

Esercitazione 1: calcolatrice RPN

Le espressioni accettate dalla calcolatrice RPN (Reverse Polish Notation) sono stringhe in cui ogni carattere è una cifra o un operatore. La valutazione di un'espressione si avvale di una pila di numeri interi, inizialmente vuota, e procede esaminando i caratteri che compongono l'espressione da sinistra verso destra eseguendo le seguenti operazioni:


- i caratteri 0, 1, ..., 9 causano l'inserimento sulla pila del numero intero corrispondente. Per esempio, il carattere 5 inserisce il numero 5 sulla pila.
- l'operatore + estrae due numeri dalla pila e inserisce la loro somma sulla pila.
- l'operatore * estrae due numeri dalla pila e inserisce il loro prodotto sulla pila.

Seguono alcuni esempi di espressioni valide:

- 12+3* calcola 9
- 52*52**252***+3+ calcola 123
- 2222222222***** calcola 1024

Risolvere i seguenti esercizi. Il simbolo  indica una potenziale difficoltà.

Esercizio 1. Compilare il codice della calcolatrice RPN visto a lezione (la versione che usa la classe `Pila`), assicurarsi di comprenderne i dettagli e verificarne il corretto funzionamento. Poi, determinare espressioni in forma RPN per calcolare:

- ✓ • $(15 + 16) \times 17$
- ✓ • il fattoriale di 10, ovvero $1 \times 2 \times 3 \times \dots \times 10$
- ✓ • un numero negativo a piacere, senza modificare il codice della calcolatrice 

Esercizio 2. Estendere la calcolatrice con gli operatori binari -, /, e % per calcolare rispettivamente la differenza, la divisione intera, ed il resto della divisione intera di due numeri. Fare in modo che:


- ✓ • l'espressione 12- calcoli -1
- ✓ • l'espressione 32/ calcoli 1
- ✓ • l'espressione 53% calcoli 2

✓ **Esercizio 3.** Estendere la classe `Calcolatrice` con un metodo `stampa` che stampi la dimensione ed il contenuto della pila, senza modificarla. Estendere la calcolatrice con un operatore # che invochi tale metodo. Usare l'operatore # in punti a piacere delle espressioni testate fino ad ora per osservare come cambia lo stato della pila durante la valutazione delle stesse.

Esercizio 4 (con soluzione). Individuare una stringa s tale che, per ogni cifra decimale n , l'espressione ns (ovvero l'espressione composta dal carattere n seguito dai caratteri della stringa s) calcoli $2n + 1$. La stringa s che risolve l'esercizio è $2*1+$. Infatti, l'espressione $n2*1+$ calcola $2n + 1$ qualsiasi sia la cifra n .

Esercizio 5. Individuare una stringa s tale che, per ogni cifra decimale n , l'espressione ns (ovvero l'espressione composta dal carattere n seguito dai caratteri della stringa s) calcoli 1 se n è pari e 0 se n è dispari. Per esempio

- ✓ • $2s$ e $0s$ devono calcolare 1 dal momento che 2 e 0 sono pari
- ✓ • $3s$ e $7s$ devono calcolare 0 dal momento che 3 e 7 sono dispari

Esercizio 6 (). Per poter definire i metodi `push` e `pop`, è opportuno che la classe `Calcolatrice` abbia i **campi** `stack` e `size`. In alternativa si potrebbe pensare di definire `stack` e `size` come variabili locali di `main` e passarle ai metodi `push` e `pop` sotto forma di parametri, in aggiunta a quelli che eventualmente già hanno. Provare a realizzare questa versione alternativa della calcolatrice RPN e argomentare sulle difficoltà che emergono.