 15 - Listagem total de máquinas utilizadas e o tempo maior para um job



Figura 16 - Escalonamento

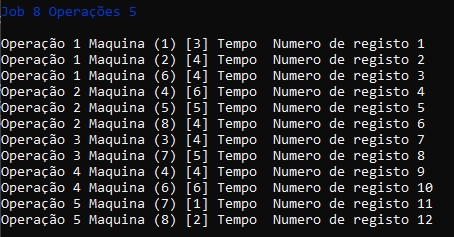


Figura 17 - Listagem de todos os jobs

# Conclusão

Um trabalho interessante, já com muito menos dificuldades na utilização de listas ligadas. Embora os problemas fossem surgindo, foram solucionados com menor dificuldade o que me deu uma grande aprendizagem em programação geral em C.

A parte de escalonamento é o que está a dar maior dificuldade, pois tem muitas operações a executar.

No geral gostei imenso de ter feito este trabalho, pois fez com que eu desenvolvesse bastante o meu raciocínio.

Ligação ao GitHub:

<https://github.com/JoseASRodrigues/EDA-fase_2>

# Bibliografia

**[cplusplus.com - C++](cplusplus.com - C++https://www.cplusplus.com)**

*[https://www.cplusplus.com](cplusplus.com - C++https://www.cplusplus.com)*