Laboratorio di Architettura degli Elaboratori A.A. 2023/2024

**Relazione sull’elaborato ASM**

Massimo Vareschi VR503167

Noemi Vellante VR503119

Indice

[Struttura del programma 2](#_Toc169789323)

[Flusso del programma 2](#_Toc169789324)

[Organizzazione dei file 2](#_Toc169789325)

[Strutture dati 2](#_Toc169789326)

[Funzioni 2](#_Toc169789327)

[Test effettuati 2](#_Toc169789328)

[Scelte progettuali 2](#_Toc169789329)

# Struttura del programma

## Flusso del programma

Il programma è suddiviso in tre fasi principali:

1. Lettura del file
2. Scelta dell’algoritmo
3. Stampa del risultato

Il *main* chiama la funzione *read\_push* che legge il file e copia i valori sullo stack.

La funzione *read\_push*

* legge il file carattere per carattere e lo converte in numero
* quando arriva alla virgola, controlla quanti campi ha letto e conseguentemente salva il valore nella variabile corrispondente (id, dur, scar, prior)
* quando arriva alla fine della riga effettua il push sullo stack e incrementa il numero di prodotti letti
* finita la lettura del file, restituisce al main il numero di prodotti letti

Il *main* chiama la funzione *print\_*menu per la stampa del menù con il quale l’utente può scegliere quale algoritmo usare.

A seconda della scelta effettuata, il *main* chiama le funzioni *edf* o *hpf*.

La funzione *edf*……………………………..

La funzione *hpf* ……………………………..

Il *main* chiama la funzione *output* che calcola e stampa a video il tempo di produzione e le eventuali penalità.

La funzione *output*

* esegue il pop di un prodotto
* stampa la pianificazione del prodotto, chiamando la funzione *itoa* per fare la conversione da numero a stringa
* calcola il tempo di produzione
* calcola le eventuali penalità
* arrivata la fine dell’ordine, stampa la durata totale e la penalità.

## Organizzazione dei file

Il programma è stato diviso in 7 file sorgente per renderne più chiaro lo sviluppo in fase progettuale e semplificare il debugging……………..

# Strutture dati

Il *main* contiene le seguenti variabili.

…….

La funzione *read\_push* contiene le seguenti variabili.

**fd**: variabile di tipo long che contiene l’indirizzo del file

**char**: variabile di tipo string utilizzata per leggere il file, carattere per carattere

**num**: variabile di tipo long utilizzata per la conversione da stringa a numero

**id**: variabile di tipo long utilizzata per salvare il valore dell’identificativo del prodotto appena letto, prima di effettuare il push sullo stack

**dur**: variabile di tipo long utilizzata per salvare il valore della durata del prodotto appena letto, prima di effettuare il push sullo stack

**scad**: variabile di tipo long utilizzata per salvare il valore della scadenza del prodotto appena letto, prima di effettuare il push sullo stack

**prior**: variabile di tipo long utilizzata per salvare il valore della priorità del prodotto appena letto, prima di effettuare il push sullo stack

**conta\_c**: variabile di tipo long utilizzata per contare quanti campi sono stati letti (conta\_c = 3 effettua il push sullo stack del prodotto appena letto)

**conta\_p**: variabile di tipo long utilizzata per contare quanti prodotti sono stati letti (parametro da passare alle funzioni di sorting)

La funzione *output* contiene le seguenti variabili.

**id**: variabile di tipo long utilizzata per salvare il valore dell’identificativo del prodotto appena preso dallo stack

**dur**: variabile di tipo long utilizzata per salvare il valore della durata del prodotto appena preso dallo stack

**scad**: variabile di tipo long utilizzata per salvare il valore della scadenza del prodotto appena preso dallo stack

**prior**: variabile di tipo long utilizzata per salvare il valore della priorità del prodotto appena preso dallo stack

**conta\_p**: variabile di tipo long utilizzata per contare quanti prodotti sono stati letti dallo stack (parametro passato dal main)

**tempo**: variabile di tipo long, utilizzata per calcolare e salvare la pianificazione dei singoli prodotti e la durata totale della produzione

**penalty**: variabile di tipo long, utilizzata per calcolare la penalità della produzione

**car**: variabile di tipo char utilizzata per la stampa

**conclusione**: variabile di tipo ascii utilizzata per la stampa dell’output

**conclusione\_len**: variabile di tipo long utilizzata per la stampa dell’output

**penalty\_str**: variabile di tipo ascii utilizzata per la stampa dell’output

**penalty\_len**: variabile di tipo long utilizzata per la stampa dell’output

# Funzioni

read\_push

* prende l’indirizzo del file dal registro EBX
* effettua il push nello stack dei valori letti dal file
* passa al main il numero di prodotti letti, tramite il registro EDI

output

* prende il numero di prodotti dal registro EDI
* per ciascun prodotto effettua il pop dallo stack e stampa ID e tempo di pianificazione
* aggiorna la variabile tempo
* calcola eventuale penalità
* continua fino a che non ha raggiunto la fine dei prodotti
* stampa la conclusione e la penalità.

# Test effettuati

Il programma è stato testato con 9 ordini:

1. EDF.txt che ha penalità uguale a zero con EDF e maggiore di zero con HPF
2. Both.txt che ha penalità uguale a zero con entrambi gli algoritmi
3. None.txt che ha penalità maggiore di zero con entrambi gli algoritmi
4. Breve.txt che è composto di meno di 10 righe
5. Lungo.txt che è composto di più di 10 righe
6. Duplicati1.txt che contiene alcuni prodotti che hanno stessa scadenza e priorità
7. Duplicati2.txt che contiene gli stessi prodotti di Duplicati1.txt, ma in ordine diverso. Dalla differenza nell’ordine iniziale dipende il risultato diverso che si ottiene.
8. Incompleto.txt che presenta una riga incompleta, portando alla generazione del relativo errore
9. ValoreExtra.txt che contiene un valore che non rispetta le specifiche del prodotto, , portando alla generazione del relativo errore.

Di seguito si riporta il contenuto dei file di test, con le penalità calcolate per entrambi gli algoritmi.

1. EDF.txt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo HPF) |
| 1 | 6 | 40 | 5 | 0 |
| 3 | 1 | 15 | 4 | 0 |
| 2 | 2 | 56 | 4 | 0 |
| 4 | 5 | 13 | 3 | 3 |
| 5 | 4 | 70 | 3 | 0 |
| 9 | 3 | 21 | 2 | 0 |
| 6 | 2 | 32 | 2 | 0 |
| 7 | 6 | 7 | 1 | 2 |
| 8 | 3 | 50 | 1 | 0 |
| 10 | 3 | 88 | 1 | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo EDF) |
| 7 | 6 | 7 | 1 | 0 |
| 4 | 5 | 13 | 3 | 0 |
| 3 | 1 | 15 | 4 | 0 |
| 9 | 3 | 21 | 2 | 0 |
| 6 | 2 | 32 | 2 | 0 |
| 1 | 6 | 40 | 5 | 0 |
| 8 | 3 | 50 | 1 | 0 |
| 2 | 2 | 56 | 4 | 0 |
| 5 | 4 | 70 | 3 | 0 |
| 10 | 3 | 88 | 1 | 0 |

1. Both.txt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo EDF) |
| 120 | 1 | 15 | 5 | 0 |
| 44 | 5 | 20 | 3 | 0 |
| 93 | 2 | 32 | 2 | 0 |
| 35 | 7 | 40 | 5 | 0 |
| 45 | 3 | 50 | 1 | 0 |
| 98 | 10 | 51 | 1 | 0 |
| 39 | 2 | 56 | 4 | 0 |
| 23 | 10 | 58 | 1 | 0 |
| 78 | 1 | 70 | 2 | 0 |
| 91 | 6 | 77 | 1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo HPF) |
| 120 | 1 | 15 | 5 | 0 |
| 35 | 7 | 40 | 5 | 0 |
| 39 | 2 | 56 | 4 | 0 |
| 44 | 5 | 20 | 3 | 0 |
| 93 | 2 | 32 | 2 | 0 |
| 78 | 1 | 70 | 2 | 0 |
| 45 | 3 | 50 | 1 | 0 |
| 98 | 10 | 51 | 1 | 0 |
| 23 | 10 | 58 | 1 | 0 |
| 91 | 6 | 77 | 1 | 0 |

1. None.txt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo HPF) |
| 23 | 7 | 21 | 5 | 0 |
| 2 | 8 | 10 | 4 | 20 |
| 33 | 10 | 33 | 4 | 0 |
| 9 | 5 | 12 | 3 | 54 |
| 17 | 6 | 89 | 3 | 0 |
| 99 | 4 | 12 | 2 | 56 |
| 67 | 4 | 45 | 2 | 0 |
| 65 | 3 | 30 | 1 | 17 |
| 44 | 2 | 77 | 1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo EDF) |
| 2 | 8 | 10 | 4 | 0 |
| 9 | 5 | 12 | 3 | 3 |
| 99 | 4 | 12 | 2 | 10 |
| 23 | 7 | 21 | 5 | 15 |
| 65 | 3 | 30 | 1 | 0 |
| 33 | 10 | 33 | 4 | 16 |
| 67 | 4 | 45 | 2 | 0 |
| 44 | 2 | 77 | 1 | 0 |
| 17 | 6 | 89 | 3 | 0 |

1. Breve.txt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo EDF) |
| 120 | 6 | 17 | 4 | 0 |
| 1 | 10 | 21 | 3 | 0 |
| 11 | 4 | 21 | 3 | 0 |
| 37 | 3 | 41 | 2 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo HPF) |
| 120 | 6 | 17 | 4 | 0 |
| 1 | 10 | 21 | 3 | 0 |
| 11 | 4 | 21 | 3 | 0 |
| 37 | 3 | 41 | 2 | 0 |

1. Lungo.txt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo EDF) |
| 90 | 1 | 5 | 3 | 84 |
| 57 | 2 | 7 | 1 | 67 |
| 32 | 3 | 10 | 3 | 78 |
| 120 | 4 | 12 | 2 | 106 |
| 1 | 7 | 18 | 3 | 75 |
| 55 | 9 | 19 | 4 | 52 |
| 56 | 4 | 29 | 5 | 0 |
| 34 | 8 | 29 | 3 | 66 |
| 58 | 7 | 43 | 2 | 58 |
| 122 | 10 | 49 | 3 | 36 |
| 24 | 6 | 50 | 5 | 0 |
| 50 | 4 | 70 | 5 | 0 |
| 30 | 9 | 71 | 5 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo HPF) |
| 56 | 4 | 29 | 5 | 0 |
| 24 | 6 | 50 | 5 | 0 |
| 50 | 4 | 70 | 5 | 0 |
| 30 | 9 | 71 | 5 | 0 |
| 55 | 9 | 19 | 4 | 52 |
| 90 | 1 | 5 | 3 | 84 |
| 32 | 3 | 10 | 3 | 78 |
| 1 | 7 | 18 | 3 | 75 |
| 34 | 8 | 29 | 3 | 66 |
| 122 | 10 | 49 | 3 | 36 |
| 120 | 4 | 12 | 2 | 106 |
| 58 | 7 | 43 | 2 | 58 |
| 57 | 2 | 7 | 1 | 67 |

1. Duplicati1.txt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo EDF) |
| 11 | 2 | 6 | 4 | 0 |
| 45 | 8 | 10 | 4 | 0 |
| 5 | 3 | 12 | 5 | 5 |
| 8 | 5 | 12 | 4 | 24 |
| 34 | 7 | 12 | 3 | 39 |
| 1 | 8 | 24 | 5 | 45 |
| 60 | 9 | 30 | 4 | 48 |
| 14 | 6 | 30 | 4 | 72 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo EDF) |
| 5 | 3 | 12 | 5 | 0 |
| 1 | 8 | 24 | 5 | 0 |
| 11 | 2 | 6 | 4 | 28 |
| 45 | 8 | 10 | 4 | 44 |
| 8 | 5 | 12 | 4 | 56 |
| 14 | 6 | 30 | 4 | 8 |
| 60 | 9 | 30 | 4 | 44 |
| 34 | 7 | 12 | 3 | 108 |

1. Duplicati2.txt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo HPF) |
| 5 | 3 | 12 | 5 | 0 |
| 1 | 8 | 24 | 5 | 0 |
| 11 | 2 | 6 | 4 | 28 |
| 45 | 8 | 10 | 4 | 44 |
| 8 | 5 | 12 | 4 | 56 |
| 60 | 9 | 30 | 4 | 20 |
| 14 | 6 | 30 | 4 | 44 |
| 34 | 7 | 12 | 3 | 108 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DURATA | SCADENZA | PRIORITA’ | PENALTY (algoritmo HPF) |
| 11 | 2 | 6 | 4 | 0 |
| 45 | 8 | 10 | 4 | 0 |
| 5 | 3 | 12 | 5 | 5 |
| 8 | 5 | 12 | 4 | 24 |
| 34 | 7 | 12 | 3 | 39 |
| 1 | 8 | 24 | 5 | 45 |
| 14 | 6 | 30 | 4 | 36 |
| 60 | 9 | 30 | 4 | 72 |

# Scelte progettuali

Si è scelto di gestire due tipi di errori: errori nell’apertura del file ed errori nel contenuto del file.

Tra gli errori di contenuto si sono considerati

* il caso in cui una o più righe non contengano esattamente 4 campi
* il caso in cui uno o più campi non rispettino le specifice (ID può andare da 1 a 127, durata può andare da 1 a 10, scadenza può andare da 1 a 100, priorità può andare da 1 a 5)