Programmazione I

dott. Sabrina Senatore

Dipartimento di Informatica

Riepilogo strutture

Definire un tipo di record

- Sintassi della dichiarazione:

typedef struct *nome_tipo* {
  Dichiarazione dei campi
} *nome_tipo*;

```
typedef struct point{
   int x;
    int y;
 } point;
```


Typedef

- Spesso si usa typedef per definire un tipo di struttura in modo da evitare l'uso dell'etichetta della struttura.
- Esempio:

```
typedef struct {
    char *face;
    char  *suit;
} Card;
```

- Creerà il tipo di struttura Card, senza la necessità di un'istruzione typedef separata

Array di strutture

- Se volessi definire una “biblioteca”, cioè un insieme di libri?

```
typedef struct libro {
 char ISBN[10];
 char titolo[40];
 char autore[25];
 char editore[25];
 int anno;
 float prezzo;
 } Libro;
;
```

```
Libro biblioteca[100];
```


Struttura impiegato

- Il record di un impiegato:

```
struct impiegato{
  char nome[20];
  char cognome[20];
  int eta;
  char sesso;
  float stipendio;
};
```


Esercizio

- Mediante input da tastiera, creare una popolazione di impiegati, usando la struttura definita predecentemente.
- Dato in input il sesso, stampare i dati anagrafici (nome, cognome, età) relativi agli impiegati inseriti

Esercizio

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define DIM 20

struct impiegato{
   char nome[DIM];
   char cognome[DIM];
   int eta;
   char sesso;
   float stipendio;
};

void stampa(struct impiegato [], char, int);
int main()
{
    struct impiegato imps[DIM];
    int i=0, j;
    char c, str[DIM], sesso;
    printf("Inserimento Impiegati (premere * per uscire)\n");
```

```
 printf("Impiegato n. 0 Nome: ");
   scanf("%s", str);
   while (i<DIM  &&  strcmp(str, "*")!=0)  {
      strcpy(imps[i].nome, str);

      printf("Cognome: ");
      scanf("%s", imps[i].cognome);
      printf("Età: ");
      scanf ("%d", &imps[i].eta);
      printf("Sesso: M/F ");
      fflush(stdin);
      scanf ("%c", &sesso);
      imps[i].sesso=toupper(sesso);
      printf("Stipendio: ");
      scanf ("%f", &imps[i].stipendio);
      i++;
      printf("Impiegato n. %d Nome: ", i);
      scanf("%s", &str);
   }
      printf("Funzione di stampa dei dipendenti per sesso. \n
         Inserisci il sesso ");

   do {
         scanf("%c", &c );
         c =  toupper(c);
      } while (c!='M' &&  c!='F') ;

       stampa (imps, c, i);
      return 0;
}
```

Svuota il buffer in entrata, cioè quell'area di memoria dedicata alle operazioni che prelevano i caratteri dalla tastiera.

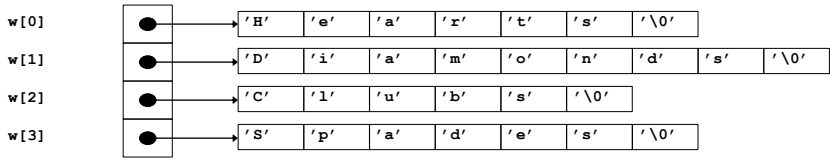
continua

```
void stampa(struct impiegato a[], char c, int num)
 {

     int i;
     printf ("SALARIO DIPENDENTI DI SESSO %c:  \n", c);
     for ( i=0; i<num; i++)
       if (a[i].sesso==c  )
         printf("%s, %s, %d, %6.1f\n", a[i].cognome, a[i].nome, a[i].eta,
                                                          a[i].stipendio);

    }
```


Liste e puntatori

Array di puntatori o liste

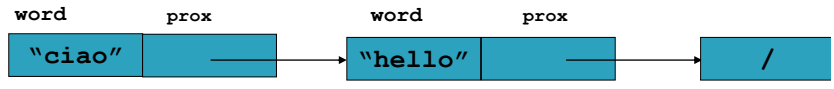
```
char *w[4] = {"Hearts", "Diamonds",
  "Clubs","Spades"};
```



Supponiamo di avere un problema più generale: vogliamo contare tutte le parole presenti in un certo input, senza definire una dimensione massima (né da programma, né da input)

Strutture ricorsive

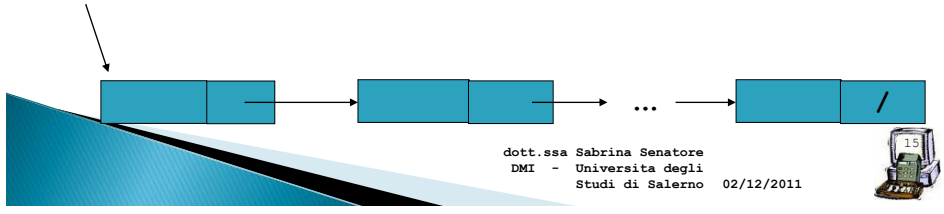
```
struct parola {
char word[20];
struct parola *prox;
};
```



prox è un puntatore a un elemento di tipo **parola.**

Lista concatenata a puntatori

- Ogni elemento di una lista concatenata è un record che ha un campo puntatore che punta al record successivo.
  - Si accede alla struttura attraverso il puntatore al primo record.
  - Il campo puntatore dell'ultimo record contiene il valore NULL



Dichiarare un tipo lista

- Due modi:

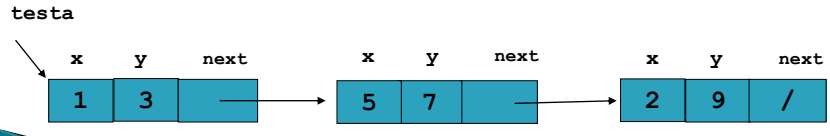
```
typedef struct nodo * lista;

typedef struct nodo {
// dichiarazione di altri
   campi
lista next;
} nodo;
```

```
typedef struct nodo {
// dichiarazione di altri
   campi
struct nodo *next;
} nodo;
```

Esempio

```
typedef struct punto{
   int x;
   int y;
   struct punto *next;
} punto;
```

Come creare una lista?



Operazioni su una lista concatenata

- Creare una lista:

```
typedef struct punto{
   int x;
   int y;
   struct punto *next;
} punto;

punto *testa = NULL;
```

- La lista è inizialmente vuota.
- La variabile **testa** è il puntatore iniziale.

Creazione di un elemento di una lista concatenata 1/2

- Per creare un elemento di una lista, bisogna:
  - Dichiarare una variabile di tipo puntatore a lista;
  - Allocare l'elemento con malloc
  - Inserire i dati
  - Associare correttamente il puntatore del nuovo elemento creato
- La creazione di un elemento equivale alla creazione di una lista di un unico elemento

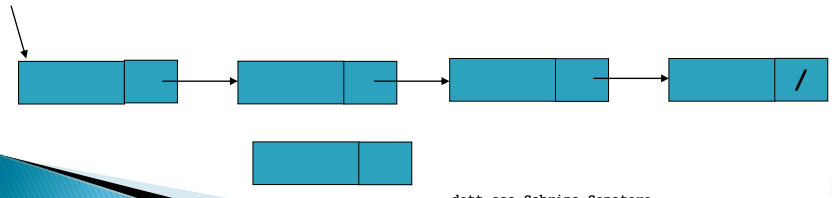
Creazione di un elemento di una lista concatenata 2/2

```
typedef struct list{
    int value;
    struct list *next;
} lista;

//inizialmente:
Lista *testa=NULL;
...

lista *testa;
testa=(lista*)malloc(sizeof(lista));
printf("Inserire valore: ");
scanf("%d",&testa->value);
testa->next=NULL;
```


Inserimento in una lista concatenata

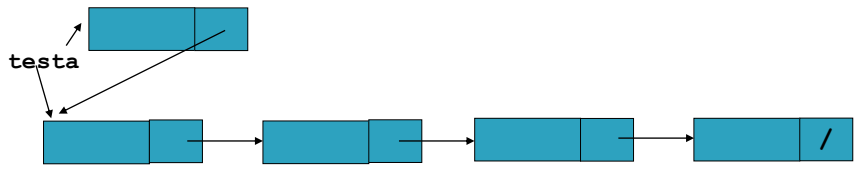
- Inserimento di un elemento in una lista concatenata.

- Ci sono tre diverse possibilità nell'inserimento
  - Inserimento in testa
  - Inserimento in coda
  - Inserimento generico



Inserimento in testa

- Passo 1. il nuovo nodo record punta dove punta testa
- Passo 2. testa punterà all'indirizzo del nuovo nodo



Un po' di codice…inserimento in testa

```
if (testa==NULL){
                testa=creaelemento();
        }
        else{
                nuovoelemento=creaelemento();
                nuovoelemento->next=testa;
                testa=nuovoelemento;
        }
```

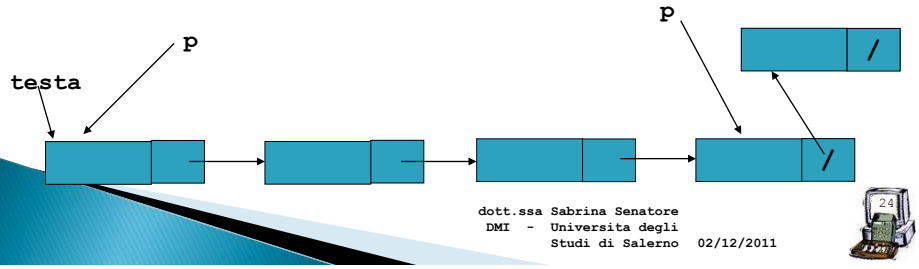
- Dove creaelemento() è una funzione con il seguente prototipo:

```
lista *creaelemento();
```

(scrivere tale funzione per esercizio)

Inserimento in coda

- Passo 1. E' necessario attraversare la lista per trovarne la fine. Si può usare un puntatore ausiliario
- Passo 2. il puntatore dell'ultimo record punterà al nuovo record.

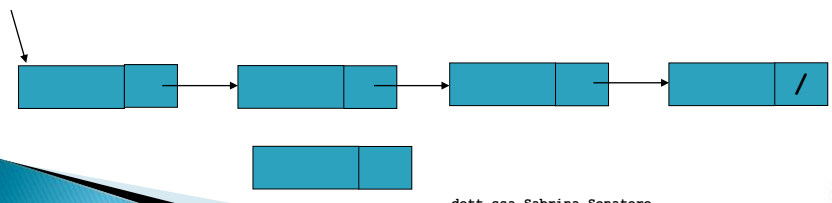


Un po' di codice…inserimento in coda

```
lista* p=testa;
lista* nuovoelemento=creaelemento();
if (testa==NULL){
        return nuovoelemento;
}
else{
        while (p->next!=NULL){
                p=p->next;
        }
         p->next=nuovoelemento;
 }
```

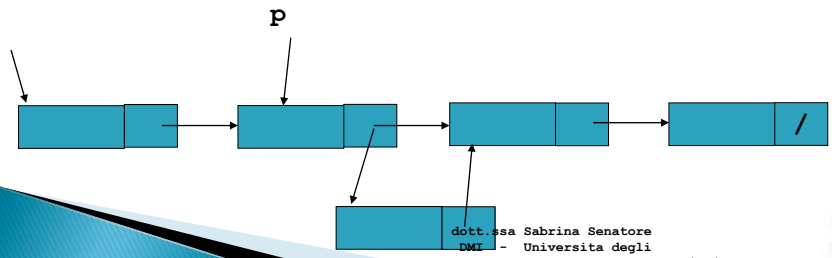

Inserimento generico in una lista concatenata

- Inserimento generico di un elemento in una lista concatenata.
  - Si può inserire un elemento in una lista concatenata come successore di uno qualunque dei record già presenti.



Passi per l'inserimento

- Notare: ho bisogno di sapere dove inserire!!
- Passo 1. creare il collegamento con il record successivo
- Passo 2. si collega il record precedente con il nuovo record



Un po' di codice…

```
...
lista* p;
...
lista* nuovoelemento=creaelemento();
        nuovoelemento->next=p->next;
        p->next=nuovoelemento;
 }
```


Qualche considerazione..

- In generale, per scorrere una lista, si utilizza un puntatore ausiliario, da aggiornare all'interno di una struttura iterativa

```
lista* p=NULL;
p=testa;
while (p!=NULL){
  … // istruzioni di computazione
  p=p->next;

}
```

- L'utilizzo del puntatore ausiliario è necessario, al fine di evitare di perdere riferimenti alla lista, man mano che si avanza (cioè si perde la concatenazione della lista)

Cancellazione

- La lista, è una struttura allocata dinamicamente. E' quindi prevista una deallocazione dei record.
- La cancellazione di elementi avviene mediante l'utilizzo del comando *free*

```
void cancella(lista *p)
{
  lista *tmp;
  tmp = p->next;
  if (tmp==NULL) return;
  p->next = temp->next;
  free(tmp);
}
```


Cancellazione

```
lista* p=testa;
    while (p!=NULL){
        p=p->next;
        free(testa);
        testa=p;
    }
```

- Cosa fa questo codice??

Esercizio

- Risolvere l'esercizio dell'impiegato usando una lista piuttosto che un array.

Esercizio

```
#include <stdio.h>
#define DIM 20

typedef struct s_impiegato{
   char nome[DIM];
   char cognome[DIM];
   int eta;
   char sesso;
   float stipendio;
   struct s_impiegato *prossimo;
}impiegato;

impiegato *creaImpiegato( char *nome, char *cognome, int eta, char sesso, float
   stipendio);
impiegato *inserisci_in_lista(impiegato * imp, impiegato *p);
void stampa(impiegato *q, char c);

int main()
{
   impiegato imp;
   impiegato *p, *testa=NULL;
   int i=0, j, eta;
   char c, d, str[DIM], nome[DIM], cognome[DIM], sesso;
   float stipendio;
```

continua

```
 printf("Inserimento Impiegati (premere $ per uscire)\n");
  printf("Impiegato n. 0 Nome: ");
  scanf("%s", &nome);

  while ( strcmp(nome, "$")!=0)  {
       p = (impiegato *) malloc(sizeof(impiegato));
       if (p==NULL)   break;
      strcpy(p->nome, nome);
       printf("Cognome: ");
       scanf("%s", &cognome);
         strcpy(p->cognome, cognome);
       printf("Età: ");
       scanf ("%d", &(p->eta));
         printf("Sesso: M/F ");
       fflush(stdin);
       p->sesso=toupper(getchar());

       printf("Stipendio: ");
       scanf ("%f",&(p->stipendio));
         p->prossimo=NULL;

       testa=inserisci_in_lista(p, testa) ;

       printf("Impiegato n. %d Nome: ", i);
       scanf("%s", &nome);

  }
```

```
 printf("Funzione di stampa dei dipendenti per sesso. \n Inserisci il
   sesso ");
   fflush(stdin);
   c =  toupper( getchar( ));

   while ( c != 'M' &&  c != 'F')
        ;
   stampa (testa, c);
   system("PAUSE");
   return 0;
}

void stampa(impiegato *q, char c)   {
   printf ("SALARIO DIPENDENTI DI SESSO %c:  \n", c);
   while ( q!=NULL) {
        if ((q->sesso)== c  )
             printf("%s, %s, %d, %6.1f\n", q->cognome, q->nome, q->eta,
                                                        q->stipendio);
        q=q->prossimo;
   }
}


impiegato *inserisci_in_lista(impiegato *imp, impiegato *testa) {
   imp->prossimo= testa;
   testa=imp;
   return testa;
}
```