Introduzione agli Algoritmi e alle Strutture Dati

Operazioni su Liste

Dr. Emanuela Merelli

Argomenti della lezione

- Tipi di Dato Astratto
  - *Lista Lineare*
  - Pila
  - Coda
- *Concetto di*
  - *Struttura dati dinamiche*

Lista Lineare

è un insieme ordinato di *n* elementi, tutti dello stesso tipo, ogni elemento ha un predecessore e un successore

Proprietà:

1. X_1 è il primo elemento della lista
2. X_i è preceduto da xi-1 e seguito da xi+1
3. X_n è l'ultimo elemento della lista
4. N= lunghezza della lista
5. N=0 $\Rightarrow$ Lista Vuota

L(A, B, C, D)

Relazione di precedenza: per accedere ad un elemento si deve accedere a tutti quelli che lo precedono

Lista Lineare come Tipo di Dato Astratto

Dominio dei valori

- Insieme dinamico basato su un elemento omogeneo

Operazioni su Lista Lineare

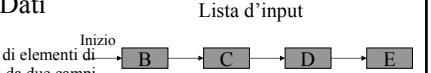
- Inserire un elemento in testa
- Inserire un elemento in coda
- Inserire un elemento dopo un elemento puntato da *P*
- Creare una lista con *N* elementi
- ….

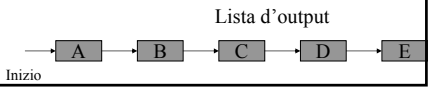
Operazioni su Liste Lineari

Inserimento di un elemento in Testa alla Lista

Struttura Dati

1. Sequenza ordinata di elementi di tipo record formati da due campi, un campo informazione e un campo puntatore ad elemento.
2. Struttura dati dinamica
3. Utilizzo del tipo puntatore

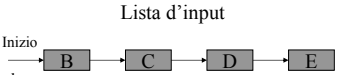


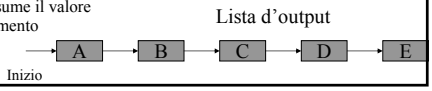


Inserimento di un elemento in testa

Algoritmo

1. Creare il nuovo elemento
2. Memorizzare l'informazione nel nuovo elemento
3. Collegare il nuovo elemento in testa alla lista esistente
   1. Il campo puntatore del nuovo elemento assume il valore di inizio lista
   2. Inizio lista assume il valore del nuovo elemento





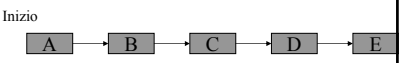
Inserimento di un elemento in testa

```
Type
 T = RECORD
        Inf : Integer;
        Next : Tp;
     END;
 Pun = ↑ T;
Var
 Primo: Pun; A: T;
Procedura Testa (Var Inizio: Pun, X: Integer)
 Var Q: Pun;
Begin
 New(Q);
 Q↑.Inf := X;
 Q↑.Next := Inizio;
 Inizio:= Q;
END;
```

Lista d'input

Nuovo elemento

Lista d'output



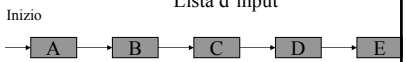
Homework

Definire una Funzione Testa di tipo puntatore che prenda in input un valore X e restituisca Testa con X come primo elemento

Eliminazione dell'elemento in testa ad una lista

```
Type
 T = RECORD
        Inf : Integer;
        Next : Tp;
     END;
 Pun = ↑ T;
Var
 Primo: Pun; A: T;
Procedura ElimTesta (Var Inizio: Pun)
Begin
 IF Inizio <> NIL THEN Inizio:= Inizio↑.Next;
END;
```

Lista d'input



Elemento eliminato

Lista d'output

Eliminazione di un elemento da una Lista

Esercizio

Definire un algoritmo che **elimini** un elemento contenente un informazione X dalla Lista

Casi possibili:

1. Elemento in test
2. Elemento in coda
3. Elemento tra due elementi

Scansione di una lista lineare

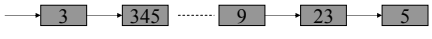
Algoritmo

1. Controllare se la lista è vuota, che equivale a controllare se il puntatore alla lista è uguale a NIL
2. Se Inizio lista uguale a NIL fine scansione, altrimenti si va al passo 3.
3. Se Inizio lista è diverso da NIL, Visita del primo elemento della lista
4. Far avanzare di un elemento il puntatore di inizio lista.
5. Ricominciare dal passo 1.

Scansione di una lista lineare

…

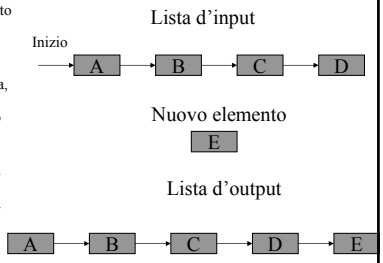
```
Procedura Scansione (Inizio: Pun)
Begin
  WHILE Inizio <> NIL  DO
     Begin
       Writeln(Inizio↑.Inf);
       Inizio:= Inizio↑.Next;
     End;
End;
```



Inserimento di un elemento in coda

Algoritmo

1. Creazione del nuovo elemento
2. Memorizzazione dell'informazione nel nuovo elemento
3. Controllare se la lista è vuota, in tal caso Inizio lista assumerà il valore del nuovo elemento, altrimenti si va al passo 4.
4. Scansione delle lista, fino ad arrivare con il puntatore "inizio" all'ultimo elemento.
5. Collegamento del nuovo tramite l'ultimo



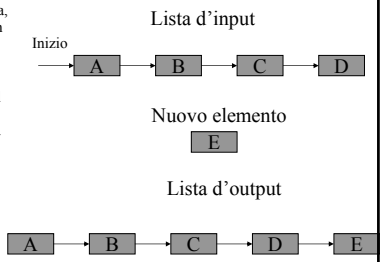
Homework

Come varia l'algoritmo precedentemente descritto, se si vuole creare direttamente il nuovo elemento utilizzando il campo puntatore dell'ultimo elemento della lista

Inserimento di un elemento in coda

Algoritmo

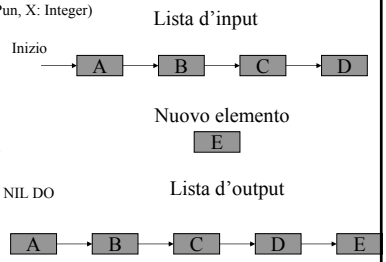
1. Controllare se la lista è vuota, in tal caso crea elemento con Inizio e memorizza l'informazione, altrimenti si va al passo 2.
2. Scansione delle lista, fino ad arrivare con il puntatore "inizio" all'ultimo elemento.
3. Creazione del nuovo elemento, utilizzando il campo puntatore dell'elemento puntato da Inizio
4. Memorizzazione dell'informazione nel nuovo elemento



Inserimento di un elemento in coda

…

```
Procedura InsCoda (Var Inizio: Pun, X: Integer)
 Var P, Q: Pun;
Begin
 New(Q);
 Q↑.Next := NIL;
 Q↑.Inf := X;
 P:= Inizio;
 IF Inizio=NIL THEN Inizio:=Q
    ELSE Begin
           WHILE P↑.Next <> NIL DO
             P:=P↑.Next;
           P↑.Next:=Q;
         END;
END;
```



Domanda

Perché il parametro Inizio della procedura Coda, deve essere passato per riferimento?

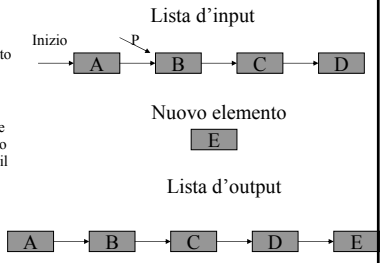
E' possibile definire un algoritmo con il parametro Inizio passato per valore?

Se si, quali modifiche o ipotesi vanno fatte?

Inserimento di un elemento dopo un elemento puntato da *P*

Algoritmo

1. Creazione del nuovo elemento
2. Memorizzazione dell'informazione nel nuovo elemento
3. Poiché siamo nell'ipotesi che l'elemento esiste ed è puntato da *P*, allora si può collegare il nuovo elemento tramite il campo NEXT dell'elemento puntato da *P*

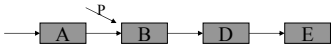


Inserimento di un elemento dopo un elemento puntato da *P*

…

```
Procedura Inserisci (P: Pun, X: Integer)
 Var Q: Pun;
Begin
  New(Q);
  Q↑.Inf := X;
  Q↑.Pun:= P↑.Next;
   P↑.Next:=Q;
END;
```

Lista d'input



Nuovo elemento

Lista d'output

Creazione di una lista di *N* elementi

ognuno contenente come informazione l'intero *n*

Algoritmo

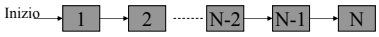
HP1: gli elementi creati vengono inseriti in testa alla Lista

HP2: la lista è ordinata in ordine crescente da 1 a N

1. Inizializzazione di Inizio
2. Creazione iterativa di un nuovo elemento
3. Memorizzazione dell'informazione nel nuovo elemento
4. Inserimento in testa alla Lista

Lista d'input

Lista d'output

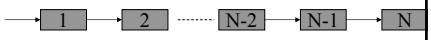


Creazione di una lista di *N* elementi

inserimento in testa

…

```
Procedura CreaT (Var Inizio: Pun, N: Integer)
 Var Q: Pun;
Begin
 Inizio:= NIL;
 WHILE N>0  DO
  Begin
     New(Q);
     Q↑.Next := Inizio;
     Inizio:= Q;
     Q↑.Inf:=N;
     N:= N-1
END;
```



Homework

Utilizzare la **funzione** inserisci in testa all'interno della procedura CreaLista

Fare le necessarie considerazioni sul passaggio dei parametri tra le procedure e le utili riflessioni sulle caratteristiche dell'ambiente locale e globale …

Creazione di una lista di *N* elementi

inserimento in coda

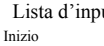
Algoritmo 1

HP1: gli elementi creati vengono inseriti in coda alla Lista

HP2: la lista è ordinata in ordine decrescente da 1 a N

1. Inizializzazione di Inizio
2. **Creazione del primo elemento in testa e memorizzazione dell'informazione**
3. Creazione iterativa di un nuovo elemento
4. Memorizzazione dell'informazione nel nuovo elemento
5. Inserimento in coda alla Lista (l'inserimento in coda implica la scansione della lista)

Lista d'input



Lista d'output

Creazione di una lista di *N* elementi

…

inserimento in coda

```
Procedura CreaC (Var Inizio: Pun, N: Integer)
 Var P, Q: Pun;
Begin
 IF N = 0 THEN Inizio:=NIL ELSE
     Begin
      new(Inizio); Inizio↑.Inf:=N
      Inizio↑.Next:=NIL
       WHILE N>1  DO
         Begin
           New(Q);
           Q↑.Inf:=N-1;
          Q↑.Next:=NIL;
           P:= Inizio;
           WHILE P↑.Next <> NIL DO
               P:=P↑.Next;
           P↑.Next:=Q;
           N:= N-1
          End;
    End
```

Creazione di una lista di *N* elementi

inserimento in coda

Algoritmo 1 Ottimizzato

HP1: gli elementi creati vengono inseriti in coda alla Lista

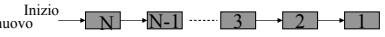
HP2: la lista è ordinata in ordine decrescente da 1 a N

1. Inizializzazione di Inizio
2. **Creazione del primo elemento in testa e memorizzazione dell'informazione**
3. Creazione iterativa di un nuovo elemento
4. Memorizzazione dell'informazione nel nuovo elemento
5. Inserimento in coda alla Lista utilizzando un puntatore che punta sempre all'ultimo elemento

Lista d'input

Inizio →

Lista d'output



Creazione di una lista di *N* elementi

inserimento in coda

```
…
Procedura CreaC (Var Inizio: Pun, N: Integer)
 Var P, Q: Pun;
Begin
 IF N = 0 THEN Inizio:=NIL ELSE
     Begin
      new(Inizio); Inizio↑.Inf:=N
      Inizio↑.Next:=NIL
      P:= Inizio;
       WHILE N>1  DO
         Begin
           New(Q);
           Q↑.Inf:=N-1;
          Q↑.Next:=NIL;
           P↑.Next:=Q;
           P:=P↑.Next;
            N:= N-1
          End;
     End
END;
```

Creazione di una lista di *N* elementi

inserimento in coda

Algoritmo 2

HP1: gli elementi creati vengono inseriti in coda alla Lista

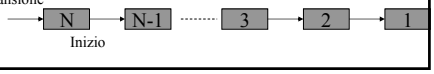
HP2: la lista è ordinata in ordine decrescente da 1 a N

1. Inizializzazione di Inizio
2. Creazione iterativa di un nuovo elemento
3. Memorizzazione dell'informazione nel nuovo elemento
4. **Se Lista Vuota Inserisci in Testa** altrimenti Inserimento in coda alla Lista (l'inserimento in coda implica la scansione della lista)

Lista d'input

Inizio →

Lista d'output



Creazione di una lista di *N* elementi

inserimento in coda

```
…
Procedura CreaC (Var Inizio: Pun, N: Integer)
 Var P,Q: Pun;
Begin
 Inizio:= NIL;
WHILE N>0  DO
 Begin
    New(Q);
    Q↑.Inf:=N;
   Q↑.Next:=Nil;
    P:= Inizio;
   IF Inizio=NIL THEN Inizio:=Q  ELSE
    Begin
      WHILE P↑.Next <> NIL DO
           P:=P↑.Next;
      P↑.Next:=Q;
    End;
   N:= N-1
 End;
END;
```

Creazione di una lista di *N* elementi

inserimento in coda

Algoritmo 3

HP1: gli elementi creati vengono inseriti in coda alla Lista

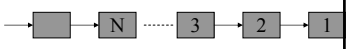
HP2: la lista è ordinata in ordine decrescente da 1 a N

1. Inizializzazione di Inizio
2. **Creazione di un elemento Testa fittizio**
3. Creazione iterative di elementi
4. Memorizzazione dell'informazione nel nuovo elemento
5. Inserimento in coda alla Lista (l'inserimento in coda implica la scansione della lista)

Lista d'input

Inizio →

Lista d'output



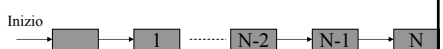
Per generalizzare l'inserimento si utilizza un elemento testa fittizio

Creazione di una lista di *N* elementi

inserimento in coda

```
…

Procedura CreaC (Var Inizio: Pun, N: Integer)
 Var P,Q: Pun;
Begin
  new(Inizio); Inizio ↑.Next:=Nil;
 WHILE N>0  DO
  Begin
     New(Q);
     Q↑.Inf:=N;
     Q↑.Next:=Nil;
     P:= Inizio;
     WHILE P↑.Next <> NIL DO
         P:=P↑.Next;
     P↑.Next:=Q;
    N:= N-1
  End;
```



Creazione di una lista di *N* elementi

inserimento in coda

CONFRONTO tra ALGORITMI

Algoritmo 1 → tratta la prima creazione come caso particolare

Algoritmo 2 → ad ogni creazione controlla se è la prima

Algoritmo 3 → utilizza un elemento testa fittizio per generalizzare la creazione

Tutti e 3 gli algoritmi effettuano una scansione della lista

L'utilizzo di un puntatore che punti sempre all'ultimo elemento potrebbe far risparmiare la scansione

Homework

Calcolare la complessità computazionale dei tre algoritmi e stabilire qual è il migliore e in quale contesto

Esercizio 1

Dato un elenco di parole (testo), si vuole costruire una **Lista Ordinata** contenente un elemento per ogni distinta parola.

Il generico elemento della Lista conterrà un campo informazione (la parola) e un contatore che indicherà il numero di occorrenze della parola nell'elenco.

Inserimento in una Lista Ordinata

ognuno contenente come informazione una parola e un intero *n*

Soluzione

Un'ovvia soluzione è costruire una lista delle parole trovate nell'elenco. La lista viene letta per ogni parola. Se la parola viene trovata, il contatore della sua frequenza viene incrementato; altrimenti la parola viene aggiunta alla lista.

I casi da considerare sono:

1. Lista vuota
2. L'elemento esiste, quindi il contatore deve essere incrementato
3. L'elemento non esiste e
   1. deve essere inserito in testa
   2. deve essere inserito dopo P
   3. deve essere inserito in coda

Inserimento in una lista Ordinata

ognuno contenente come informazione una parola e un intero *n*

Definire un algoritmo il più possibile generale

I possibili casi da considerare sono:

1. Lista vuota → l'elemento è inserito in testa
2. Lista non vuota
   1. Si cerca l'elemento
   2. L'elemento esiste nella lista → si incrementa il contatore
   3. L'elemento non esiste
      1. È il più piccolo → deve essere inserito in testa
      2. È compreso tra due valori → deve essere inserito dopo P
      3. È il più grande → deve essere inserito in coda

Ricerca di un elemento in una Lista

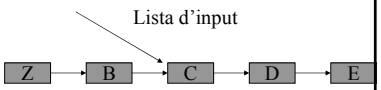
Algoritmo

HP1: gli elementi sono memorizzati in un Lista non ordinata

Domanda

La lista è ordinata o no?

1. Controllare se la lista è vuota
2. Scansione della Lista
   1. Elemento trovato
   2. Raggiunta fine lista



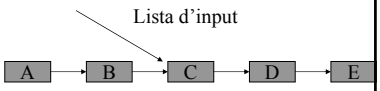
Elemento da trovare

C

Ricerca di un elemento in una Lista
non ordinata

…

```
Function Ricerca (Var Inizio: Pun, X: Info): Boolean
 Var Q: Pun; B:Boolean;
Begin
  Q:=Inizio; B:=true;
  WHILE Q<>NIL AND B  DO
       IF Q↑.Inf:=X THEN B:= false ELSE Q:=Q↑.Next
  If B then Ricerca:=false ELSE Ricerca:=true
 END;
```



Elemento da trovare

C

Osservazioni sulla Condizione While

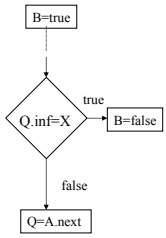
HP:

B= true elemento non trovato

Q<>NIL scansione non terminata

Q<>NIL AND B → elemento non trovata e la lista non è vuota, quindi continuare la scansione

B	Q<>NIL	B and (Q<>NIL)	
1	1	1	continua
0	1	0	elemento trovato, non abbiamo raggiunto la fine
1	0	0	elemento non trovato, abbiamo raggiunto la fine
0	0	0	non si verifica mai, PERCHÉ?



Ricerca di un elemento in una Lista
ordinata

…

```
Function Ricerca (Var Inizio: Pun, X: Info): Boolean
 Var Q: Pun; B:Boolean;
Begin
  Q:=Inizio; B:=true;
  WHILE Q<>NIL AND b  DO
       IF Q↑.Inf >=X THEN B:= false ELSE Q:=Q↑.Next
  If B then Ricerca:=false ELSE
                          If Q↑.Inf=X  THEN  Ricerca:=true
                                       ELSE  Ricerca:=false
 END;
```

Homework

Definire una funzione di tipo puntatore che restituisca il valore NIL se l'elemento non esiste e restituisca il puntatore all'elemento con informazione X se esiste

Inserimento in Liste Ordinate

```
Type
  stringa = …
  Pun  =↑ Parola;
  Parola = RECORD
              Inf : Stringa;
              Conta: Integer
              Next : Pun;
           END;
Var
K: Stringa; Radice:Pun;
Procedura Ricerca (Var Inizio: Pun, X: stringa)
 Var W1,W2 : Pun;
Begin
  W2:= Inizio; W1:=W2 ↑.Next;
  IF W1 = NIL THEN Inserisci (NIL) ELSE
    Begin
      WHILE (W1.Inf < X AND (W1↑.Next <> NIL)
          DO Begin
               W2:=W1;
                 W1:=W1 ↑.Next
             End;
      IF   W1↑.Inf = X     THEN
                 W1 ↑. Conta:= W1 ↑.Conta+1
           ELSE  Begin
                   IF W1↑.Next:= NIL THEN
                      Begin
                        W2:=W1;
                        W1:= NIL
                      End
             Inserisci (W1)
    End

Procedura Inserisci (W:Pun)
  Var W3: Pun;
  Begin
    New (W3);
    W3↑.Inf:= X;  W3 ↑.Conta:= 1;
    W3↑.Next:=W
    W2↑.Next:=W3
  End ;

Begin   %programma principale%
  New(Radice); Radice ↑.Next=NIL; Read(K);
  While K <> "Fine" DO Begin
                          Ricerca(Radice, K)
                          Read(K)
                       End;
End.
```

Osservazione

Le operazioni di inserimento di un nuovo elemento in una Lista possono essere variate in modo da passare come parametro l'elemento già creato tramite una procedura *Crea Elemento*

…

```
Procedura CreaElem (Var P: Pun, X: Info)
Begin
   new(P);
   P↑.Inf:=X;
    P↑.Next:=NIL
End;
```

Operazioni definite su Lista

- Creazione di un nuovo elemento *CreaEl(R,X)*
- Inserimento di un elemento in testa *InsTesta(R,E)*
- Inserimento di un elemento in coda *InsCoda(R,E)*
- Inserimento di un elemento dopo P *Inserisci(P,E)*
- Eliminazione dell'elemento in testa *ElimTesta(R)*
- Eliminazione di un elemento in coda *ElimCoda(R)*
- Eliminazione di un elemento dopo P *Elimina(P)*
- Scansione della lista *Scansione(R)*
- Creazione di una lista di N elementi *CreaT(R,N)*
- Creazione di una lista di N elementi *CreaC(R,N)*

Ordinamento di una Lista

Esercizio

Data una lista di N elementi ordinare gli elementi, in ordine decrescente del campo informazione supposto essere di tipo Integer