1

Programmazione e Strutture Dati (PR&SD)

I° ANNO – Informatica

Prof. V. Fuccella

Operazioni su Array

Eliminazione

Visita

Ricerca

2

Eliminazione di un elemento

Analisi

- Dati di ingresso: Array a di n elementi, posizione pos
  - Precondizione: $0 <= pos < n$
- Dati di uscita: Array a1 di n1 elementi
  - Postcondizione: $\forall i \in [0, pos-1]\ a1[i] = a[i]$ AND
    $\forall j \in [pos, n1-1]\ a1[j] = a[j+1]$ AND $n1 = n-1$

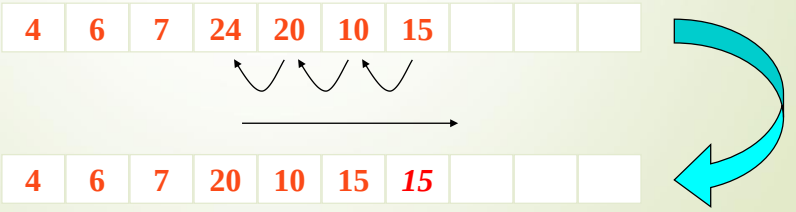
Dizionario dei dati

Identificatore	Tipo	Descrizione
a	array	array di interi in input
n	intero	numero di elementi nell'array a
pos	intero	indice dell'elemento da eliminare
a1	array	array di interi in output
n1	intero	numero di elementi nell'array a1

Eliminazione di un elemento
Progettazione

- Si spostano indietro tutti gli elementi dell'array *a* compresi tra le posizioni *pos+1* e *n*-1.
- Si decrementa *n*

Esempio: eliminare l'elemento in posizione 3



Errore tipico

- Un errore tipico consiste nell'eseguire il ciclo a conteggio scandendo le posizioni in senso decrescente
- il risultato è il ricopiamento a sinistra dell'elemento che si trova in posizione finale

Eliminazione di un elemento
Implementazione

```
/* la precondizione: 0 <= pos < *n deve essere
   controllata dalla funzione chiamante */

void elimina(int a[], int *n, int pos)
{
   int i;
   for (i = pos; i < (*n)-1; i++)
        a[i] = a[i+1];    // shift a sinistra

   (*n)--;
}
```


Visita degli elementi di un array

- ***Visita totale***: vengono analizzati tutti gli elementi
  - **In questo caso bisogna usare un ciclo a conteggio e visitare gli elementi dell'array in un verso o l'altro**
  - **for(i=0; i<n; i++) oppure for(i=n-1; i>=0; i--)**
- ***Visita finalizzata***: la visita termina quando un elemento dell'array verifica una certa condizione
  - **bisogna usare *due condizioni di uscita*:**
    - **una sull'indice di scansione (visitati tutti gli elementi si esce comunque dal ciclo)**
    - **l'altra che dipende dal problema specifico ...**
  - **Esempio: la ricerca termina se è stato trovato l'elemento o è stato visitato tutto l'array**

7

Ricerca di un elemento
Analisi

- Dati di ingresso: Array a di n interi, elemento el
  - Precondizione: n > 0
- Dati di uscita: Intero pos
  - Postcondizione: se el è contenuto in a allora pos è la posizione della prima occorrenza di el in a altrimenti pos = -1

Dizionario dei dati

Identificatore	Tipo	Descrizione
a	array	array di interi in input
n	intero	numero di elementi nell'array
el	intero	elemento da ricercare
pos	intero	indice dell'elemento trovato o -1

8

Ricerca di un elemento in un array
Progettazione

- Si scorre l'array di input finché non si trova l'elemento o non si raggiunge la fine dell'array (visita finalizzata)
  - Se l'elemento è stato trovato allora si restituisce la posizione corrente
    - in questo caso all'uscita del ciclo l'indice dell'array corrisponde a quello dell'elemento cercato
  - altrimenti si restituisce -1

9

Ricerca di un elemento in un array
Progettazione

- Si utilizzano:
  - un indice per scorrere l'array
  - una variabile booleana che ci dice se abbiamo trovato un elemento che soddisfa la condizione di uscita dal ciclo (preferibile rispetto all'uso dell'istruzione *break*)

```
int i = 0;
int trovato = 0;
```

- La condizione di permanenza nel ciclo sarà:

```
(i < n &&  !trovato)
```

- trovato sarà posta a 1 quando l'elemento corrente soddisfa la condizione definita dal problema

```
while(i < n &&  !trovato)
 if (a[i] soddisfa condizione)
          trovato = 1;
 else i++;
```

10

Ricerca di un elemento
Implementazione

```
int  ricerca(int a[], int n, int elem)
{
   int i = 0;              /* indice dell'array */
   int trovato = 0;   /* nello schema di visita indica che è stato trovato */

   while(i < n &&  !trovato)   /* visita finalizzata */
        if (a[i] == elem)
              trovato = 1;    /* permette di uscire dal ciclo   */
       else i++;               /* se non trovato incrementa l'indice  */

   /* se trovato restituisce la posizione i dell'elemento
       altrimenti restituisce -1                                          */

    return (trovato ? i : -1);
}
```

11

Short circuit evaluation

- Invece dello schema di visita finalizzata precedente

```
while(i < n &&  !trovato)
        if (a[i] == elem) trovato = 1;
        else i++;
```

- I programmatori C spesso utilizzano il seguente

```
while(i < n &&  a[i] != elem)
        i++;
```

  - Nota che la condizione (i < n && a[i] != elem) è corretta in C grazie alla short-circuit evaluation …
  - Infatti quando i == n la condizione a sinistra dell'operatore && è falsa e la condizione a destra dell'operatore && non viene valutata (se venisse valutata a[i] sarebbe indefinito)
  - In molti linguaggi di programmazione non c'è la short circuit evaluation

12

Ricerca lineare in un array ordinato

- Se l'array è ordinato in senso crescente, non è necessario arrivare alla fine dell'array per stabilire che l'elemento non è stato trovato ...
- … ci si può fermare appena si trova un elemento maggiore (o uguale) di quello dato …
  - maggiore --> non trovato !
  - uguale --> trovato !
- Ovviamente, se l'elemento è maggiore di tutti quelli presenti nell'array, allora si visiterà l'intero array (caso peggiore) ...

Esempio

- Dato l'array ...

4	6	7	10	12	15	18	20	24	30

La ricerca di 13 e la ricerca di 15 terminano
quando l'elemento corrente è 15

La ricerca di 40 termina quando si sono visitati tutti gli elementi dell'array

Ricerca lineare in un array ordinato
Con short-circuit evaluation

```
int  ricercaord(int a[], int n, int elem)
{
   int i = 0;

   while(i < n &&  a[i] < elem)   // visita finalizzata
         i++;

    return (a[i]==elem ? i : -1);
}
```

15

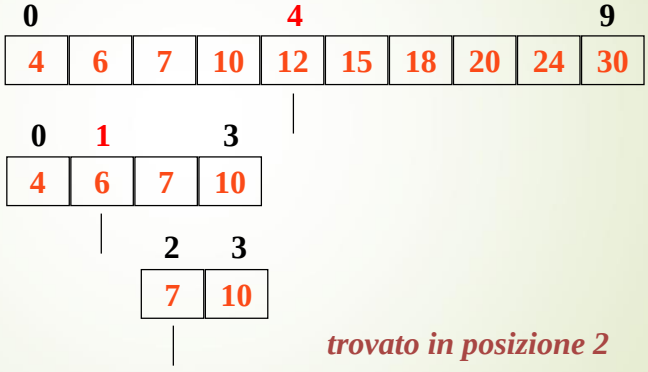
Ricerca binaria in un array ordinato

- Consiste nel dividere l'array in due metà e confrontare l'elemento da cercare con l'elemento centrale dell'array
  - **uguali --> trovato (… e ci si ferma)**
  - **elemento da cercare minore--> continuare la ricerca nella prima metà dell'array**
  - **elemento da cercare maggiore--> continuare la ricerca nella seconda metà dell'array**
- Se l'elemento non è presente, l'array si ridurrà ad un solo elemento, non divisibile in due (terminazione)
  - **nel caso peggiore si visitano $\log_2 n$ elementi dell'array …**

16

Esempio

- Cercare l'elemento 7 nell'array …



… e se invece si fosse cercato 8 ? …

Ricerca binaria

```
int ricercabin(int num, int arr[], int n){

    int begin = 0, end = n-1, center;
    while(end >= begin){
            center = (begin+end)/2;
            if(num == arr[center])
                    return center;
            else if (num < arr[center])
                    end = center - 1;
            else if (num > arr[center])
                    begin = center + 1;
    }
    return -1;
}
```