Sequenze

Marco Alberti

Dipartimento
di Matematica
e Informatica

Università
degli Studi
di Ferrara

Programmazione e Laboratorio, A.A. 2020-2021

Ultima modifica: 19 ottobre 2020

Sequenze

- Sequenza: elenco, finito o infinito, di elementi dello stesso tipo
- Tipo degli elementi: qualsiasi
- Concetto presente in varie forme e con vari nomi. Noi vedremo sequenze di:
  - valori di variabile, espressione, input, output;
  - aree di memoria contigue (array);
  - dati memorizzati in modo consecutivo su dispositivo di memorizzazione di massa (file di record)
  - aree di memoria non contigue (lista collegata);

L'essenza di una sequenza è che si possono prendere i suoi elementi uno alla volta. Alcune *astrazioni* di linguaggi di alto livello (interfacce, moduli, classi di tipi) catturano questa essenza.

Operazioni su sequenze

- Creazione (ma a volte la sequenza ci viene data come input)
- Sulla possibilità di prendere un elemento alla volta si basano le operazioni complesse:
  - map
  - filter
  - reduce

In certi linguaggi le operazioni su sequenze sono esplicite (funzioni di libreria). In C sono (generalmente) implicite: pattern.

Dobbiamo sapere dove implementare queste operazioni sulle sequenze che incontreremo.

Dovremmo anche, dato un problema, sapere riconoscere quali sono le sequenze in gioco e quali operazioni ci servono per risolvere il problema; spesso la formulazione ci aiuta con parole chiave.

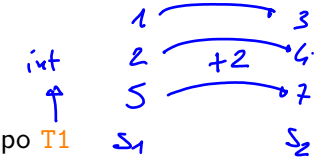
Creazione di sequenze

1. Scrivere un ciclo la cui variabile di controllo assuma tutti i valori compresi fra 1 e 10.
2. Scrivere un ciclo la cui variabile di controllo assuma decrescendo tutti i valori compresi fra 10 e 5.

1
2
3
4
5
6
7
8
9
10

10
9
8
7
6
5

map



Dati

- una sequenza S_1 di elementi di tipo `T1`
- una operazione o che si applica a un elemento di tipo `T1` e produce un elemento di tipo `T2`

map(o, $S1$) produce la sequenza S_2 di elementi di tipo `T2` ottenuta applicando `o` a ogni elemento di S_1

Doppio

Stampare la sequenza dei doppi dei numeri interi compresi fra 1 e 10.

```
for ( i=1 ; i<=10 , i++ ) {
// variabile assume tutti gli elementi di s1
printf("%d", i*2) , // operazione o
}
```

i assume i valori

1
2
3
4
5
6
7
8
9
10

s1

2
4
6
8
10
12
14
16
18
20

s2

Dati

- una sequenza S_1 di elementi di tipo `T1`
- una operazione p che si applica a un elemento di tipo `T1` e produce vero o falso

filter$(o, S1)$ produce la sequenza S_2 di elementi di tipo `T1` formata dagli elementi di S_1 per i quali p è vero.

Numeri pari

Stampare la sequenza dei numeri pari compresi fra 1 e 20.

ciclo

```
for(i=1; i<=20; i++){
    // i assume la sequenza 1,2,3 . .20 = S1
    if (i%2==0) // predicato i è pari
        { // S2 // i assume la sequenza 2,4,6,8,10 .. 20
          printf("%d\n",i);
        }
}
```

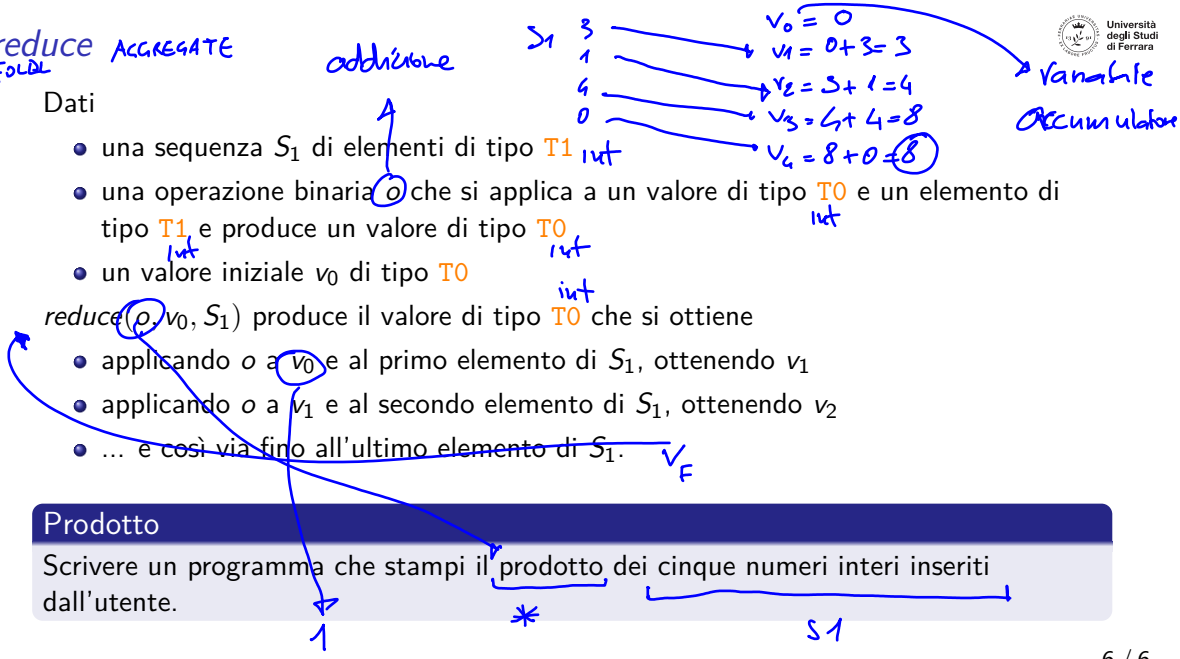
reduce

Dati

- una sequenza S_1 di elementi di tipo `T1`
- una operazione binaria o che si applica a un valore di tipo `T0` e un elemento di tipo `T1` e produce un valore di tipo `T0`
- un valore iniziale v_0 di tipo `T0`

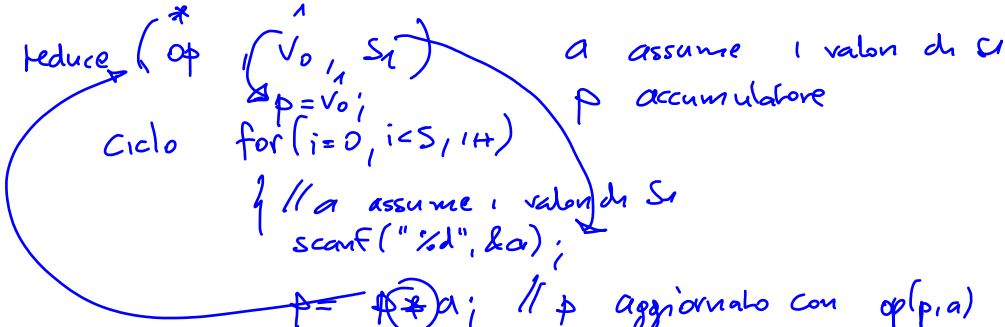
reduce(o, v_0, S_1) produce il valore di tipo `T0` che si ottiene

- applicando o a v_0 e al primo elemento di S_1, ottenendo v_1
- applicando o a v_1 e al secondo elemento di S_1, ottenendo v_2
- ... e così via fino all'ultimo elemento di S_1.



Prodotto

Scrivere un programma che stampi il prodotto dei cinque numeri interi inseriti dall'utente.


reduce (* , V0 , Sf)
a assume i valori di Sf
p accumulatore
p = V0 ;
Ciclo for (i = 0 ; i < Sf ; i++)
{ // a assume i valori di Sf
scanf ("%d" , &a) ;
p = p * a ; // p aggiornato con op(p,a)
}
// p vale di reduce (op , V0 , Sf)
printf ("%d\n" , p) ;