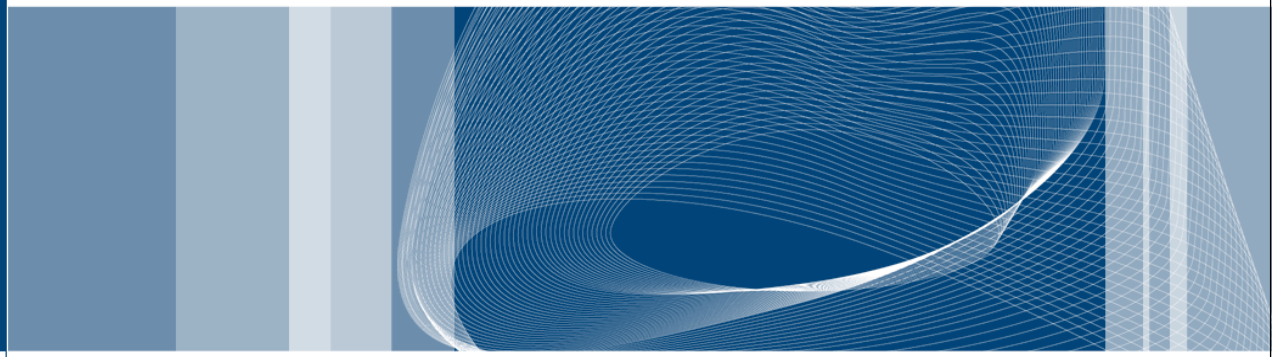
POLITECNICO DI MILANO



Fondamenti di Informatica 2013-2014

Allocazione Dinamica

Paola Mussida
Area Servizi ICT

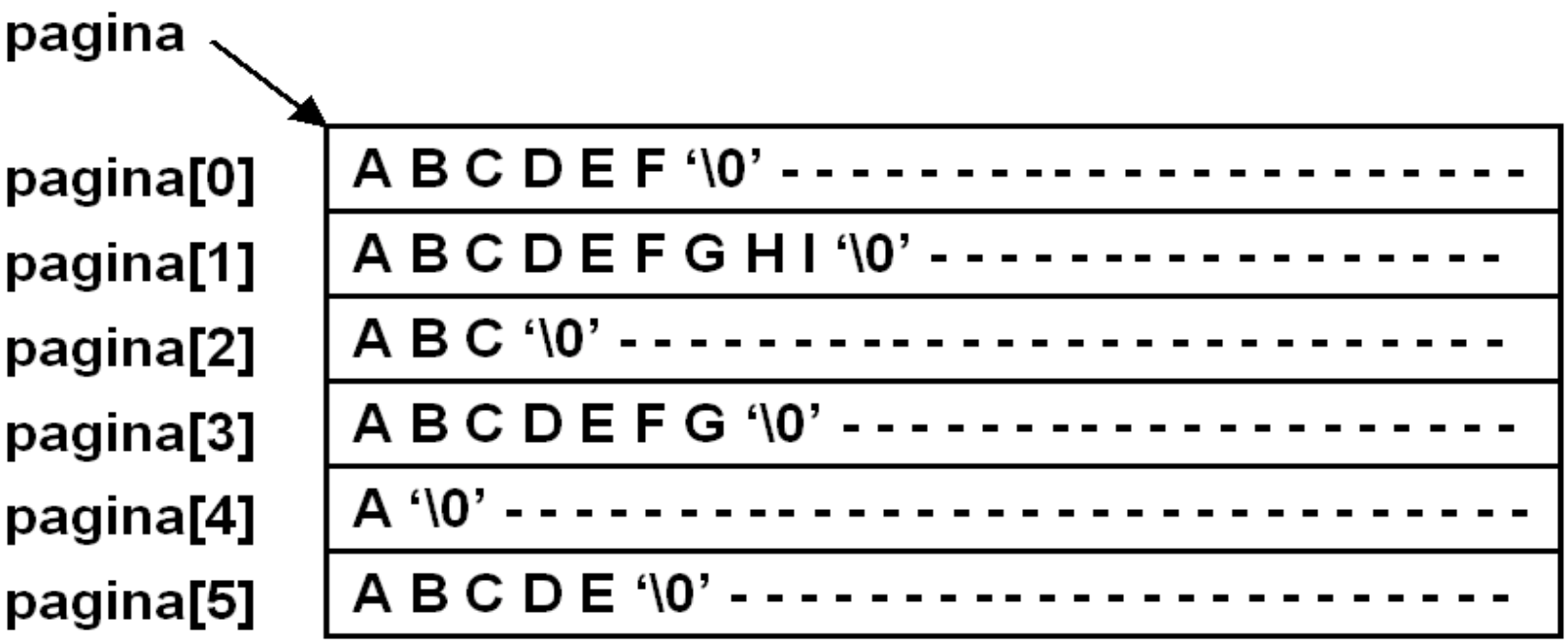
Confronto fra vettore di stringhe (matrice di caratteri sovradimensionata) e vettore di puntatori a stringhe.

Esercizio A_1 - CharArr vs CharStarPo

matrice di caratteri sovradimensionata

Struttura Dati:

char pagina[6][81];



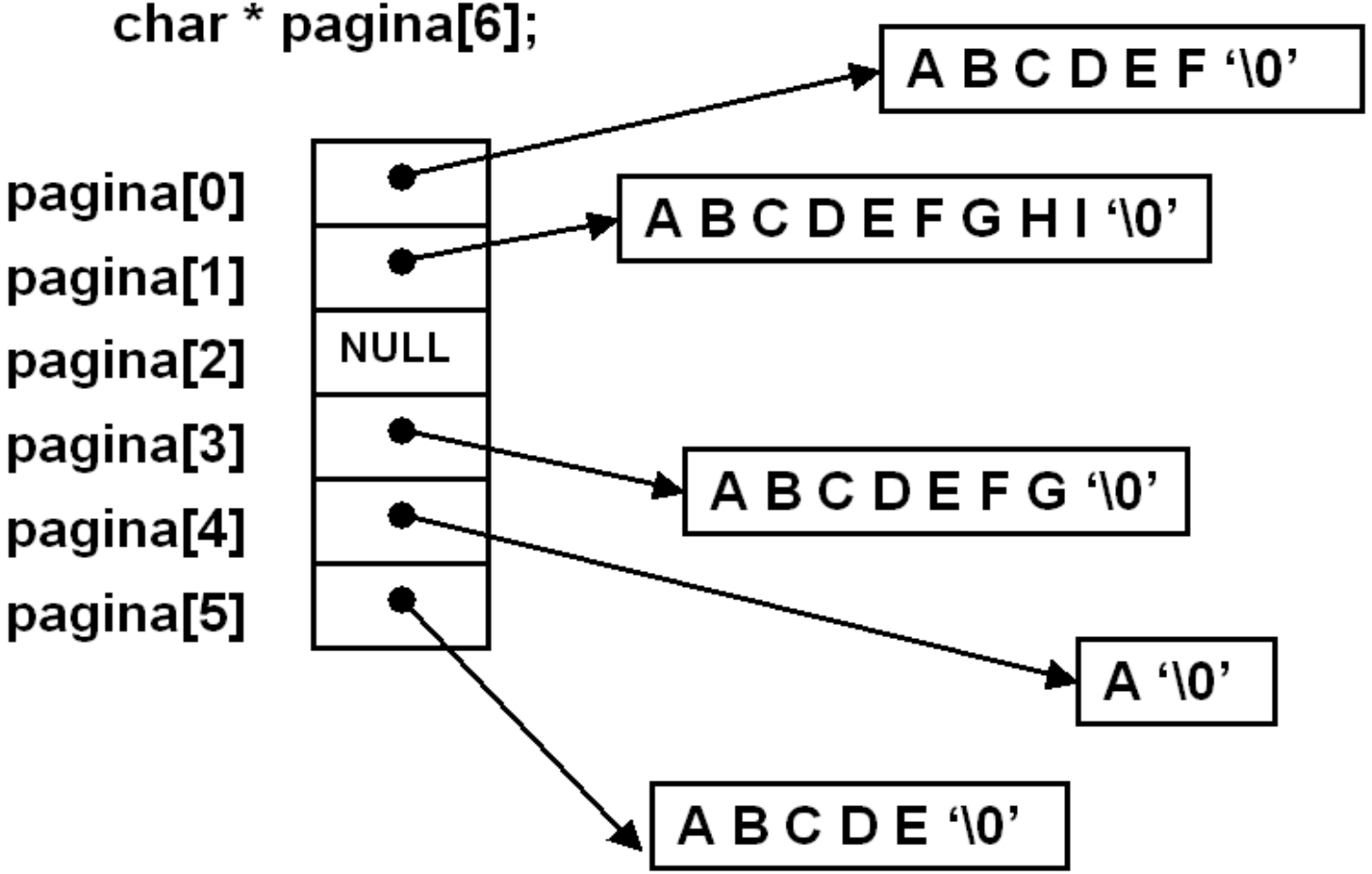
Esercizio A_1 - CharArr vs CharStarPo


```
#define MAXR 6
#define MAXC 81

void main() {
char pagina[MAXR][MAXC];        /* matrice sovradimensionata */

        int r=0,i=0;
        char riga[MAXC];           /* stringa temporanea */

        gets(riga);     /* memorizzazione dell'input */

        while ((strcmp(riga,"")!=0) && (r<MAXR)) {

             strcpy(pagina[ r ],riga); /* copia della stringa inserita */
             r++;

             /* prossima stringa */
             gets(riga);

        }

        /* visualizzazione delle stringhe memorizzate */
        while (i!=r)
        printf("\n%s",pagina[i++]);
 }
```

Esercizio A_1 - CharArr vs CharStarPo

Struttura Dati:

char * pagina[6];



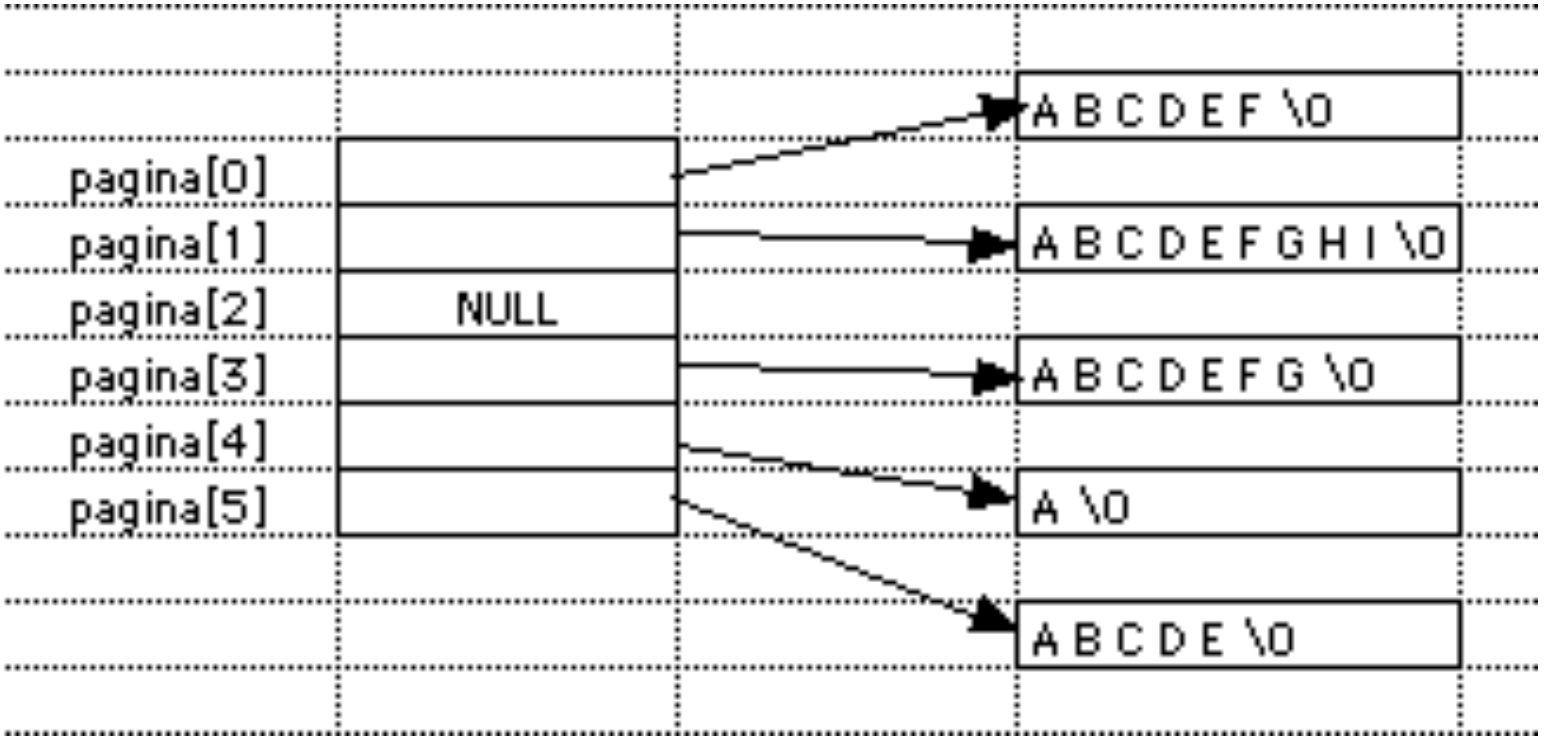
Esercizio A_1 - CharArr vs CharStarPo


```
#define MAXR 6
#define MAXC 81

void main() {
 /* puntatori alle stringhe inserite */
char *pagina[MAXR];

char riga[MAXC];        /* stringa temporanea */
int r=0,i=0;
        gets(riga);     /* memorizzazione dell'input */

        while ((strcmp(riga,"")!=0) && (r<MAXR)) {

            /* memorizzazione della stringa */
            pagina[r]=mem_str(riga);
            /* prossima stringa */
            r++;
            gets(riga);

        }
        /* visualizzazione e rilascio delle stringhe inserite */
        while (i!=r) {
            elim_str(&pagina[i]); /* pagina[i] = elim_str2(pagina[i]); */
            i++;
        }
}
```

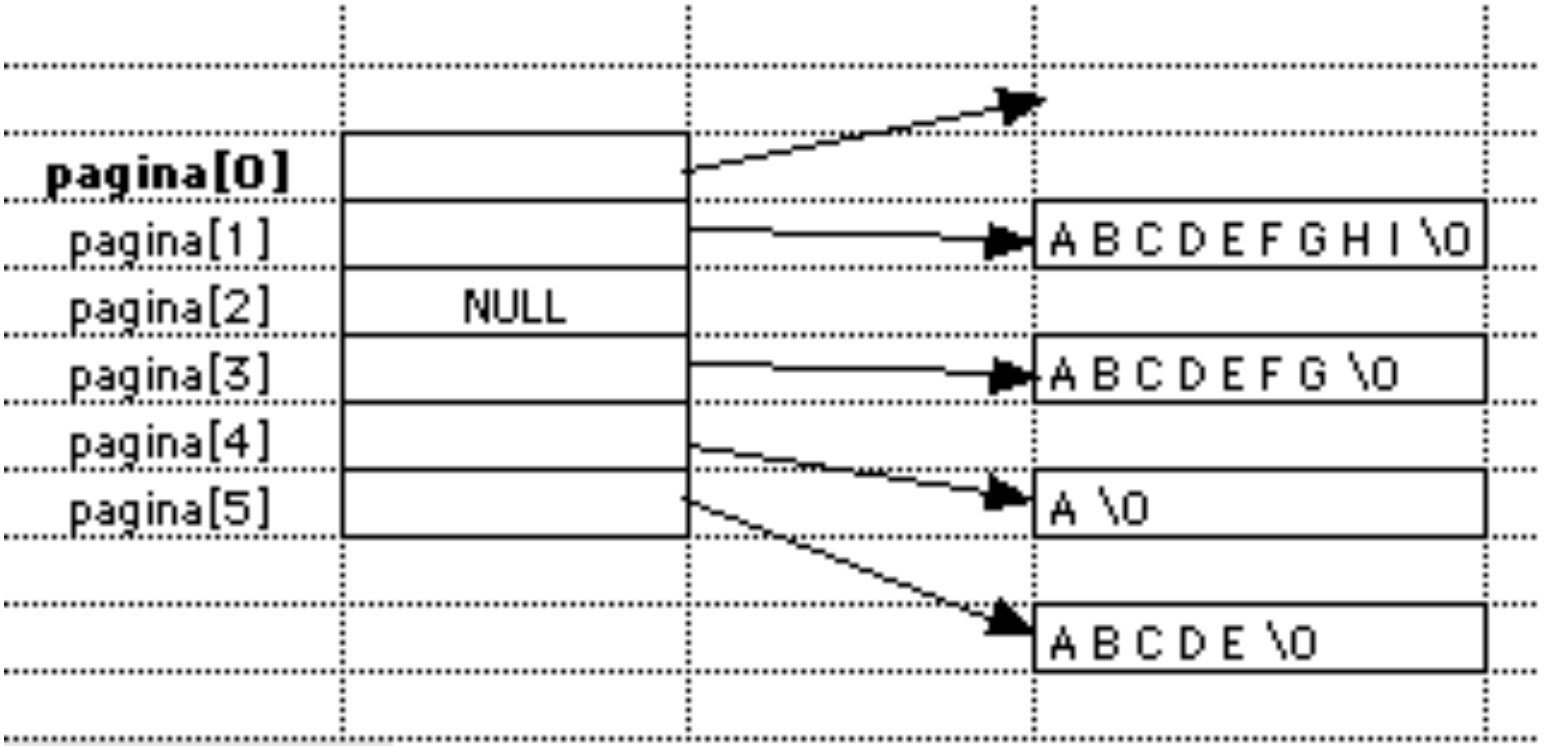
Esercizio A_1 - CharArr vs CharStarPo


```
/* alloca la memoria e memorizza la stringa passata come
parametro */
char *mem_str(char str[])
{
        char *punt;

        /* allocazione della memoria */

        punt=(char *) malloc( (strlen(str)+1) * sizeof(char) );

        if (punt==NULL)
              printf("\nErrore in allocazione!");
        else
              /* copia della stringa inserita */
              strcpy(punt,str);

        /* indirizzo di memoria restituito */
        return (punt);
}
```

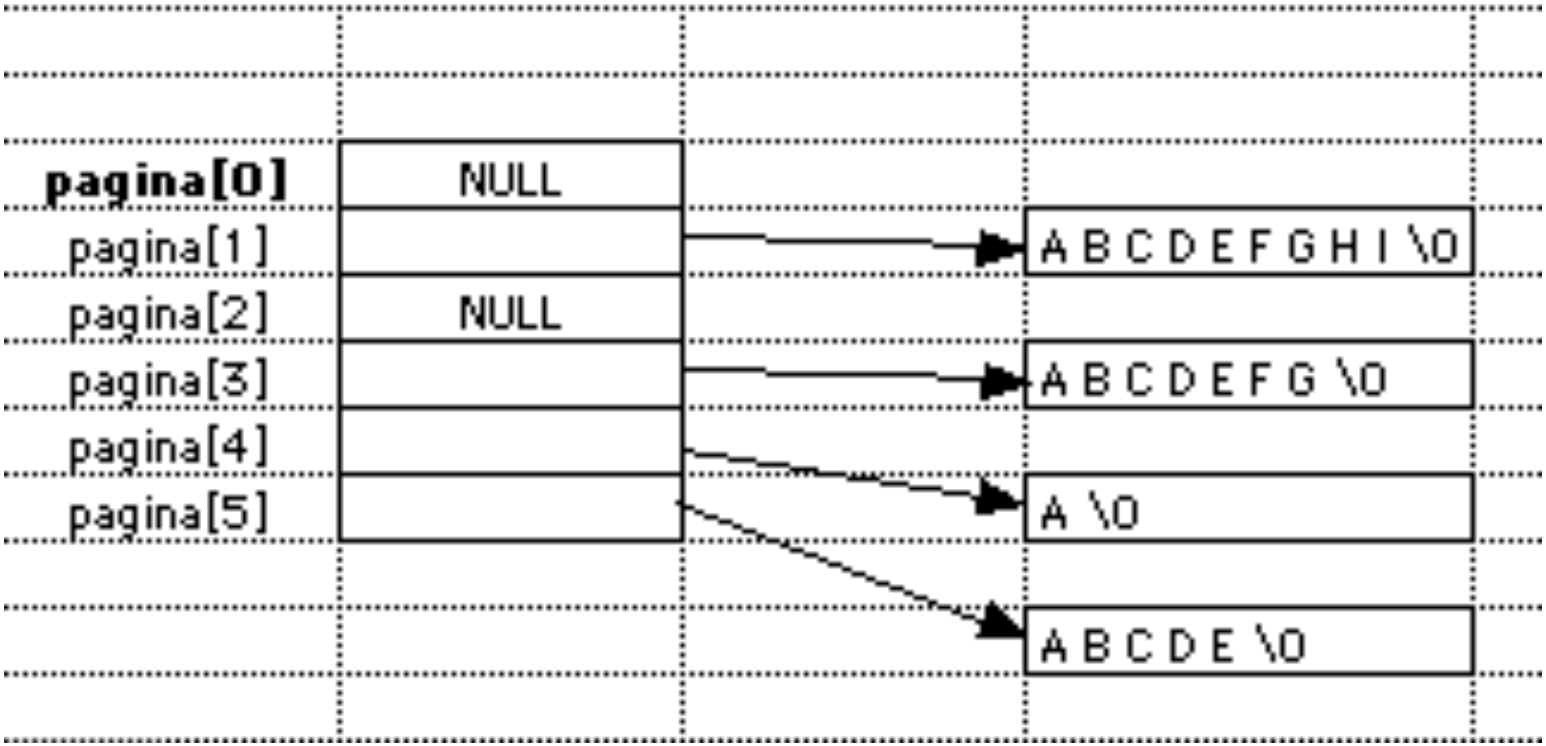
Esercizio A_1 - CharArr vs CharStarPo



Esercizio A_1 - CharArr vs CharStarPo



Esercizio A_1 - CharArr vs CharStarPo



Esercizio A_1 - CharArr vs CharStarPo


```
/* visualizzazione e rilascio delle stringhe inserite */
void elim_str(char **str)
{
        /* visualizzazione */

        printf("\n%s",*str);



        /* rilascio della memoria */
        free(*str);
        *str=NULL;




}
```

Inserimento e cancellazione da un array di (puntatori a) strutture.

Esercizio A_2 - Anagrafica

Si gestisca un archivio di dati anagrafici di un certo numero di persone (MAX predefinito).

Le operazioni da implementare su tale archivio sono:

- ✓ inserimento;
- ✓ eliminazione;
- ✓ visualizzazione.

L'archivio deve essere gestito allocando dinamicamente la memoria, in modo da ottimizzare l'utilizzo di questa risorsa.

Definire le strutture dati necessarie e implementare le funzioni richieste.

Esercizio A_2 - Anagrafica

```
#define MAX 10

typedef struct
{
                char    cognome[20];
                char    nome[20];
                int     eta;

} cella;
```

Esercizio A_2 - Anagrafica


```
void main() {

    /* vettore di puntatori alle strutture */
    cella *vett[MAX];

    /* inizializzazione */
    init(vett);
                              ...

    /* inserimento della nuova anagrafica */
    if (ins(vett, str1, str2, numero)==-1)
        printf("Non ci sono celle disponibili!");
                              ...

    /* eliminazione in base al cognome specificato */
    if (del(vett, str1)==-1)
        printf("Il cognome richiesto non è stato trovato!");
                              ...

    /* rilascio della memoria */
    ril(vett);
}
```

Esercizio A_2 - Anagrafica


```
/* inizializzazione */
void init(cella *lista[])
{


        int i;

                for (i=0;i<MAX;i++)
                        lista[i]=NULL;




}
```

Esercizio A_2 - Anagrafica


```
/* inserimento di un'anagrafica in una cella disponibile */
int ins(cella *seq[], char *c, char *n, int numero) {
        int i;

        /* ricerca cella disponibile */
        for(i=0;i<MAX;i++) {
                if (seq[i]==NULL) {
                        /* allocazione */
                        seq[i] = (cella *) malloc (sizeof(cella));

                        /* inserimento */
                        strcpy(seq[i]->cognome,c);
                        strcpy(seq[i]->nome,n);
                        seq[i]->eta=numero;

                        return (i);
                }
        }

        return (-1);
}
```

Esercizio A_2 - Anagrafica


```
/* eliminazione di una anagrafica */
int del(cella *array[], char c[]) { /* del(cella **p, char c[]) */
        int i;

        /* ricerca persona nell'array */
        for(i=0;i<MAX;i++) {
                if ((array[i]!=NULL) && (strcmp(array[i]->cognome,c)==0))
                {
                        /* eliminazione */
                        /* eventualmente leggere prima il valore */
                        free(array[i]);
                        array[i]=NULL;

                        return (i);
                }
        }

        return (-1);

}
```

Esercizio A_2 - Anagrafica

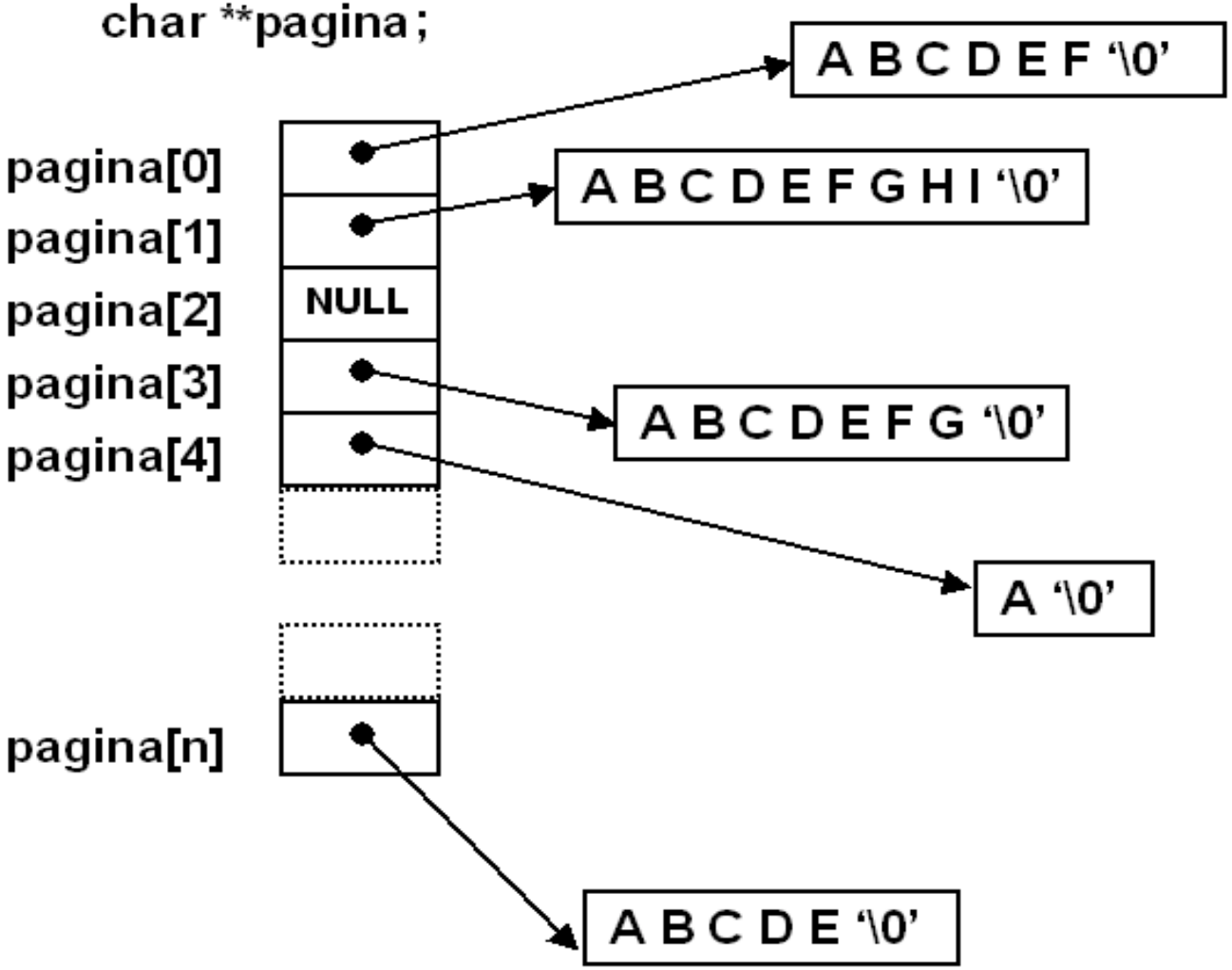
```
/* rilascio finale della memoria */
void ril(cella *lista[])
{
        int i;

        for (i=0;i<MAX;i++)
                if (lista[i]!=NULL)
                {
                        free(lista[i]);
                        lista[i] = NULL;
                }

}
```

Vettore di puntatori a stringhe allocato dinamicamente.

Esercizio A_3 - Dynamic CharStarPo

Struttura Dati:

char **pagina;



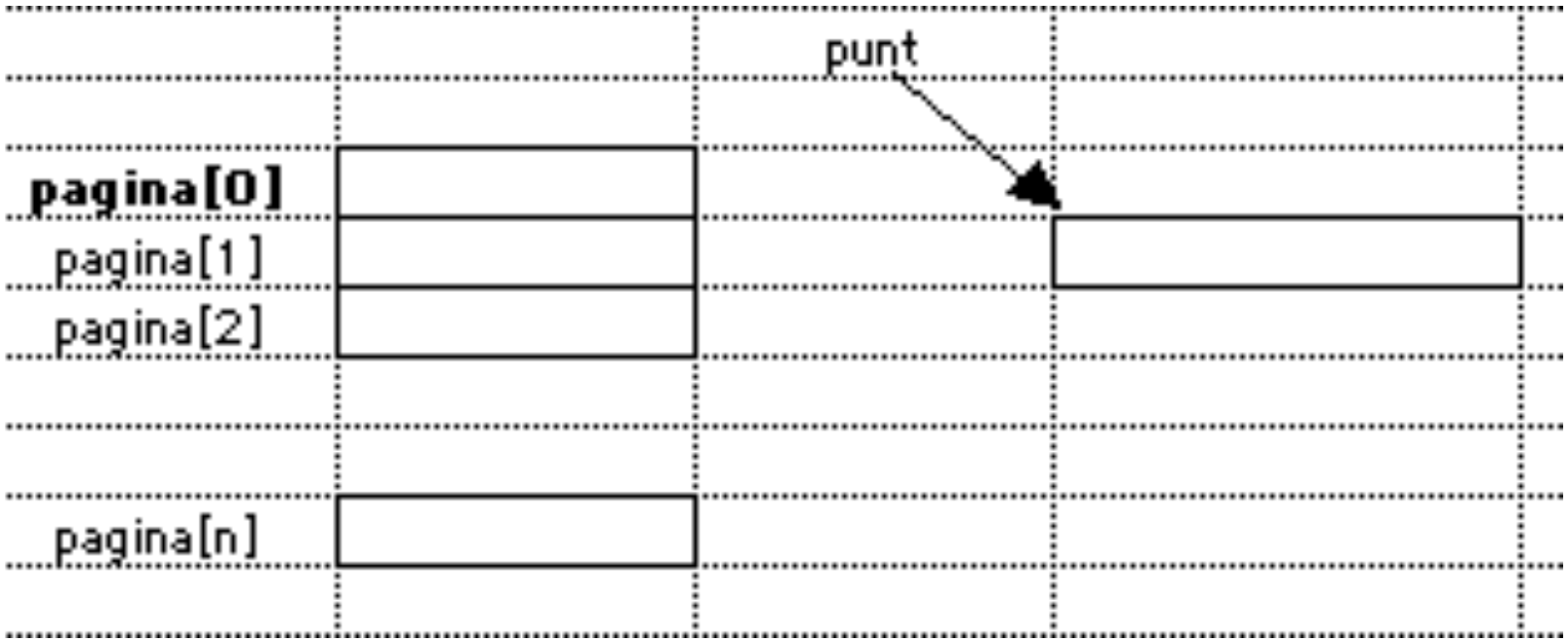
Esercizio A_3 - Dynamic CharStarPo


```
#define MAXC 81

typedef char * pts;

void main() {
        /* puntatore all'inizio del vettore dinamico */
        pts *pagina; //char **pagina;
        /*ogni elemento è a sua volta un puntatore a char*/
        char riga[MAXC];          /* stringa temporanea */
        int r=0,i=0, n;

        /* inserimento numero di righe */
        scanf("%d", &n);

        /* allocazione del vettore di puntatori */
        pagina = (pts *) malloc (sizeof(pts) * n);
        /*pagina = (char **) malloc (sizeof(char *) * n);*/
                              /*...*/
```

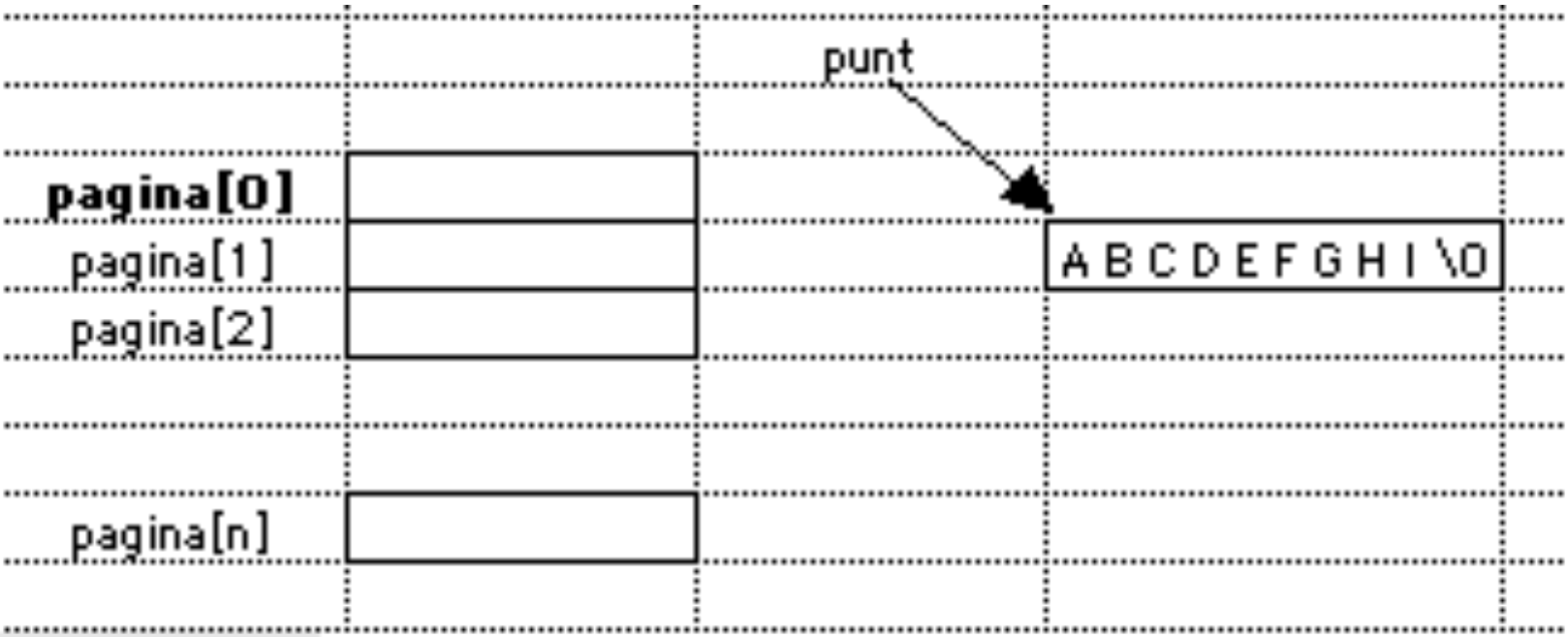
Esercizio A_3 - Dynamic CharStarPo

```
                                /*...*/

        /* memorizzazione dell'input (singole stringhe) */
        gets(riga);

        while ((strcmp(riga,"")!=0) && (r< n))  {

                /* memorizzazione della stringa */
                pagina[ r ]=mem_str(riga);
                r++;

                /* prossima stringa */
                gets(riga);
        }

        /* visualizzazione e rilascio delle stringhe inserite */
        while (i!=r) {
                elim_str(&pagina[i]);
                i++;
        }
}
```

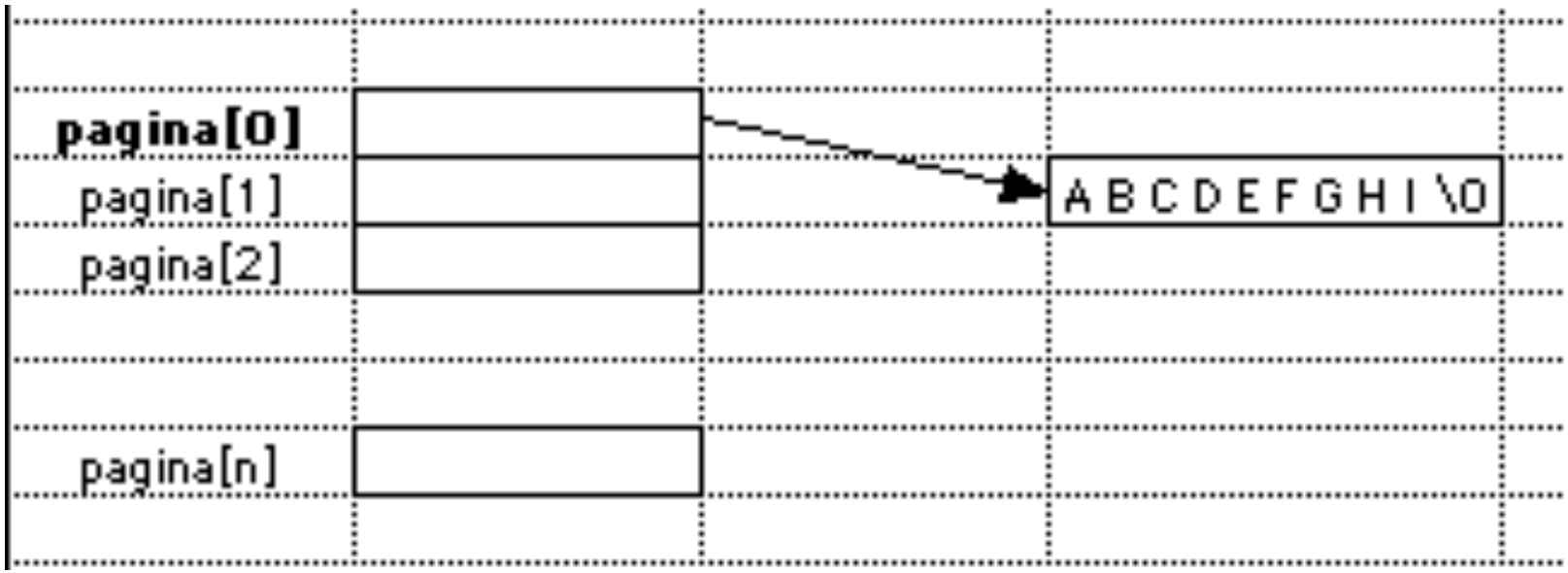
Esercizio A_3 - Dynamic CharStarPo



Esercizio A_3 - Dynamic CharStarPo



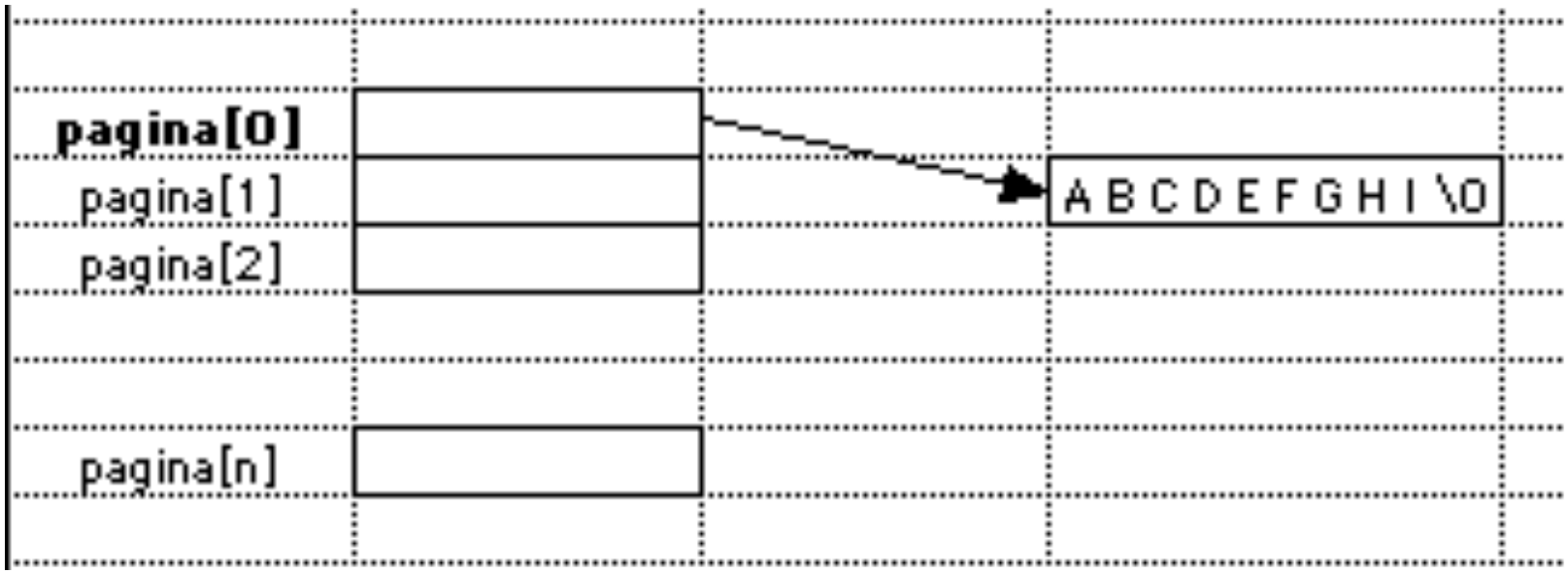
Esercizio A_3 - Dynamic CharStarPo



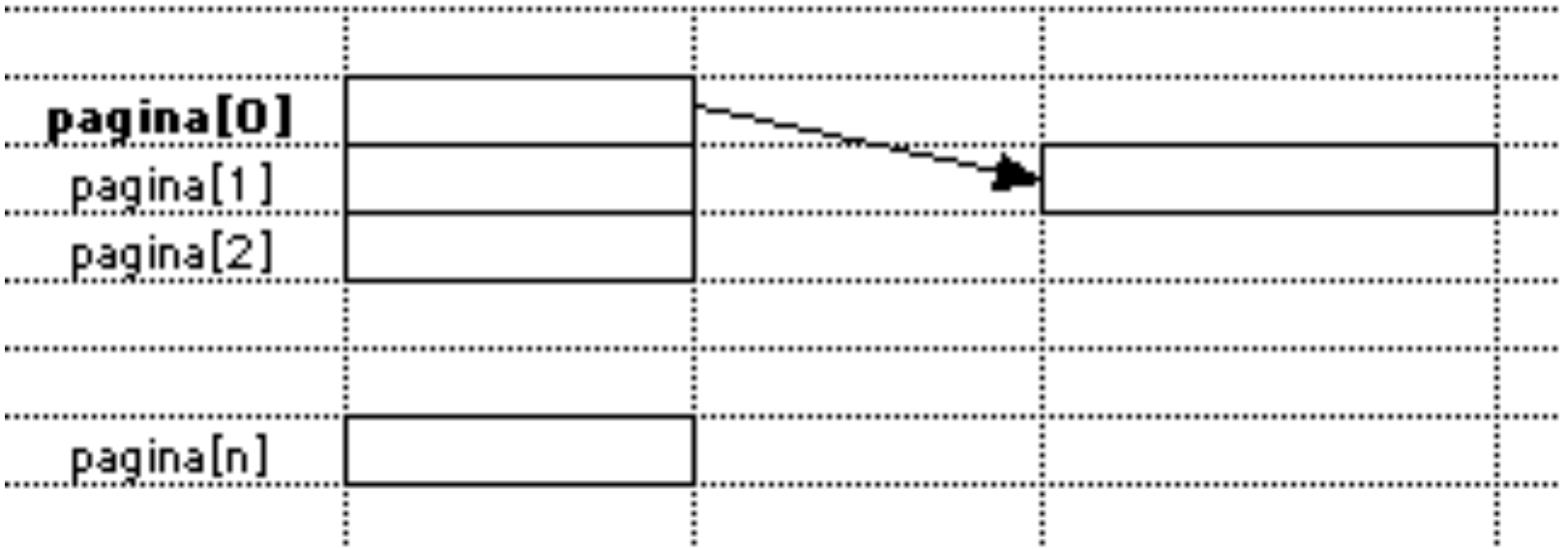
Esercizio A_3 - Dynamic CharStarPo


```
/* alloca la memoria e memorizza la stringa passata come
parametro */
char *mem_str(char str[])
{
        char *punt;

        /* allocazione della memoria */
        punt=(char *) malloc( (strlen(str)+1) * sizeof(char) );

        if (punt==NULL)
                printf("\nErrore in allocazione!");
        else
                /* copia della stringa inserita */
                strcpy(punt,str);


        /* indirizzo di memoria restituito */
        return (punt);


}
```

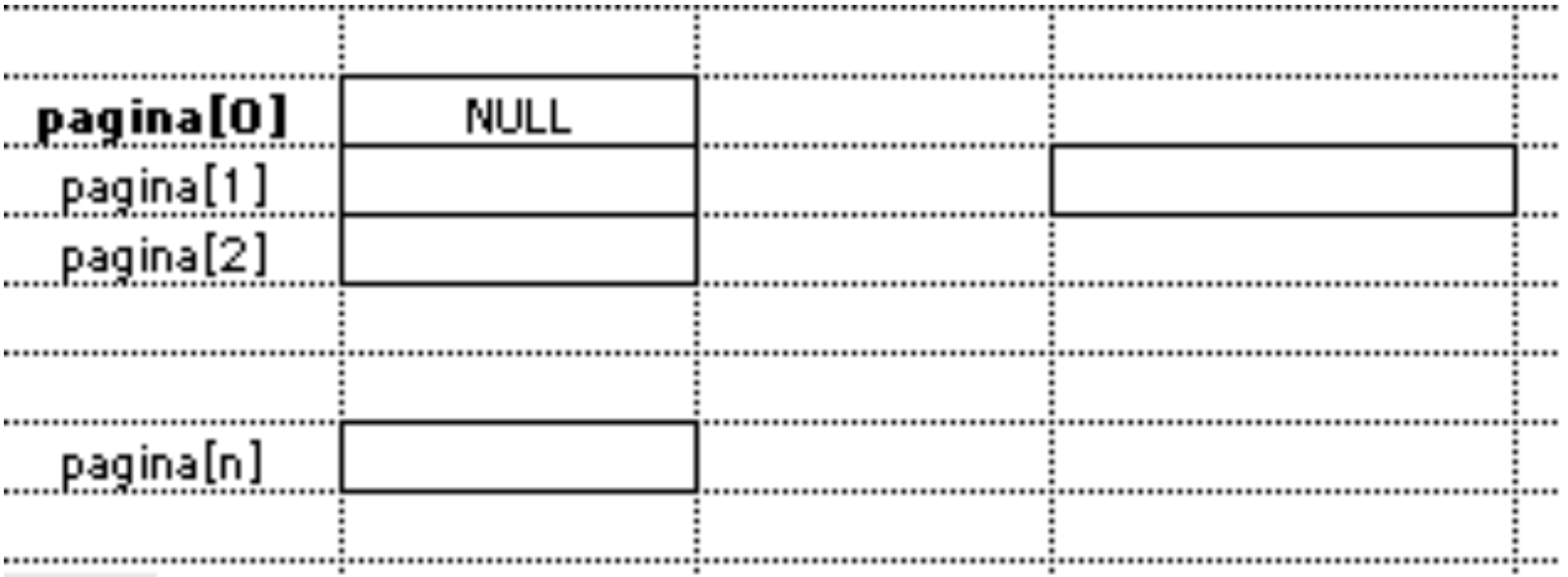
Esercizio A_3 - Dynamic CharStarPo



Esercizio A_3 - Dynamic CharStarPo



Esercizio A_3 - Dynamic CharStarPo



Esercizio A_3 - Dynamic CharStarPo


```
/* visualizzazione e rilascio delle stringhe inserite */
void elim_str(char *str[])
{



        /* visualizzazione */
        printf("\n%s",*str);

        /* rilascio della memoria */
        free(*str);
        *str=NULL;









}
```