C++ · Esercizi 2012

Fondamenti
di informatica

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

Esercitazione 1

- Blocchi condizionali
- Cicli semplici

Ken Thompson, Dennis Ritchie (Unix)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

1.1 Hello, user!

- Compilare ed eseguire il programma “*Hello world*”
- In una versione successiva del programma...
- Chiedere il nome all'utente e aggiungere tale nome al messaggio di saluto
- Se il nome dell'utente è “*admin*”, allora salutarlo con il messaggio speciale “*At your command*”

1.2 Sfera

- Chiedere all'utente il raggio di una sfera e la sua densità (in Kg/m^3)
- Calcolare e visualizzare la superficie, il volume e il peso della sfera

Definire le variabili necessarie (nomi in minuscolo)

Definire una costante PI = 3.14159

Formule utili: $V = (4/3)\ \pi\ R^3$; $S = 4\ \pi\ R^2$

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

1.3 Caratteri ASCII

- Leggere un carattere
- Se è una lettera minuscola, visualizzare la corrispondente lettera maiuscola
- Altrimenti visualizzare il carattere immutato

Nel codice non usare esplicitamente i codici ASCII, ma simboli tra apici singoli (`'a'`, `'z'` ecc.)

La distanza di una lettera minuscola dalla corrispondente maiuscola è una costante che vale `'a' - 'A'`

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

1.4 Resistenze, ciclo

- Leggere, attraverso un ciclo, una sequenza di valori di resistenze elettriche
- La sequenza termina quando l'utente immette il valore 0
- Alla fine, visualizzare la resistenza equivalente, sia nel caso di resistenze disposte in serie, che in parallelo

$$R_s = \sum R_i$$

$$1/R_p = \sum (1/R_i)$$

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

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

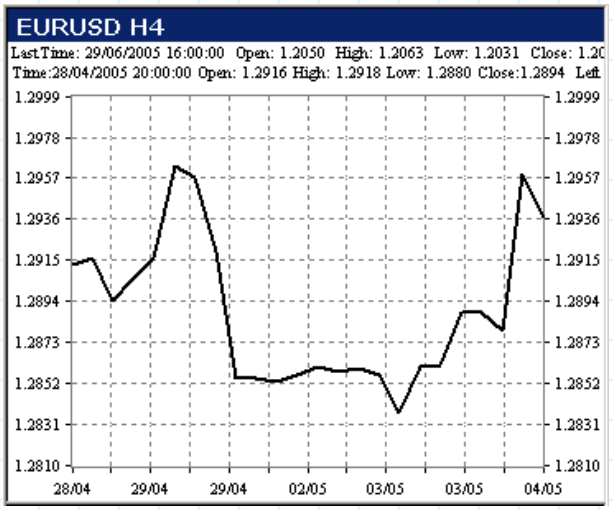
1.5 Condizioni

- Chiedere all'utente tre numeri interi: `a`, `b`, `c`
- Determinare qual è il massimo e qual è il minimo tra i tre numeri

Controllare prima di tutto se `a` è minore degli altri due, altrimenti controllare quale è il minore tra `b` e `c`...

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

1.6 Massimo e minimo

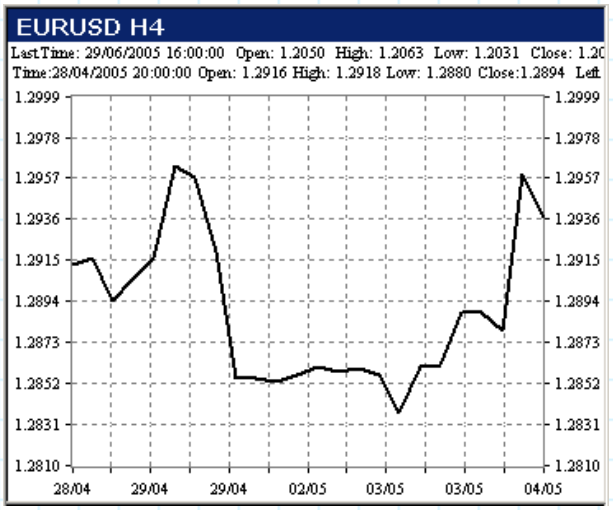


- Leggere, attraverso un ciclo, una sequenza di numeri interi
- La sequenza termina quando l'utente inserisce il valore 0
- Visualizzare il valore massimo e quello minimo tra i numeri inseriti

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

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

1.7 Chiusura stream



- Leggere, attraverso un ciclo, una sequenza di numeri interi
- La sequenza termina quando l'utente chiude lo stream della console
- Visualizzare il valore massimo e quello minimo tra i numeri inseriti

Usare un ciclo while

L'utente chiude lo stream della console premendo:
CTRL-Z (Windows)
CTRL-D (Unix ecc.)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

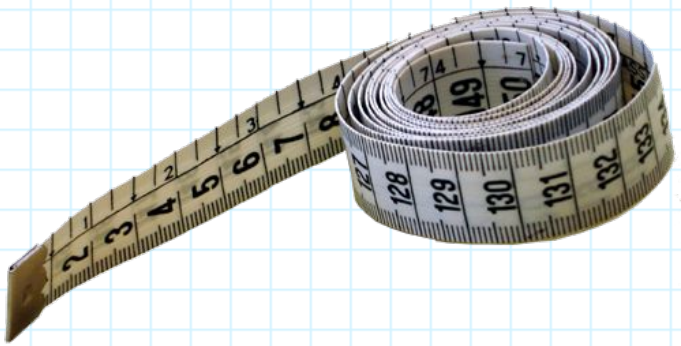
Esercitazione 2

- Cicli annidati
- Numeri pseudo-casuali
- Testo

Bjarne Stroustrup (C++)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

2.1 Lunghezza parole



- Leggere una sequenza di parole, attraverso un ciclo
- La sequenza termina quando l'utente chiude lo stream
- Visualizzare la lunghezza della parola più lunga e di quella più corta

Lunghezza di una variabile `word`
di tipo stringa: `word.size()`

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

2.2 Conteggio a ritroso

- Leggere un numero positivo `n`
- Per ciascun valore `y` tra `n` ed `1`...
- Stampare una riga con `y` ripetizioni di `y`

```
4444
333
22
1
```

Usare due cicli for annidati

All'inizio fissare y pari ad un certo valore
e scrivere una sola riga; es.: y = 4 → "4444"

Poi racchiudere tutto in un ciclo for esterno, per variare y:
y parte da n, arriva ad 1, ad ogni passo decresce di 1

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

2.3 Conteggio spazi

- Leggere una riga di testo
  - `string line;`
    `getline(cin, line);`
- Contare il numero di spazi presenti
- Se si incontra un carattere di tabulazione, contarlo come 4 spazi

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

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

2.4 Parentesi

- Leggere una riga di testo
- Scrivere in output solo le porzioni di testo comprese tra parentesi graffe
- Scrivere ciascuna porzione di testo su una riga

Es. "Bla bla {Hei!} bla {How boring!}" →
Hey!
How boring!

Segnare in un bool se si è all'interno delle parentesi

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

2.5 Tre carte

- All'inizio l'utente ha 10$
- Ad ogni turno...
  - Viene generato un numero casuale segreto tra 1 e 3
  - L'utente sceglie quanto puntare e su quale numero
  - Se idovina, gli viene sommato l'importo puntato, altrimenti gli viene sottratto lo stesso importo
- Il gioco termina quando l'utente perde tutto o si ritira

Generazione numero pseudo-casuale: `rand()`

Per avere sequenze diverse, aggiungere all'inizio del programma:
`srand(time(NULL))`

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

2.6 Calendario a muro

- Chiedere all'utente:
  - Il giorno della settimana (lunedì=0 … domenica=6) corrispondente al primo del mese
  - La lunghezza del mese (28-31)
- Visualizzare il calendario del mese
  - In prima colonna la data
  - In seconda colonna il giorno della settimana (0-6)

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

Se si invia su `cout` *il manipolatore* `setw(3)`*, l'output successivo occuperà almeno 3 posti (richiede la libreria* `iomanip`*)*

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

2.7 Calendario da tavolo

- Chiedere all'utente:
  - Il giorno della settimana (lunedì=0 … domenica=6) corrispondente al primo del mese
  - La lunghezza del mese (28-31)
- Visualizzare il calendario del mese
  - Disporre i giorni in una griglia, una riga alla volta
  - 7 colonne, la prima per i lunedì, l'ultima per le domeniche

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

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Esercitazione 3

- File
- Vector

Richard Stallman (GNU, FSF, Emacs, gcc)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

3.1 Righe su file

- Leggere testo da console, una riga alla volta
- Riscrivere il testo in un file, ma...
- Escludere le righe che contengono dei numeri

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

3.2 Valori da file

- Chiedere all'utente di definire un range di valori interessanti (fornendo max e min)
- Leggere da un file di testo una sequenza di valori interi (fino alla fine del file)
- Scartare i valori esterni al range definito dall'utente
- Scrivere a console i valori utili, ma in ordine inverso rispetto al file

Memorizzare tutti i valori utili in un vector

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

3.3 Agenda, vector

- Gestire una agenda per la giornata corrente (per ogni ora ci può essere una sola attività)
- Ciclicamente, l'utente può:
  - Inserire una nuova attività (indicandone l'ora e la descrizione)
  - Oppure rimuovere una attività dalla lista (indicandone l'ora)
- Al termine, visualizzare la lista delle attività, con ora e descrizione (omettendo le ore libere)

Usare un vector di 24 stringhe

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

3.4 Agenda su file

- In aggiunta all'esercizio precedente
- Prima della chiusura del programma, scrivere la lista di attività in un file (come a console, omettendo le ore libere)
- Ricaricare automaticamente la lista da file, se esistente, all'avvio del programma

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

3.5 Mescolamento vector

- Leggere due valori, `w` e `h`; calcolare `n = w * h`
- Creare un vector di lunghezza `n`
- Riempirlo con valori crescenti, a partire da 0
- Visualizzare le celle a console, su `h` righe
- Mescolare il vector
- Visualizzare nuovamente le celle su `h` righe
- Salvare il vector in un file, sempre su `h` righe

Mescolamento: per ogni cella, scegliere casualmente una nuova posizione; scambiare quindi il valore nella cella di origine con quello nella cella di destinazione

3.6 Compleanni

- Un file elenca i compleanni degli amici. Ogni riga contiene:
  - Il giorno del compleanno (testo con possibili spazi)
  - Dopo TAB, il nome dell'amico (testo con possibili spazi)
  - Dopo TAB, un commento (testo arbitrario, da scartare)
- Caricare tutti i dati utili in una `map`
- Ciclicamente, chiedere all'utente il nome di un amico e mostrare il giorno del compleanno

Per leggere nome ecc.: `getline(file, name, '\t')`

Si può anche analizzare ogni singola riga con `istringstream`

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

3.7 Calendario trasposto

- Chiedere all'utente:
  - Il giorno della settimana (lunedì=0 … domenica=6) corrispondente al primo del mese
  - La lunghezza del mese (28-31)
- Visualizzare il calendario del mese
  - Disporre i giorni in una griglia, una colonna alla volta
  - 7 righe, la prima per i lunedì, l'ultima per le domeniche

Console usata sempre come macchina da scrivere (→, ↓)
Ma ∀ (x, y) si calcola il numero da visualizzare

Bastano due cicli for annidati, senza vector

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Esercitazione 4

- Funzioni
- Classi
- Matrici

James Gosling (Java)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

4.1 Funzione, potenza dissipata

- Scrivere una funzione per calcolare la potenza dissipata da un resistore
- Parametri: resistenza e differenza di potenziale applicata, entrambi come `float`
- Risultato: potenza dissipata come `float`

Nel `main`: chiedere all'utente due valori, poi invocare la funzione con questi parametri e visualizzare infine (sempre nel `main`) il risultato restituito dalla funzione ($P = V^2/R$)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

4.2 Classe degli esami

- Scrivere una classe `Exam` per rappresentare gli esami sostenuti in un corso di studio
- Parametri del costruttore: *nome esame*; *numero crediti*; *data* (es. “2012-10-14”); *voto* (da 18 a 30)
- Fornire un metodo pubblico `estimateWork` per stimare le ore di studio, supponendo che…
  - Ad ogni credito corrispondano ~ 25h di studio
  - Il voto sia direttamente proporzionale allo studio

Nel `main`, istanziare un esame con valori forniti all'utente, invocare il metodo e mostrare il valore stimato.

Tenere `cin` e `cout` fuori dalla classe!

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

4.3 Battaglia navale

- Allocare una matrice `rows×cols` (dimensioni scelte dall'utente), ex.:
  - `vector< vector<char> > matrix(rows, vector<char>(cols, '-'));`
  - `vector<char> matrix(rows*cols, '-');`
- Ripetutamente...
  - Chiedere all'utente un numero `size`
  - Riempire con `'+'` un numero `size` di celle adiacenti (direzione e posizione di partenza casuali)
  - Accettabili inserimenti sovrapposti o parziali, fino a bordo
  - Mostrare la matrice risultante
- Al termine salvare la matrice in un file di testo

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

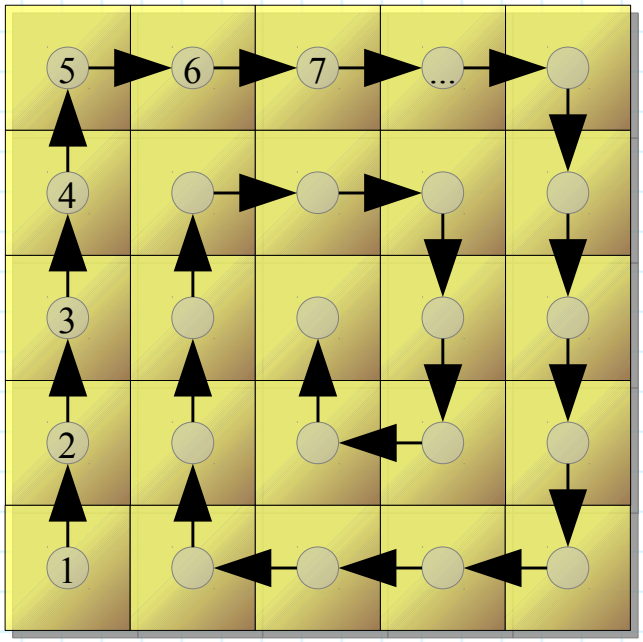
4.4 Lettura matrice da file

- Supponendo che il programma precedente salvi nel file solo il contenuto della matrice, riga per riga...
- Leggere i caratteri dal medesimo file e memorizzarli in una matrice
- Inferire le dimensioni della matrice (`rows×cols`) in base a:
  - Numero di righe del file
  - Lunghezza di ciascuna riga

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

4.5 Spirale

- Scrivere una funzione per riempire di numeri crescenti una matrice quadrata (o rettangolare)
- Seguire il percorso a spirale suggerito nella figura a fianco
- Dimensioni della matrice indicate dall'utente a runtime



Tenere traccia della direzione attuale (Δy, Δx)
Avanzare fino al bordo o ad una cella già visitata,
poi cambiare la direzione in senso orario

4.6 Elenco di esami

- Gestire un vettore di (puntatori a) esami
  - `vector<Exam*> exams;`
- Ripetutamente chiedere all'utente di scegliere tra:
  - Aggiunta di un nuovo esame
    ```
    Exam* ex = new Exam("FI", 9, "2013", 28);
    exams.push_back(ex);
    ```
  - Eliminazione di un esame esistente (deallocazione memoria + rimozione puntatore)
    ```
    delete exams[i];
    exams.erase(exams.begin()+i);
    ```
  - Calcolo delle ore complessive di studio
- Al termine di ogni operazione dell'utente, visualizzare l'elenco aggiornato di tutti gli esami

4.7 Esami in ordine

- In aggiunta all'esercizio 4.6, ordinare (e visualizzare) gli esami in base al tempo di studio stimato
  - Esiste una funzione standard per l'ordinamento
  - `#include <algorithm>`
  - `sort(exams.begin(), exams.end(), compare);`

Occorre definire `compare`*, funzione booleana con due param.:*

```
bool compare(Exam* ex1, Exam* ex2) { /*...*/ }
```

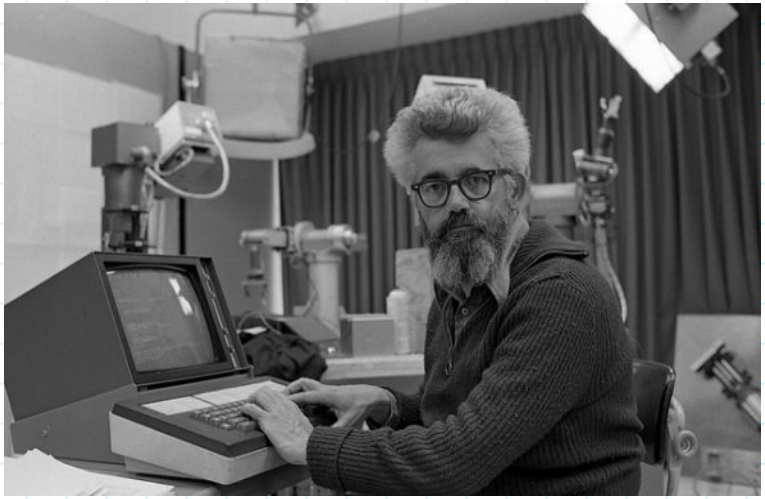
La funzione dice se i due parametri sono in ordine,
ossia se il primo parametro deve precedere il secondo

Non viene invocata direttamente, ma serve all'algoritmo sort

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Esercitazione 5

- Classi
- Ricorsione



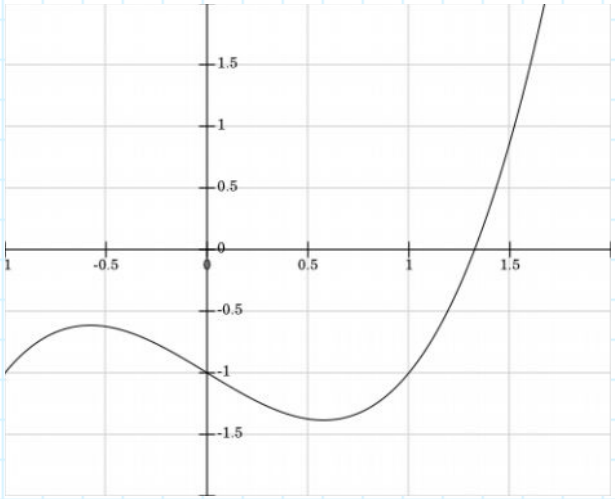
John McCarthy (LISP)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

5.1 Battaglia in classe

- Classe per incapsulare le funzionalità dell'es. 4.3
- Il costruttore riceve `rows`, `cols` e alloca la matrice
  - Es. campo privato: `vector<char> matrix;`
  - Nel costruttore: `matrix.assign(rows*cols, '.')`
- Metodo per l'inserimento casuale di una nave di dimensione `size` (`size` parametro del metodo)
- Metodo per la stampa dello stato su generico stream
  - Parametro `ostream & out`, passato per riferimento
- In `main`: creare un oggetto, inserire diverse navi a scelta dell'utente, infine scrivere la matrice su file

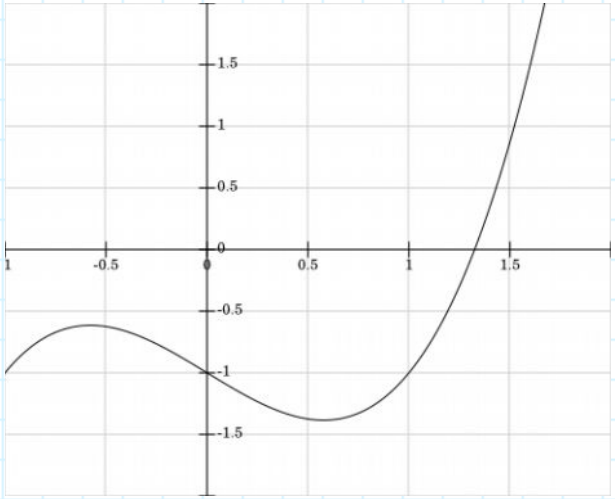
5.2 Bisezione, iterazione



- Trovare lo zero di una funzione
  - $f(x) = x^3 - x - 1$, *per* $1 \leq x \leq 2$
  - Trovare x *t.c.* $|f(x)| < 0.001$
- Dimezzare ripetutamente l'intervallo considerato:
  - Variabili per inizio e fine intervallo di ricerca
    All'inizio: `min = 1; max = 2;`
  - Si pone la stima `x` a metà intervallo e si calcola `f(x)`
  - Se l'errore è accettabile, `x` è il risultato
  - Altrimenti se `f` cambia segno tra `min` ed `x`, si ripete il ciclo considerando solo la prima metà dell'intervallo
  - Altrimenti solo la seconda metà (da `x` ad `max`)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

5.3 Bisezione, ricorsione



- Trovare lo zero di una funzione
  - $f(x) = x^3 - x - 1$, *per* $1 \leq x \leq 2$
  - Trovare x *t.c.* $|f(x)| < 0.001$
- Definire una funzione ricorsiva `findZero`
  - Parametri necessari: *inizio intervallo* di ricerca, *fine intervallo* di ricerca
- Bisezione: invocare ad ogni livello la funzione su un intervallo dimezzato

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

5.4 Contenimento, studenti ed esami

- Scrivere una classe `Student`
  - `name`, `surname`, `code`: `string`
  - `exams`: vettore di puntatori ad `Exam`
  - Costruttore: dati anagrafici come parametri
  - Metodi per aggiunta e rimozione esami
- Creare nel `main` un oggetto studente
- Permettere all'utente di gestire gli esami dello studente
  - Aggiunta
  - Rimozione

5.5 Composizione, studenti ed esami

- `Exam`, metodi di lettura/scrittura stato su stream
  - `void read(istream& in);`
  - `void write(ostream& out);`
- Aggiungere alla classe `Student`:
  - Metodo per lettura e metodo per scrittura su stream: leggere/scrivere tutti i dati anagrafici e tutti gli esami
  - Metodo per stimare il tempo totale di studio
  - Distruttore, cancella anche tutti gli esami
- Istanziare uno studente, caricando i dati da file
  - Permettere all'utente di aggiungere/rimuovere esami
  - Infine salvare dati anagrafici ed esami su file

5.6 Navi ben poste

- Aggiungere all'es. 5.1...
- Un metodo intelligente per il posizionamento casuale di navi
- La nave è posizionata solo se ci sono sufficienti celle libere, senza sovrapposizioni

Provare a realizzare il metodo sfruttando la ricorsione (ciascuna cella è riempita solo se è possibile riempire quella successiva)

Altrimenti basta contare le celle libere prima di riempirle

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

5.7 Labirinto

- Leggere da file un labirinto, con muri, punto di partenza e punto di arrivo
- Incapsulare la matrice (di `char`?) in una classe
- Fornire un metodo: `void print(ostream& out);`
- Fornire un metodo per avanzare di un passo, alla ricerca dell'uscita: `void move();`
- In `main`: chiamare ciclicamente `move` e `print`

Es. Avanzare seguendo il muro alla propria sinistra

http://blockly-demo.appspot.com/blockly/demos/maze/index.html

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Progetto 1

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

Ole-Johan Dahl e Kristen Nygaard (Simula)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Othello

- Due giocatori, bianco e nero
  - All'inizio: 2 pedine bianche e 2 nere nelle celle centrali, incrociate
- A turno, ciascun giocatore aggiunge una pedina
  - Se, tra la nuova pedina ed un'altra pedina dello stesso giocatore, c'è una **fila continua** di pedine avversarie, queste cambiano tutte colore (vale nelle 8 direzioni)
  - Si può mettere una pedina solo in una cella vuota, solo se si **catturano** pedine avversarie
- Se non ci sono mosse, si passa il turno
  - Ma, se ci sono mosse, non si può passare
- Vince chi alla fine ha più pedine

Othello - Metodi base per...

- Giocare: inserire una nuova pedina in una cella
  - Verificare se la mossa è possibile
  - Cambiare colore a tutte le pedine catturate
  - (Sfruttare, se possibile, la ricorsione)
- Scrivere lo stato del gioco su uno stream generico
- *Aggiungere inoltre le seguenti funzionalità...*
  - Poter passare il turno, se non ci sono mosse
  - Giocare su tavole di diverse dimensioni
  - Leggere lo stato del gioco da uno stream generico

Partita tra due utenti, non contro il computer!

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Othello - Bozza della classe

```
class Othello {
public:
  Othello(int rows, int cols); // custom board
  bool play(int y, int x);     // place disc and capture
  void write(ostream & out);

  bool pass();
  bool isFinished();
  void read(istream & in);

  // consider: char get(int y, int x)
  // getCurrentPlayer, getRows, getColumns …

private:
  // consider: void set(int y, int x, char value) …
}
```

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Othello - GUI

- Aggiungere una interfaccia grafica al progetto
  - Creare una sottoclasse di `QWidget` o `QMainWindow`
  - Evidenziare il giocatore di turno e le mosse possibili
  - Permettere di salvare/caricare una partita in un file
  - Interfaccia adattabile a dimensioni diverse del gioco
- Riuso - Classe di modello definita in modo generico
  - Usabile sia da interfaccia grafica che da console

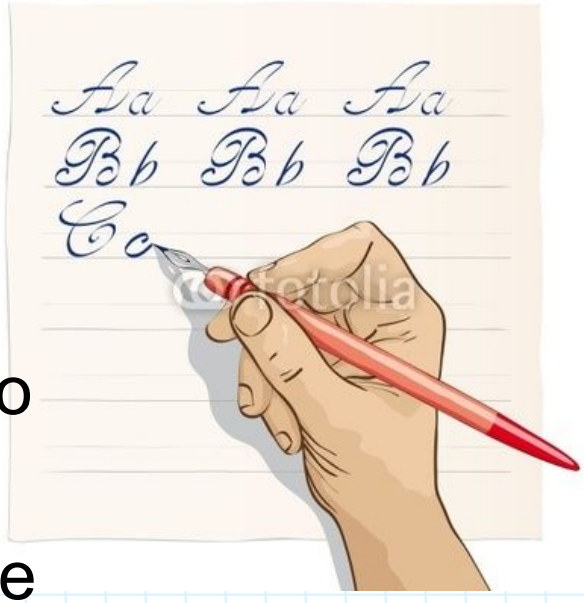
M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Othello - Bozza della GUI

```
class OthelloGui : public QWidget
{
    Q_OBJECT

public:
    OthelloGui(Othello* game);

private slots:
    void controlButtons(int i);

private:
    void updateAllButtons();
    void checkFinished();
    // ...

    QButtonGroup* buttons;
    Othello* game;
};
```

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

“Bella copia”



- Codice leggibile e ben organizzato
- Usare costanti, anziché valori numerici “*magici*” sparsi nel codice
- Nomi semplici ed esplicativi + regole di stile:
  - `nomeVariabile`, `nomeFunzione`, `NomeClasse`, `NOME_COSTANTE`
  - Indentazione, parentesi ecc.
- No *copia&incolla* del codice
  - Funzioni parametrizzate e/o cicli
- Programmazione strutturata
  - Un solo `return`, a fine funzione; niente `break` nei cicli

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Estensioni opzionali

- Suggerire / giocare mossa migliore
  - Valutare numero di pedine
  - Pedine + sicure: in angolo e collegate, sui bordi
  - Ricorsione minimax per analisi albero delle mosse
- Migliorare l'interfaccia utente
  - Menù e/o barra degli strumenti (`QMainWindow`)
  - Comandi per suggerimenti (di diverso livello)
- 5 pedine rosse “Stop!” per ogni giocatore
  - Sostituiscono la nuova pedina al termine della mossa
  - Bloccano catture successive (muro neutro)

Mossa migliore: minimax

```
int Othello::minimax(int depth) {
  int bestScore = -999; // bestY = -1; bestX = -1;
  if (!isFull() && depth > 0) {
    for (int y...) for (int x...) if canPlay(y, x) {
      Othello copy = *this; // copy the whole game
      copy.play(y, x); // play and change player
      // recursion, change the sign of the result
      int score = -copy.minimax(depth - 1);
      if (score > bestScore) {
        bestScore = score; // bestY = y; bestX = x;
      }
    } // if no moves: bestScore is very low!
  } else {
    bestScore = heuristic(); // how strong is current plr?
  }
  return bestScore;
}
```

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

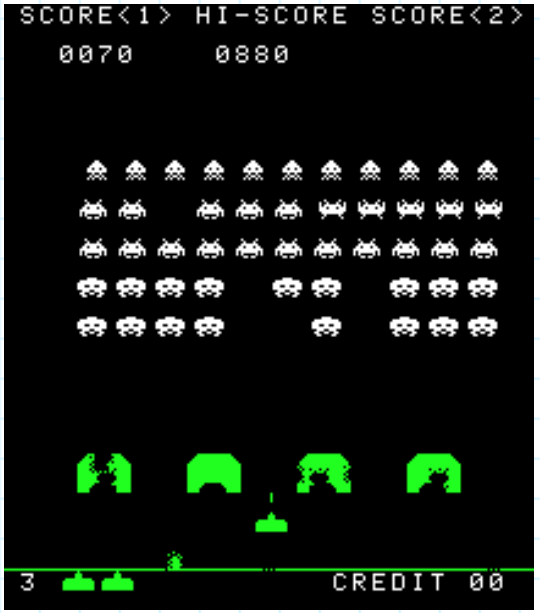
Progetto 2

- 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 (Wolfenstein 3D, Doom, Quake)

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Space invaders



- Cannone: guidato dall'utente
  si sposta in orizzontale ← / →

  Può sparare verso l'alto,
  contro gli alieni
- Alieni: si muovono tutti nella stessa direzione ← / →

  Se uno di loro arriva al bordo, al *turno successivo*
  scendono tutti di un passo e invertono la direzione

  Possono sparare verso il basso, contro il cannone
- Muri: si ditruggono lentamente dove colpiti

È fornito un semplice framework di esempio

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Space invaders - Opzionale

- Movimento dei personaggi libero e fluido
  (non su una griglia rigida, ma su qualsiasi pixel)
- Il cannone rimane sempre ancorato a terra...

- `QGraphicsScene` + `Item` + `View`, oppure...
- `QPainter` + `QWidget::paintEvent`

Mantenere un ciclo simile a `Game::moveAll`
e il polimorfismo di `Actor::move` *e* `Actor::touchedBy`

Non mischiare nelle stesse classi la logica
e la rappresentazione del gioco

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/

Space invaders - Estensioni

- Due cannoni contemporaneamente in gioco
- I personaggi possono occupare più celle
  - Es. 2 celle per gli alieni e 3 per i cannoni
- Icone al posto del testo nelle label (`QPixmap`, o stile)
- Altri personaggi *a fantasia*, per esempio:
  - Base madre aliena, se colpita dà vita e/o bonus
  - Alieni speciali, distrutti solo dopo alcuni colpi
- Completamento di varie missioni
  - Conteggio del tempo e assegnazione bonus finale
  - Vari bonus *a fantasia*, per punti, tempo, vite ecc.

M. Tomaiuolo – Fondamenti di informatica
Dip. Ingegneria dell'informazione – UniPR
http://www.ce.unipr.it/people/tomamic/