Politecnico di Torino
Dipartimento di Automatica e Informatica

Laboratorio 1

FLOWCHART E PSEUDOCODICE

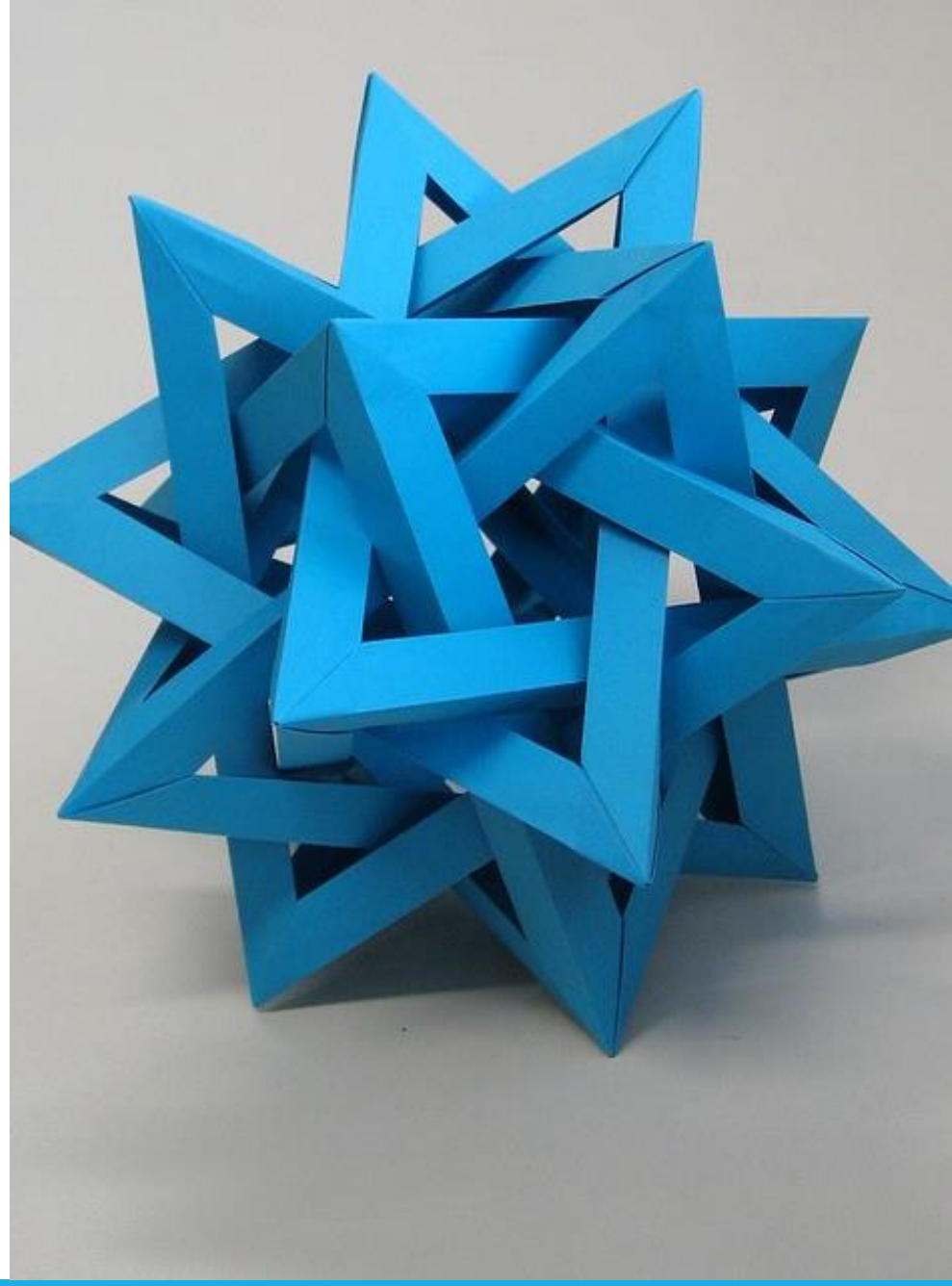


Figura 8

Procedimento per lo sviluppo di un programma

Esercizio 1

Esercizio 1. Volete calcolare la percentuale di utilizzo della vostra automobile per uso personale e, separatamente, per recarvi al lavoro. Conoscete la distanza tra casa vostra e il vostro luogo di lavoro e, dato un periodo di tempo, avete registrato il valore riportato dal contachilometri all'inizio ed alla fine del periodo; inoltre è noto il numero di giorni in cui vi siete recati al lavoro in tale periodo. Scrivete un algoritmo per risolvere questo problema (sotto forma di flow chart o pseudo codice) e calcolare la percentuale citata. [R1.16]

Valori d'esempio:
21gg, km iniziali: 9605, km finali: 9980, giorni lavorativi: 15, distanza casa-lavoro: 10 km.

Esercizio 1

- Determinare i dati disponibili e i risultati da produrre
- Scomporre il problema in compiti più semplici
- Descrivere ciascun sotto-problema mediante pseudocodice (o flowchart)
- Collaudare lo pseudocodice (o il flowchart) con valori d'esempio

Esercizio 1 - *Determinare i dati disponibili e i risultati da produrre*

Dati disponibili:

- Distanza casa-lavoro
- Numero giorni lavorativi
- Numero giorni totali sui quali calcolare la %
- Valore iniziare contachilometri
- Valore finale contachilometri

Risultati da produrre:

- Percentuale di utilizzo dell’auto per uso personale
- Percentuale di utilizzo dell’auto per uso lavorativo

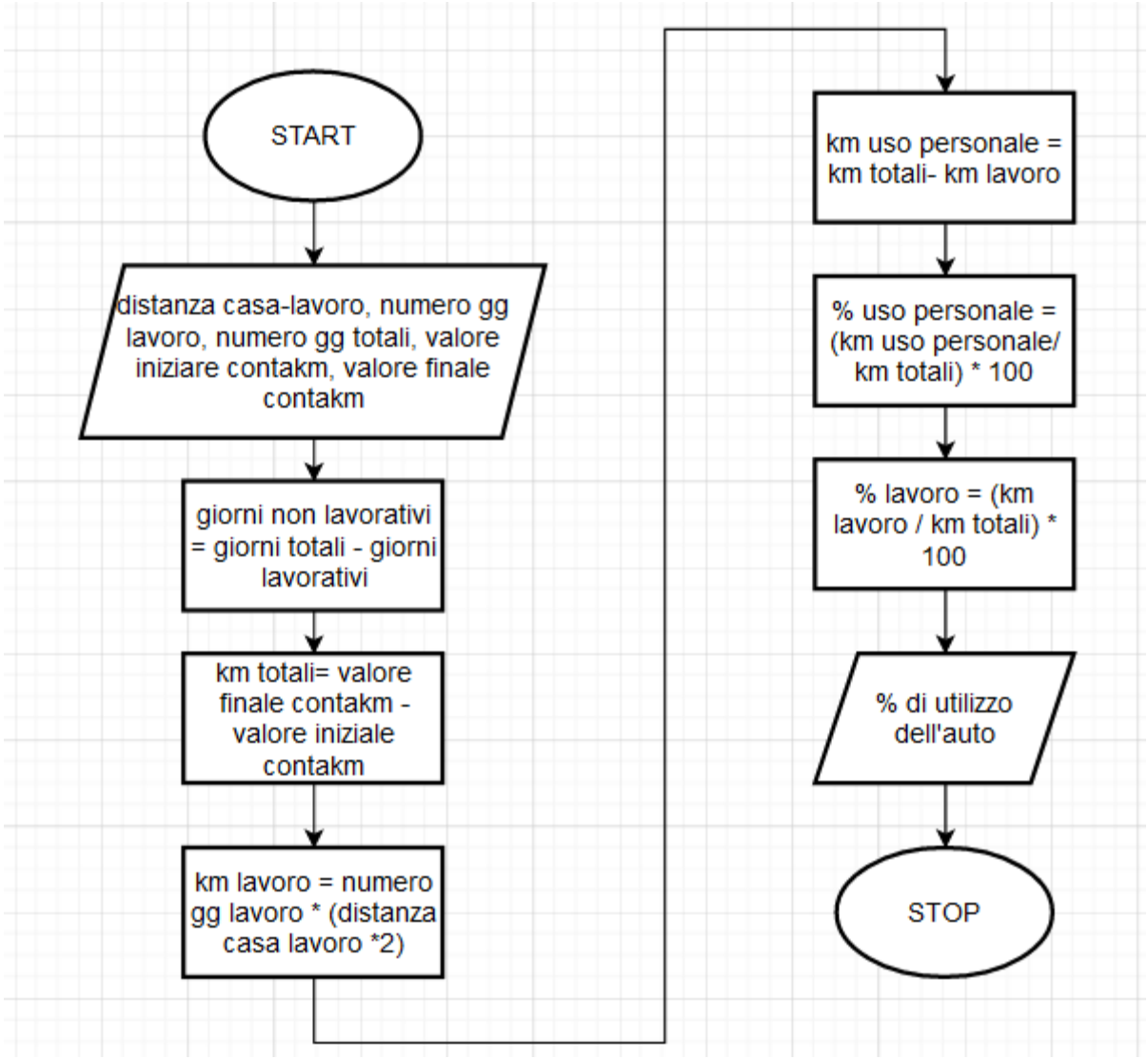
Esercizio 1 – *Scomporre il problema in compiti più semplici*

- Calcolare i giorni non lavorativi
- Calcolare i chilometri percorsi
- Calcolare i chilometri percorsi per lavoro (andata e ritorno)
- Calcolare i chilometri percorsi per uso personale
- Calcolare la percentuale di utilizzo dell'auto per lavoro e per uso personale

Esercizio 1 – *Descrivere ogni sotto-problema con pseudocodice*

- `giorni non lavorativi = giorni totali – giorni lavorativi`
- `km totali = valore finale contakm – valore iniziale contakm`
- `km lavoro = giorni lavorativi * (2 * distanza casa/lavoro)`
- `km uso personale = km totali – km lavoro`
- `% uso lavoro = (km lavoro / km totali) * 100`
- `% uso personale = (km uso personale / km totali) * 100`

Esercizio 1 – *Descrivere ogni sotto-problema con flowchart*



Esercizio 1 – *Collaudare con valori d'esempio*

- giorni non lavorativi = giorni totali – giorni lavorativi
- km totali = valore finale contakm – valore iniziale contakm
- km lavoro = giorni lavorativi * (2 * distanza casa/lavoro)
- km uso personale = km totali – km lavoro
- % uso lavoro = (km lavoro / km totali) * 100
- % uso personale = (km uso personale / km totali) * 100

→ giorni non lavorativi = 21 – 15 = 6

→ km totali = 9980 – 9605 = 375

→ km lavoro = 15 * (2 * 10) = 300

→ km uso personale = 375 – 300 = 75

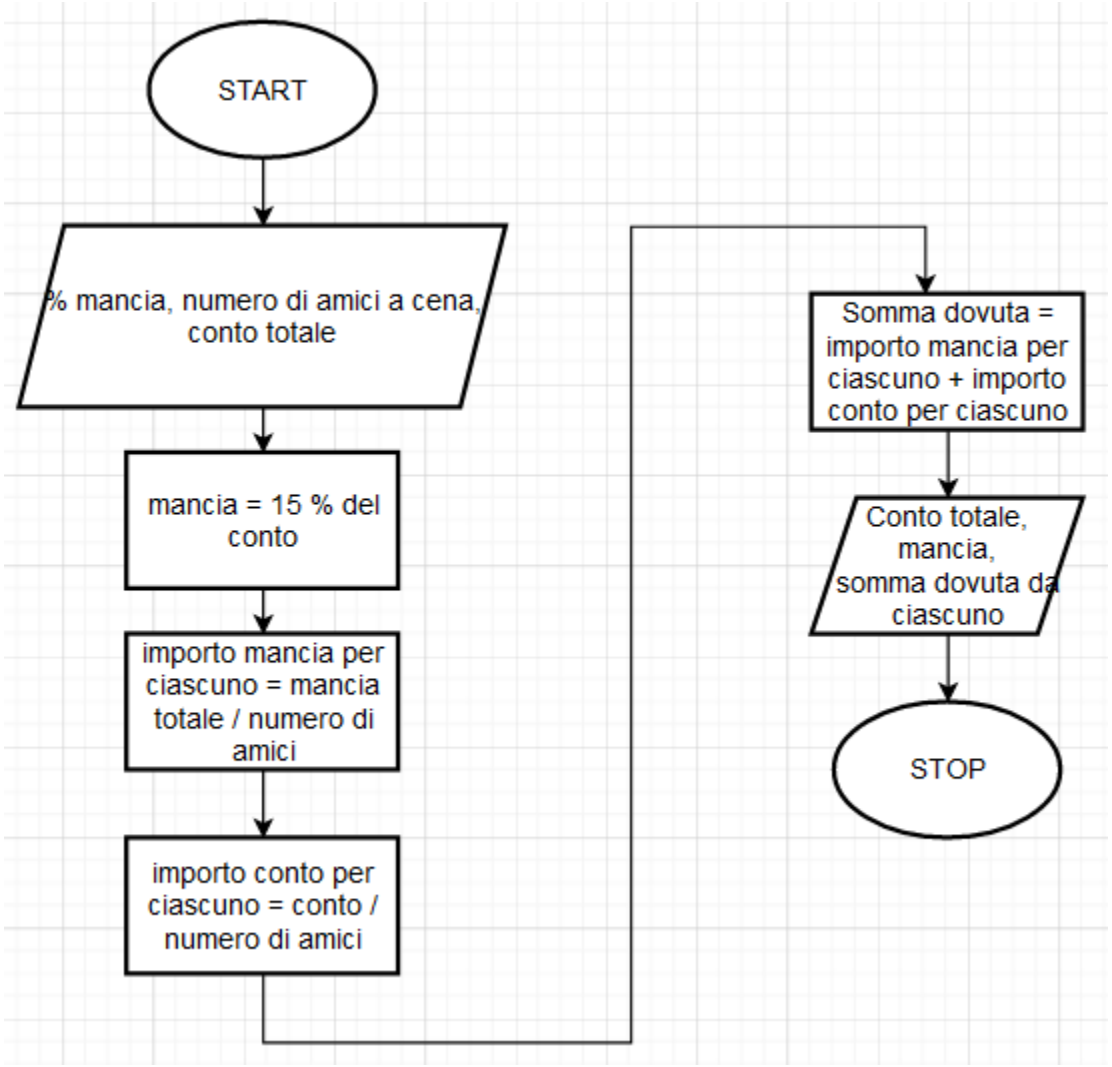
→ % uso lavoro = (300 / 375) * 100 = 80

→ % uso personale = (75 / 375) * 100 = 20

Esercizio 2

Esercizio 2. Supponete di andare in un ristorante di lusso insieme ad alcuni amici e che, al momento di pagare il conto, vogliate dividerlo in parti uguali, compresa la mancia del 15%. Descrivete, mediante un diagramma di flusso, un algoritmo che calcoli la somma dovuta da ciascuno. Il programma, noto l'importo del conto ed il numero degli amici, deve visualizzare l'importo del conto, la mancia, il costo totale e la somma dovuta da ciascuno; inoltre, deve mostrare quanta parte della somma dovuta da ciascuno serve per pagare il conto e quanta per la mancia. [R1.20]

Esercizio 2 - *flowchart*



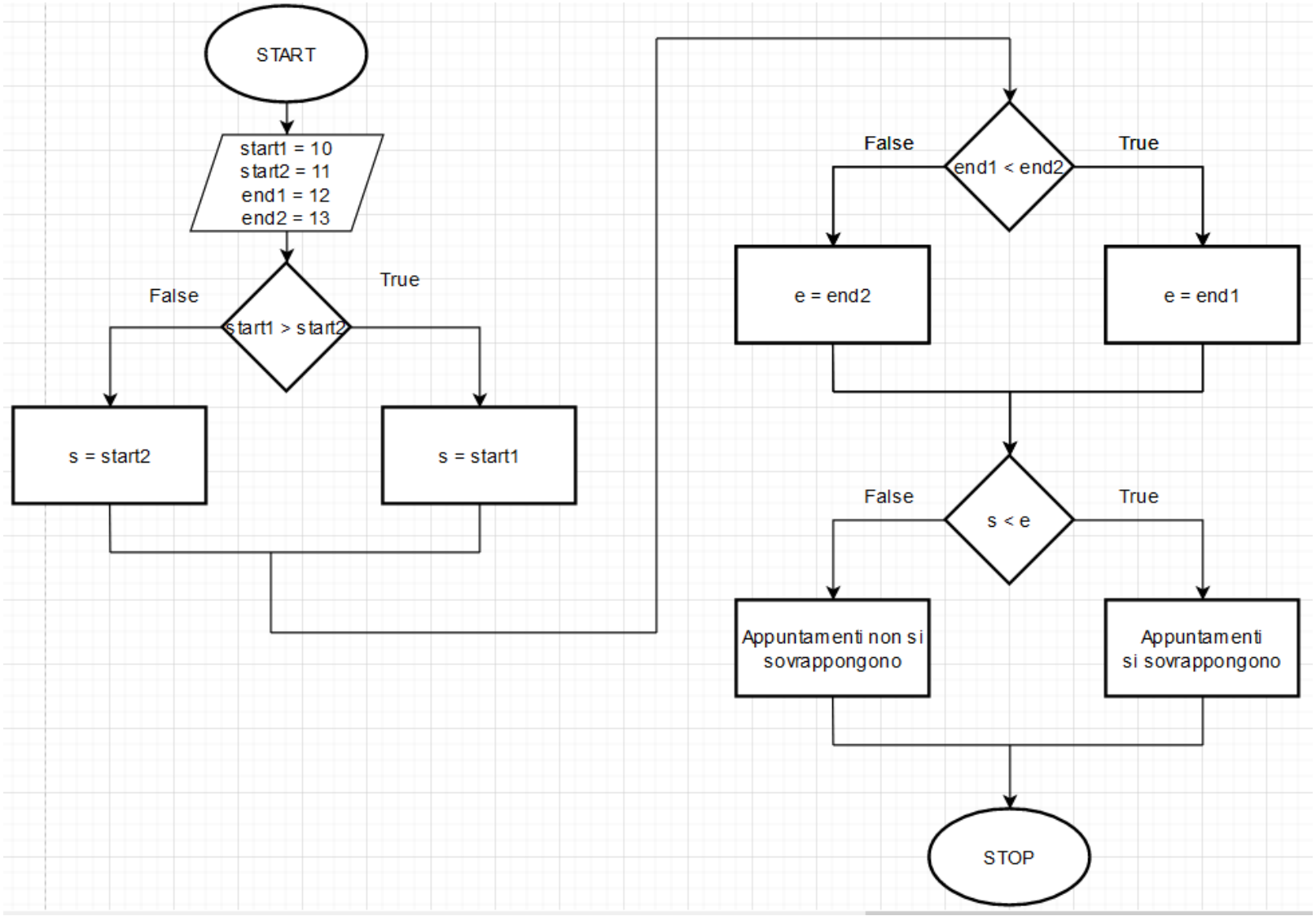
Esercizio 3

Esercizio 3. In un programma di calendarizzazione (scheduling) di eventi si deve verificare se due appuntamenti della stessa giornata si sovrappongono. Per semplicità, ipotizziamo che gli appuntamenti inizino sempre a un'ora esatta (senza minuti) e usiamo l'orario militare (cioè con le ore che vanno da 0 a 23). Lo pseudocodice seguente descrive un algoritmo che determina se l'appuntamento che inizia all'ora start1 e termina all'ora end1 si sovrappone all'appuntamento che inizia all'ora start2 e termina all'ora end2.

```
Se start1 > start2
        s = start1
Altrimenti
        s = start2
Se end1 < end2
        e = end1
Altrimenti
        e = end2
Se s < e
        Gli appuntamenti si sovrappongono
Altrimenti
        Gli appuntamenti non si sovrappongono
```

i. Seguite passo dopo passo l'esecuzione dello pseudo-codice con gli appuntamenti 10–12 e 11–13, poi con gli appuntamenti 10–11 e 12–13
ii. Disegnate il diagramma di flusso per l'algoritmo
iii. Spiegate il funzionamento dell'algoritmo e verificate se esso sia corretto [R3.12]

Esercizio 3 - *flowchart*



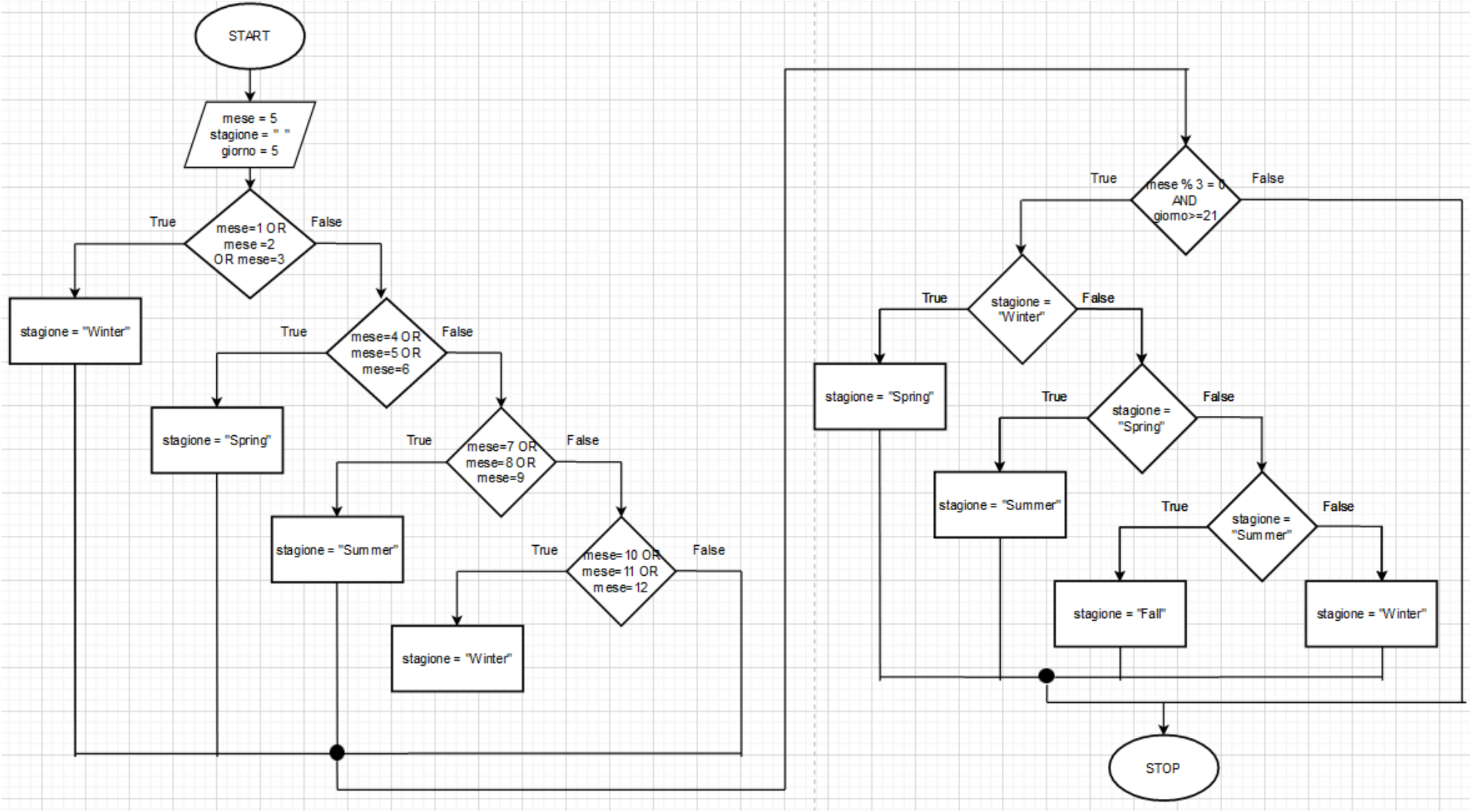
Esercizio 4

Esercizio 4. L'algoritmo seguente individua la stagione (Spring, Summer, Fall o Winter, cioè, rispettivamente, primavera, estate, autunno o inverno) a cui appartiene una data, fornita come mese e giorno.

Se mese è 1, 2 o 3, stagione = "Winter"
Altrimenti se mese è 4, 5 o 6, stagione = "Spring"
Altrimenti se mese è 7, 8 o 9, stagione = "Summer"
Altrimenti se mese è 10, 11 o 12, stagione = "Fall"
Se mese è divisibile per 3 e giorno >= 21
    Se stagione è "Winter", stagione = "Spring"
    Altrimenti se stagione è "Spring", stagione = "Summer"
    Altrimenti se stagione è "Summer", stagione = "Fall"
    Altrimenti stagione = "Winter"

Disegnate il diagramma di flusso per l'algoritmo. Verificate se l'algoritmo si comporta correttamente con una serie di date di prova. [R3.13]

Esercizio 4 - *flowchart*



Esercizio 5

Esercizio 5. Si consideri il seguente algoritmo di tipo matematico:

Dato un valore $\alpha > 0$, un algoritmo per approssimare $\sqrt{\alpha}$ comunemente usato è conosciuto come metodo babilonese e sfrutta gli stessi principi poi codificati nel metodo di Newton. Questo metodo funziona nel modo seguente:

1. Poni $n = 1$ e inizia con un valore arbitrario positivo x_n (quanto più esso è prossimo alla radice, tanto migliore è la convergenza dell'algoritmo)
2. sostituisci x_n con la media di x_n e α/x_n
3. aumenta n e vai al punto 2

Questo algoritmo può essere rappresentato da

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{\alpha}{x_n}\right)$$

da cui si ricava $\lim_{n\to\infty} x_n = \sqrt{\alpha}$.

Disegnare il diagramma di flusso relativo a tale algoritmo e provare ad eseguirlo passo-passo per calcolare $\sqrt{3}$.

Esercizio 5 - *flowchart*

a1pha = numero del quale calcolare la radice

xn = approssimazione della radice

n = contatore del ciclo

Precisione definita con **ε**:

L'uso di una costante EPSILON

- Si può utilizzare una valore molto piccolo per verificare se la differenza tra due numeri floating-point è '*abbastanza piccola*'
  - Considero i due numeri uguali se la loro differenza è minore di un certo limite EPSILON
  - Cioè x e y sono «uguali» se:

$$|x - y| < \varepsilon$$

Quindi l'approssimazione in questo caso si considera precisa (dopo un numero massimo di iterazioni, quindi $n<n_{max}$) se $|x_n - \alpha/x_n| < \varepsilon$.