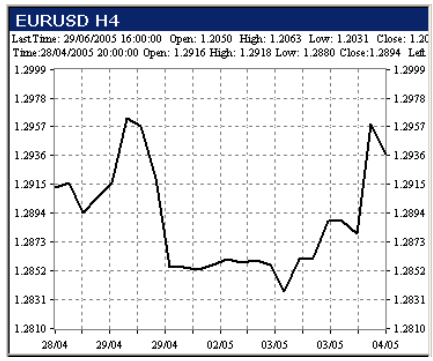
Esercitazione 11 (12-02 *)

11.1 Massimo e minimo (C++)

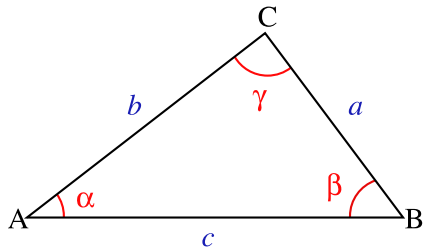
- Leggere, attraverso un ciclo, una sequenza di numeri interi positivi
- La sequenza termina quando l'utente inserisce il valore 0
- Visualizzare il valore massimo e quello minimo tra i numeri inseriti



Basta un ciclo, senza `vector`

11.2 Funzione, Erone (C++)

- Definire una funzione `heron` per il calcolo dell'area di un triangolo
  - Parametri: tre lati come `float`
  - Risultato: area come `float`
- Definire poi una funzione `main`
  - Chiedere all'utente tre valori
  - Poi chiamare `heron` con questi parametri
  - Infine mostrare all'utente il risultato



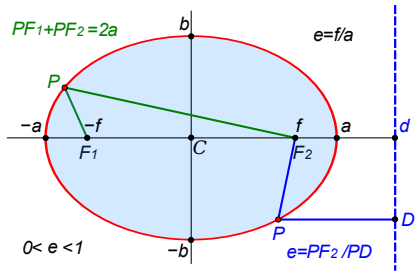
Formula di Erone: `area = sqrt(s * (s - a) * (s - b) * (s - c))`
Con `s = (a + b + c) / 2`, *semiperimetro*

`sqrt` *definita in libreria* `<cmath>`

http://en.wikipedia.org/wiki/Heron%27s_formula

11.3 Ellisse (C++)

- Classe che modella un'ellisse
- Campi privati (parametri del costruttore)
  - Semiassi: `a, b`
- Metodi pubblici per ottenere...
  - Area: `π*a*b`
  - Distanza focale: $2*\sqrt{|a^2 - b^2|}$
- Nel corpo principale del programma...
  - Creare un oggetto con dati forniti dall'utente
  - Visualizzare area e distanza focale dell'ellisse



Definire `PI = 3.14159`*, oppure usare* `acos(-1)`*, dalla libreria* `<cmath>`

11.4 Testo, lettere (C++)

- Leggere una riga di testo
- Contare quante lettere maiuscole ci sono in tutto
- Contare quante lettere minuscole ci sono in tutto

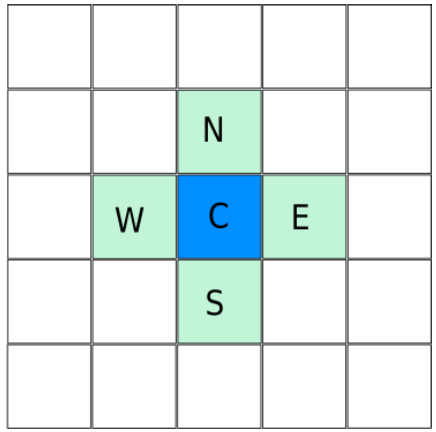
11.5 Risultati casuali (C++)

- Simulare n lanci di una coppia di dadi
  - n scelto dall'utente
- Contare quante volte si presenta ciascun risultato
  - Risultati possibili: da 2 a 12
  - Somma dei due dadi

Per conteggiare i vari risultati, usare un `vector` *di (almeno) 11 valori*

11.6 Funzione di smooth (C++)

- Scrivere una funzione `smooth`
  - Parametro: matrice iniziale, di `float`
  - Risultato: nuova matrice con *smooth*
- **Smooth**: per ogni cella in matrice iniziale
  - Il risultato è la *media* dell'intorno
  - 5 valori: cella stessa e 4 adiacenti
- Attenzione alle celle esterne
  - Sommare e contare solo i valori disponibili
  - 4 valori ai bordi, 3 valori agli angoli
- Verificare la funzione con alcune matrici di test



11.7 Hitori, backtracking

PY

```
Hitori::solve_recursive(i: int) -> bool:
    ## self.mark_auto()  # mark all obvious cells
    ## if self.wrong(): return False  # unsolvable
    # find first undecided cell, starting from i
    while i < len(self._annots) and self._annots[i] != CLEAR:
        i += 1
    if i < len(self._annots):
        saved = self._annots[:]  # save current status
        for a in (BLACK, CIRCLE):
            self._annots[i] = a
            if self.solve_recursive(i + 1):
                return True
            self._annots = saved  # backtracking
    return self.finished()
```

Suggerita l'implementazione di `mark_auto` *(es. 10.7) e* `wrong` *(11.7)*