Fondamenti di Informatica A
Laboratorio 9
Strutture dati in C

Danilo Pianini
danilo.pianini@unibo.it
Mirko Viroli
mirko.viroli@unibo.it

Alma Mater Studiorum—Università di Bologna

15 maggio 2013

Operazioni preliminari

- Accendere il computer
- Effettuare il log-in con le proprie credenziali istituzionali
  - Normalmente sono simili a:
    `nome.cognome@studio.unibo.it`
- Aprire Google Chrome, o altro browser
- Collegarsi al sito `http://bit.ly/finfa2013-e9`
- Scaricare il pacchetto `http://bit.ly/finfa2013-e9-code`
- Scompattare il pacchetto in una cartella a piacimento: questa sarà la vostra directory di lavoro di oggi.

`struct` e `typedef`

Si consideri il contenuto della cartella `es0`

- Si osservi con attenzione `point_base.h`. Cosa descrive?
- Si osservi con attenzione `use_point_base.c`. Solo dopo averne compreso appieno il funzionamento, si compili e si esegua.
- Si osservi ora `point.h`. In cosa differisce da `point_base.h`?
- Si osservi con attenzione `use_point.c`. Qual è il risvolto pratico dell'uso della keyword `typedef`? Solo dopo averne compreso appieno il funzionamento, si compili e si esegua.

Una libreria C

Si consideri il contenuto della cartella `es1`.

- `complex.c` implementa una libreria di operazioni sui numeri complessi
- `complex.h` è un file header che contiene la definizione della struttura dati per rappresentare numeri complessi e il prototipo di tutte le funzioni sui complessi implementate dalla libreria.
- `usecomplex.c` contiene un main che effettua alcune prove di funzionamento sulla libreria.
- Si comprenda il funzionamento della libreria, si compili e si esegua.
  - Per compilare la libreria:
    `gcc -c -Wall complex.c -o complex.o`
  - Per compilare il programma che usa la libreria:
    `gcc -Wall complex.o usecomplex.c -o usecomplex`
  - Per eseguire:
    usecomplex.exe (Windows)
    ./usecomplex (UNIX)

Una libreria C

Si consideri il contenuto della cartella `es1`.

- Si decommenti completamente il `main` di `usecomplex.c`
- Si implementi la funzione
  `Complex *complex_find_biggest(Complex *, int)`
  che, dato in ingresso un array di `Complex` ed un `int` rappresentante il numero di elementi nell'array, torna in uscita un puntatore al numero complesso di modulo maggiore.
- Il `main` decommentato può essere utilizzato per testare la vostra funzione.

Uso della libreria esportata

Data la libreria list fornita con i file `list.c` e `list.h`, definire un file `uselist.c` contenente un main nel quale realizzare i seguenti test

- Creazione di una lista (10,20,30,40):
  1. creare un array di interi contenente i 4 elementi specificati
  2. creare la lista utilizzando la funzione `List *list_from_array(int [],int)` al quale passare in ingresso l'array di interi appena creato, e che restituirà in uscita il puntatore alla lista appena creata
  3. Assegnare il puntatore a una variabile di tipo `List *`;
- Creare ora una lista (50,60), questa volta utilizzando la funzione `List *list_cons(int, List *)`, funzione che rappresenta il costruttore di una lista e che accetta in ingresso un intero e un puntatore ad una lista.

Uso della libreria esportata

- Stampare su console la rappresentazione delle due liste appena create utilizzando la funzione `char *list_to_string(List *)` che accetta in ingresso il puntatore a una lista e restituisce l'array di caratteri contenente la stringa rappresentante la lista.
- **SUGGERIMENTO**: per stampare la lista su console utilizzare la funzione `printf` e il parametro `%s`, come visto più volte a lezione.
- Stampare su console la dimensione delle due liste avvalendosi della funzione `int list_length(List *)`.
- Ora appendere alla lista la la seconda lista lb: utilizzare la funzione `void list_append_to(List *,List *)` nel seguente modo: `list_append_to(la,lb)`. In coda alla lista `la` si troverà ora la lista `lb`: accertarsene stampando su console la stringa rappresentante la lista e la sua dimensione.
- Compilare ed eseguire.

Uso della libreria esportata

- Si crei la funzione `void list_cut_zero(List *)` che, data una lista, la taglia al primo zero che incontra.
- Si testi tale funzione nel `main`.

Uso degli argomenti del programma

- Così come in Java, anche in C è possibile passare degli argomenti ad un programma. Per utilizzarli, il main deve avere la signature:
  `int main(int, char **)`
  Il primo parametro che viene inserito è il numero di elementi presenti nel secondo argomento, che rappresenta il comando che è stato eseguito nel suo intero.
- Si osservi il file `echo.c` nella cartella `es3`. Cosa realizza? Si compili e si testi il programma, invocando ad esempio
  `./echo mi piace informatica`
- **ATTENZIONE**: Non si nomini il programma `echo` (o `echo.exe`), ma si compili con altro nome (e.g. `gcc echo.c -Wall -o programma`). Questo perché `echo` è anche un programma di sistema, ed il sistema potrebbe invocare quello di sistema invece del vostro.
- Cosa accade se la variabile i nel main di `echo.c` viene inizializzata a 0 e non a 1? Si eseguano delle prove.

Conversione da stringhe a numeri

Gli argomenti passati sono sempre in formato stringa (char *). Così come Java, C mette a disposizione delle utility di libreria in `stdlib.h` per convertire tali stringhe in numeri (`int`, `long`, `double`) in modo da poterli usare internamente. Tali funzioni sono:

1. `int atoi (const char * str)` — Converte in `int`
2. `long int atol (const char * str)` — Converte in `long`
3. `double atof (const char* str)` — Converte in `double`

- Si osservi il contenuto di `input.c`. Cosa realizza?
- Si compili `input.c` e si provi ad utilizzare il programma con diversi argomenti. Cosa accade se si passa un letterale?
- Prendendo spunto da `input.c`, si realizzi in `sum.c` un programma che, dati in ingresso un numero arbitrario di argomenti di tipo intero, ne stampi a video la somma.

Puntatori a funzione

In C è possibile passare una funzione come argomento di una funzione. Si consideri il contenuto della cartella `es4`.

- Si osservi il contenuto di `functptr.c`. Cosa realizza?
- Si compili e si esegua `functptr.c`.

Puntatori a funzione

Si modifichi la libreria delle liste data prima come segue:

- In `list.h`, definisca una nuova funzione
  `void apply_to_all(List *, void (*)(int *))`
  che torna `void` e prende in ingresso
  1. un puntatore ad una `List`
  2. una funzione che prende in ingresso un puntatore ad `int` e torna `void`

  E applica la funzione f a tutti gli elementi della lista
- Si implementi in `list.c` tale funzione
- **SUGGERIMENTO**: Si scorra la lista fintanto che non si trova la lista vuota, e si applichi la funzione f al puntatore alla testa dell'elemento corrente.
- Si utilizzi la versione modificata di `uselist.c` fornita nella cartella `es4`
- Si definisca all'interno di `uselist.c` una funzione `void filter(int *)` che, dato in ingresso un puntatore ad un intero, modifica l'elemento puntato nel seguente modo:
  - Se l'elemento è pari, lo incrementa di 1
  - Se l'elemento è dispari, lo azzera
- Si osservi attentamente come viene utilizzata la `apply_to_all` nel `main`. Si compili e si esegua.

Esercizi aggiuntivi

Come al solito, se terminate in anticipo tutti gli esercizi precedenti, vi consigliamo di allenarvi con quelli nella cartella `aggiuntivi`.