Esercizi di programmazione

Fondamenti
di informatica

Michele Tomaiuolo
tomamic@ce.unipr.it
http://www.ce.unipr.it/people/tomamic

Esercitazione 1

- Espressioni e variabili
- Blocchi condizionali
- Cicli semplici

Ken Thompson, Dennis Ritchie

1.1 Conversione gradi

- Ricevere un numero che rappresenta una temperatura espressa in gradi Celsius
- Fornire la temperatura in gradi Fahrenheit

Partire dalla formula

$$F = 32 + \frac{9}{5} C$$

1.2 Condizioni

- Calcolare due radici di un'equazione di secondo grado espressa nella forma:

$$ax^2 + bx + c = 0$$

- I valori dei parametri `a`, `b` e `c` saranno immessi dall'utente

Per il calcolo della radice quadrata, utilizzare la funzione `sqrt` *nella libreria* `<cmath>`

$$x_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

1.3 Ciclo

- Leggere, attraverso un ciclo, una sequenza di numeri interi
- La sequenza termina quando l'utente immette il numero 0
- Visualizzare la somma dei numeri
- Visualizzare la loro media aritmetica

Provare ad usare sia il ciclo while
che il ciclo do-while

1.4 Carattere ASCII *

- Leggere un numero
- Visualizzare il carattere ASCII corrispondente

Fare il cast da `int` *a* `char`
prima di mandare il carattere su `cout`

1.5 Conversioni di tipo *

- Definire le seguanti variabili:
  - `int i, j; float x , y, z;`
- Ricevere in ingresso i valori di j, x , y
- Visualizza il valore delle seguenti espressioni:
  - `z = x + y; i = x + y; i = y / j;`
    `z = y / j; z = i / j;`

Attenzione agli arrotondamenti!

1.6 Triangolo *

- Chiedere all'utente le lunghezze dei tre lati
- Controllare se il triangolo è:
  - Equilatero
  - Isoscele
  - Scaleno
  - Oppure se i tre lati non formano affatto un triangolo
- Visualizzare il responso

1.7 Conteggio cifre *

- Leggere un numero intero positivo
- Determinare di quante cifre è composto

Quante volte riusciamo a dividerlo per 10, prima che si annulli?

Esercitazione 2

- Cicli
- Cicli annidati
- Stringhe

Bjarne Stroustrup

2.1 For, tabella ASCII

- Visualizzare la tabella dei codici ASCII
- Riportare in prima colonna il codice ASCII
- In seconda colonna il carattere corrispondente
- Limitare l'intervallo tra 32 a 126 (due costanti)

Procedere una riga alla volta (ltr →, ttb ↓)
Se si invia su `cout` *il manipolatore* `setw(3)`*,*
l'output successivo occuperà almeno 3 posti

2.2 For annidati, tabelline

- Visualizzare le tabelline
  - Da 1 a 10 (o 12...)
  - Scrivere una tabellina su ogni riga

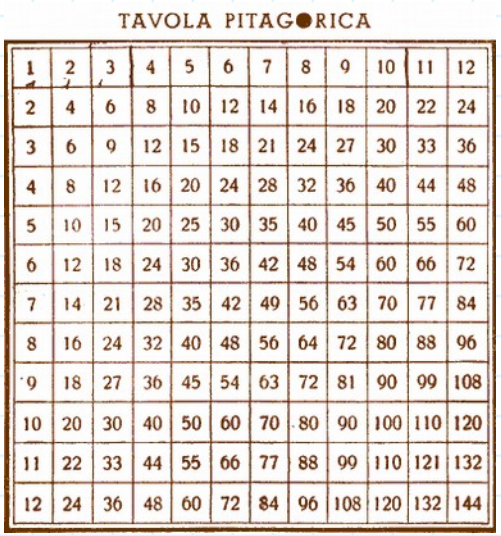


TAVOLA PITAGORICA

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

Suggerimento: provare a fissare per es. `y=3`*, e scrivere su una riga solo quella tabellina (ciclando su* `x`*)*
Aggiungere quindi un ciclo `for` *esterno, per variare la* `y`

Procedere sempre una riga alla volta (ltr →, ttb ↓)

2.3 Stringhe, cifre

- Leggere una riga di testo
  - `string line; getline(cin, line);`
- Riscrivere in output la riga, carattere per carattere, ma...
- Escludere tutte le cifre (0-9)

`line.size()` *è la lunghezza di* `line`
`line[i]` *è il* `char` *in posizione* i
Attenzione: indice da 0 *a* $size-1$

2.4 Numero casuale

- Generare un intero “segreto” a caso tra 1 e 90
- Chiedere ripetutamente all'utente di immettere un numero, finché non indovina quello generato
- Ogni volta dire se il numero immesso è maggiore o minore del numero segreto

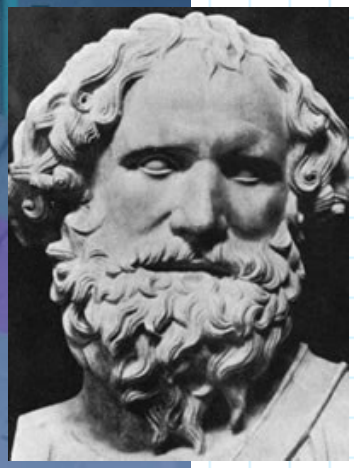
Generazione numero pseudo-casuale: `rand()`

Per avere valori diversi, eseguire all'inizio:
`srand(time(NULL))`

Librerie necessarie: `<cstdlib>`, `<ctime>`

2.5 Numero primo *

- Leggere un numero intero positivo
- Stabilire se esso è primo oppure no



Fermarsi a $\sqrt{n}$, saltando i pari

Eratostene: sia p il più piccolo divisore di n

$$n = p \cdot r,\ r \geq p$$

$$n \geq p \cdot p \rightarrow p \leq \sqrt{n}$$

2.6 Stringhe, maiuscolo *

- Leggere una riga di testo
- Riscrivere la riga, carattere per carattere, ma...
- Trasformare le minuscole (a-z) in maiuscole (A-Z)

Sottrarre al codice di una minuscola la costante `'a'-'A'`
Le minuscole infatti vengono dopo le maiuscole

2.7 Conteggio caratteri *

- Leggere da tastiera una stringa di testo
- Leggere un'altra stringa di caratteri
- Controllare quante volte i caratteri della seconda stringa sono presenti nella prima (totale complessivo)

Usare due cicli for annidati

Esercitazione 3

- Stringhe
- Array
- Allocazione dinamica

Richard Stallman

3.1 Stringhe, minuscolo

- Leggere da tastiera un testo generico
- Riscrivere il testo a console, ma...
- Convertire tutte le lettere in minuscolo
- Trasformare spazi e punteggiatura (*opzionalmente*, anche ripetuti) in un singolo trattino

Usare le stringhe C++
Segnare in un bool se l'ultimo char scritto è un trattino
Es. “It's 2010, yet” → “it-s-2010-yet”

3.2 Array

- Leggere una sequenza di N numeri
- Leggere un ulteriore numero
- Contare quante volte il numero è presente nella sequenza

Utilizzare un array

3.3 Matrice, scitala spartana

- Leggere un testo
- Inserire i primi M×N caratteri in una matrice di M colonne × N righe (M, N prefissati)
  - `char matrix[N][M];`
- Procedere a riempire la matrice ltr e ttb (→, ↓)
- Leggere dalla matrice
  e scrivere su console
  procedendo ttb e ltr (↓, →)

3.4 ASCII su più colonne

- Visualizzare la tabella dei codici ASCII
- Codici da 32 a 126
- Sistemare 10 coppie codice/carattere per ogni riga
- *Opzionalmente*: sistemare le coppie su 10 colonne, ma con i codici in progressione verticale, prima che orizzontale

Si procede sempre come macchina da scrivere (→, ↓)
Ma ∀ (x, y) si calcola il codice desiderato: $code = 32 + y \cdot w + x$
Bastano due cicli for annidati, senza array

3.5 Sottostringhe *

- Leggere da tastiera un testo generico
- Estrarre le parti comprese tra `'<'` e `'>'`
- Riprodurre in output, ciascuna su una riga, le parti selezionate

Usare le stringhe C++
Es. “Scrivete a <john@example.com> per informazioni”
→ “john@example.com”

3.6 Lettura file *

- Chiedere all'utente un nome di file di testo
- Chiedere all'utente un carattere
- Scorrere il file
- Contare tutte le occorrenze del carattere scelto

3.7 Allocazione dinamica *

- Come esercizio 3.3, ma `m` (colonne) ed `n` (righe) forniti dall'utente a tempo d'esecuzione
- Usare una (*o più*) delle seguenti soluzioni:

1. Allocazione statica, matrice ampia “a sufficienza”
2. Buffer di `m×n` caratteri (uni- o bi-dimensionale), allocato dinamicamente tramite `new`
3. Vettore di vettori; basta cambiare *una sola riga*:
   - ```
     vector< vector<char> >
       matrix(n, vector<char>(m));
     ```
     ```

Esercitazione 4

- Linea di comando
- Funzioni
- Ricorsione

James Gosling

4.1 Linea di comando

- Leggere una riga alla volta da un file di testo
- Copiare in un altro file le righe che iniziano con il carattere `'*'`
- Ricevere entrambi i nomi di file da linea di comando
- Altrimenti leggere i nomi dei file da tastiera

4.2 Cifrario di Cesare

- Leggere un testo da un file e riscriverlo, una lettera alla volta, in un altro file, ma...
- Trasformare tutte le lettere in maiuscolo
- Sostituire ciascuna lettera con quella che la segue dopo K posizioni nell'alfabeto
- Effettuare una rotazione analoga per le cifre
- Scartare ogni altro carattere

Usare l'operatore % per limitare i valori in un intervallo

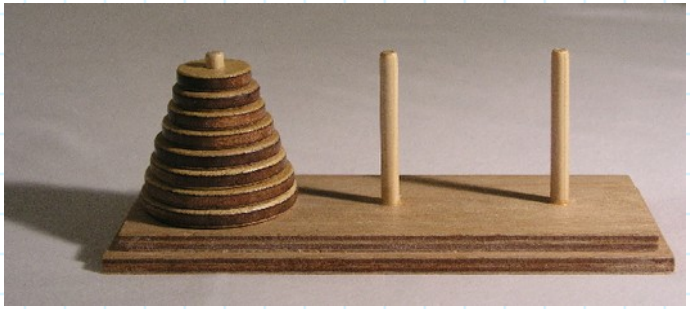
4.3 Ricorsione, MCD

- Leggere due numeri
- Calcolare il loro Massimo Comun Divisore
- Visualizzare il risultato

Provare ad usare sia l'iterazione che la ricorsione

Euclide: MCD(m, n) = MCD(n, m mod n), se n > 0;
MCD(m, n) = m, se n = 0

4.4 Torre di Hanoi



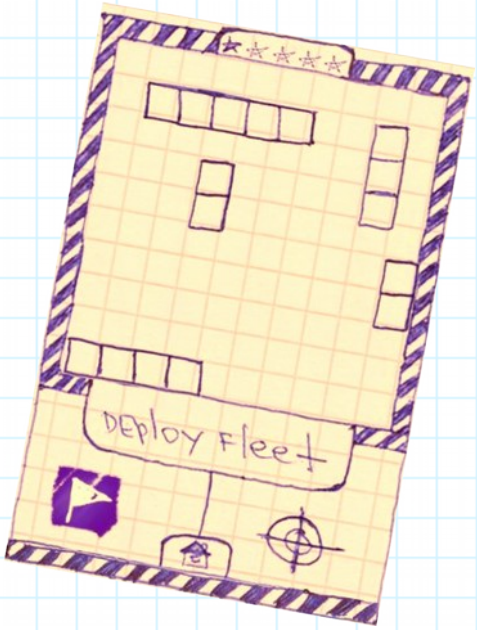
- Tre paletti + N dischi di diametro decrescente
- Portare tutti i dischi dal primo all'ultimo paletto
- Si può spostare solo un disco alla volta
- Non si può mettere un disco su uno più piccolo

Usare la ricorsione. Immediato spostare un solo disco.
N dischi: spostarne N-1 sul piolo né origine né dest.,
spostare l'ultimo disco sul piolo giusto,
spostare ancora gli altri N-1 dischi.

4.5 Hex dump *

- Leggere un file binario, un byte alla volta
- Scrivere i byte in esadecimale a console, disponendone 16 su ciascuna riga
- Dopo i 16 byte, visualizzare sulla stessa riga, più a destra, la stringa testuale corrispondente (sostituire i char non stampabili con spazi)

4.6 Battaglia navale *

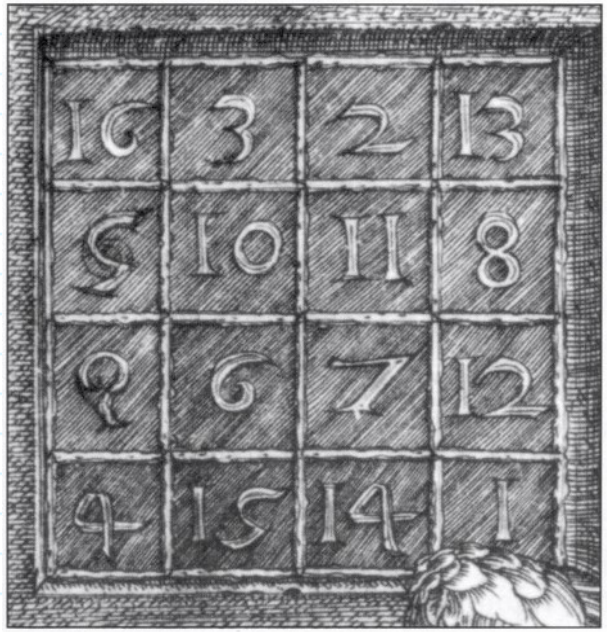


- Disporre a caso “navi” di varia lunghezza su una matrice
- Una nave occupa una striscia orizz. o vert. di celle adiacenti
- Concedere all'utente N “tiri”
- Se un tiro va a segno:
  comunicare all'utente la lunghezza della nave colpita; concedergli un tiro in più

Definire varie funzioni: per controllare la disponibilità di una striscia di celle, per disporre effettivamente una nave ecc.

4.7 Quadrato magico *

- Leggere N
- Creare e visualizzare una matrice N×N con tutti i numeri tra 1 e N^2
- Mantenere la somma costante su righe, colonne e due diagonali



$$M(n) = \frac{1}{n} \sum_{k=1}^{n^2} k = \frac{1}{2} n(n^2+1)$$

Ad ogni livello di ricorsione, provare ad inserire nella prima casella libera, uno alla volta, tutti i numeri rimanenti

Progetto 1

- Classe per incapsulare dati e regole di un gioco
  - Campi privati + metodi pubblici *(e privati)*
- Ciclo principale e interazione con l'utente (I/O, tramite console), ma *al di fuori della classe*

Ole-Johan Dahl e Kristen Nygaard

“Prato fiorito” - Base

- N fiori nascosti a caso in una tabella rettangolare
- L'utente sceglie una casella da scoprire, ad ogni turno
  - Se c'è fiore, partita persa
  - Altrimenti, conteggio fiori nelle caselle adiacenti
- Se restano solo caselle con fiori, partita vinta

“Prato fiorito” - Ricorsione

- Prato di varie dimensioni
- Quando viene scoperto uno “*0*”, si scoprono anche le celle adiacenti (ricorsione)
- L'utente può marcare una cella con una bandiera; se ci sono N bandiere:
  - Partita vinta (tutte su fiori)
  - Altrimenti persa
- Si può salvare una partita su file e poi ricaricarla

“Prato fiorito” - Gui

- Aggiungere una interfaccia grafica al progetto
  - Creare una sottoclasse di `QWidget`
  - Rendere l'interfaccia adattabile a dimensioni diverse del campo di gioco
- Riuso - Definire la classe di modello per le partite in modo generico
  - Usabile sia da interfaccia grafica che da console

Sono fornite due classi per gestire anche i click con il tasto destro del mouse

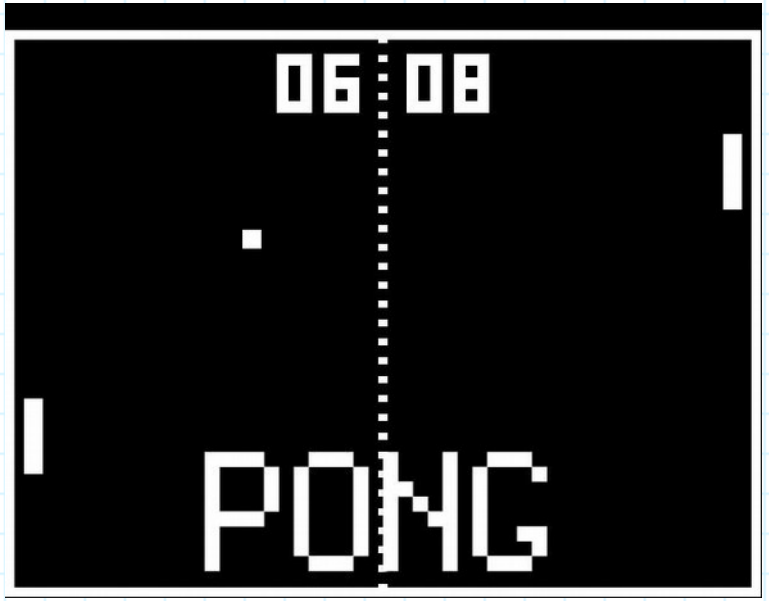
Progetto 2

- Realizzare un gioco con diversi personaggi
  - Classe per partita; classe base astratta (e sottoclassi) per personaggi
- Polimorfismo nei vari personaggi, per:
  - Il movimento
  - L'interazione reciproca

John Carmack

Pong - Base

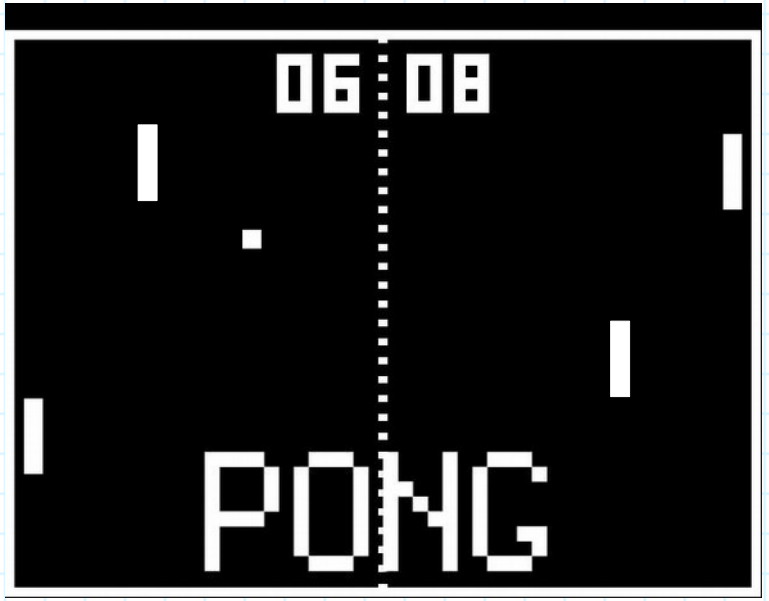
- Campo rettangolare
- Pallina: si muove a 45°, rimbalza su bordi lunghi e barrette
- Barrette: si muovono solo verticalmente
- Punti: segnati quando la pallina esce dal campo



È fornito un semplice framework di esempio

Pong - Doppio

- Sottoclasse di barretta: movimento automatico
- Partita di “doppio”
  - 2 squadre, 2 barrette per ogni squadra
  - Barrette distanziate (una più avanti e una più dietro)
  - Movimento sempre e solo in verticale
- A piacimento: ulteriori palline, ostacoli, bonus...



Pong - Opzionale

- Garantire ai personaggi un movimento libero, pixel a pixel (posizione x, y non fissata su una griglia rigida)
- Angolo di rimbalzo della palla sulla barretta dipendente da distanza dal centro della barretta
- Usare un oggetto `QPainter` nel metodo `paintEvent`, oppure...
- Una `QGraphicsScene` (con gli associati `QGraphicsItem` e `QGraphicsView`)