Approfondimenti sul ciclo while

Andrea Marin

Università Ca' Foscari Venezia
Laurea in Informatica
Corso di Programmazione part-time

a.a. 2011/2012

Introduzione

In questa lezione...

- Mostriamo semplici esempi di uso dei cicli
- Ciascuno di questi esempi mostra un metodo d'uso dei cicli che va capito a fondo e memorizzato
- Altri e nuovi problemi che vedremo in futuro potranno essere risolti riconducendosi ai questi *pattern*

Esempio 1

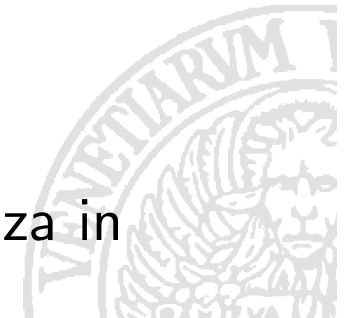
- Leggere una sequenza di numeri di standard input che termini con lo 0 e calcolarne la somma

```
int a;
int acc;  /*accumulatore*/
int main(){
   acc = 0;
   scanf("%d", &a);
   while (a != 0) {
      acc = acc + a;
      scanf("%d", &a);
   }
   printf("Somma=%d", a);
   return 0;
}
```

Inizilizzazione dell'accumulatore

Aggiornamento dell'accumulatore

- Applicare il pattern per calcolare la media della sequenza in input (che termina con lo zero).



Esempio 2 (proprietà universale)

- Leggere una sequenza di interi in input (che termina con lo 0) e stampare in output se sono tutti positivi (eccetto l'ultimo) oppure se ci sono anche dei numeri negativi
- Questo pattern è caratterizzato dal verificarsi di una proposizione su tutti gli elementi di un insieme
  - Siano i_1, i_2, ..., i_N, 0 la sequenza in input
  - Dobbiamo verificare la proposizione $\forall j \in [1, N], i_j > 0$
- Soluzione: assumiamo che la proprietà richiesta sia verificata, e cerchiamo un controesempio, cioè un elemento della sequenza in input che non soddisfi la proprietà richiesta

Soluzione

```
int a;
int flag;  /*stato della condizione*/
int main(){
   flag = 1;
   scanf("%d", &a);
   while (a != 0) {
      if (a < 0) {
         flag = 0;
      }
      scanf("%d", &a);
   }
   if (flag) {
       printf("Tutti i numeri sono poisitivi");
   }
   else printf("Almeno un numero è negativo");
   return 0;
}
```

Assumiamo la proprietà soddisfatta (flag con valore true)

Trovato un controesempio

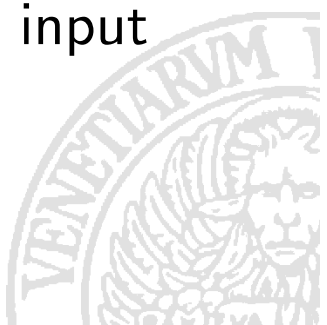
Soluzione ERRATA

```
int a;
int flag;  /*stato della condizione*/
int main(){
   flag = 1;
   scanf("%d", &a);
   while (a != 0) {
      if (a < 0) {
         flag = 0;
      }
      else {
         flag = 1;
      }
      scanf("%d", &a);
   }
   if (flag) {
        printf("Tutti positivi");
   }
   else printf("Almeno un numero è negativo");
   return 0;
}
```

Questo è l'errore, perchè?

Esempio 3 (proprietà esistenziale)

- Leggere una sequenza di interi in input (che termina con lo 0) e stampare in output se almento uno è positivo
- Questo pattern è caratterizzato dal verificarsi di una proposizione su almeno un elemento di un insieme
  - Siano $i_1, i_2, \ldots, i_N, 0$ la sequenza in input
  - Dobbiamo verificare la proposizione $\exists j \in [1, N], i_j > 0$
- Soluzione: assumiamo che la proprietà richiesta non sia verificata, e cerchiamo quell'elemento della sequenza in input che la soddisfi



Soluzione

```
int a;
int flag;  /*stato della condizione*/
int main(){
   flag = 0;
   scanf("%d", &a);
   while (a != 0) {
      if (a > 0) {
         flag = 1;
      }
      scanf("%d", &a);
   }
   if (flag) {
       printf("Almeno un numero è poisitivo");
   }
   else printf("Tutti i numeri sono negativi");
   return 0;
}
```

Assumiamo la proprietà non soddisfatta (flag con valore false)

Trovato un esempio

Esempio 4 (successivi di una sequenza)

- Leggere una sequenza di interi da standard input e fermarsi quando si leggono due valori consecutivi uguali.
- una soluzione errata:

```
int a;
int b;
int main(){
   a = 0;
   b = 1;
   while (a != b) {
      scanf("%d", &a);
      scanf("%d", &b);
   }
   return 0;
}
```

- Dare due esempi di sequenza per le quali il precedente programma non funziona

Soluzione

```
int a;
int prec; /*penultimo valore letto*/
int main(){
   scanf("%d", &prec);
   scanf("%d", %a);
   while (a != prec) {
      prec = a;
      scanf("%d", &a);
   }
   return 0;
}
```

Altri cilci

- Il linguaggio C mette a disposizione altre due strutture di controllo cicliche oltre al while
  - il ciclo `do .. while`
  - il ciclo `for`
- I cicli hanno un uso preferenziale, ma se un programma è scritto usando cicli `for` o `do .. while` allora può essere riscritto usando solo il ciclo `while`
  - vedi slide successive

Il ciclo `do .. while`

- Sintassi:

```
do
    blocco;
while (exp);
```

- Semantica:
  1. Esegue il codice di `blocco`
  2. Valuta l'espressione `exp`. Se è `true` si ritorna al punto 1. altrimenti si termina il ciclo
- A differenza del `while` questo ciclo esegue sempre almeno una volta le istruzioni di `blocco`

- Esempio: ripetere la lettura di un valore intero da standard input fino a quando se ne acquisisce uno positivo

```
int a;
int main() {
    do {
        scanf(``%d'', &a);
    } while (a <= 0);
    return 0;
}
```

- Attenzione: la condizioni dei cicli in C sono sempre quelli di permanenza

Confronto tra `while` e `do .. while`

do .. while	while

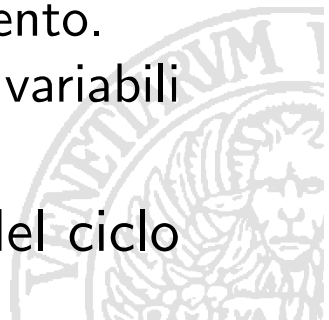
`blocco0;`
`do {`
`  blocco1;`
`} while (exp);`
`blocco2;`
	`blocco0;`
	`blocco1;`
	`while (exp) {`
	`  blocco1;`
	`}`
	`blocco2;`

Ciclo `for`

- Sintassi:

```
for(inizializzazione; condizione; incremento)
    blocco;
```

- Semantica:
  1. Valuta l'inizializzazione. Di solito sono degli assegnamenti che stabiliscono lo stato delle variabili all'inizio del ciclo. Questa parte può essere assente
  2. Si valuta la condizione. Se risulta false il ciclo termina immediatamente altrimenti si eseguono le istruzione di `blocco`
  3. Al termine dell'esecuzione di `blocco` si valuta l'incremento. Questa parte di solito indica come vanno aggiornate le variabili che determinano il numero di iterazioni del ciclo
- Buon uso del ciclo `for`: dalla lettura dell'intestazione del ciclo si deve sapere quante iterazioni saranno fatte



- Esempio: leggere 10 numeri da standard input e stamparne la somma in standard output
  - Conosco il numero di iterazioni!!

```
int i;
int a;
int somma;
int main() {
    somma = 0;
    for (i = 0; i < 10; i = i+1) {
        scanf(''%d'', &a);
        somma = somma + a;
    }
    printf(''La somma e %d \n'', somma);
    return 0;
}
```

Confronto tra `while` e `for`

for

```
blocco0;
for (inizializza; condizione; aggiorna) {
    blocco1;
}
blocco2;
```

while

```
blocco0;
inizializza;
while (exp) {
    blocco1;
    aggiorna;
}
blocco2;
```

goto

- Il linguaggio C mette a disposizione anche la possibilità di salto incondizionato
- L'idea è quella di etichettare delle istruzioni:
  - `identificatore: istruzione;`
- Quindi da qualsiasi parte del programma è possibile raggiungere un'istruzione etichettata con il goto:
  - `goto identificatore;`
- L'uso del goto è deprecato!!

Dijkstra, 1968

The unbridled use of the go to statement has as an immediate consequence that it becomes terribly hard to find a meaningful set of coordinates in which to describe the process progress. ... The go to statement as it stands is just too primitive, it is too much an invitation to make a mess of one's program.

Programmazione strutturata

- La programmazione strutturata emerge alla fine degli anni 60
- È alla base di molti moderni paradigmi di programmazione (inclusa la programmazione ad oggetti)
- Rifiuta l'uso del `goto` e si affida alle strutture di controllo: sequenza, if, while, do while, for
- Teorema di Böhm Jacopini: qualsiasi programma scritto usando il goto può essere riscritto senza, a patto di avere a disposizione altri tre tipi di strutture di controllo: sequenza, ripetizione e alternativa