Programmazione

Docenti: Giovanni Da San Martino
Francesco Gavazzo
Lamberto Ballan
<lamberto.ballan@unipd.it>

Programmazione, A.A. 2023/24
SCQ0093758 - LT Informatica

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Recap: le liste

- Una lista è una successione finita di elementi di un tipo
- L'informazione codificata dalla lista riguarda:
  - La successione di elementi (valori)
  - La relazione di ordine tra gli elementi stessi

- La lista è qualificata non solo dai valori che rappresenta ma anche dalle operazioni che ci si eseguono
  - Inserimento e cancellazione (in testa, coda, intermedia)
  - Visita / ricerca
  - Inizializzazione

Recap: le liste

- Una lista può essere rappresentata:
  - In forma *sequenziale*: gli elementi sono rappresentati in un array e il loro ordine è codificato in modo implicito dalla posizione
  - In forma *collegata*: in questo caso la relazione è resa esplicita e ogni elemento è associato all'informazione che identifica il successore (questa può essere un indice o un puntatore)

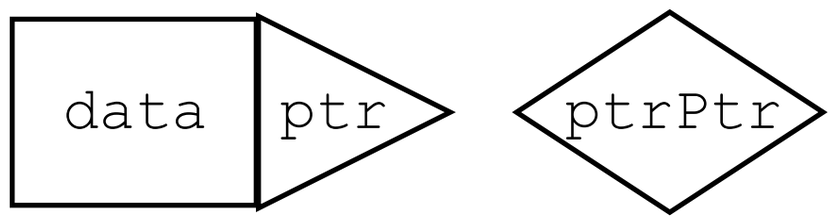
- Una lista collegata con puntatori è una successione di elementi (nodi) connessi da puntatori

Liste collegate con puntatori

- Le liste collegate non offrono un accesso immediato ai loro elementi, ma inserimento/cancellazione di elementi possono seguire diverse strategie
  - LIFO, FIFO, in posizione “generica”
- L’elemento atomico di una lista è il *nodo* che può essere definito da una struttura “autoreferenziale”:

```
struct nodo {
   int data;
   struct nodo *nextPtr;
};
```

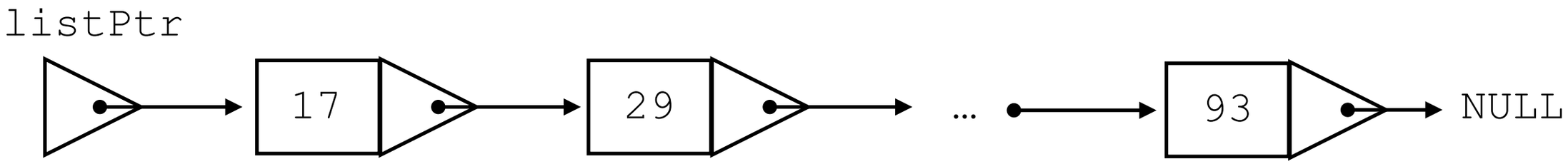
NB: useremo questa “convenzione grafica”



Liste collegate con puntatori

- Esempio / rappresentazione grafica di una lista collegata:

Lista di elementi: {17, 29, …, 93}




```
struct nodo {
   int data;
   struct nodo *nextPtr;
};
```

Funzioni su liste collegate

- L'utilizzo di liste collegate richiede la definizione di alcune funzioni di base:
  - inizializzazione
  - visita (solitamente stamperà i valori dei nodi)
  - ricerca di un elemento
  - inserimento
  - cancellazione
- Oggi inizieremo a:
  - Sviluppare funzioni di inserimento, visita, etc. su liste (vedremo implementazioni di tipo iterativo)

terminal:~$

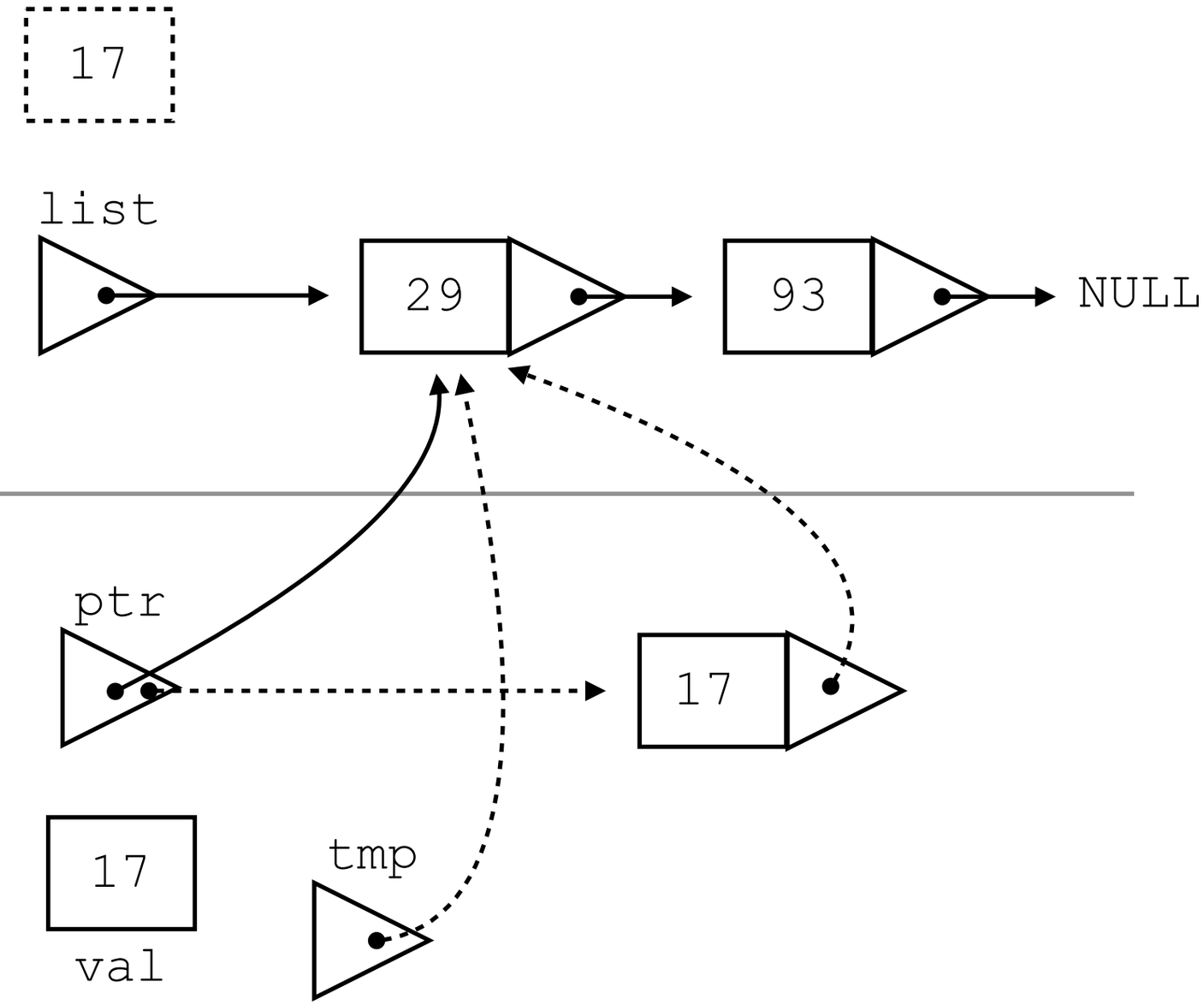
Lavoriamo “online” e alla lavagna…

Liste: funzione di inserimento

- Funzione *wrong*_pre_insert()

```c
int main(void) {

    struct node *list = NULL;

    //...

    pre_insert(list, 17);

    //...
}

void pre_insert(struct node *ptr, int val) {
    struct node *tmp;

    tmp = ptr;
    ptr = malloc(sizeof(struct node));
    ptr->value = val;
    ptr->nextPtr = tmp;
}
```

Inserimento *in testa*

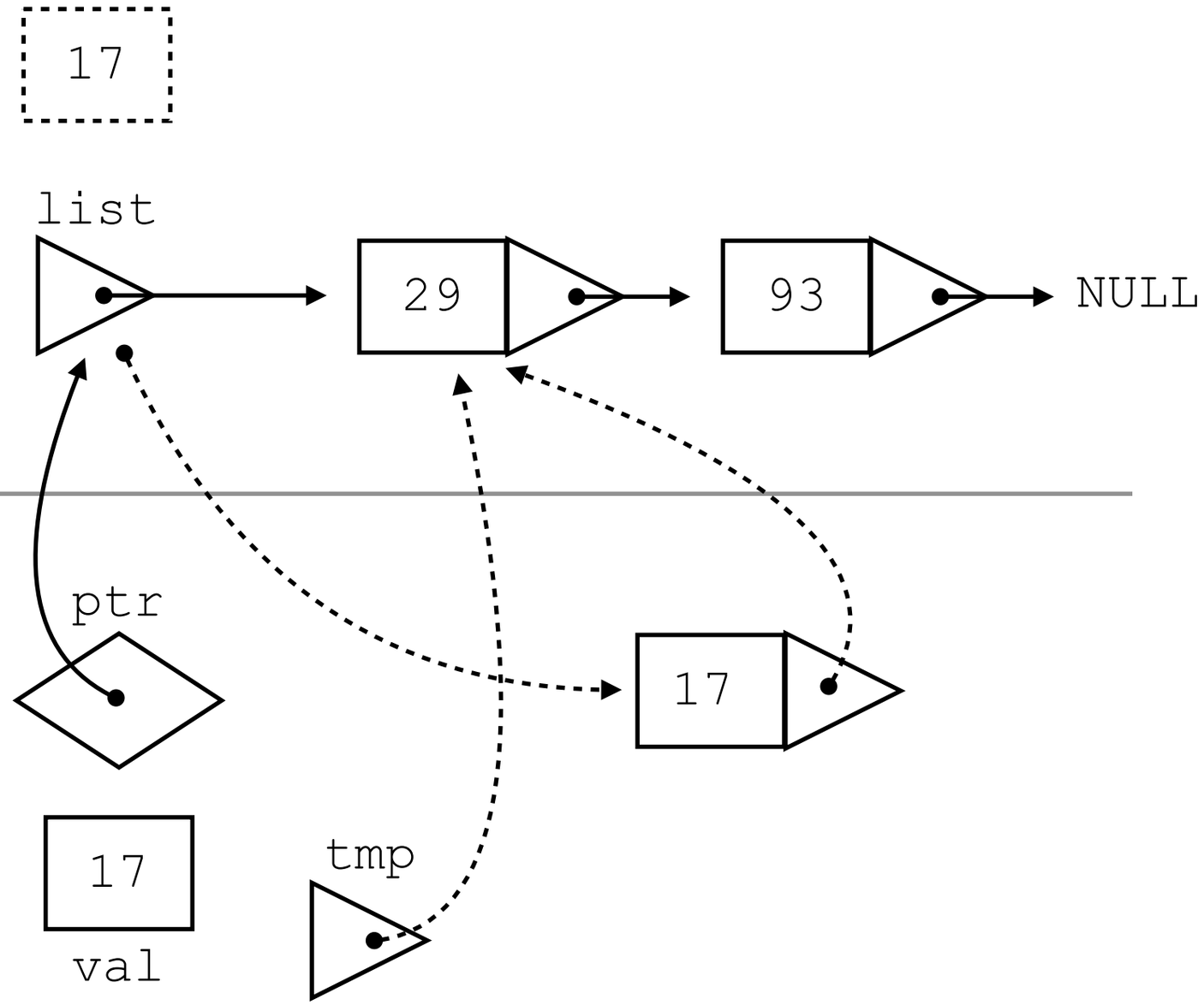


Liste: funzione di inserimento

- Funzione `pre_insert()`

```c
int main(void) {
    struct node *list = NULL;
    //...
    pre_insert(&list, 17);
    //...
}

void pre_insert(struct node **ptr, int val) {
    struct node *tmp;

    tmp = *ptr;
    *ptr = malloc(sizeof(struct node));
    (*ptr)->value = val;
    (*ptr)->nextPtr = tmp;
}
```

Inserimento *in testa*

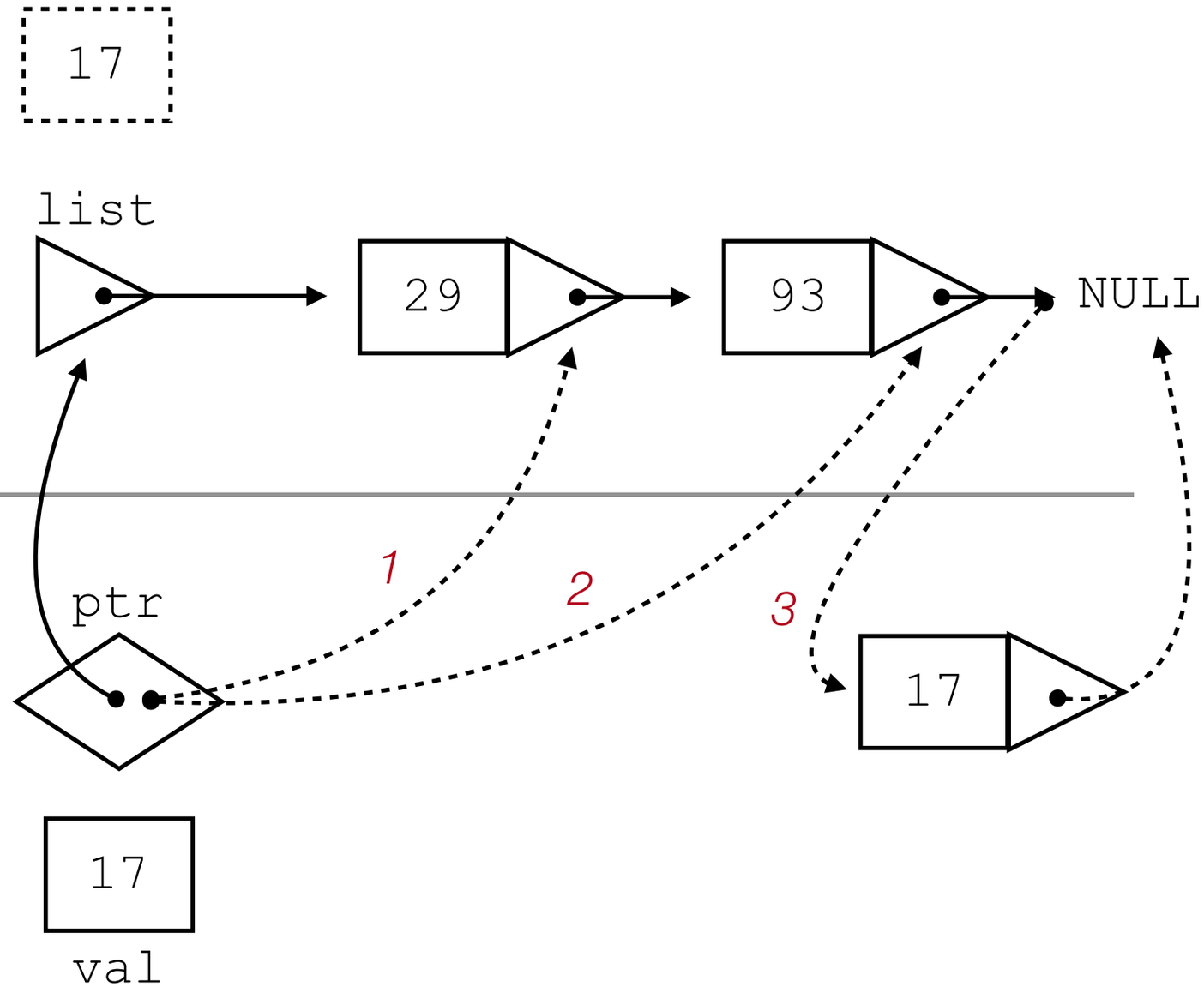


Liste: funzione di inserimento

- Funzione `suf_insert()`

```c
int main(void) {
    struct node *list = NULL;
    //...
    suf_insert(&list, 17);
    //...
}

void suf_insert(struct node **ptr, int val) {
    while(*ptr != NULL) {
        ptr = &((*ptr)->nextPtr);
    }
    pre_insert(ptr, val);
}
```

Inserimento *in coda*



Liste: versioni “alternative” / ricorsive

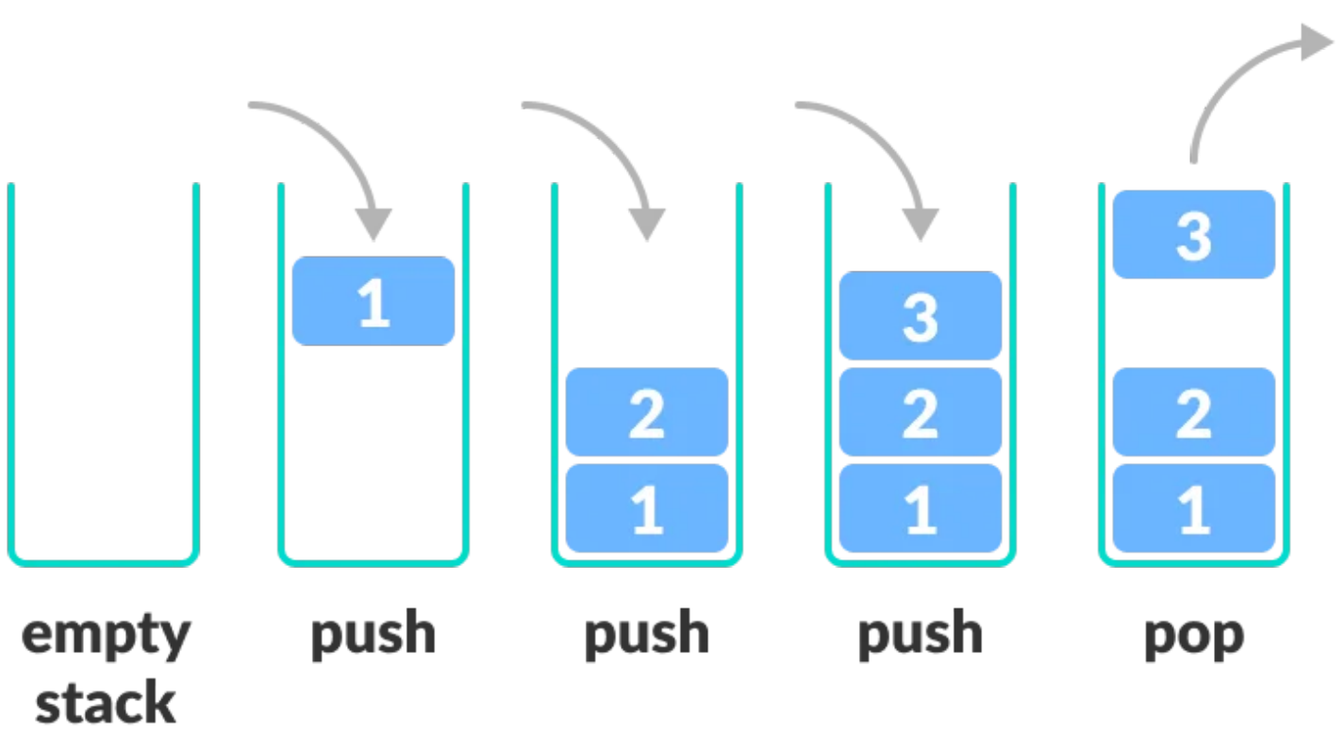
- Adesso vedremo una versione alternativa ad alcune delle funzioni già viste, ed analizzeremo il caso ricorsivo
- In particolare:
  - Vedremo innanzitutto l’implementazione senza doppio ptr (quindi con ritorno ptr)
  - Discuteremo brevemente alcuni aspetti dell’utilizzo di `typedef`
  - Introdurremo una nuova funzione d’inserimento `ord_insert` (che dovrà garantire un inserimento ordinato degli elementi)
  - Introdurremo le versioni ricorsive di alcune delle funzioni già viste in precedenza

terminal:~$

Lavoriamo “online” e alla lavagna…

Liste: implementiamo una pila (stack)

- Una pila (o stack) è una struttura dati lineare di tipo LIFO (last in first out)
  - Il classico esempio / analogia è quello di una pila di piatti
  - Le funzioni di inserimento e cancellazione di un elemento in una pila sono comunemente chiamate `push` e `pop`



terminal:~$

Lavoriamo “online” e alla lavagna…

Contatti

- **Ufficio:** Torre Archimede, ufficio 6CD3
- **Ricevimento:** ~~Venerdì 9:00-11:00;~~ fino alla fine delle lezioni: Giovedì 8:30-10:30 *(inviare cmq e-mail per conferma)*

✉ lamberto.ballan@unipd.it
🏠 http://www.lambertoballan.net
🏠 http://vimp.math.unipd.it
@ twitter.com/lambertoballan