
## UNIVR Dipartimento di Informatica

---

## Elaborato Assembly

## Laboratorio Architettura degli Elaboratori

---

# Pianificatore delle attività di un sistema produttivo

A.A. 2023/2024

Matricole:
Quaresima Luca VR500114
Pedretti Simone VR500040

**INDICE**

## Specifiche

Si sviluppi **in Assembly (sintassi At&t)** un software per la pianificazione delle attività di un sistema produttivo, per i successivi prodotti, fino ad un massimo di 10 prodotti, nelle successive 100 unità di tempo dette “slot temporali”. Il sistema produttivo può produrre prodotti diversi, ma produce un prodotto alla volta. La produzione è suddivisa in slot temporali uniformi, e durante ogni slot temporale solo un prodotto può essere in produzione. Ogni prodotto è caratterizzato da quattro valori interi:

- *Identificativo*: il codice identificativo del prodotto da produrre. Il codice può andare da 1 a 127;
- *Durata*: il numero di slot temporali necessari per completare il prodotto. La produzione di ogni prodotto può richiedere 1 a 10 slot temporali;
- *Scadenza*: il tempo massimo, espresso come numero di unità di tempo entro cui il prodotto dovrà essere completato. La scadenza di ciascun prodotto può avere un valore che va da 1 a 100;
- *Priorità*: un valore da 1 a 5, dove 1 indica la priorità minima e 5 la priorità massima. Il valore di priorità indica anche la penalità che l’azienda dovrà pagare per ogni unità di tempo necessaria a completare il prodotto oltre la scadenza.

Per ogni prodotto completato in ritardo rispetto alla scadenza indicata, l’azienda dovrà pagare una penale in Euro pari al valore della priorità del prodotto completato in ritardo, moltiplicato per il numero di unità di tempo di ritardo rispetto alla sua scadenza.

Ogni file non può contenere più di 10 ordini.
Una volta letto il file, il programma mostrerà il menu principale che chiede all’utente quale algoritmo di pianificazione dovrà usare.
L’utente potrà scegliere tra i seguenti due algoritmi di pianificazione:

1. **Earliest Deadline First (EDF)**: si pianificano per primi i prodotti la cui scadenza è più vicina, in caso di parità nella scadenza, si pianifica il prodotto con la priorità più alta.
2. **Highest Priority First (HPF)**: si pianificano per primi i prodotti con priorità più alta, in caso di parità di priorità, si pianifica il prodotto con la scadenza più vicina.

L’utente dovrà inserire il valore 1 per chiedere al software di utilizzare l’algoritmo EDF, ed il valore 2 per chiedere al software di utilizzare l’algoritmo HPF.
Una volta pianificati i task, il software dovrà stampare a video:

1. L’ordine dei prodotti, specificando per ciascun prodotto l’unità di tempo in cui è pianificato l’inizio della produzione del prodotto. Per ogni prodotto, dovrà essere stampata una riga con la seguente sintassi: ID:Inizio Dove ID è l’identificativo del prodotto, ed Inizio è l’unità di tempo in cui inizia la produzione.
2. L’unità di tempo in cui è prevista la conclusione della produzione dell’ultimo prodotto pianificato.
3. La somma di tutte le penalità dovute a ritardi di produzione.

Una volta stampate a video le statistiche, il programma tornerà al menù iniziale in cui chiede all'utente se vuole pianificare la produzione utilizzando uno dei due algoritmi.
L'uscita dal programma potrà essere gestita in due modi: si può scegliere di inserire una voce apposita (esci) nel menu principale, oppure affidarsi alla combinazione di tasti ctrl-C.
In entrambi i casi però, tutti i file utilizzati dovranno risultare chiusi al termine del programma.

# Variabili

### Pianificatore.s

Array:

- ID→ contiene gli ID dei vari oggetti.
- Inizio→contiene i valori (calcolati) di inizio della produzione degli oggetti.
- Durata→ contiene i valori della durata di produzione degli oggetti.
- Deadline→contiene i valori delle scadenze degli oggetti.
- Priority→contiene i valori della priorità degli oggetti
- Fine→contiene i valori di Fine produzione oggetti.

**conclusione**: usata per stampare il valore di conclusione della produzione.
**penalty**: variabile usata per contenere la penalità di ogni oggetto, poi copiata nell'indice corrispondente di tale oggetto.
**lines**: contiene il numero di oggetti nel file (ossia le righe)
**salva_num**: variabile in cui salvo temporaneamente il numero mentre lo converto da stringa a intero.
**i :** contatore ciclo.
**j:** contatore ciclo.

**newline:** contiene il valore **\n** codificato in ASCII.
**virgola:** contiene il valore '**,**' codificato in ASCII.
**buffer**: serve per la lettura delle stringhe da file.

### Penalty.s

**A_2**: stringa che serve per la stampa di ogni cifra.
**contatore_cifre:** conta il numero di cifre presenti in ogni numero.

### Risultati.s

**salvo_resto**: serve per salvare il resto della divisione che mi serve a stampare una singola cifra.
**salvo_seconda_cifra:** salva la seconda cifra di un eventuale numero a più cifre.

# File

Il codice è diviso in 3 file:

- **Pianificatore.s:** è il file principale, contiene la maggior parte del codice, e l'implementazione di:
    - Algoritmi di ordinamento.
    - Calcolo penalty.
    - Calcolo fine produzione degli oggetti.
    - Calcolo inizio produzione degli oggetti.

- **risultati.s:** file ausiliario che serve per la stampa degli oggetti prodotti (**ID:Inizio**) e ordinati secondo l'algoritmo scelto.

- **Penalty.s**: file ausiliario che serve per la stampa della penalità.

# Logica del Codice

**Algoritmi di Ordinamento:**

Per l'implementazione degli algoritmi di ordinamento è stato utilizzato l'algoritmo **Bubble Sort.** Sono stati realizzati due cicli for innestati, in modo simile a come si implementerebbe l'algoritmo nel linguaggio C.

**Algoritmo EDF**:
Esempio di iterazione:

Viene confrontata la Deadline[i] con la Deadline[i+1].
Se maggiore, scambio ID, Durata, Deadline, Priority dei due oggetti.
Se uguale, guardo chi ha la priorità maggiore e agisco di conseguenza.

Una volta implementato l'algoritmo **EDF**, l'algoritmo **HPF** è risultato essere identico da implementare, ordinando gli oggetti secondo priorità.

**Calcolo Fine:**

La fine della produzione di un oggetto non è altro che la somma delle durate dei vari oggetti, compresa quella dell'oggetto in questione.
La fine di produzione dell'ultimo oggetto risulterà essere anche il tempo di Conclusione della produzione.

**Calcolo Inizio:**

Per l'inizio della produzione di un oggetto, non è stato necessario effettuare alcun calcolo aggiuntivo.
L'inizio della produzione di un oggetto, infatti, altro non è che la fine della produzione dell'oggetto precedente.

Naturalmente, il primo oggetto da produrre inizia a t=0, e di conseguenza siamo partiti a copiare l'array **Fine** nell'array **Inizio** dall'indice i=0 per **Fine** e j=1 per **Inizio** (la fine della produzione dell'oggetto all'indice k corrisponde all'inizio della produzione dell'oggetto all'indice k+1).

**Calcolo Penalty:**

Il calcolo della penalità di un oggetto all'indice **i** risulta essere (**Fine[i]** -**Deadline[i]**) * **Priorità[i].**
Viene naturalmente calcolata solo nell'eventualità che **Fine[i]-Deadline[i]** sia maggiore di 0.

La somma delle penalità di tutti gli oggetti è il totale da pagare.

## File di Verifica

I file utilizzati per il testing del corretto funzionamento del programma sono:

- EDF.txt→Penalità con EDF=0 e HPF>0
- Both.txt→Con entrambi gli algoritmi la penalità è 0
- None.txt→Con nessun algoritmo la penalità è 0.
- Piu_di_dieci.txt→test che verifica che venga segnalato un errore nel momento in cui si inseriscono più di 10 input.
- Input_errati.txt→verifica che, se si inseriscono input al di fuori dal range previsto, venga stampata una stringa di errore: ad esempio, se si inserisce un ID>127.

## Scelte progettuali

- Per la realizzazione del programma sono stati utilizzati **vettori** in byte, visto che i valori erano tutti compresi tra 0 e 255.

- La realizzazione di funzioni con i vettori byte sarebbe risultata macchinosa, abbiamo quindi “preferito” mantenere tutti i principali algoritmi nel file principale.

- Si è utilizzato l’algoritmo di ordinamento **bubble sort** al fine di ordinare gli oggetti in quanto è l’algoritmo che è stato utilizzato più di frequente nel corso di **Programmazione**

- Nel caso vengano inseriti input errati o più input di quelli previsti dalla consegna, viene stampato un messaggio di errore.