Alcune note per iniziare a lavorare

Preparato per: studenti classe 1, corso di Programmazione 1
Preparato da: docenti del corso di Programmazione 1
Data: 26 settembre 2016
Revisione: 3

Lista delle revisioni

- Revisione 1 (19.09.2016)
- Revisione 2 (23.09.2016)
- Revisione 3 (26.09.2016)

Ingredienti necessari per superare l'esame

1) seguire il corso (ovvio)

2) studiare di volta in volta (ovvio)

3) esercitarsi, esercitarsi, esercitarsi…

OSSIA

scrivere codice, scrivere codice, scrivere codice…

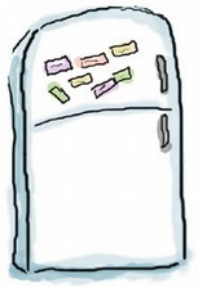
Dove? Quando?

In laboratorio (non serve *GUARDARE* il proprio collega che digita sulla tastiera, ma devi farlo anche tu!) e a casa.

Ecco quindi alcune note per poter lavorare a casa e in laboratorio, che ovviamente saranno integrate e commentate a lezione dai propri docenti.

COMUNICAZIONE. Gli studenti immatricolati riceveranno nel mese di ottobre una e-mail nella loro casella di posta @STUDENTI.UNISA.IT inviata dall'Università-Dipartimento di Informatica. Il Dipartimento offre agli studenti un abbonamento al programma Microsoft Imagine, che consente di scaricare software Microsoft gratuitamente.

Per accedere al negozio online bisogna completare la registrazione dell'account secondo quanto indicato nella e-mail. Quindi…. Occhio



Cut this out and stick it on your refrigerator.

Here’s what YOU can do to bend your brain into submission

So, we did our part. The rest is up to you. These tips are a starting point; listen to your brain and figure out what works for you and what doesn’t. Try new things.

1. **Slow down. The more you understand, the less you have to memorize.**

   Don’t just *read*. Stop and think. When the book asks you a question, don’t just skip to the answer. Imagine that someone really *is* asking the question. The more deeply you force your brain to think, the better chance you have of learning and remembering.

2. **Do the exercises. Write your own notes.**

   We put them in, but if we did them for you, that would be like having someone else do your workouts for you. And don’t just *look* at the exercises. **Use a pencil.** There’s plenty of evidence that physical activity *while* learning can increase the learning.

3. **Read “There Are No Dumb Questions.”**

   That means all of them. They’re not optional sidebars, ***they’re part of the core content!*** Don’t skip them.

4. **Make this the last thing you read before bed. Or at least the last challenging thing.**

   Part of the learning (especially the transfer to long-term memory) happens *after* you put the book down. Your brain needs time on its own, to do more processing. If you put in something new during that processing time, some of what you just learned will be lost.

5. **Talk about it. Out loud.**

   Speaking activates a different part of the brain. If you’re trying to understand something, or increase your chance of remembering it later, say it out loud. Better still, try to explain it out loud to someone else. You’ll learn more quickly, and you might uncover ideas you hadn’t known were there when you were reading about it.

6. **Drink water. Lots of it.**

   Your brain works best in a nice bath of fluid. Dehydration (which can happen before you ever feel thirsty) decreases cognitive function.

7. **Listen to your brain.**

   Pay attention to whether your brain is getting overloaded. If you find yourself starting to skim the surface or forget what you just read, it’s time for a break. Once you go past a certain point, you won’t learn faster by trying to shove more in, and you might even hurt the process.

8. **Feel something.**

   Your brain needs to know that this *matters*. Get involved with the stories. Make up your own captions for the photos. Groaning over a bad joke is *still* better than feeling nothing at all.

9. **Write a lot of code!**

   There’s only one way to learn to program in C: **write a lot of code**. And that’s what you’re going to do throughout this book. Coding is a skill, and the only way to get good at it is to practice. We’re going to give you a lot of practice: every chapter has exercises that pose a problem for you to solve. Don’t just skip over them—a lot of the learning happens when you solve the exercises. We included a solution to each exercise—don’t be afraid to **peek at the solution** if you get stuck! (It’s easy to get snagged on something small.) But try to solve the problem before you look at the solution. And definitely get it working before you move on to the next part of the book.

Passi fondamentali per programmare

1. Scrivere il programma (edit)
2. Compilarlo (compile)
3. Eseguirlo (run)

--Nel seguito si useranno, senza distinzione, i termini shell/finestra di comandi/Terminale--

Per lavorare in LABORATORIO

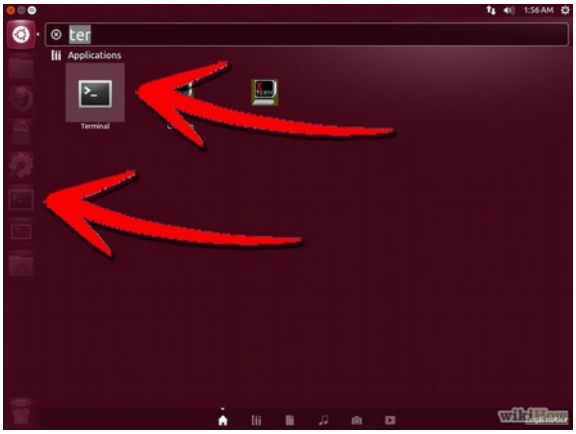
1. Richiedere le credenziali (login/password) per accedere alle postazioni dei laboratori didattici. Le modalità sono riportate a questo indirizzo: http://www.unisa.it/dipartimenti/dip_informatica/didattica/laboratori

2. Le postazioni sono dotate di DUAL-BOOT per la scelta del sistema operativo (all'accensione della macchina, viene proposta la scelta tra Windows e Ubuntu-Linux). Selezionare Ubuntu. Questo sarà anche il sistema operativo che utilizzerete per l'esame. Inoltre, è utile prendere familiarità con questo sistema operativo che studierete/userete anche per altri esami.

3. Ubuntu è dotato di ambiente grafico, che non lo discosta poi molto da Windows. Per esercitarsi per il nostro corso, è sufficiente aprire sul desktop (Scrivania) due finestre: una per il Terminale, una per un Editor di testo (qualsiasi).

Servirà per eseguire i comandi (compilare, eseguire,

Servirà per scrivere il file contente il codice in C



Dopo aver scritto il file .c, da finestra del terminale occorre COMPILARLO e poi ESEGUIRLO

Fase in cui si controlla la correttezza sintattica del codice scritto (comando

Fase in cui il codice scritto “prende vita” e consente di produrre l’output per cui era

Alcuni comandi utili da linea di comando sono riassunti a fine testo.

Per lavorare a CASA

- Non c'è un vincolo sullo strumento da utilizzare a casa per esercitarsi.
- Non c'è preferenza tra tipologia di sistema operativo (Windows, Linux, MacOs).
- *E' invece fortemente sconsigliato l'uso di piattaforme o IDE (Integrated Developments Enviroments, Ambienti di sviluppo integrato), come Code::Blocks o XCode(per Mac), Dev C-C++, o simili.*

PERCHE'?

Questi ambienti "aiutano troppo" nella scrittura corretta del file C, effettuando il completamento delle scrittura di parole codice, ad esempio. In questa fase di apprendimento iniziale, non è utile essere aiutati. Occorre IMPARARE… Un motivo non banale: durante l'esame questi strumenti non saranno disponibili. E' **necessario** quindi che ogni studente impari a scrivere file in C utilizzando esclusivamente un editor di testi qualsiasi e una finestra di terminale, per compilare ed eseguire i file. Per poter compilare i programmi in C, occorre che sia presente sul computer "gcc", il compilatore.

Text editor

Con il text editor si scrive il programma (in un file di testo con estensione .c) e poi con il compilatore si traduce il programma scritto nel linguaggio C in un programma scritto in linguaggio macchina detto eseguibile. Infine, l'eseguibile può essere eseguito come una qualsiasi altra applicazione sulla piattaforma (Linux, Windows, Mac OS X, ecc.) che si sta usando.

I text editor e i compilatori C che si possono usare dipendono dalla piattaforma, anche se ce ne sono molti che sono disponibili per diverse piattaforme. Qualsiasi applicazione che permette di scrivere file di testo semplice (plain text) va bene per scrivere programmi in C. Ad esempio i text editor disponibili di default sulle diverse piattaforme sono: vi/gedit sotto Linux/Unix, Notepad (Blocco note) sotto Windows, TextEdit sotto Mac OS X. Alcuni editor (come NotePad++ sotto Windows) colorano automaticamente il testo di un programma, in modo da aiutare nella scrittura. Questi sono consentiti.

Installazione e uso del compilatore C

Windows. Se si ha un computer con sistema operativo Windows, è possibile ottenere gcc in vari modi (di fatto simili). Ad esempio, indichiamo due possibilità, simili.

- Cygwin (https://www.cygwin.com/). E' un insieme di strumenti e programmi GNU[1] che consentono di utilizzare funzionalità Linux sotto Windows.

Per installarlo: Seguire la procedura di installazione mediante il programma setup.exe[2]. Così viene creata un'icona sul desktop che consente di eseguire una shell in cui è possibile usare gcc (e alcuni comandi Linux), senza altre operazioni di configurazione.

- MinGW (Minimalist GNU for windows - http://www.mingw.org/wiki/Getting_Started).

Per installarlo:

1. cliccare sul file ***mingw-get-setup.exe*** (https://sourceforge.net/projects/mingw/files/latest/download).
2. Come cartella di destinazione specificare C:\
3. Selezionare "Continue" e spuntare mingw32-base.
4. Completata la selezione (e marcatura del box associato), aprire il menù "Installazione" e selezionare l'operazione "Apply Changes".

Il compilatore gcc si trova nella directory C:\mingw32\bin (il nome mingw potrebbe essere scritto con lettere maiuscole/minuscole diverse da queste). Se si usa il Prompt dei comandi come shell, alcuni comandi Linux non vengono riconosciuti. L'essenziale è gcc, già utilizzabile.

Se si desidera invece avere una shell simile a quella di Linux, installare anche il pacchetto **MSYS Base System**. Qualsiasi altro pacchetto può essere successivamente installato utilizzando MinGw Installer, precedentemente scaricato.

Per usare gcc da finestra terminale in qualsiasi cartella (altrimenti i file dovrebbero essere salvati nella directory dove è presente il file gcc per poter essere compilati), occorre:

1. Selezionare la risorsa C: e, tenendo premuto il tasto destro, selezionare Proprietà>Avanzate>Variabili d'ambiente>Path.
2. Aggiungere alla fine della linea ;C:\mingw32\bin
3. Selezionare OK per ogni finestra aperta.
4. Affinché tutto vada a buon fine, occorre riavviare il computer.

---

1 GNU è un sistema operativo di tipo Unix. Il nome "GNU" è un acronimo di "GNU's Not Unix" ("GNU" si pronuncia gh-nu). Link: https://www.gnu.org/

2 Per eseguire un'installazione guidata, si consiglia la guida predisposta in questo documento: http://web.diegm.uniud.it/pierluca/public_html/teaching/fpac/strumenti/compilatore/istruzioni_per_installazione.pdf

Piccolo consiglio E' utile creare un collegamento sul desktop sia per NotePad, sia per il Prompt dei comandi o per la shell (si trova in C:\mingw32\msys\1.0\msys.bat).
Per creare il collegamento, trovare l'icona relativa al file/applicazione e selezionare "Crea collegamento - create shorcut" tenendo premuto il tasto destro del mouse.

Linux. In ambiente Linux il compilatore gcc è integrato, cioè è già presente. Seguire le stesse istruzioni date per il laboratorio per il suo utilizzo.

Mac OS X. Occorre scaricare il pacchetto "*Xcode Developers Tools*" dal sito https://developer.apple.com/ (per accedere registratevi o usate il vostro *Apple-ID*).
Attenzione: XCode prevede anche un IDE per scrivere i programmi. Si consiglia di non usarlo, ma di lavorare sempre da linea di comando (terminale).

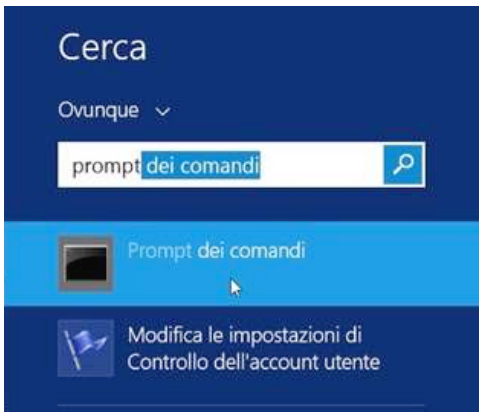
Per installarlo:

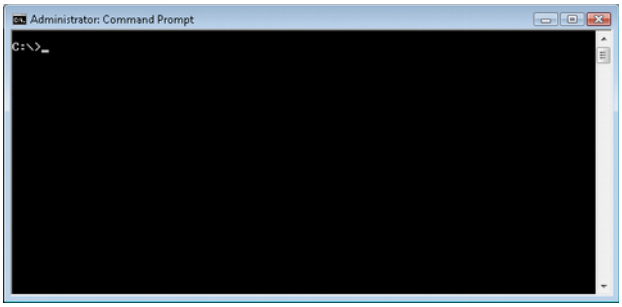
1. Installare Xcode.
2. Dal menù del programma, andare su Preferenze > Downloads > "Tool a linea di comando" > "Installa"
3. Dopo che il file .dmg è stato scaricato, verrà visualizzata una piccola finestra di dialogo con il nome del file – "Command Line Tools.mpkg". Fare doppio click su esso, seguire la guida per l'installazione.
4. Al termine, lanciare una finestra Terminale e digitare"gcc -v" . Se tutto è andato a buon fine, sarà visualizzata la versione del compilatore installata.

ALCUNI COMANDI DA LINEA DI COMANDO (FINESTRA TERMINALE)

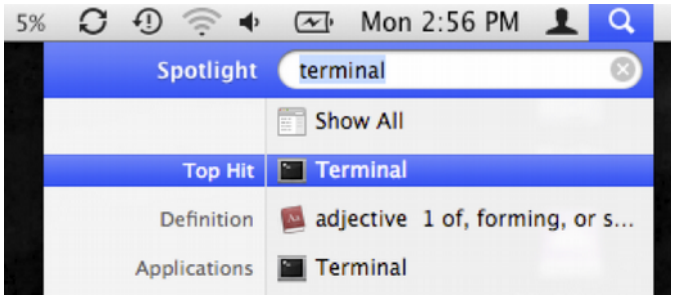
(IN CASA O IN LABORATORIO)

Dalla shell è possibile interagire con il sistema operativo scrivendo alcuni comandi, dopo il prompt (che può essere “>”, ad esempio).





Shell in Windows



Per Mac, basta cercare nello Spotlight “terminale” e verrà individuata l’applicazione

Un breve elenco dei comandi che saranno utilizzati per il nostro corso.

Comando	effetto
man	Mostra a video la documentazione per un programma (ad es. man ls mostrerà la documentazione sul comando ls)
ls	Elenca tutti i file presenti nella directory corrente. ls –l mostra i file specificando data e permessi di accesso al file
mkdir X	Crea una nuova directory di nome X come sottodirectory di quella corrente
rmdir X	Elimina la directory X (se vuota)

cd X	Scende nella sottodirectory X. Se usato senza argomenti, torna nella directory home. Se usato cd .. sale di un livello nelle directory
pwd	Mostra l'intero path della directory in cui ci si trova
mv old new	Rinomina il file old nel file new. Può spostare anche tra directory diverse
cp old new	Crea una copia del file old ottenendo il file new.
rm old	Cancella (definitivamente) il file old
chmod X	Cambia i permessi di accesso (lettura/scrittura/esecuzione) al file/directory X: vedere il manuale per i codici da scrivere.

Per interrompere un processo: se un programma in esecuzione non termina (ahimè, ne capiteranno tanti…), usare la combinazione dei tasti CTRL+C per interromperlo.

Per eseguire i programmi dopo aver scritto il codice in un file (salvato con estensione .c), occorre prima compilare, e poi eseguire. Per le tastiere italiane, il simbolo "{" (parentesi graffa aperta) si ottiene con la combinazione dei tasti CTRL+SHIFT+Alt+è (oppure AltGr+SHIFT+è), mentre il simbolo "}" (parentesi graffa chiusa) si ottiene con la combinazione dei tasti CTRL+SHIFT+Alt+* (oppure AltGr+SHIFT+*).

Supponiamo di aver scritto un file mio.c

1. Per effettuare la compilazione, aprire la shell, posizionarsi nella directory che contiene il file mio.c (utilizzando il comando cd seguito dal percorso per raggiungere la directory) e digitare

gcc mio.c

Nella finestra del terminale saranno visualizzati eventuali messaggi di "errore", che dovranno essere corretti dal programmatore (vedi la sezione dedicata ai messaggi, a fine documento). Dopo ogni correzione, il file dovrà essere nuovamente compilato, perché spesso gli errori sono legati uno all'altro. Finchè non saranno corretti tutti gli errori, non si potrà usare il programma. Potranno poi essere visualizzati dei messaggi di "warning", che sebbene non corretti, consentiranno ugualmente di procedere. Occorre però tenerli presenti, perché, se non corretti, potrebbero generare un comportamento anomalo nell'esecuzione del programma.

Se non ci sono messaggi di errore, a seguito del comando gcc mio.c, viene creato un file eseguibile nella stessa directory del file mio.c. Questo file eseguibile si chiamerà a.out oppure a.exe (a seconda del sistema operativo). Si può anche specificare il nome dei file eseguibile, digitando questo comando

gcc mio.c –o mio

A seguito di questo comando, viene creato il file eseguibile **mio.**

2. Per eseguire, occorre digitare

```
./a.out
```

oppure

```
./a.exe
```

oppure

```
./mio
```

a seconda del nome del file eseguibile.

In tal modo viene mostrato sul terminale, l'esecuzione del file.

Per conoscere la versione del compilatore presente sul proprio computer, digitare

```
gcc --version
```

I messaggi del compilatore

Il compilatore invia messaggi all'utente, che possono essere di due tipologie Questi messaggi si possono classificare in due famiglie: **warning messagges** – messaggi di avvertimento, e **error messagges** – messaggi di errore. I messaggi di avvertimento indicano la presenza di parti di codice presumibilmente mal scritte o di problemi che potrebbero avvenire in seguito, durante l'esecuzione del programma. I messaggi di avvertimento non interrompono comunque la compilazione.I messaggi di errore invece indicano qualcosa che deve essere necessariamente corretto .

Impareremo, usando, a leggere (capire) i messaggi di errore, fondamentali per correggere il codice scritto.

Questo file sarà aggiornato durante il corso, man mano che si aggiungeranno funzionalità più avanzate.

Buon lavoro!!!