UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Dispense del corso di Fondamenti di Informatica 1

Presentazione del corso

Ultimo aggiornamento: 18/09/2020

Docenti

- **Prof. Costantino Grana**

  Dipartimento di Ingegneria «Enzo Ferrari»

  e-mail: costantino.grana@unimore.it

  Telefono: 059 205 6265

  Ricevimento studenti: Lunedì 15.00-17.00 **su appuntamento**

Prerequisiti

- Conoscenza di base dell'uso del calcolatore (accenderlo, spegnerlo, utilizzare tastiera e mouse, sapere qual è la differenza tra file e cartella, sapere che cos'è il nome del file e l'estensione, sapere che cos'è un browser internet, che cosa significa ridurre a icona)
- Competenze matematiche fondamentali quali:
  - l'aritmetica
  - il procedimento per eseguire le moltiplicazioni e le divisioni a più cifre
  - divisione tra numeri interi con quoziente e resto
  - i numeri decimali, razionali e reali
  - multipli e divisori
  - minimo comune multiplo e massimo comun divisore (davvero, dovete sapere che cosa sono)
  - il concetto di espressione e l'utilizzo di incognite
  - il simbolo matematico di sommatoria e il suo utilizzo

Programma

- Rappresentazione dell'informazione, sistema posizionale, basi e conversioni di base, rappresentazione in complemento, virgola fissa e mobile
- Il calcolatore elettronico: struttura e piccolo esempio didattico
- Il linguaggio C
  - Struttura di un programma: le funzioni
  - I tipi fondamentali
  - Variabili e costanti
  - Il processo di compilazione
  - Operatori fondamentali
  - Istruzioni di controllo
  - Puntatori
  - Array
  - Strutture
  - La libreria standard del C
- L'ambiente Microsoft VisualStudio

Note sulla modalità di insegnamento

- Introduzione delle funzionalità di input/output il più tardi possibile
  - Vantaggi: concentrarsi sulla sintassi del linguaggio; avere prima gli elementi di base e poi utilizzarli
  - Svantaggi: non si «vede» nulla → poca soddisfazione; i programmi devono essere ricompilati per modificarne il comportamento (fino a quando non introdurremo l'I/O)

  → mi direte se l'approccio ha funzionato o no (nei questionari di valutazione alla fine del corso)
- Non esistono «lezioni integrative». Quando troverete questa domanda nel questionario rispondete «non previste» (tanto vi dimenticherete).
- Le lezioni in laboratorio sono parte integrante del corso e saranno proprio quello su cui sarete valutati.
- Per superare l'esame non si deve «studiare» per giorni e giorni. Bisogna «programmare» per giorni e giorni (mesi e mesi…).
- L'unico programma corretto è quello che funziona. Non quello che «quasi» funziona. L'uso del debugger sarà essenziale.
- **Durante le lezioni potete fare domande: fatelo!**

Il materiale didattico

- Libro di testo:

  Tullio Facchinetti, Cristiana Larizza, Alessandro Rubini
  Programmare in C
  Maggioli Editore

- Libro degli esercizi:

  Costantino Grana, Marco Manfredi
  Esercizi di Programmazione in Linguaggio C: 80 problemi e soluzioni commentate

- I libri di testo vanno considerati come un punto di partenza e di riferimento, ma non possono essere il limite oltre il quale non si può andare. Traduco: le frasi «questo non c'è sul testo» o «questo non l'ha spiegato a lezione» non sono una scusa durante l'esame.

Il materiale didattico

- Sito web del corso su: http://dolly.ingmo.unimore.it/2020
- Sito con gli esercizi di programmazione per le prove di esame:
  https://olj.ing.unimore.it
- Alternative al libro di testo pubblicamente disponibili e gratuite presenti anche sul sito del corso.
- Qualsiasi altro libro sul C.
- Wikipedia e wikibooks:
  http://en.wikipedia.org/wiki/C_(programming_language)
  http://en.wikipedia.org/wiki/C_syntax
  http://it.wikibooks.org/wiki/C
  http://en.wikibooks.org/wiki/C_Programming
- Reference alla libreria standard e alla sintassi del C:
  http://cppreference.com
  http://cplusplus.com

Orario delle lezioni

- Lunedì: 09:00-11:00
- Martedì: 11:00-13:00
- Mercoledì: 14:00-16:00
- Giovedì: 14:00-16:00

- Alcune lezioni, soprattutto dalla seconda metà del corso in avanti, verranno svolte dividendo gli studenti casualmente in gruppi a cui sarà chiesto di svolgere autonomamente esercizi.
- Questo servirà a simulare le lezioni in laboratorio e speriamo sia efficace.
- Chiaramente la composizione casuale dei gruppi, causerà qualche problema di relazione, ma deve essere uno stimolo ad instaurare rapporti di lavoro corretti e a gestire lavori in gruppo.

Variazioni sull'orario

- Già da ora sappiamo che avremo queste modifiche:
  - 22 Ottobre: no lezione (Lauree)
  - 4 e 5 Novembre: no lezione (sospensione delle lezioni)
  - 3 Dicembre: no lezione (Lauree)
  - 7 e 8 Dicembre: no lezione (ponte dell'Immacolata)

- Eventuali ulteriori cambiamenti verranno comunicati durante il corso delle lezioni.
- L'ultima lezione è prevista per Giovedì 17 Dicembre.

Modalità di esame (1)

- Esame scritto (1 ora) con test a crocette su
  - le basi numeriche e operazioni con questi numeri (no calcolatrice!)
  - rappresentazione numeriche (interi e floating point)
  - aspetti del funzionamento dell'architettura di esempio (ADE8)
  - sintassi del linguaggio C
- Prova di programmazione in laboratorio (2 ore) su tutto quello che è stato visto a lezione.
- Entrambe le prove vengono valutate con un punteggio massimo di 33. La sufficienza, come da standard universitari, si raggiunge con un voto maggiore o uguale a 18.
- L'esame scritto **deve** essere superato (voto >=18) per poter accedere alla prova di programmazione.
- Il voto finale sarà la media pesata (arrotondata all'intero più vicino, 0,5 per eccesso) dei voti delle due prove, dove lo scritto pesa 1 e il laboratorio pesa 2.
- La lode si ottiene se il voto finale è 31 o più.

Modalità di esame (2)

- La prova scritta contiene 20 domande a crocette con 4 possibilità, delle quali solo una corretta. Ogni risposta corretta viene valutata 1,65 punti, ogni risposta sbagliata viene valutata -0,55 punti, ogni risposta non data 0 punti. Il voto finale è arrotondato all'intero più vicino, 0,5 per eccesso.
- La prova scritta, una volta superata rimane valida fino all'inizio del nuovo corso a Settembre 2021.
- Pertanto chi supera la prova scritta al primo appello, può sostenere il laboratorio molte volte.
- Chi supererà la prova scritta all'ultimo appello (quello di Settembre 2021) avrà una sola possibilità per sostenere il laboratorio.
- **Fate l'esame il prima possibile: non aspettate l'ultima possibilità.**
- In ogni anno accademico, sono previsti 6 appelli (ovvero possibilità di sostenere l'esame), di cui 5 obbligatoriamente a Gennaio, Febbraio, Giugno, Luglio e Settembre. Il sesto è a discrezione del docente e verrà probabilmente messo a Febbraio.
- **Non ci saranno altri appelli**, quindi non chiedetene.

Modalità di esame (3)

- Dopo ogni scritto il voto viene pubblicato sul sito del corso.
- Chi è risultato sufficiente ad uno scritto può partecipare al laboratorio.
- Dopo ogni laboratorio il voto viene pubblicato sul sito del corso.
- Inoltre la media (pesata come detto precedentemente) viene inserita su ESSE3 e inviata sul proprio libretto.
- A questo punto ogni studente riceve una mail e potete rifiutare il voto. Se non lo rifiutate entro la scadenza indicata (una settimana tipicamente), il voto viene verbalizzato e non può **mai più** essere modificato (per legge).

- Una volta superati lo scritto e il laboratorio, è possibile rifiutare la verbalizzazione (senza perdere i voti precedenti) e sostenerli nuovamente (NON FATELO!) per migliorare (O PEGGIORARE) il voto.

Frequently Asked Questions

- Il termine Frequently Asked Questions, abbreviato FAQ, significa «domande poste frequentemente».
- Sul sito trovate una serie di risposte a questo tipo di domande. Prima di contattare i docenti, **dovete** leggere questo file.
- Se rispondo ad una vostra email con «FAQ» e basta, sapete che cosa fare.

Perché il linguaggio C fa schifo

- Non esiste un tipo per le stringhe
- Quasi ogni operazione, per quanto semplice, richiede una funzione
- Gli array devono avere una dimensione fissata al momento della compilazione (sono praticamente inutili)
- Tante funzioni della libreria standard sono vulnerabili al buffer overflow
- Gli interi «sforano» senza avvisare e non c'è modo (nello standard) di controllare se questo accade
- Gli errori segnalati dal compilatore non sono sempre molto comprensibili
- La libreria di funzioni è vecchia e poco chiara; molte funzioni hanno varie versioni di dubbia utilità e rarissimo utilizzo
- È un linguaggio vecchio e gli mancano tutte le novità introdotte dopo: garbage collection, supporto per le istruzioni SIMD, supporto per i processori multicore o per cluster di macchine.

Allora perché lo dobbiamo studiare?!?!

- Linguaggi come il C++, il C# e il Java sono bellissimi e hanno costrutti eleganti, comodi e semplici. Peccato che per capirli sia necessario prima **capire le basi** e tutti questi linguaggi hanno le proprie basi nel C, a partire dalla sintassi.
- Velocità di esecuzione: **i programmi scritti in C sono i più veloci**. Per battere il C è necessario passare all'assembly (linguaggio macchina) e quindi addio alla portabilità.
- Gran parte dei sistemi operativi moderni sono scritti in C: Windows e Linux per fare un esempio. I device driver vengono scritti in C.
- Per i sistemi embedded il C è la scelta fatta dalla maggior parte dei progettisti.
- Pur se di basso livello, consente di programmare utilizzando una elegante programmazione strutturata.
- Pur se di alto livello, richiede di capire i concetti di memoria, indirizzo, allocazione e deallocazione, richiesta di risorse e rilascio delle stesse.

Popolarità dei linguaggi di programmazione

- http://www.tiobe.com/tiobe-index
- TIOBE Programming Community Index for September 2020:

Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	15.95%	+0.74%
2	1	↓	Java	13.48%	-3.18%
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	↑	PHP	2.49%	+0.62%
9	19	⇈	R	2.37%	+1.33%
10	8	↓	SQL	1.76%	-0.19%

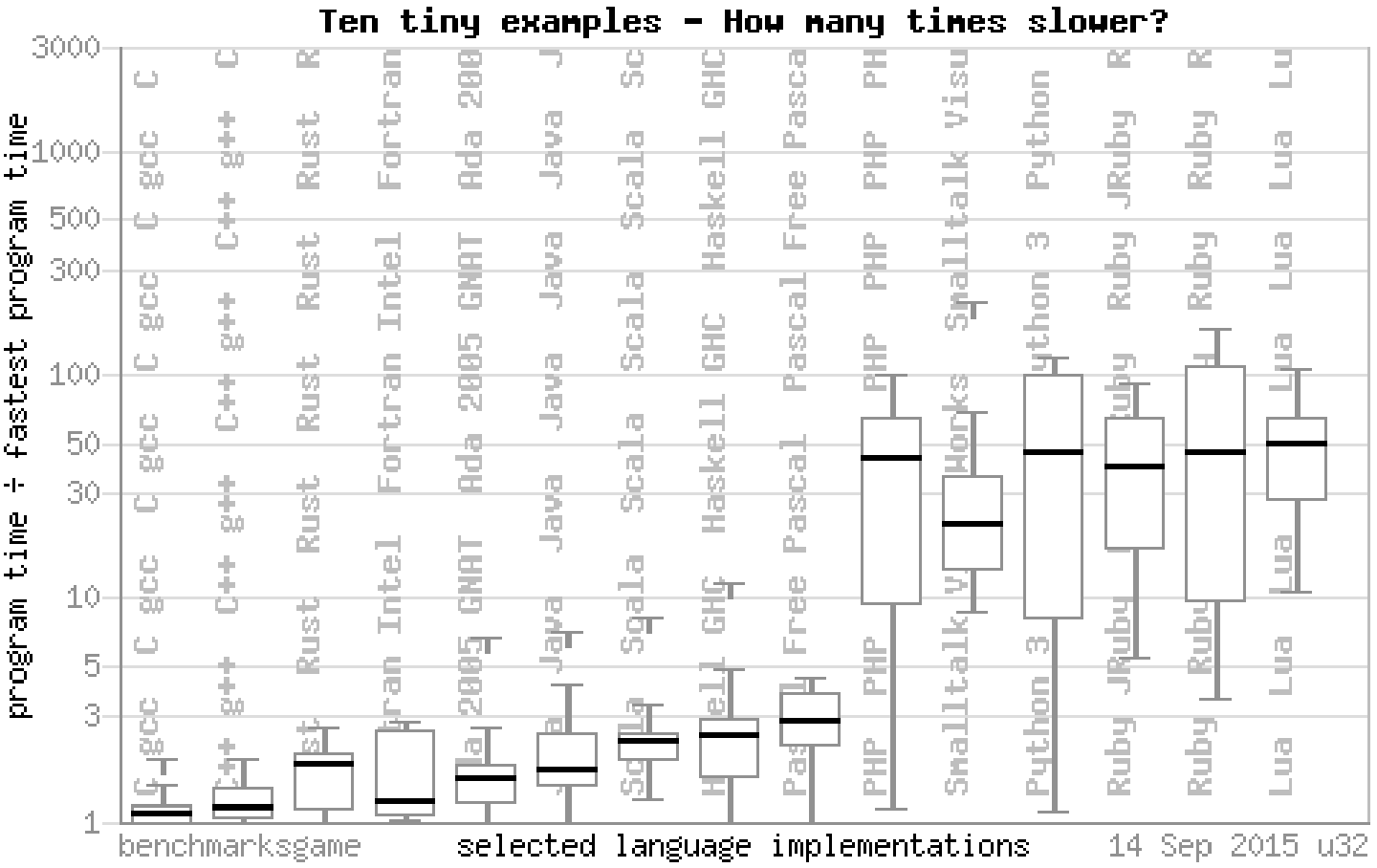
Popolarità dei linguaggi di programmazione

- http://pypl.github.io/PYPL.html
- PYPL PopularitY of Programming Language (Settembre 2020)

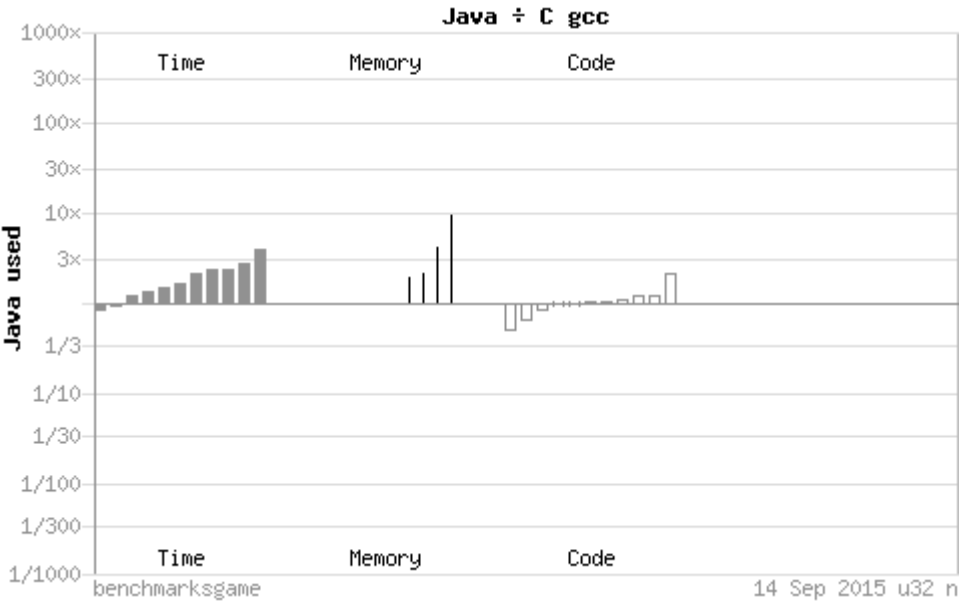
Rank	Change	Language	Share	Trend
1		Python	31.56 %	+2.9 %
2		Java	16.4 %	-3.1 %
3		Javascript	8.38 %	+0.3 %
4		C#	6.5 %	-0.8 %
5		PHP	5.85 %	-0.5 %
6		C/C++	5.8 %	+0.0 %
7		R	4.08 %	+0.3 %
8		Objective-C	2.79 %	+0.2 %
9		Swift	2.35 %	-0.1 %
10		TypeScript	1.92 %	+0.1 %

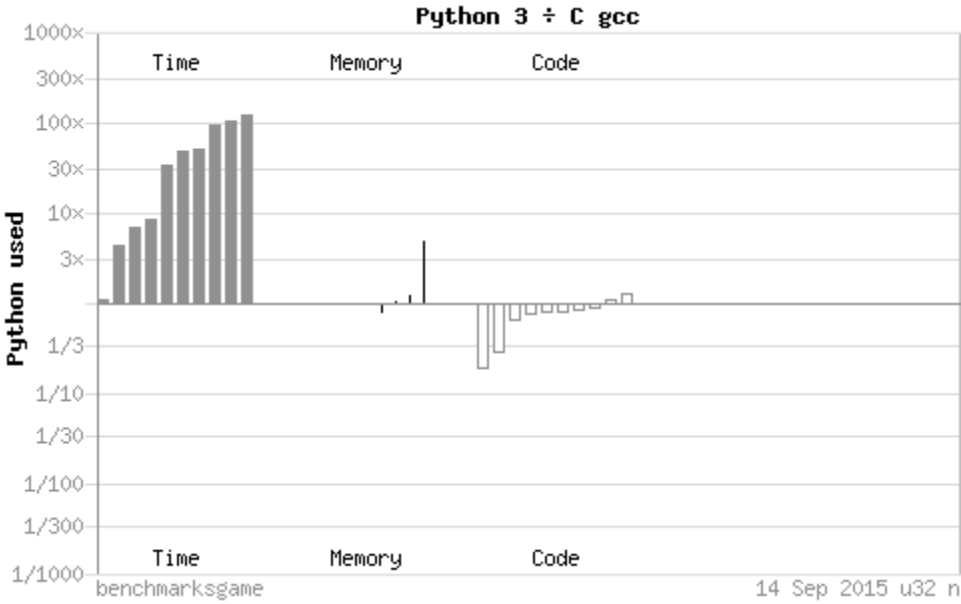
The PYPL PopularitY of Programming Language Index is created by analyzing how often language tutorials are searched on Google.

Performance dei linguaggi di programmazione



Performance dei linguaggi di programmazione





Difficoltà di Fondamenti di Informatica 1 e Lab.

- L'anno scorso (a.a. 2019/2020) la prova scritta è stata svolta da 286 studenti ed è stata superata da 246 studenti: 86%.
- Mediamente ogni studente ha provato 1,46 volte, massimo 5 volte.
- 2 studenti erano sufficienti, hanno rifatto lo scritto e sono risultati insufficienti, perdendo quindi il voto.
- Dei 246, 230 hanno affrontato la prova di laboratorio (93% di quelli che hanno fatto lo scritto) e 190 la hanno superata: 83%.

- Però solo 150 di questi erano iscritti al primo anno.
- Sui 324 studenti iscritti nell'a.a. 2019/2020, 51 hanno rinunciato agli studi, quindi gli studenti attualmente «attivi» sono 273. Di questi quindi il 53% ha superato l'esame.

- Per superare questo esame bisogna **programmare a casa** e fare tanto esercizio. Venire alle lezioni di laboratorio **non basta**!

Sapete già programmare in C? Un problema di esempio da risolvere in massimo 30 minuti.

- La società BigBurger Inc. vuole vedere se è fattibile avere una sola persona al bancone sia a prendere gli ordini sia a servirli. In ogni BigBurger, i clienti arriveranno e si metteranno in fila. Quando arrivano alla cassa fanno il loro ordine, che sarà preparato e servito. Poi lasciano la fila e la prossima persona in fila sarà in grado di ordinare. Abbiamo bisogno di sapere quanto tempo un cliente può essere costretto ad aspettare prima di poter effettuare un ordine.
- Dato uno script che elenca ogni cliente per una giornata tipo, vogliamo calcolare il tempo massimo di attesa del cliente. Ogni cliente nello script è caratterizzato da un orario di arrivo (misurato in minuti, dall'apertura del negozio) e una durata di servizio (il numero di minuti dall'ordine al servizio).
- Creare una funzione maxWait che riceve gli array di int *arrivi* e *servizio*, e descrive tutti i clienti. La funzione restituisce il tempo massimo trascorso da un cliente tra arrivo e l'ordine.

Un problema di esempio

- Gli elementi corrispondenti di *arrivi* e *servizio* sono riferiti allo stesso cliente, e sono dati nell'ordine in cui arrivano al negozio (*arrivi* è in ordine non decrescente).
- Se più clienti arrivano contemporaneamente, essi accederanno alla fila assieme, nell'ordine indicato negli array.

- Dichiarazione della funzione da realizzare:

```
int maxWait (const int* arrivi, const int* servizio, int n);
```

- *arrivi* conterrà da 1 a 50 elementi (estremi inclusi)
- *servizio* conterrà lo stesso numero di elementi di *arrivi*
- gli elementi di *arrivi* saranno in ordine non decrescente
- ogni elemento di *arrivi* sarà compreso tra 1 e 720 (estremi inclusi)
- ogni elemento di *servizio* sarà compreso tra 1 e 15 (estremi inclusi)

Esempi

- **Esempio 1:**
- arrivi = {3,3,9} - servizio = {2,15,14}
- Ritorno: 11
- Due clienti arrivano al tempo 3. Il primo aspetta 0 minuti, ordina e viene servito dopo 2 minuti e libera la fila al tempo 5. Il secondo ordina e viene servito al tempo 20. Nel frattempo un cliente arriva al tempo 9 e aspetta fino a che il secondo cliente non se ne va. Quest'ultimo cliente quindi ordina al tempo 20 e viene servito al tempo 20+14 = 34. Il primo cliente ha aspettato 0 minuti, il secondo 2 minuti (dal tempo 3 al tempo 5), e l'ultimo cliente ha aspettato 11 minuti (dal tempo 9 al tempo 20).

- **Esempio 2:**
- arrivi = {182} - servizio = {11}
- Ritorno: 0
- Il primo (e unico) cliente non deve aspettare.

Esempi

- **Esempio 3:**
- arrivi = {2,10,11} - servizio = {3,4,3}
- Ritorno: 3
- Il terzo cliente deve aspettare dal tempo 11 al tempo 14. Nessuno degli altri deve aspettare.

- **Esempio 4:**
- arrivi = {2,10,12} - servizio = {15,1,15}
- Ritorno: 7
- Il secondo è quello che aspetta di più.