# Allocazione dinamica della memoria (2)



## Allocazione dinamica e funzioni



Con il seguente programma si vorrebbe allocare dinamicamente un vettore all'interno di una funzione (leggi_vettore)

```
#include <stdio.h>
#include <stdlib.h>

void leggi_vettore(int *Vf, int *num);

void main (void)
{  int *V,i,numero=0;

  V=NULL;
  leggi_vettore(V,&numero);
   for(i=0;i<numero;i++)
           printf("%d ",V[i]);
}

void leggi_vettore(int *Vf, int *num)
{ int i;

  printf("Quanti numeri vuoi leggere ?");
  scanf("%d",num);
  Vf=malloc((*num)*sizeof(int));
  for(i=0;i<*num;i++)
         { printf("Nuovo numero ");
      scanf("%d",&Vf[i]);
    }
}
```

# Allocazione dinamica e funzioni



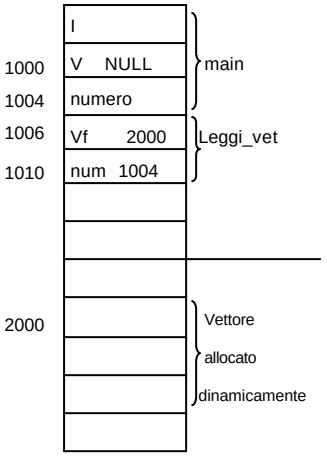
Tuttavia il precedente programma si comporta in modo diverso rispetto a quanto si vorrebbe ottenere.

Come si può vedere nella rappresentazione della memoria, quando viene invocata la funzione leggi_vettore, vengono allocate le locazioni per il parametro formale Vf, in cui viene copiato il valore del parametro attuale V, e per il parametro num, in cui viene copiato l'indirizzo del parametro attuale numero.

Durante l'esecuzione della funzione viene allocato un vettore il cui indirizzo iniziale viene assegnato al parametro Vf e vengono letti i valori degli elementi del vettore.

Al termine della funzione l'area associata a Vf viene liberata e il valore iniziale del vettore allocato viene perso.

La ragione è che la funzione non modifica il valore del parametro attuale che è stato passato per valore.






# Passaggio per indirizzo del puntatore



E' necessario passare il puntatore V per indirizzo.

Il prototipo della funzione leggi_vettore è il seguente:

```
void leggi_vettore(int **Vf, int *num);
```

In questo modo, all'attivazione della funzione leggi_vettore, nel parametro formale Vf viene copiato l'indirizzo del parametro attuale e ogni modifica attuata alla locazione *Vf si ripercuote in una modifica del parametro attuale V.

## Versione corretta del programma con passaggio per indirizzo del puntatore



```
#include <stdio.h>
#include <stdlib.h>

void leggi_vettore(int **Vf, int *num);

void main (void)
{  int *V,i,numero=0;

  V=NULL;
  leggi_vettore(&V,&numero);
   for(i=0;i<numero;i++)
     printf("%d ",V[i]);
}

void leggi_vettore(int **Vf, int *num)
{ int i;

  printf("Quanti numeri vuoi leggere ?");
  scanf("%d",num);
  *Vf=malloc((*num)*sizeof(int));
  for(i=0;i<*num;i++)
   { printf("Nuovo numero ");
      scanf("%d",&(*Vf)[i]);
    }
}
```



## Allocazione dinamica di una tabella con un numero fisso di righe e variabile di colonne

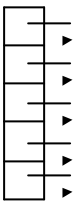


Per poter allocare una tabella di cui è noto a priori il numero delle righe, ma non è noto il numero di colonne è necessario dichiarare un vettore i cui elementi sono dei puntatori agli elementi della riga.

Es.

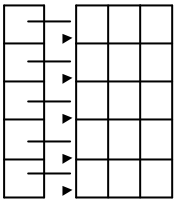
Se si vuole allocare una tabella di interi con 5 righe, di cui non si conosce a priori il numero di colonne, è necessario fare la seguente dichiarazione:

```
int *T[5];
```



Il vettore T viene allocato staticamente quando il programma viene mandato in esecuzione.

Il passo successivo è l'allocazione dinamica di ciascuna riga mediante la malloc().



```
for(i=0;i<5;i++)
  T[i]=malloc(sizeof(int)* dim_riga);
```

## Allocazione dinamica di una tabella con un numero fisso di righe e variabile di colonne



- Al termine dell'esecuzione, per liberare lo spazio allocato precedentemente per la tabella è necessario liberare lo spazio allocato per ciascuna riga.
  Es.

```
for(i=0;i<5;i++)
  free(T[i]);
```



## Allocare una tabella con 5 righe e un numero di colonne letto da tastiera



```
#include <stdio.h>
#include <stdlib.h>

void main (void)
{
 int *T[5];
 int riga,colonna,dim_riga;

 printf("Inserire il numero di colonne\n");
 scanf("%d",&dim_riga);
 /* Allocazione delle righe */
 for(riga=0;riga<5;riga++)
    T[riga]=malloc(sizeof(int)*dim_riga);
 /* Assegnamento degli elementi */
 for(riga=0;riga<5;riga++)
  for(colonna=0;colonna<dim_riga;colonna++)
      T[riga][colonna]=riga*colonna;

 for(riga=0;riga<5;riga++)
   for(colonna=0;colonna<dim_riga;colonna++)
        printf("%d\t",T[riga][colonna]);
 /* Deallocazione della memoria */
 for(riga=0;riga<5;riga++)
  free(T[riga]);
}
```

## Allocare dinamicamente una matrice con 11 righe, pari al numero massimo di esami, e un numero variabile di colonne (1)



```
/* Questo programma legge da file coppie di variabili (matricola,
   esami) e inserisce in una tabella di 11 righe (pari al numero
   massimo di esami) il numero di matricola nella riga di indice pari
   al valore della variabile esami.
   Per effettuare l'inserimento in una riga è necessario incrementarne
   la dimensione mediante la funzione realloc.
   Per tenere traccia della dimensione di ogni riga si fa uso di un
   vettore D i cui elementi D[i] indicano la dimensione della riga
   S[i]. */

#include <stdio.h>
#include <stdlib.h>

main (void)
{
 FILE *pf;
 long *S[11],*Vaux,matricola;
 int riga,colonna,D[11],esami;

/* Inizializzazione dei vettore */
 for(riga=0;riga<11;riga++)
    { S[riga]=NULL;
      D[riga]=0;
    }
```



## Allocare dinamicamente una matrice con 11 righe, pari al numero massimo di esami, e un numero variabile di colonne (2)



```
pf=fopen("studenti.txt","r");
 if(pf)
  { while(!feof(pf))
    { /*Lettura da file */
      fscanf(pf,"Matricola: %ld\n",&matricola);
      fscanf(pf,"Esami: %d\n",&esami);
      /* Incremento della dimensione della riga S[esami] */
      Vaux=realloc(S[esami],(D[esami]+1)*sizeof(long));
      if(Vaux) /* L'allocazione ha avuto successo */
            { S[esami]=Vaux;
              S[esami][D[esami]]=matricola;
               D[esami]++;
            }
      else {printf("Memoria esaurita\n");
             break;
            }
    }
   fclose(pf);
  }
```

## Allocare dinamicamente una matrice con 11 righe, pari al numero massimo di esami, e un numero variabile di colonne (3)



```
for(riga=0;riga<11;riga++)
    { printf("\nElenco studenti con %d esami\n",riga);
       for(colonna=0;colonna<D[riga];colonna++)
          printf("%ld\t",S[riga][colonna]);
     }

 for(riga=0;riga<11;riga++)
  free(S[riga]);

}
```



## Allocare dinamicamente una matrice con 11 righe (massimo numero di esami) e un numero variabile di colonne (V2) (1)



```
/* Questo programma è la versione con le funzioni del programma
   precedente*/

#include <stdio.h>
#include <stdlib.h>

void leggi_da_file(FILE *pf, int **Vf,int *Df);
void visualizza(int **Vf,int *Df);

void main (void)
{ FILE *pf;
  long *S[11];
  int riga,colonna,D[11];

for(riga=0;riga<11;riga++)
    { S[riga]=NULL;
      D[riga]=0;
    }
  pf=fopen("studenti.txt","r");
  if(pf)
  { leggi_da_file(pf,S,D);
    fclose(pf);
  }
  else printf("Errore in lettura");
  visualizza(S,D);
  for(riga=0;riga<11;riga++)
    free(S[riga]);
}
```

## Allocare dinamicamente una matrice con 11 righe (massimo numero di esami) e un numero variabile di colonne (V2) (2)

Prof. G. Ascia

```
void leggi_da_file(FILE *pf, int **Vf,int *Df)
{ long *Vaux,matricola;
  int esami;

   while(!feof(pf))
    { fscanf(pf,"Matricola: %ld\n",&matricola);
      fscanf(pf,"Esami: %d\n",&esami);
      /* Incremento della dimensione della riga S[esami] */
      Vaux=realloc(Vf[esami],(Df[esami]+1)*sizeof(long));
      if(Vaux) /* L'allocazione ha avuto successo */
            { Vf[esami]=Vaux;
              Vf[esami][Df[esami]]=matricola;
               Df[esami]++;
            }
      else {printf("Memoria esaurita\n"); break;
             }
    }
}

void visualizza(int **Vf,int *Df)
{ int riga,colonna;

  for(riga=0;riga<11;riga++)
    { printf("\nElenco studenti con %d esami\n",riga);
       for(colonna=0;colonna<Df[riga];colonna++)
          printf("%ld\t",Vf[riga][colonna]);
     }
}
```



## Leggere da tastiera corso di laurea, matricola ed esami e inserire in una tabella con un numero di righe pari al numero di corsi di laurea (1)

Prof. G. Ascia

```
#include <stdio.h>
#include <stdlib.h>

struct studente {
   long matricola;
   int esami;    };

void inserisci(struct studente **VC,int *DC);
void visualizza(struct studente **VC,int *DC);

main (void)
{
 FILE *pf;
 struct studente *CL[2];
 int corso,D[2];

 for(corso=0;corso<2;corso++)
    { CL[corso]=NULL;
      D[corso]=0;
    }
  inserisci(CL,D);
  visualizza(CL,D);

 for(corso=0;corso<2;corso++)
  free(CL[corso]);

}
```

## Leggere da tastiera corso di laurea, matricola ed esami e inserire in una tabella con un numero di righe pari al numero di corsi di laurea (2)



```
void inserisci(struct studente *VC[],int *D)
{  struct studente *Vaux,A;
   int esami;

/* Vengono inseriri nuovi dati fino a quando la matricola è non nulla
   */
  do
    { printf("Matricola: ");
      scanf("%ld",&A.matricola);
      if(A.matricola)
       { printf("Esami: ");
         scanf("%d",&A.esami);
         printf("Corso di Laurea (0=Ambientale, 1=Telematica)\n");
         scanf("%d",&esami);
         Vaux=realloc(VC[esami],(D[esami]+1)*sizeof(struct studente));
          if(Vaux)
            { VC[esami]=Vaux;
              VC[esami][D[esami]]=A;
                  D[esami]++;
            }
          else {printf("Memoria esaurita\n");
                break;
                }
        }
       else printf("Fine inserimento\n");
   }
  while(A.matricola);
}
```



## Leggere da tastiera corso di laurea, matricola ed esami e inserire in una tabella con un numero di righe pari al numero di corsi di laurea (3)



```
void visualizza(struct studente *VC[],int *DC)
{
 int r,c;

 for(r=0;r<2;r++)
    { if(r==0)
         printf("\nElenco studenti di Ingegneria Ambientale\n");
      else printf("\nElenco studenti di Ingegneria Telematica\n");
      for(c=0;c<DC[r];c++)
        printf(" Matr: %ld Esami: %d\n",VC[r][c].matricola,VC[r][c].esami);
     }
}
```

## Allocazione dinamica di una tabella di cui non è noto a priori il numero delle righe e delle colonne



Per allocare un tabella di cui non è nota a priori la dimensione delle righe e delle colonne è necessario:

1) Allocare dinamicamente un vettore i cui elementi sono dei puntatori a un tipo base. Ad es. (int *) o (long *).
2) Allocare dinamicamente ogni riga mediante la malloc() o la realloc().

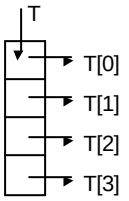
Per potere realizzare il primo punto è necessario definire un puntatore a un elemento di tipo puntatore.

Es. Volendo allocare una tabella di **int** è necessario definire un puntatore ad elementi di tipo (int *) ovvero:

```
(int *) *T;
```

o più semplicemente:

```
int **T;
```




L'allocazione del vettore di puntatori a int viene fatta nel seguente modo:

```
T=malloc(numero_righe*sizeof(int *));
```

Gli elementi T[i] sono i puntatori al primo elemento delle righe da allocare dinamicamente.



## Allocazione dinamica di una tabella di cui non è noto a priori il numero delle righe e delle colonne



L'allocazione dinamica delle righe avverrà con le stesse modalità viste per le tabelle con un numero predefinito di righe ovvero:

```
for(i=0;i<numero_righe;i++)
   T[i]=malloc(dim_riga*sizeof(int));
```

Per liberare la memoria precedentemente allocata per l'intera tabella è necessario;

1) Liberare tutte le righe precedentemente allocate;
2) Liberare il vettore di puntatori alle righe

Es.

```
for(i=0;i<numero_righe;i++)
     free(T[i]);
free(T);
```

## Allocare una tabella il cui numero di righe e colonne è letto da tastiera



```
#include <stdio.h>
#include <stdlib.h>

void main (void)
{int **T=NULL,riga,colonna,num_righe,num_colonne;

 printf("Inserire il numero di righe e di colonne\n");
 scanf("%d %d",&num_righe,&num_colonne);

/* Allocazione del vettore di puntatori */
  T=malloc(num_righe*sizeof(int *));
 /* Allocazione delle righe */
 for(riga=0;riga<num_righe;riga++)
    T[riga]=malloc(sizeof(int)*num_colonne);

 for(riga=0;riga<5;riga++)
  for(colonna=0;colonna<dim_riga;colonna++)
      T[riga][colonna]=riga*colonna;

 for(riga=0;riga<5;riga++)
   for(colonna=0;colonna<dim_riga;colonna++)
         printf("%d\t",T[riga][colonna]);

/* Deallocazione della memoria */
 for(riga=0;riga<5;riga++)
   free(T[riga]);
 free(T);
}
```



## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V1) (1)



```
/* Questo programma prevede l'allocazione di una tabella che ha tutte
   le righe della stessa dimensione (17 caratteri per contenere il cod.
   fiscale), ma che ha un variabile il numero di righe.
   Per potere realizzare le operazioni richieste è necessario definire
   la tabella come un puntatore a (char *) ovvero un puntatore a
   puntatore di char (char **CF).
   Poiche' nelle funzioni di inserimento e di cancellazione può essere
   modificato il valore di CF dalla realloc(), tali funzioni ne
   restituiscono il nuovo valore. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void visualizza(char **V, int num);
char **inserimento (char **V,int *pnum);
char **cancellazione(char **V,int *pnum);

main(void)
{  char **CF=NULL;
   int numero=0,i,scelta;

   do {
      printf("Scegli una delle seguenti opzioni\n");
      printf("1) Visualizzazione\n");
      printf("2) Inserimento\n");
      printf("3) Cancellazione\n");
      printf("0) Fine\n");
      scanf("%d",&scelta);
```

## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V1) (2)



```
   switch(scelta) {
          case 1: visualizza(CF,numero);        break;
          case 2: CF=inserimento(CF,&numero);   break;
          case 3: CF=cancellazione(CF,&numero); break;
          default:;    }
   }     while(scelta);
   for(i=0;i<numero;i++)
      free(CF[i]);
  free(CF);
}


void visualizza(char **V, int num)
{ int i;

  for(i=0;i<num;i++)
    printf("%s\n",V[i]);
}
```



## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V1) (3)



```
char **inserimento (char **V,int *pnum)
{
  char **Vaux;

  /* Incremento del numero delle righe */
  Vaux=realloc(V,(*pnum+1)*sizeof(char *));
  if(Vaux)
    { V=Vaux;
      /*Allocazione del vettore di caratteri */
      V[*pnum]=malloc(17*sizeof(char));
      if(V[*pnum])
        {  printf("Codice fiscale: ");
           scanf("%s",V[*pnum]);
            (*pnum)++;

        }
     else {printf("Memoria esaurita\n");
           /* Se non e' possibile allocare il vettore di caratteri
           viene ripristinata la dimensione del vettore di puntatori */
            V=realloc(V,(*pnum)*sizeof(char*));
           }
    }
   else  printf("Memoria esaurita");
  return V;
}
```

## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V1) (4)



```
char **cancellazione(char **V,int *pnum)
{
  char codfiscale[17];
  int pos,i;

  printf("Inserire il codice fiscale da eliminare\n");
  scanf("%s",codfiscale);
  /* Ricerca della posizione della stringa da eliminare */
  pos=0;
  while (pos<*pnum)
     if(!strcmp(codfiscale,V[pos])) break;
      else pos++;
/* Eliminazione della stringa */
 if(pos<*pnum)
 {
  for(i=pos;i<*pnum-1;i++)
     strcpy(V[i],V[i+1]);
  /* Liberazione della memoria allocata per l'ultima riga */
  free(V[*pnum-1];
  /* Riduzione del numero di elementi del vettore di puntatori */
  V=realloc(V,(*pnum-1)*sizeof(char*));
  (*pnum)--;
 }
  return V;
}
```



## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V2) (1)



```
/* In questa nuova versione del programma precedente le funzioni di
   inserimento e di cancellazione modificano il parametro attuale CF
   all'interno delle funzioni. Pertanto tali parametri vengono passati
   per indirizzo. Cio' richiede che il parametri formati siano definiti
   come puntatori di puntatori di puntatore a char ovvero char ***V. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void visualizza(char **V, int num);
void inserimento (char ***V,int *pnum);
void cancellazione(char ***V,int *pnum);

main()
{  char **CF=NULL;
   int numero=0,i;
    int scelta;

   do {
      printf("Scegli una delle seguenti opzioni\n");
      printf("1) Visualizzazione\n");
      printf("2) Inserimento\n");
      printf("3) Cancellazione\n");
      printf("0) Fine\n");
      scanf("%d",&scelta);
```

## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V2) (2)



```
switch(scelta) {
          case 1: visualizza(CF,numero);
                  break;
          case 2: inserimento(&CF,&numero);
                  break;
          case 3: cancellazione(&CF,&numero);
                  break;
          default:;    }
      }
    while(scelta);
  for(i=0;i<numero;i++)
   free(CF[i]);

  free(CF);

}
void visualizza(char **V, int num)
{ int i;

  for(i=0;i<num;i++)
    printf("%s\n",V[i]);
}
```



## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V2) (3)



```
void inserimento (char ***V,int *pnum)
{
  char **Vaux;

  Vaux=realloc(*V,(*pnum+1)*sizeof(char *));
  if(Vaux)
    { *V=Vaux;
      /* Allocazione del vettore di caratteri */
      (*V)[*pnum]=malloc(17*sizeof(char));
      if((*V)[*pnum]) /* Allocazione OK */
       {  printf("Codice fiscale: ");
          scanf("%s",(*V)[*pnum]);
           (*pnum)++;
       }
     else {printf("Memoria esaurita\n");
           /* Se non e' possibile allocare il vettore di caratteri
           viene ripristinata la dimensione del vettore di puntatori */
            (*V)=realloc(V,(*pnum)*sizeof(char*));
           }
    }
   else  printf("Memoria esaurita");
}
```

## Programma con un menù con tre operazioni: visualizzazione, inserimento e cancellazione di codici fiscali (V2) (4)



```
void cancellazione(char ***V,int *pnum)
{
  char codfiscale[17];
  int pos,i;

  printf("Inserire il codice fiscale da eliminare\n");
  scanf("%s",codfiscale);
  /* Ricerca della posizione della stringa da eliminare */
  pos=0;
  while (pos<*pnum)
     if(!strcmp(codfiscale,(*V)[pos])) break;
      else pos++;
 /* Eliminazione della stringa */
 if(pos<*pnum)
 {  for(i=pos;i<*pnum-1;i++)
       strcpy((*V)[i],(*V)[i+1]);
 /* Liberazione della memoria allocata per l'ultima riga */
    free((*V)[*pnum-1];
  /* Riduzione del numero di elementi del vettore di puntatori */
  (*V)=realloc(*V,(*pnum-1)*sizeof(char*));
  (*pnum)--;
 }
}
```