Alcuni errori tipici

Assegnamento 1 -- 2016

rand() ed srand()....

- Per generare davvero una sequenza casuale
  - La **`srand()`** va chiamata una sola volta (per impostare il primo valore della sequenza)
  - Tutti i valori successivi vanno generati chiamando la **`rand()`**
  - Quindi il modo corretto di procedere per generare una sequenza casuale lunga N e'

```
srand (42);
for (i= 0; i < N; i ++)
{... x = rand(); .... }
```

| e || -- & e &&

- Alcuni hanno utilizzato le operazioni bit a bit invece degli operatori booleani
  - Attenzione il significato e' simile ma non bisogna fare confusione
  - Meglio usare solo gli operatori booleani (|| e &&) nei test delle condizione e usare gli operatori bit a bit quando stiamo davvero manipolando i bit

```
int a = 2, b = 4; /* 010 e 100 */
A & B vale 0
Mentre A && B vale 1
/* prova per esercizio ... */
```

Attenzione agli array!!!

- Alcuni hanno dichiarato array troppo piccoli
  - Quindi alla fine sono usciti dai bound dell'array
  - Anche se il programma funziona fate attenzione a come usate gli array e i loro indici
  - Se gli array sono allocati sullo heap potete usare valgrind per verificare i bound, lo vedremo nelle esercitazioni

Commenti

- Non devono ripetere passo passo quello che è evidente dal codice!
- è preferibile:
  - scrivere un commento esauriente all'inizio o alla fine del file, magari spiegando a parole l'algoritmo utilizzato
  - commentare sinteticamente all'inizio di una blocco di 5-15 istruzioni per spiegare cosa sta per succedere
  - commentare bene le variabili globali (inclusa la politica di modifica)
  - commentare brevemente ogni variabile locale dal significato non ovvio

Costanti esplicite nel codice

- *è PESSIMA norma* usare costanti numeriche o caratteri cablate nel codice

  Es:

```
char a[37], b[35], c[33];

for (i=0 ; i< 32; i++)
      a[i] = ' ';
```

  Questo rende difficile leggere e manutenere il codice (i valori sono legati fra di loro o scollegati ? Quel'e' il loro uso ?)

Costanti esplicite nel codice

- *è PESSIMA norma* usare costanti numeriche o caratteri cablate nel codice

  Es:

```
#define N 37
char a[N], b[N-2], c[N-4];

for (i=0 ; i< N-5; i++)
      a[i] = ' ';
```

  Bisogna fattorizzare le costanti intere e stringa con opportune **`#define`**, dando loro nomi significativi ed usare tali nomi consistentemente nel codice per esplicitare i legami

  – Ad esempio erac MEGLIO usare delle define per Nord Sud Est Ovest invece che 1,2,3,4

Costanti esplicite nel codice

- *è una PESSIMA IDEA usare direttamente gli interi dei codici ASCII invece delle costanti char*

  Es:

```
for (i=0 ; i< N-5; i++)
      a[i] = 32;
```

  – Bisogna usare invece la versione con la costante, più leggibile agli umani

```
for (i=0 ; i< N-5; i++)
      a[i] = ' ';
```

Organizzazione del codice

- Molti hanno scritto un codice lunghissimo esplicitando tutti i casi
  - Bisogna FATTORIZZARE il codice replicato raggruppando i casi uguali e NON copiare il codice replicandolo in tutti i punti che servono

Organizzazione del codice

- Variabili GOLOBALI: non c'era nessuna necessità di usare delle variabili globali nell'assegnamento
  - Eppure molti le hanno usate
  - E' importante abituarsi ad usare le globali solo quando non abbiamo altra scelta, altrimenti il codice scritto sara' caotico e confuso!
  - Prima di tutto cerchiamo di risolvere usando le locali, poi le static e solo come ultima ratio le globali

Tipo del main

- *Il main deve avere TIPO*

  Non sono accettabili tipi diversi da:

```
int main (void)
int main (argc, char * argv[])
```

  In particolare bisogna sempre ritornare l'esito del programma

  – Questo semplifica la vita a chi deve testare automaticamente il vostro codice (ad esempio io) ed e' quello che fanno tutti I programmi C

  – Alcuni non passavano I test sol operche' non restituivano 0

Tipo del main

- *In particolare*

  Non sono accettabili

```
int main () /* significa accetta un numero di
  ragomenti qualsiasi e disattiva il controll
  odel compilatore */
main () /* compila ed e' equivalente a quello
  sopra visto che il C se non conosce una
  funzione assume che ritorni un int */
```

  – Si iusavano tranquillamente nei vecchi programmi C ma sono sconsigliati (deprecated)

Miscellanea

- Il codice va INDENTATO correttamente se no si legge malissimo e non va farcito di ocmmenti lunghissimi
- La convenzione C e' che le macro usino nomi maiuscoli e le variabili minucoli.
  - Seguiamola – questo rende il codice più leggibile a voi ed agli altri