UNIVERSITY OF TRENTO - Italy

Laboratorio 18
FIFO con liste semplicemente concatenate

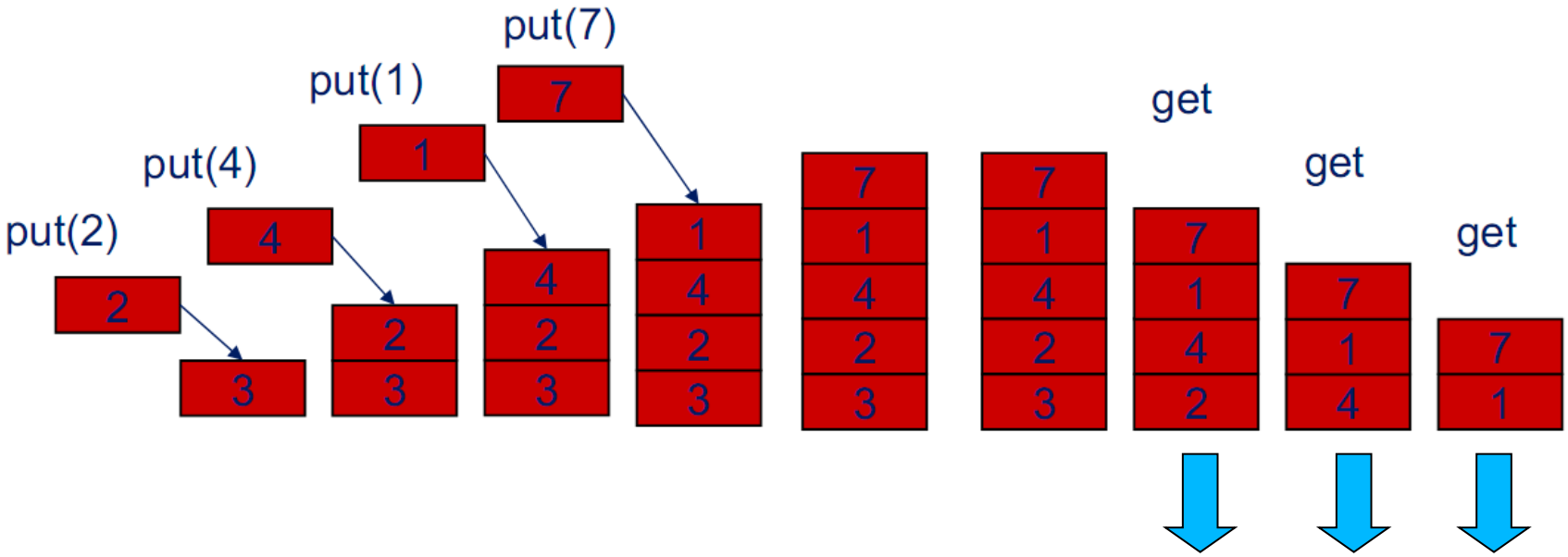
DISI – aa 2024/25

Pierluigi Roberti
Carmelo Ferrante

Università degli Studi di Trento

ADT: Coda-FIFO

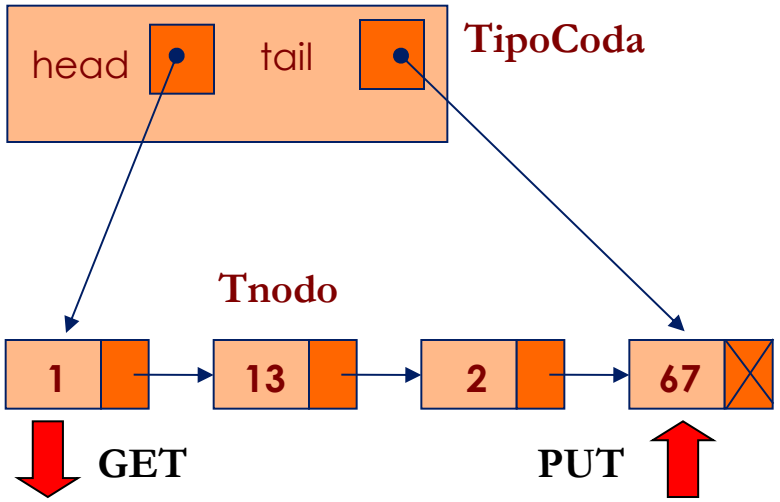
- Uno **coda,** o lista **FIFO** (first in first out, come in una coda in biglietteria) è una struttura dati astratta (ADT) che consente due operazioni:
  - Inserimento di un nuovo elemento in fondo alla coda (**put**)
  - Estrazione di un elemento dalla testa della coda (**get**)



Esercizio2 - Code con lista concatenata

- Vogliamo scrivere un programma che implementa una **coda** usando le liste concatenate

```
typedef struct Tnodo {
 Tdato dato;
 Tnodo * next;
 Tnodo() { next = NULL; }
 Tnodo (Tdato x, Tnodo * n) {
    dato = x; next = n;
 }
} Tnodo;
typedef struct TipoCoda {
 Tnodo * head;
 Tnodo * tail;
 TipoCoda() {
    head = NULL; tail=NULL;
 }
}TipoCoda;
typedef struct TipoCoda Coda;
typedef struct TipoCoda* CodaPtr;
```



Implementare il tipo "**Tdato**" che contiene:

- **cognome** array di caratteri
- **nome** array di caratteri
- **eta** numero intero senza segno
- costruttore 0 parametri
- costruttore 3 paramentri (eta opzionale).

Creare un file denominato "**tipo_dati.h**"

Esercizio 2- Code con lista concatenata

- Andremo a definire le seguenti funzioni (è possibile creare anche dei metodi dentro tipostruct TipoCoda!):
  - **void put (CodaPtr p, Tdato d)** : inserisce x in coda
  - **Tdato get (CodaPtr p)** : estrae l'elemento in testa alla lista
  - **void stampa (CodaPtr p)** : stampa il contenuto della lista (dall'elemento di testa fino a quello di coda)
  - **bool cerca (CodaPtr p, Tdato d)**: cerca elemento d nella lista
  - **bool daticmp (Tdato d1, Tdati d2);** confronta 2 elementi
- Creare i file **FIFO.h** e **FIFO.cpp** che implementano la lista e le funzioni per usarla

Esercizio2 - Code con lista concatenata

- Domandare all'utente di scegliere tra:
  - 1 => Leggere da tastiera valori di una variabile di tipo “Dati”
  - 2 => Inserisce nella coda FIFO (posizione tail) il dato letto (PUT)
  - 3 => Cerca nella lista un dato che ha gli stessi valori
  - 4 => Stampare la coda
  - 5 => Legge dalla coda FIFO (posizione head) il primo elemento (GET)
  - 6 => Esce dal programma

Esercizio2 - metodi + funzioni

FUNZIONE

```
void put(CodaPtr  p, Tdato d) {
    Tnodo* n = new Tnodo();
    n->dato = d;
    n->next = NULL;
    if (p->head==NULL){
        p->head = n;
        p->tail = n;
    } else {
        p->tail->next = n;
        p->tail = n;
    }
}
```

METODO

Dentro Struct TipoCoda

```
void put(Tdato d) {
 Tnodo* n = new Tnodo();
 if(head==NULL){
    head = n;
    tail = n;
 } else {
     tail->next = n;
     tail = tail->next;
  }
}
```

MAIN

```
void main(){
 CodaPtr miaCoda;
 miaCoda = new Tipocoda();
 Tdato d;  //costr. 0 param
 put(miaCoda,d);
 miaCoda->put(d);
}
```

Esercizio2 - metodi + funzioni

FUNZIONI

```
Tdato get (CodaPtr p){
  Tdato d;
  if (p->head==NULL) {
     return d;
  }
  d = p->head->dato;
  if (p->head->next==NULL){
     delete p->head;
     p->head=NULL;
     p->tail=NULL;
  } else {
     Tnodo* q = p->head;
     p->head=p->head->next;
     delete q;
  }
  return d;
}
```

```
Tdato read (CodaPtr p){
   Tdato d;
   if(p->head!=NULL){
     d = p->head->dato;
   }
   return d;
}
```

METODI

```
Tdato get () {
 Tdato d;
 if (head==NULL){
    return d;
 }
 d = head->data;
 if(head->next==NULL){
    delete head;
    head=NULL;
    tail=NULL;
 } else {
    Tnodo* p=head;
    head=head->next;
    delete p;
 }
 return d;
}
```

```
Tdato read (){
   Tdato d;
   if(head!=NULL){
     d = head->dato;
   }
   return d;
}
```

Dentro Struct TipoCoda

Di solito il controllo è fatto nal main

MAIN

```
void main(){
 CodaPtr miaCoda;
 miaCoda=new Tipocoda();
 Tdato d;
 //…
 if (miaCoda!=NULL) {
  d=get(miaCoda);
  //oppure
  d= miaCoda->get();
 }
}
```

Esercizio2 - metodi + funzioni

FUNZIONE

```
void stampa (TipoCoda* s){
 Tnodo* p = s->head;
 while(p!=NULL){
    p->dato.stampa();
    p=p->next;
 }
}


 void main(){
   CodaPtr miaCoda;
   stampa(miaCoda);
   //…
 }
```

METODO

```
void stampa (){
   Tnodo* p = head;
   while(p!=NULL){
      p->dato.stampa();
      // p->stampa();
      p=p->next;
   }
}


void main(){
   CodaPtr miaCoda;
   miaCoda.stampa();
   //…
}
```

Dentro Struct TipoCoda

Si suppone esista un metodo **stampa** dentro la Struct Tdato

Esercizio2 - Struttura dati

#define MAXCHAR 20

```
typedef struct Tdato {
  char nome[MAXCHAR];
  char cognome[MAXCHAR];
  int eta;
  Tdato () {
    nome[0]='\0'; cognome[0]='\0'; eta=0;
   }
  Tdato (char _nome[MAXCHAR],char _cognome[MAXCHAR], int _eta){
    strcpy(nome,_nome); strcpy(cognome,_cognome); eta=_eta;
   }
  void stampa() const{
    cout << "nome:"<<nome<<" cognome:"<<cognome<<" eta:"<<eta <<
   endl;
  }
  bool compare(Tdato d){
    if ( (strcmp(nome,d.nome)==0) &&
         (strcmp(cognome,d.cognome)==0) &&  (eta==d.eta) ) {
       return true;
    } else {
       return false;
   }
}Tdato;
```

Esercizio2 - Struttura nodo e Tipocoda

```
typedef struct Tnodo {
  Tdato dato;
  Tnodo* next;
  Tnodo (Tdato d,Tnodo* n){
    dato=d; next=n;
  }
  void stampa()const {
    data.stampa();
  }
} Tnodo;
```

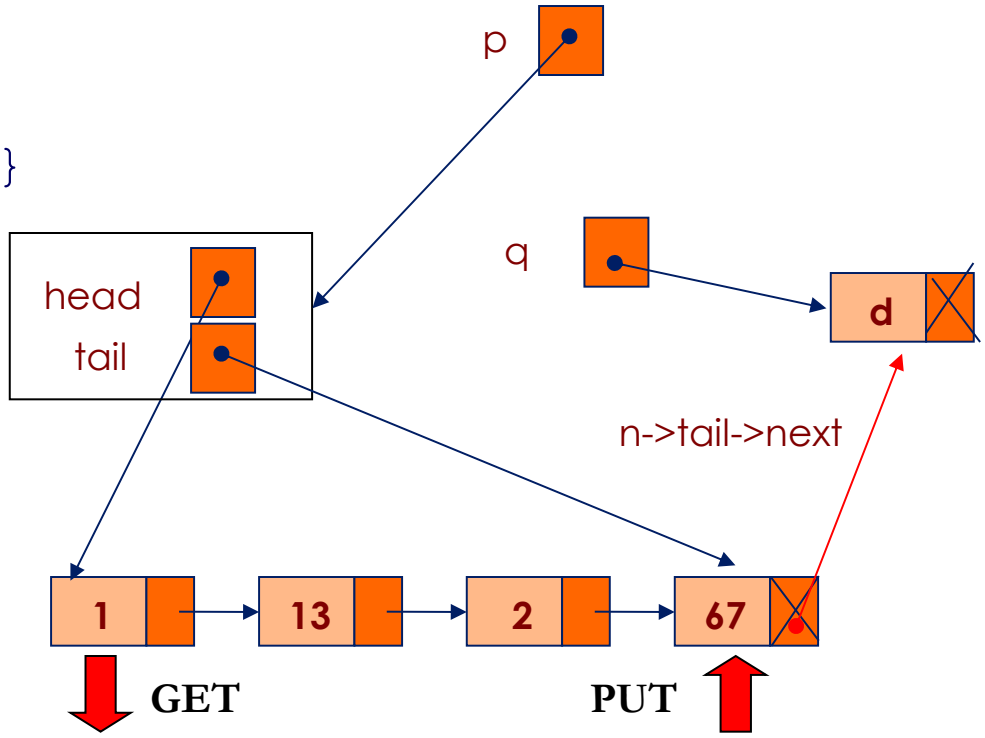
ALIAS

```
typedef TipoCoda Coda;
typedef TipoCoda* CodaPtr;
```

```
typedef struct TipoCoda {
  Tnodo* head;
  Tnodo* tail;
  TipoCoda()  {
     head=NULL; tail=NULL;
  }
  void stampa() const {
   Tnodo* s = head;
   while (s!=NULL){
      s->stampa();
      s= s->next;
    }
    cout << endl;
  }
} TipoCoda;
```

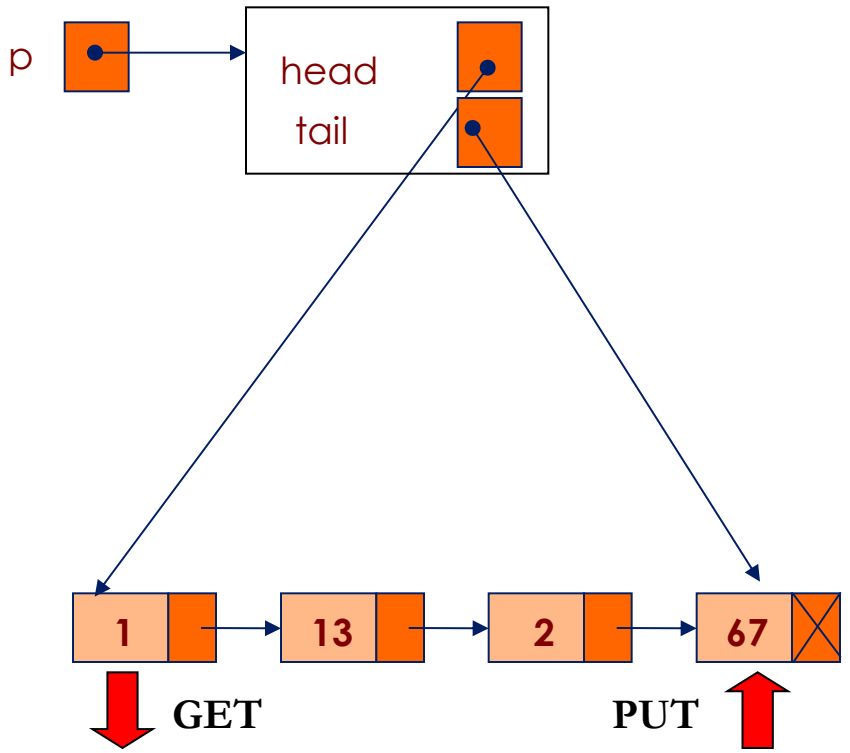
Esercizio2 - Funzione PUT

```
//void put (TipoCoda* p, Tdato d)
void put (CodaPtr p, Tdato d) {
  CodaPtr n;
  n = p;
  Tnodo* q =new Tnodo(d,NULL);
  if (n->tail==NULL) //lista vuota
   { n->head = q; }
  else
   { n->tail->next =q; }
  n->tail = q;
  p = n;
}
```



Esercizio2 - Funzione READ

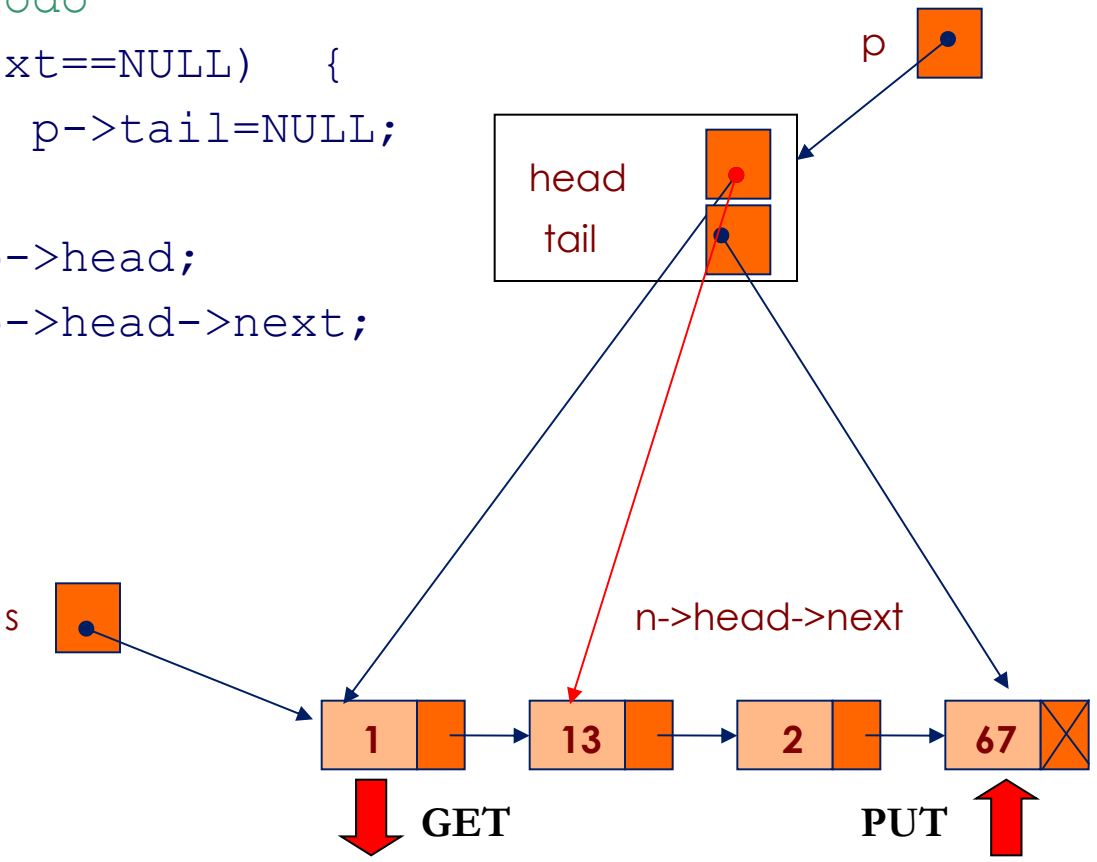
```
Tdato read (CodaPtr p){
    Tdato d;
    if(p->head!=NULL){
      d = p->head->dato;
    }
    return d;
}
```



Esercizio2 - Funzione GET

```
Tdato get (CodaPtr p){
  Tdato d;  //invoco costruttore default
  if (p->head==NULL) //lista vuota
   { return d; } //devo comunque ritornare qualcosa
  d = p->head->dato;
  //caso 1 solo nodo
  if (p->head->next==NULL)  {
    p->head=NULL; p->tail=NULL;
  } else {
      Tnodo* s= p->head;
      p->head = p->head->next;
      delete s;
  }
  return d;
}
```

È meglio controllare prima se la lista è vuota



Esercizio2 - Versione alternativa

```
void put (CodaPtr p, Tdato d){
  if (p->head==NULL){
     p->tail = new Nodo(d, NULL);
     p->head = p->tail;
  } else {
     p->tail->next = new Nodo(d, NULL);
     p->tail = p->tail->next;
  }
}
```

```
Tdato get (CodaPtr p){
  Tdato d;
  d = p->head->dato;
  //caso particolare: 1 solo elemento
  if (p->head->next==NULL) {
     delete p->head; //dealloco la memoria
     p->head = NULL;
     p->tail = NULL;
  } else {
     Nodoptr s = p->head;
     p->head = p->head->next;
     delete s;
  }
  return d;
}
```

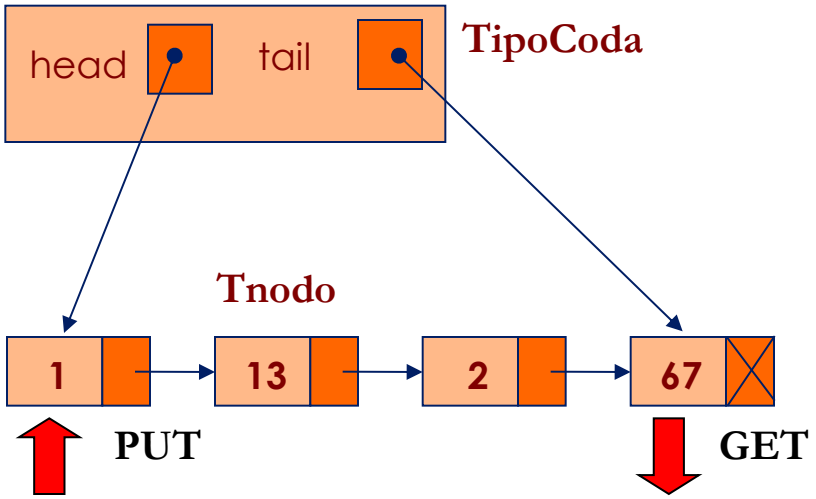
```
int main() {
CodaPtr fifo= new Coda();
   Tdato d;
   put(fifo, Tdato(4,45));
   put(fifo, Tdato(3,3.78));
   put(fifo, Tdato(7,1.11));
   stampa(fifo->head);

   d = get(fifo);
   cout << "dato letto:"<<endl;
   d.stampa();
   cout << endl;
   stampa(fifo->head);
}
```

Esercizio2bis-Code con lista concatenata

- Vogliamo scrivere un programma che implementa una **coda** usando le liste concatenate (PUT e GET invertiti!)

```
typedef struct Tnodo {
 Tdato dato;
 Node * next;
 Tnodo() { next = NULL; }
 Tnodo(Tdati x, Tnodo * n) {
   dato = x; next = n;
 }
} Tnodo ;
typedef struct TipoCoda {
 Tnodo * head;
 Tnodo * tail;
 TipoCoda() {
    head = NULL; tail=NULL;
 }
} TipoCoda ;
typedef struct TipoCoda Coda;
typedef struct TipoCoda* CodaPtr;
```



Implementare il tipo "**Tdato**" che contiene:

- **cognome** array di caratteri
- **nome** array di caratteri
- **eta** numero intero senza segno
- costruttore 0 parametri
- costruttore 3 paramentri (eta opzionale).

Creare un file denominato "**tipo_dati.h**"

Esercizio2 bis – Metodi

METODO

```
void put (Tdato d){
  Node* n = new Node(d,head);
  head  = n;
}

Tdato get (){
  Tnodo* q = head;
  if(q->next==NULL){
     Tdati d = q->data;
     delete q;
     head = NULL;
     return d;
  }
  while(q->next->next!=NULL){
      q = q->next;
  }
  Tdati d = q->next->data;
  delete q->next;
  q->next=NULL;
  return d;
}
```

```
void print ()const{
   Tnodo * q = head;
   while(q!=NULL){
     q->data.print();
     q = q->next;
   }
}
```

Dentro Struct Tipocoda

MAIN

```
void main(){
  CodaPtr miaCoda = new Coda();
  Tdato d(”Mario”, ”Rossi”,23);  //costr 3 param
  miaCoda->put(d);
  if(miaCoda->head!=NULL){
     dd=codaPoste->get();
     dd.print();
   }
  delete miaCoda;
}
```

Esercizio3 – gestione multi-code

- Dichiarare un array di 3 code FIFO di tipo **Coda**
- Ripetere per **K** volte (dichiarata costante)
  - Generare numero casuale tra 0 e 2
  - Inserire (PUT) nella coda selezionata il dato letto da tastiera (per nome e cognome) e generato in modo casuale (tra 1 e 50) per età.
  - Stampare tutte le 3 code FIFO
  - Generare numero casuale tra 0 e 2
  - Prelevare dalla coda selezionata un dato (GET)
  - Stampare tutte le 3 code FIFO