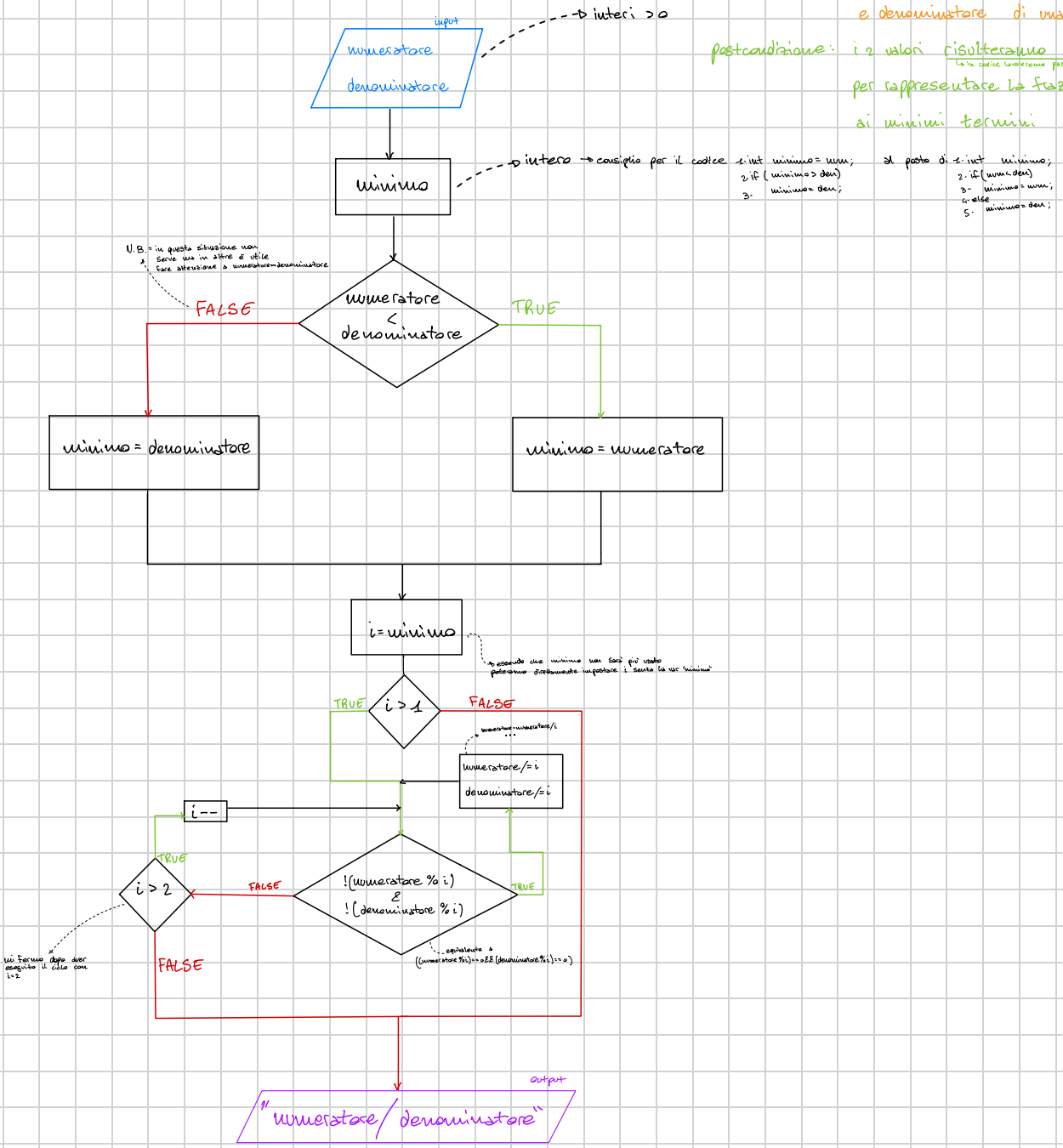
MINIMI TERMINI

precondizione: 2 interi >0 rappresentanti numeratore e denominatore di una frazione

postcondizione: i 2 valori risulteranno modificati (a tutti codice successivo passando puntatori alla funzione) per rappresentare la frazione ridotta ai minimi termini



Roba in più:

- Se si vuole effettuare l'esercizio per tutti i numeri interi (positivi/negativi/nulli)

1) controllare a inizio funzione se:

   a) numeratore = 0 → ritorno immediatamente (tramite 'return;' oppure valuto se ritornare $\frac{0}{1}$ → esempio: $\frac{0}{42} = \frac{0}{1}$)

   b) denominatore = 0 → ritorno immediatamente

   c) (numeratore > 0 && denominatore < 0) || (numeratore < 0 && denominatore > 0)
      → so che la frazione finale dovrà essere negativa → soluzione: creo un intero flag
      binario → se uno dei 2 è negativo imposto flag=1 e poi (a fine esercizio) imposto denominatore o numeratore a negativo (indifferente)
      non binario (...) → se num < 0 → flag = 1 (a fine es. imposto num a >0)
      den < 0 → flag = 2 (a fine es. imposto den a >0)

2) imposto entrambi i valori a positivi (se num < 0 → num *= -1, se den < 0 → den *= -1) → dunque se num < 0 && den < 0 a fine esercizio rimarranno entrambi positivi (es. $\frac{-4}{-12} = \frac{1}{3}$)
   → non obbligatorio ma sicuramente più semplice da gestire

3) Svolgo l'algoritmo come sopra

4) a fine algoritmo controllo flag
   - binario: se flag == 1 → numeratore *= -1 (esempi: $\frac{-4}{16} = \frac{-1}{4}$, $\frac{2}{-4} = \frac{-1}{2}$, ...)
   - non bin.: se flag == 1 → numeratore *= -1 / se flag == 2 → denominatore *= -1 ($\frac{-4}{16} = \frac{-1}{4}$, $\frac{2}{-4} = \frac{1}{-2}$, ...) → meglio questo credo

- Firma funzione: `void minimi_termini(int *num, int *den);`

Nel main (o in altre funzioni): `minimi_termini(&n, &d);`

All'interno della funzione uso num e den scrivendo sempre *num e *den