Corso di Informatica

Modulo T1

2-Iterazione

Prerequisiti

- Salto condizionato
- Salto incondizionato
- Espressione logica

Introduzione

In alcuni casi si presenta la necessità di eseguire un ciclo, ossia di ripetere più volte (centinaia, migliaia, o più!!) un gruppo di istruzioni.

Questa situazione, richiede nella programmazione a basso livello, la presenza di un salto condizionato e di uno incondizionato.

Con un terzo tipo di struttura di controllo, possiamo evitare l'uso dei salti e rendere l'algoritmo molto più leggibile.

In questa Unità introduciamo le **strutture di controllo iterative**, con le quali è possibile realizzare cicli all'interno di algoritmi strutturati.

Iterazione

Problema: Leggere, per ciascuno di 4 studenti, il nome ed i 3 voti riportati. Stampare, per ciascuno, il nome e la media aritmetica dei suoi voti.

ATTIVITA': scrivere l'analisi del testo (**Fase 1**)

ATTIVITA': disegnare la tabella delle variabili

	Nome	**Tipo**	**Significato**
INPUT	nome	**Stringa**	Nome studente, sequenza di caratteri
	voto1	**Intero**	Primo voto
	voto2	**Intero**	Secondo voto
	voto3	**Intero**	Terzo voto
OUTPUT	nome	**Stringa**	Nome studente, sequenza di caratteri
	media	**Reale**	Media dei voti

Iterazione

Algoritmo MediaProve
Inizio
  Leggi (nome);
  Leggi (voto1, voto2, voto3);
  media=(voto1+voto2+voto3)/3;
  Stampa (nome, media);
  Leggi (nome);
  Leggi (voto1, voto2, voto3);
  media=(voto1+voto2+voto3)/3;
  Stampa (nome, media);
  Leggi (nome);
  Leggi (voto1, voto2, voto3);
  media=(voto1+voto2+voto3)/3;
  Stampa (nome, media);
  Leggi (nome);
  Leggi (voto1, voto2, voto3);
  media=(voto1+voto2+voto3)/3;
  Stampa (nome, media);
Fine.

ATTIVITA': scrivere l'algoritmo (**Fase 2**)

OSSERVAZIONI:

E' efficiente l'algoritmo?

No, lo *stesso gruppo di istruzioni* è scritto molte volte

Cosa succede se cambia il numero di studenti?

L'algoritmo non va più bene e deve essere modificato.

E se non si sa a priori il numero di studenti?

L'algoritmo non va più bene.

Vediamo allora come si può migliorare l'algoritmo

Iterazione

Algoritmo MediaProve
Inizio
  Ciclo:
  Leggi (nome);
  Leggi (voto1, voto2, voto3);
  media=(voto1+voto2+voto3)/3;
  Stampa (nome, media);
  Vai a Ciclo;
Fine.

Salto incondizionato

OSSERVAZIONI:

Il controllo usa etichette (ad esempio "**Ciclo**"), invece dei numeri dei passi

Tramite il controllo **Vai a** …il programma è più sintetico, ma…..

….**non termina mai**

Come risolvere questo ulteriore problema?

Iterazione

Salto condizionato

Algoritmo MediaProve
Inizio
  Ciclo:
  Se *studenti finiti* **Vai a Uscita;**
  Leggi (nome);
  Leggi (voto1, voto2, voto3);
  media=(voto1+voto2+voto3)/3;
  Stampa (nome, media);
  Vai a ciclo;
  Uscita:
Fine.

Salto incondizionato

OSSERVAZIONI:
Tramite il controllo
Se *condizione….*
il programma può terminare quando sono finiti i dati da esaminare

Iterazione

Per evitare la programmazione a salti, quando si deve eseguire ripetutamente un blocco di istruzioni, si usano le **strutture di controllo iterative.**

- Le strutture di controllo iterative servono ad eseguire i **cicli**.
- Un ciclo viene eseguito diverse volte in base al valore di una espressione logica (o condizione) che funziona come un interruttore.

Disponiamo di 3 s.d.c. iterative:

- **iterazione predefinita**
- **iterazione precondizionata**
- **iterazione postcondizionata**

Iterazione predefinita

Un primo tipo di sintassi è la seguente:

Per i = *valore-iniziale* **a** *valore-finale* **fai**
istruzione;

- Questa s.d.c. fa subire alla variabile *i* una variazione unitaria da un *valore-iniziale* ad un *valore-finale* e, per ognuno di questi valori, esegue l'*istruzione*.
- La condizione di uscita dal ciclo avviene quando la variabile *i* supera il *valore-finale*.
- La s.d.c. prende il nome di **s.d.c. iterativa predefinita**, poiché si usa quando è noto a priori il numero di cicli da eseguire.

Iterazione predefinita

- La variabile *i* viene detta **contatore** perché ha il compito di tenere traccia del numero di volte che il ciclo viene effettuato.
- La variazione unitaria può essere un **incremento** (aumento) o **decremento** (diminuzione) a seconda del tipo di problema considerato.
- In questa s.d.c. vengono automaticamente eseguiti:
  - l'**inizializzazione** del contatore a *valore-iniziale*;
  - la verifica della condizione di uscita.
  - l'incremento (o il decremento) del contatore;

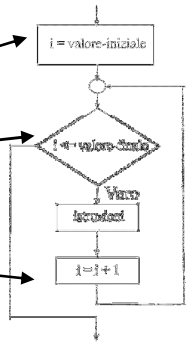
Iterazione predefinita

Mediante i ddf, l'iterazione predefinita si rappresenta come segue:

Si noti che nei ddf, le operazioni di:

- l'**inizializzazione** del contatore a *valore-iniziale*;
- la <u>verifica</u> della condizione di uscita
- <u>l'incremento</u> (o il <u>decremento</u>) del contatore;

<u>vanno scritti esplicitamente.</u>



Iterazione predefinita

Problema: Calcolare e stampare la somma dei primi *nmax* numeri interi, con *nmax* letto da Input

ATTIVITA': scrivere l'analisi del testo

Analisi del testo (Fase 1)

Indichiamo con l'intero *nmax* (dato esplicito) la quantità dei numeri, e con *somma* (dato esplicito) il valore della loro somma. Per risolvere il problema è sufficiente saper svolgere un'addizione. Il problema è solubile e i dati sono sufficienti. Per la verifica basta osservare che *somma* contiene la somma dei numeri da 1 a *nmax*.Il dato implicito è dovuto alla capacità di incremento del contatore.

ATTIVITA': disegnare la tabella delle variabili

	Nome	**Tipo**	**Significato**
INPUT	nmax	**Intero**	Quantità dei numeri da sommare
OUTPUT	somma	**Intero**	Somma degli *nmax* valori
	MSG	**Stringa**	"La somma è "

Iterazione predefinita

ATTIVITA': scrivere l'algoritmo in NLS (**Fase 2**)

Algoritmo SommaPrimiNumeri
Costante MSG "La somma è "
Inizio
  Intero nmax, numero, somma;
  somma=0;
  Leggi(nmax);
  Per numero =1 **a** nmax **fai**
    somma=somma+numero;
  Stampa(MSG, somma);
Fine

contatore

La stampa produce in Output il messaggio *MSG* seguito dal valore dell'accumulatore *somma*

Passo	numero	somma	nmax	msg
1		0		"La somma è "
2		0	3	"La somma è "
3	1	0	3	"La somma è "
4	1	1	3	"La somma è "
5	2	1	3	"La somma è "
6	2	3	3	"La somma è "
..	..	..	..	..
10	4	6	3	"La somma è "

ATTIVITA': disegnare la tabella di traccia per l'istanza *nmax=3*.

La variabile *somma* prende il nome di **accumulatore**, poiché all'interno di essa si accumula via via la somma dei vari valori progressivi di *numero.*

Iterazione precondizionata

Quando **NON** si conosce a priori il numero di volte che un ciclo deve essere ripetuto, si usa la seguente s.d.c:

Fintantochè (*espressione-logica*) **fai**

istruzione;

- Questa s.d.c. prende il nome di **s.d.c. iterativa precondizionata** in quanto il valore di *espressione-logica* viene valutato prima dell'esecuzione di *istruzione*.
- Il ciclo termina quando *espressione-logica* assume valore **FALSO.**

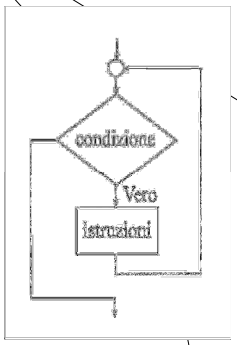
Iterazione precondizionata

Fintantochè (*espressione-logica*) **fai**
istruzione;

- Le operazioni di **inizializzazione** del contatore e del suo **incremento** o **decremento** *vanno scritte esplicitamente nel programma*, a differenza della s.d.c. predefinita.
- Poiché il controllo della condizione è posto all'inizio, *istruzione* potrebbe non essere MAI eseguito se la condizione si presenta subito con valore **FALSO**.
- Questa struttura di controllo esegue il ciclo fintantochè *espressione-logica* risulta **VERA**. Esce quando diventa **FALSA**.

Iterazione precondizionata

In ddf, questa struttura di controllo è molto simile a quella della precondizionata ed ha la struttura mostrata a fianco.



Iterazione precondizionata

Problema: Calcolare e stampare la somma di valori interi letti da input.

ATTIVITA': scrivere l'analisi del testo

Analisi del testo (Fase 1)

Usiamo la variabile intera *numero* per leggere i valori da input. La quantità dei numeri non si conosce, per cui il problema è solubile se si stabilisce un criterio di terminazione, ad es. quando *numero* vale 0. In tal caso, per risolvere il problema è sufficiente saper svolgere un'addizione nell'accumulatore *somma*.I dati sono sufficienti e per la verifica basta osservare che *somma* contiene la somma dei valori letti.

ATTIVITA': disegnare la tabella delle variabili

	Nome	**Tipo**	**Significato**
INPUT	numero	**Intero**	Numero immesso da Input
OUTPUT	somma	**Intero**	Somma dei valori immessi
	MSG	**Stringa**	"La somma è "

Iterazione precondizionata

ATTIVITA': scrivere l'algoritmo in NLS (**Fase 2**)

```
Algoritmo SommaValori
Costante MSG "La somma è "
Inizio
  Intero numero, somma;
  somma=0;
  Leggi (numero);
  Fintantochè (numero!=0) fai
  Inizio
    somma=somma+numero;
    Leggi (numero);
  Fine;
  Stampa(MSG, somma);
Fine
```

Condizione

La stampa il messaggio *MSG* seguito dal valore di *somma*

numero	**somma**	**msg**
	0	"La somma è "
7	7	"La somma è "
-2	5	"La somma è "
19	24	"La somma è "
0	24	"La somma è "

ATTIVITA': disegnare la tabella di traccia per l'istanza *7, -2, 19, 0.*

La variabile *somma* è l'**accumulatore** poiché all'interno di essa si accumula via via la somma dei vari valori progressivi di *numero.*

Iterazione postcondizionata

In altri casi, e sempre quando **NON** si conosce a priori il numero di volte che un ciclo deve essere ripetuto, si usa la seguente s.d.c:

Ripeti

istruzione

Fintantochè (*espressione-logica*);

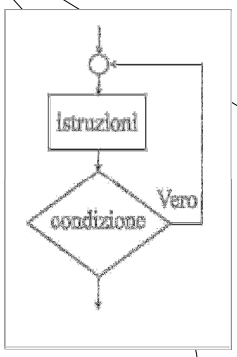
- Questa s.d.c. prende il nome di **s.d.c. iterativa postcondizionata**, in quanto il controllo di *espressione-logica* avviene dopo l'esecuzione di *istruzione*
- Anche qua, come nella iterazione precondizionata, le operazioni di **inizializzazione** del contatore, il suo **incremento** o **decremento** vanne scritte esplicitamente nel programma.

Iterazione postcondizionata

- Poiché il controllo della condizione avviene in fondo, *istruzione* sarà eseguita sempre ALMENO UNA VOLTA
- La s.d.c. iterativa postcondizionata è caratterizzata dalle parole **Ripeti-Fintantochè**.
- La s.d.c. iterativa postcondizionata ripete *istruzione* fintantochè *espressione-logica* ha valore **VERO**; esce dal ciclo quando *espressione-logica* assume valore **FALSO**.

Iterazione postcondizionata

Vediamo, infine, questa s.d.c. come si rappresenta nel linguaggio dei ddf.



Iterazione postcondizionata

Problema: Eseguire una lettura ripetuta da input di un valore reale e quando si immette un valore positivo calcolarne la radice quadrata.

ATTIVITA': scrivere l'analisi del testo

Analisi del testo (Fase 1)

Usiamo la variabile reale *numero* per leggere i valori da input e *radiceq* reale per calcolare la radice. Il criterio di terminazione dell'algoritmo è che *numero* sia positivo o nullo. La quantità dei numeri che vengono letti non è richiesta. In tal caso, per risolvere il problema è sufficiente saper svolgere l'operazione richiesta. I dati sono sufficienti e la verifica immediata.

ATTIVITA': disegnare la tabella delle variabili

	Nome	**Tipo**	**Significato**
INPUT	numero	**Reale**	Numero immesso da Input
OUTPUT	radiceq	**Reale**	Radice quadrata di numero
	MSG	**Stringa**	"Radice quadrata = "

Iterazione postcondizionata

ATTIVITA': scrivere l'algoritmo in NLS (**Fase 2**)

Algoritmo RadiceQuadrata

Costante MSG "Radice quadrata = "

Inizio

  Reale numero, radiceq

  Ripeti

    Leggi(numero);

  Fintantochè (**numero < 0**);

  radiceq =**Sqrt** (numero);

  Stampa(MSG, radiceq);

Fine

Condizione

numero	radiceq	msg
		"Radice quadrata"
-7.2		"Radice quadrata"
-2		"Radice quadrata"
25.0	5.0	"Radice quadrata"

ATTIVITA': disegnare la tabella di traccia per l'istanza *-72, -2, 25*

Il ciclo esegue una lettura ripetuta di *numero* fino al momento in cui questo assume valore maggiore o uguale a 0. *A* questo punto si calcola la radice di numero mediante la **funzione Sqrt()** e la si assegna a *radiceq*

La stampa produce in Output il messaggio *MSG* seguito dal valore della radice quadrata di *numero*

Lettura filtrata

La s.d.c. iterativa postcondizionata è utile, in questi casi per creare una parte di codice chiamata **filtro di input** o **lettura filtrata**, che ha la seguente struttura:

Ripeti

  Lettura dato;

Fintantochè (*dato non valido*);

Esecuzione a ciclo continuo

Un altro frequente utilizzo della s.d.c. postcondizionata è quando si desidera che il programma, terminata l'esecuzione e prodotti i risultati, chieda all'utente se intende ripetere il procedimento con altri dati. Ad esempio:

```
Carattere risposta;          /* variabile di tipo carattere */
….
Ripeti                       /* esecuzione a ciclo continuo */
   Esecuzione intero programma;
   Stampa ("Ripetere l'esecuzione [S/N] ?");
   Leggi (risposta);
Fintantochè (risposta!='N');
….
```


Il teorema di Jacopini-Bohm

Uno dei risultati più interessanti nell'ambito della programmazione è il **Teorema di Jacopini-Bohm** (1966) che afferma che:

> **Qualunque algoritmo descritto mediante la tecnica a salti, può essere sempre trasformato in un altro, equivalente, che faccia esclusivamente uso delle strutture di controllo sequenza, selezione ed iterazione.**

Argomenti

- Iterazione
- Iterazione predefinita
- Iterazione precondizionata
- Iterazione postcondizionata
- Lettura filtrata
- Esecuzione a ciclo continuo
- Il teorema di Jacopini-Bohm