PROGRAMMI

IN

LINGUAGGIO C

Corso di **Programmazione Modulo B**
Prof. GIULIANO LACCETTI

Studentessa
Bellino Virginia
Matr. 408466
Corso di Diploma in Informatica

http://www.dicecca.net

Indice

1. Package per la gestione di una pila
Premessa
Illustrazione della libreria di funzioni
Funzioni principali
- Funzione Push
- Funzione Pop

Funzioni secondarie
- Definizione delle costanti utilizzate nelle diverse funzioni
- Definizione della struttura dati che costituisce ciascun elemento dello stack.
- Funzione che crea lo stack vuoto
- Funzione che verifica lo stato dello stack
- Funzione che riceve l'elemento da inserire nello stack
- Funzione che restituisce l'elemento in cima alla pila
- Funzione che visualizza l'elemento in cima alla pila

2. Package per la gestione di una linked-list
Premessa
Illustrazione della libreria di funzioni
- Funzione inserthead
- Funzione insertmiddle
- Funzione insertend
- Funzione erase
- Funzione displaylista

3. Package per la gestione di un albero binario
Premessa
Illustrazione della libreria di funzioni
- Definizione della struttura dati che costituisce ciascun elemento dell'albero
- Funzione init_albero
- Funzione insertnode
- Funzione ricerca
- Funzione stampa_in_order
- Funzione stampa_pre_order
- Funzione stampa_post_order

- Funzione conta nodi
- Funzione profondità

4. Package per la gestione di un database studenti

Premessa

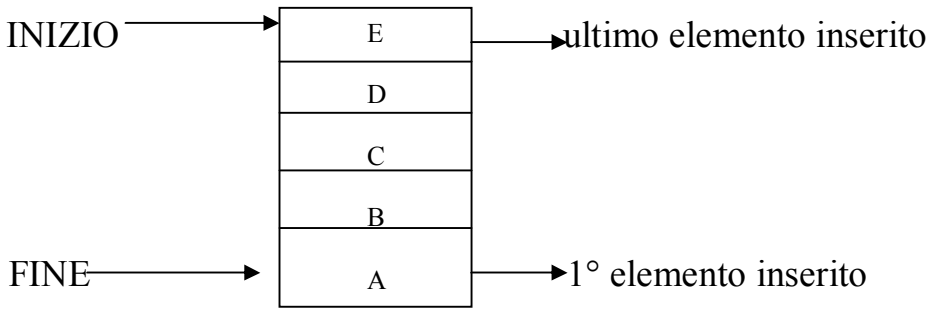
Illustrazione della libreria di funzioni

- Funzione ins_head
- Funzione inserisci
- Funzione doveinserire
- Funzione erase
- Funzione ricerca
- Funzione database output

PACKAGE PER LA GESTIONE DI UNA PILA

Premessa

La PILA è una struttura dati aperta di tipo lineare, in cui è possibile inserire o eliminare elementi soltanto all'inizio della struttura.
Per questo motivo essa viene detta ***struttura LIFO (Last-In-First-Out).***



Il tipo strutturato che rappresenta la pila è lo **stack**, ma poiché esso non esiste come struttura dati, la sua simulazione può essere effettuata ricorrendo ad un array oppure ad una linked-list.
Tuttavia, la seconda ipotesi è da preferirsi poiché non crea eccessivi problemi se la dimensione dello stack aumenta più del previsto durante l'esecuzione del programma.

Il seguente package si compone di due funzioni che effettuano le operazioni principali consentite sullo stack (*inserimento* ed *eliminazione*), e di alcune funzioni secondarie atte a consentirne una più agevole gestione.

FUNZIONI SECONDARIE

Definizione delle costanti utilizzate nelle diverse funzioni

```
#define ERR -1          /* costante che serve per segnalare un errore */
#define EMPTY 1    /* costante che segnala se lo stack è vuoto  */
#define FULL 2      /* costante che segnala se lo stack è pieno  */
#define OTHER 3  /* altro */
```

Definizione della struttura dati che costituisce ciascun elemento dello stack

```
typedef struct
{
        tipo_el *vett;  /*puntatore al prossimo elemento */
        int dim;
        int top;  /*puntatore alla cima dello stack */

}stack;
```

FUNZIONE CHE CREA LO STACK VUOTO

Documentazione esterna

- **Scopo**: la funzione crea uno stack vuoto.

- **Specifiche:**

  stack * initstack (int n)

- **Descrizione:** avvalendosi dell'ausilio di alcune costanti e ricevendo in input un puntatore, la funzione verifica se lo stack è in grado di accogliere elementi.

- **Lista dei parametri:**

  n: intero. Dimensione massima dello stack.

Function initstack
In C:

```
stack * initstack (int n)
{
        stack *tmp;

        tmp =(stack *) malloc (sizeof(stack));
        tmp ->vett=(int *) calloc (n, sizeof(int));
        tmp ->top=0;
        tmp ->dim=n;
        return tmp;
}
```

FUNZIONE CHE VERIFICA LO STATO DELLO STACK

Documentazione esterna

- **Scopo**: la funzione verifica lo stato dello stack, e quindi se esso può accogliere elementi.

- **Specifiche:**

  int stato_stack (stack *stackpunt)

- **Descrizione:** la function riceve in input la dimensione massima n dello stack, alloca n posizioni di memoria e restituisce al programma chiamante un puntatore allo stack vuoto.

- **Lista dei parametri:**

  ***stackpunt:** puntatore alla struttura che rappresenta ciascun elemento dello stack.

Function stato_stack
In C:

```
int stato_stack (stack *stackpunt)
{
/*se lo stack non è stato creato viene rimandato un messaggio di errore */
        if (stackpunt == NULL)
                return ERR;

        else if (( stackpunt ->top) == 0) /*se il puntatore alla cima è 0 allora… */
                return EMPTY;  /*..lo stack esiste ma è vuoto */

        else if ((stackpunt ->top) >= (stackpunt ->dim)) /*se il puntatore alla cima è
>= alla.. */
                return FULL; /*..dim massima allora lo stack è pieno */

        else return OTHER; /*in questo caso si può inserire */
}
```

FUNZIONE CHE RICEVE L'ELEMENTO DA INSERIRE NELLO STACK

Documentazione esterna

- **Scopo**: la funzione accoglie l'elemento che deve essere inserito nello stack.

- **Specifiche:**

  int inserimento (void)

- **Descrizione:** la funzione stampa il messaggio che indica all'utente quando inserire l'elemento.

Function inserimento

In C:

```
int inserimento (void)
{
        int n; /*variabile che indica l'elemento da inserire */
        printf ("\n Inserire l'elemento: ");
        scanf ("%d", &n);

        return n;
}
```

FUNZIONE CHE RESTITUISCE L'ELEMENTO IN CIMA ALLA PILA

Documentazione esterna

- **Scopo:** restituire il valore dell'elemento in cima alla pila:

- **Specifiche:**

  int top (stack *st)

- **Descrizione:** dopo aver verificato se lo stack contiene elementi, in caso affermativo, la funzione restituisce al programma chiamante il valore dell'elemento posto in cima alla pila.

- **Lista dei parametri:**

  ***st:** puntatore alla struttura che rappresenta ciascun elemento dello stack.

- **Funzioni ausiliarie:**

  int stato_stack (stack *stackpunt)

Function top

In C:

```
int top (stack *st)
{
        int stato;
        stato = stato_stack (st);
/*se lo stack contiene elementi, allora la funzione restituisce il valore dell'elemento in
cima alla */     /* pila */
        if (stato != ERR && stato != EMPTY)
        return st ->vett[st ->top-1];
}
```

FUNZIONE CHE CANCELLA LO STACK

Documentazione esterna

- **Scopo:** deallocazione dello spazio di memoria utilizzato.

- **Specifiche:**

  void deletestack (stack *st)

- **Lista dei parametri:**
  ***st:** puntatore alla struttura che rappresenta ciascun elemento dello stack.

Function deletestack

In C:

```
void deletestack (stack *st)
{

        free ( st );
}
```

FUNZIONE CHE VISUALIZZA L'ELEMENTO IN CIMA ALLA PILA

Documentazione esterna

- **Scopo:** visualizzare l'elemento in cima alla pila:
- **Specifiche:**

  void stampa_el (int el)

- **Lista dei parametri:**

  el: intero. Valore dell'elemento da visualizzare.

Function stampa_el
In C:

```
void stampa_el (int el)
{

        printf ("L'ultimo elemento della pila é: %d \n", el );
}
```

Esempio di programma chiamante che utilizza l'intero insieme di funzioni documentate in precedenza, organizzate sotto forma di una libreria utente denominata *stack.h*

```
#include "stack.h"

main ( )
{
/* variabili */
stack *st;
int n, c, j;
int terminato=0;  /* variabile che indica quando uscire dal programma */

printf ("\n Dimensione Massima dello stack: \n ");
scanf ("%d", &n);

/*creazione dello stack e allocazione di n posizioni di memoria */
        st = initstack (n);

/* menu di scelte possibili */
while (!terminato)
{
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf (" °° PROGRAMMA PER LA GESTIONE DI UNA PILA °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf (" 1 --> Inserimento di un elemento \n");
printf (" 2 --> Estrazione di un elemento \n");
printf (" 3 --> Visualizzazione dell'elemento in cima alla pila \n");
printf (" 4 --> Fine \n");
printf ("\n Eseguire una scelta: \n");
scanf ("%d", &c);

switch (c)
{
        case 1:
                if (stato_stack(st) != FULL)
                        push ( st, inserimento (  ) );
                else
                        printf ("\n Stack pieno \n");
                printf("\n");
                system("pause");
                break;

        case 2:
                if (stato_stack(st) == EMPTY)
                        printf ("\n Stack vuoto \n");
                else
                {
                        j = pop (st);
```

```
                printf ("\n L'elemento estratto è: %d \n", j);
            }
            printf("\n");
            system("pause");
            break;

        case 3:
            if (stato_stack(st) == EMPTY)
                printf ("\n Stack vuoto \n");
            else
                stampa_el (top(st));
                printf("\n");
            system("pause");
            break;

        case 4:
            terminato=1;
            printf("\n Fine del programma \n");
            break;

        default:
            system ("cls");
            printf("Funzione non associata \n \n");
            system ("pause");
            break;

}/*end switch */

}/*end while */

        deletestack (st);
        free (st);

}/*end main */
```

esempio di esecuzione

Dimensione Massima:
3
°°
°°° PROGRAMMA PER LA GESTIONE DI UNA PILA °°
°°
1 --> Inserimento di un elemento
2 --> Estrazione di un elemento
3 --> Visualizzazione dell'elemento in cima alla pila
4 --> Fine

Eseguire una scelta:
1
Inserire l'elemento: 4
Premere un tasto per continuare...

Eseguire una scelta:
1

Inserire l'elemento: 9
Premere un tasto per continuare...

Eseguire una scelta:
1
Inserire l'elemento: 6
Premere un tasto per continuare...

Eseguire una scelta:
1
Stack pieno
Premere un tasto per continuare...

Eseguire una scelta:
3
L'ultimo elemento della pila è: 6
Premere un tasto per continuare…

Eseguire una scelta:
2
L'elemento estratto é: 6
Premere un tasto per continuare...

Eseguire una scelta
7
Funzione non associata
Premere un tasto per continuare...

Eseguire una scelta
4
Fine del programma

LIBRERIA UTENTE STACK.H

(Listato completo in linguaggio c, comprensivo di documentazione interna)

```
#include <stdio.h>
#include <stdlib.h>
#define ERR -1 /*costante che serve per segnalare un errore */
#define EMPTY 1 /*costante che segnala se lo stack è vuoto  */
#define FULL 2 /*costante che segnala se lo stack è pieno  */
#define OTHER 3 /*altro */

/* record che rappresenta ogni elemento dello stack */
typedef struct
{
        tipo_el *vett; /*puntatore al prossimo elemento */
        int dim;
        int top; /*puntatore alla cima dello stack */


}stack;

/* FUNZIONE CHE VERIFICA LO STATO DELLO STACK */
int stato_stack (stack *stackpunt)
{
/*se lo stack non è stato creato viene rimandato un messaggio di errore */
        if (stackpunt == NULL)
                return ERR;

        else if (( stackpunt ->top) == 0) /*se il puntatore alla cima è 0 allora… */
                return EMPTY;  /*..lo stack esiste ma è vuoto */

        else if ((stackpunt ->top) >= (stackpunt ->dim)) /*se il puntatore alla cima è >= alla */
                return FULL; /*dim massima allora lo stack è pieno */

        else return OTHER; /*in questo caso si può inserire */
}

/* FUNZIONE CHE CREA LO STACK VUOTO*/
stack * initstack (int n)
{
        stack *tmp;

        tmp =(stack *)malloc(sizeof(stack));
        tmp ->vett=(tipo_el *)calloc(n,sizeof(tipo_el));
        tmp ->top=0;
        tmp ->dim=n;
        return tmp;
}


/*FUNZIONE CHE ESEGUE L'INSERIMENTO DI UN ELEMENTO NELLO STACK */
void push (stack *st, int el)
```

```
{
        int stato; /* variabile che valuta se lo stack può accogliere elementi */

/*chiamata della funzione che valuta lo stato dello stack */
        stato = stato_stack ( st );
/*se lo stack risulta creato e non pieno allora viene eseguito l'inserimento */
        if (stato != ERR && stato != FULL)
        {
        st->vett[st->top] = el;
        (st->top)++; /*aggiornamento del puntatore alla cima dello stack */
        }
}

/*FUNZIONE CHE ESEGUE L'ELIMINAZIONE DI UN ELEMENTO DALLO STACK */
int pop (stack *st)
{
        int stato, k;

/*chiamata della funzione che valuta lo stato dello stack */
        stato = stato_stack (st);

/*esecuzione dell'estrazione */
        if ((stato != ERR)&&(stato != EMPTY))
{
        k = st->vett[st->top-1];
        (st->top)--; /*aggiornamento del puntatore alla cima dello stack */
}

        return k;
}


/* FUNZIONE CHE RESTITUISCE L'ELEMENTO IN CIMA ALLA PILA */
int top (stack *st)
{
        int stato;
        stato=stato_stack (st);

        if (stato != ERR && stato != EMPTY)
        return st->vett[st->top-1];
}


/* FUNZIONE CHE CANCELLA LO STACK */
void deletestack (stack *st)
{

        free ( st ->vett);
}
```

```
/* FUNZIONE CHE RICEVE L'ELEMENTO DA INSERIRE NELLO STACK */
int inserimento (void)
{
        int n; /*variabile che indica l'elemento da inserire */
        printf ("\n Inserire l'elemento: ");
        scanf ("%d", &n);

        return n;
}


/* FUNZIONE CHE VISUALIZZA L'ELEMENTO IN CIMA ALLA PILA*/
void stampa_el (tipo_el el)
{

        printf ("L'ultimo elemento della pila é: %d \n", el );
}
```

FUNZIONE PUSH
Documentazione esterna

- **Scopo:** la funzione esegue l'inserimento di un dato nello stack.

- **Specifiche:**

  void push (stack *st, int el)

- **Descrizione:** servendosi di alcune costanti simboliche, di una function ausiliaria che valuta lo stato dello stack e di un'altra function che consente di ricevere l'elemento da inserire nello stack, la funzione, utilizzando un ciclo if, dapprima inserisce un elemento nello stack, e poi aggiorna il puntatore alla cima.

- **Lista dei parametri:**

  ***st:** puntatore alla struttura che rappresenta ciascun elemento dello stack.
  el: intero. Rappresenta il dato che viene inserito nello stack.

- **Raccomandazioni d'uso:** prima di utilizzare la funzione push, è necessario definire la struttura che rappresenta ciascun elemento dello stack e il puntatore ad essa. La creazione dello stack vuoto può essere invece effettuata utilizzando l'apposita funzione inclusa nel pacchetto.

- **Funzioni ausiliarie:**

  int stato_stack (stack *stackpunt)
  int inserimento (void)

Funzione push
In C

```
void push (stack *st, int el)
{
        int stato; /* variabile che valuta se lo stack può accogliere elementi */

/*chiamata della funzione che valuta lo stato dello stack */
        stato = stato_stack ( st );
/*se lo stack risulta creato e non pieno allora viene eseguito l'inserimento */
        if (stato != ERR && stato != FULL)
        {
        st->vett[st->top] = el;
        (st->top)++; /*aggiornamento del puntatore alla cima dello stack */
        }
}
```

FUNZIONE POP
Documentazione esterna

- **Scopo:** la funzione esegue l'eliminazione di un dato dallo stack.

- **Specifiche:**

  int pop (stack *st)

- **Descrizione:** servendosi di alcune costanti simboliche e di una function ausiliaria che valuta lo stato dello stack, la funzione, utilizzando un ciclo if, dapprima elimina un elemento dallo stack, e poi aggiorna il puntatore alla cima.

- **Lista dei parametri:**

  ***st:** puntatore alla struttura che rappresenta ciascun elemento dello stack.

- **Raccomandazioni d'uso:** l'utilizzo della funzione pop è strettamente collegato con quello della funzione push, poiché ovviamente bisogna prima creare lo stack ed inserirvi elementi per poi poter eliminare.

- **Funzioni ausiliarie:**

  int stato_stack (stack *stackpunt)

Funzione pop
In C:

```
int pop (stack *st)
{
        int stato, k;

/*chiamata della funzione che valuta lo stato dello stack */
        stato = stato_stack (st);

/*esecuzione dell'estrazione */
        if ((stato != ERR)&&(stato != EMPTY))
{
        k = st->vett[st->top-1];
        (st->top)--; /*aggiornamento del puntatore alla cima dello stack */
}

        return k;
}
```

PACKAGE PER LA GESTIONE DI UNA LINKED-LIST

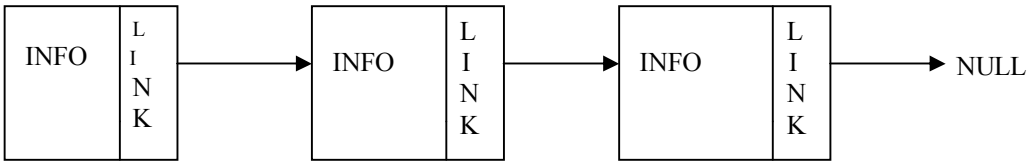
Premessa

La LISTA, (comunemente denominata linked-list), è una struttura dati (aperta o chiusa) di tipo lineare, in cui è possibile inserire o eliminare elementi in qualsiasi posizione .

Tale tipo di struttura risulta molto utile per gestire un insieme di componenti che deve essere continuamente aggiornato con inserimenti ed eliminazioni.

Nella linked-list, ciascun componente, detto NODO, è costituito da un record con due campi:

- *info*: posizione dove sono contenuti i dati veri e propri. Tale campo può essere di tipo scalare (es. int, float….), oppure di tipo strutturato statico (array, record).
- *link*: variabile puntatore contenente l'indirizzo del successivo elemento della lista.



Inoltre, in una lista, l'ordine logico dei componenti non coincide quasi mai con il loro ordine fisico. Infatti, ciascun elemento è univocamente individuato dal link che lo collega al suo predecessore, e quando si effettuano inserimenti ed eliminazioni sono esclusivamente i link ad essere modificati.

ILLUSTRAZIONE DELLA LIBRERIA DI FUNZIONI

FUNCTION INSERTHEAD

Documentazione esterna

- **Scopo:** inserire un elemento in cima ad una linked-list.

- **Specifiche:**

  void inserthead (LINK head)

- **Descrizione:** dopo aver allocato dinamicamente lo spazio di memoria necessario per creare il nuovo nodo da inserire nella lista, la funzione crea tale nodo e lo inserisce in cima alla lista.

- **Lista dei parametri:**

  head: puntatore alla cima della lista.

- **Routine ausiliarie**

  void insertmiddle (LINK head);
  void insertend (LINK head);
  void displaylista (LINK head);

- **Esempio d'uso**

esempio di programma chiamante

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

/* prototipi */
    void inserthead ( LINK head );
    void insertmiddle ( LINK head );
    void insertend ( LINK head );
    void displaylista ( LINK head );

/*definizione della struttura che costituisce ciascun elemento della lista */
        struct data
        {
        char nome[40];
        struct data *NEXT;  /* NEXT punta al prossimo elemento della lista */
        };

typedef struct data LINK; /* definizione del puntatore alla struttura */
typedef struct data LISTANOMI;
```

```
 /* Puntatori:                                  */
 /*    Inizio Lista ( Head )                    */
 /*    Nuovo Elemento della Lista (Newelem)     */
 /*    Posizione corrente nella Lista (Current) */
 /*    Elemento precedente della Lista ( Prec )  */

/*creazione della lista vuota */
 LINK head  = NULL; /* ricordo che LINK e' un tipo puntatore */
 LINK newelem  =  NULL;
 LINK current  = NULL;
 LINK  prec = NULL;

main(  )
{
/* variabili */
        int terminato = 0;
        int c;

while (!terminato)
{
system("cls");
printf ("\n");
printf ("Scegliere l'operazione da effettuare: \n \n");
printf ("1.   Inserimento in testa \n");
printf ("2.   Inserimento nella lista \n");
printf ("3.   Inserimento in coda \n");
printf ("4.   Visualizzazione \n");
printf ("5.   Esci \n");

printf ("\n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch (c)
{

        case 1 :
        inserthead (head);
        displaylista (head);
        system ("pause");
        break;

        case 2:
        insertmiddle(head );
        displaylista ( head);
        system ("pause");
        break;

        case 3:
        insertend( head );
        displaylista ( head);
```

```
        system ("pause");
        break;

        case 4:
        displaylista(head);
        system ("pause");
        break;

        case 5:
        terminato=1;
        printf(" \n Fine del programma...\n \n ");
        break;

        default:
        printf(" \n Funzione non associata , riprovare \n \n");
        system ("pause");
        break;

}/* end switch */

}/* end while */

}/* end main */
```

esempio di esecuzione

```
Scegliere l'operazione da effettuare:
1. Inserimento in testa
2. Inserimento nella lista
3. Inserimento in coda
4. Visualizzazione
5. Esci

Eseguire una scelta:
1

Inserire nome da inserire in cima alla lista:
Peter

_____________________
| Lista corrente... |
 -------------------------------
>  Peter
 ____________________________________

Premere un tasto per continuare...
********************************************************************************
Eseguire una scelta:
1
Inserire nome da inserire in cima alla lista:
Alex

_____________________
| Lista corrente... |
 -------------------------------
>  Alex
>  Peter
 ____________________________________

Premere un tasto per continuare...
```

FUNCTION INSERTHEAD

In C:

```
void inserthead ( LINK head )
{
/*puntatore all'array di tipo char che costituisce il nuovo elemento da inserire */
        char *elemento;

/* creazione del nuovo nodo e suo inserimento nella lista */
        newelem = (LINK)malloc(sizeof(LISTANOMI));
        newelem -> NEXT = head;
        head = newelem;

/* inserimento del nome nel nuovo nodo appena creato */
        elemento = (char*)malloc(sizeof(char)*40);
        puts ("\n Inserire nome in cima alla lista: ");
        gets (elemento);

        strcpy (newelem -> nome, elemento);
        printf("\n");

}
```

FUNCTION INSERTMIDDLE

Documentazione esterna

- **Scopo:** inserire un elemento in cima ad una linked-list.

- **Specifiche:**

  void insertmiddle (LINK head)

- **Descrizione:** dopo aver allocato dinamicamente lo spazio di memoria necessario per creare il nuovo nodo da inserire nella lista, la funzione crea tale nodo e lo inserisce nel mezzo della lista.

- **Lista dei parametri:**

  head: puntatore alla cima della lista.

- **Routine ausiliarie:**

  void insertmiddle (LINK head);
  void insertend (LINK head);
  void displaylista (LINK head);

- **Esempio d'uso**

esempio di programma chiamante

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

/* prototipi */
    void inserthead ( LINK head );
    void insertmiddle ( LINK head );
    void insertend ( LINK head );
    void displaylista ( LINK head );

/*definizione della struttura che costituisce ciascun elemento della lista */
        struct data
        {
        char nome[40];
        struct data *NEXT;  /* NEXT punta al prossimo elemento della lista */
        };

typedef struct data LINK; /* definizione del puntatore alla struttura */
typedef struct data LISTANOMI;
```

```
 /* Puntatori:                                  */
 /*    Inizio Lista ( Head )                    */
 /*    Nuovo Elemento della Lista (Newelem)     */
 /*    Posizione corrente nella Lista (Current) */
 /*    Elemento precedente della Lista ( Prec )    */

/*creazione della lista vuota */
 LINK head  = NULL; /* ricordo che LINK e' un tipo puntatore */
 LINK newelem  =  NULL;
 LINK current  = NULL;
 LINK  prec = NULL;

main(  )
{
/* variabili */
        int terminato = 0;
        int c;

while (!terminato)
{
system("cls");
printf ("\n");
printf ("Scegliere l'operazione da effettuare: \n \n");
printf ("1.   Inserimento in testa \n");
printf ("2.   Inserimento nella lista \n");
printf ("3.   Inserimento in coda \n");
printf ("4.   Visualizzazione \n");
printf ("5.   Esci \n");

printf ("\n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch (c)
{

        case 1 :
        inserthead (head);
        displaylista (head);
        system ("pause");
        break;

        case 2:
        insertmiddle( head);
        displaylista ( head);
        system ("pause");
        break;

        case 3:
        insertend( head );
```

```
            displaylista ( head);
            system ("pause");
            break;

            case 4:
            displaylista(head);
            system ("pause");
            break;

            case 5:
            terminato=1;
            printf(" \n Fine del programma...\n \n ");
            break;

            default:
            printf(" \n Funzione non associata , riprovare \n \n");
            system ("pause");
            break;

}/* end switch */

}/* end while */

}/* end main */
```

esempio di esecuzione

```
Scegliere l'operazione da effettuare
1. Inserimento in testa
2. Inserimento nella lista
3. Inserimento in coda
4. Visualizzazione
5. Esci

Eseguire una scelta:
2

Inserire nome da integrare alla lista:
Dana

 _____________________
| Lista corrente... |
 -------------------------------
>  Alex
>  Dana
>  Peter
 ____________________________________

Premere un tasto per continuare...
```

FUNCTION INSERTMIDDLE

In C:

```
void insertmiddle(LINK head )
{

/*puntatore all'array di tipo char che costituisce il nuovo elemento da inserire */
        char *elemento;

/* creazione del nuovo nodo e suo inserimento nella lista */
        newelem = (LINK)malloc(sizeof(LISTANOMI));
        newelem -> NEXT = head -> NEXT;
        head -> NEXT = newelem;

/* inserimento del nome nel nuovo nodo appena creato */
        puts ("\n Inserire nome da integrare alla lista: ");
        elemento = (char*)malloc(sizeof(char)*40);
        gets (elemento);
        strcpy(newelem -> nome, elemento);
        printf("\n");
}
```

FUNCTION INSERTEND

Documentazione esterna

- **Scopo:** inserire un elemento in coda ad una linked-list.

- **Specifiche:**

  void insertend (LINK head);

- **Descrizione:** partendo dalla testa della lista, la funzione, utilizzando un ciclo di tipo while, scorre tutti gli elementi finché non trova l'ultima posizione (cioè un elemento che punta a NULL), e qui viene inserito il nuovo elemento creato.

- **Lista dei parametri:**

  head: puntatore alla cima della lista.

- **Routine ausiliarie:**

  void insertmiddle (LINK head);
  void inserthead (LINK head);
  void displaylista (LINK head);

- **Esempio d'uso**

esempio di programma chiamante

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

/* prototipi */
    void inserthead ( LINK head );
    void insertmiddle ( LINK head );
    void insertend ( LINK head );
    void displaylista ( LINK head );

/*definizione della struttura che costituisce ciascun elemento della lista */
        struct data
        {
        char nome[40];
        struct data *NEXT;  /* NEXT punta al prossimo elemento della lista */
        };

typedef struct data LINK; /* definizione del puntatore alla struttura */
typedef struct data LISTANOMI;
```

```
 /* Puntatori:                                  */
 /*    Inizio Lista ( Head )                    */
 /*    Nuovo Elemento della Lista (Newelem)     */
 /*    Posizione corrente nella Lista (Current) */
 /*    Elemento precedente della Lista ( Prec )  */

/*creazione della lista vuota */
 LINK head  = NULL; /* ricordo che LINK e' un tipo puntatore */
 LINK newelem  =  NULL;
 LINK current  = NULL;
 LINK  prec = NULL;

main(  )
{
/* variabili */
        int terminato = 0;
        int c;

while (!terminato)
{
system("cls");
printf ("\n");
printf ("Scegliere l'operazione da effettuare: \n \n");
printf ("1.   Inserimento in testa \n");
printf ("2.   Inserimento nella lista \n");
printf ("3.   Inserimento in coda \n");
printf ("4.   Visualizzazione \n");
printf ("5.   Esci \n");

printf ("\n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch (c)
{

        case 1 :
        inserthead (head);
        displaylista (head);
        system ("pause");
        break;

        case 2:
        insertmiddle( head);
        displaylista ( head);
        system ("pause");
        break;

        case 3:
        insertend( head );
        displaylista ( head);
```

```
        system ("pause");
        break;

        case 4:
        displaylista(head);
        system ("pause");
        break;

        case 5:
        terminato=1;
        printf(" \n Fine del programma...\n \n ");
        break;

        default:
        printf(" \n Funzione non associata , riprovare \n \n");
        system ("pause");
        break;

}/* end switch */

}/* end while */

}/* end main */
```

esempio di esecuzione

```
Scegliere l'operazione da effettuare:
1. Inserimento in testa
2. Inserimento nella lista
3. Inserimento in coda
4. Visualizzazione
5. Esci

Eseguire una scelta:
3

Inserire nome da accodare alla lista:
Harry

 _____________________
| Lista corrente... |
 -------------------------------
>  Alex
>  Dana
>  Peter
>  Harry
 ___________________________________

Premere un tasto per continuare...
```

FUNCTION INSERTEND

In C:

```
void insertend ( LINK head )
{
        char *elemento;
/*la posizione corrente è inizialmente la testa della lista */
        current = head;

/* Scorrimento della lista fino al raggiungimento dell'ultima posizione */
        while (current -> NEXT != NULL)
        { current = current -> NEXT; }
/*creazione del nuovo nodo e suo inserimento nella lista */
        newelem=(LINK)malloc(sizeof(LISTANOMI));
        current -> NEXT = newelem;
        newelem -> NEXT = NULL;

/* immissione del nuovo elemento da inserire nella lista */
        puts("\nInserire nome da accodare alla lista: ");
        elemento = (char*)malloc(sizeof(char)*40);
        gets(elemento);
        gets(elemento);

/*copia dell'elemento inserito nel campo info del nuovo nodo creato */
        strcpy(newelem -> nome, elemento);
        printf("\n");

}
```

FUNCTION ERASE

Documentazione esterna

- **Scopo:** eliminazione di un elemento da una linked-list.

- **Specifiche:**

  void erase (LINK head);

- **Descrizione:** grazie ad alcuni cicli if innestati e ad un ciclo while, la funzione prevede diverse opzioni: elemento trovato, elemento non trovato, e il caso in cui la lista sia vuota.

- **Raccomandazioni d'uso:** l'uso della funzione erase è strettamente correlato con quello delle funzioni di inserimento, poiché per poter eliminare elementi da una lista, e ovviamente necessario averli preventivamente inseriti.

- **Funzioni ausiliarie:**

  void inserthead (LINK head);
  void insertmiddle (LINK head);
  void insertend (LINK head);
  void displaylista (LINK head);

- **Esempio d'uso**

esempio di programma chiamante

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

/*prototipi */
        void insertmiddle (LINK head );
        void inserthead ( LINK head );
        void insertend ( LINK head );
        void erase( );
        void displaylista ( );

/*definizione della struttura che costituisce ciascun elemento della lista */
        struct data
        {
                char nome[40];
                struct data *NEXT; /* NEXT punta al prossimo elemento della lista */
        };

        typedef struct data *LINK;
        typedef struct data LISTANOMI;
```

```
  /* Puntatori:                          */
  /*    Inizio Lista (Head)                 */
  /*    Nuovo Elemento della Lista (Newelem)      */
  /*    Posizione corrente nella Lista (Current)  */
  /*    Elemento precedente della Lista (Prec)    */

/*creazione della lista vuota */
        LINK head = NULL;   /* ricordo che LINK e' un tipo puntatore */
        LINK newelem = NULL;
        LINK current = NULL;
        LINK prec = NULL;
main( )
{
/* variabili */
        int terminato = 0;
        int c;

while (!terminato)
{
system("cls");
printf ("\n");
printf ("Scegliere l'operazione da effettuare: \n \n");
printf ("1.   Inserimento in testa \n");
printf ("2.   Inserimento nella lista \n");
printf ("3.   Inserimento in coda \n");
printf ("4.   Eliminazione di un elemento \n");
printf ("5.   Visualizzazione \n");
printf ("6.   Esci \n");

printf (" \n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch (c)
{

        case 1 :
        inserthead ( head);
        displaylista ( head);
        system (" pause");
        break;

        case 2:
        insertmiddle(head);
        displaylista (head);
        system (" pause ");
        break;

        case 3:
        insertend (head);
        displaylista  ( head);
        system (" pause ");
```

```
        break;


        case 4:
        erase (head);
        displaylista  (head);
        system (" pause ");
        break;


        case 5:
        displaylista(head);
        system (" pause ");
        break;

        case 6:
        terminato=1;
        printf("\n Fine del programma...\n \n");
        break;

        default:
        printf   (" \n Funzione non associata , riprovare \n \n");
        system (" pause");
        break;
  } /*end switch */

} /*end while */

} /*end main */
```

esempio di esecuzione

```
Scegliere l'operazione da effettuare:
1.  Inserimento in testa
2.  Inserimento nella lista
3.  Inserimento in coda
4.  Eliminazione di un elemento
5.  Visualizzazione
6.  Esci
Eseguire una scelta:
5
 _____________________
| Lista corrente... |
 ---------------------------------
>  Alex
>  Dana
>  Peter
>  Harry
 ____________________________________
Premere un tasto per continuare...
Eseguire una scelta:
4
Nominativo da eliminare: Dana
L'elemento Dana é stato cancellato
 _____________________
| Lista corrente... |
 ---------------------------------
>  Alex
>  Peter
>  Harry
 ____________________________________

Premere un tasto per continuare...
```

FUNZIONE ERASE

In C:

```
void erase( head)
{
 /* variabili */
        int trovato=0; /* indica quando è stato trovato l'elemento da eliminare */
        char elemento; / puntatore all'elemento da eliminare */

/* si parte dall'inizio della lista */
        current = head;
        prec = head;
        if (head == NULL) /* caso in cui la lista è vuota */
        {
                printf(" La lista è vuota!!! Cosa elimino? \n");

        }
        else
        { /* Inserimento del nominativo da eliminare */
                printf(" \n Nominativo da eliminare: ");
                elemento = (char*)malloc(sizeof(char)*40);
                gets(elemento);
                gets(elemento);

/* Scansione della lista per ricercare l'elemento da eliminare */
                while ((current != NULL) && (trovato == 0))
                {
                        if (strcmp (current -> nome, elemento) == 0)
                        {
                                if (current == head)
                                {
                                        head = current -> NEXT;
                                        printf(" L'elemento %s è stato cancellato \n ",elemento);
                                        trovato = 1;
                                }
                                else
                                {
                                        prec -> NEXT = current -> NEXT;
                                        printf(" L'elemento %s è stato cancellato \n", elemento);
                                        trovato = 1;
                                }
                        }
        prec = current;
        current = current -> NEXT;  /* Avanza nella ricerca */
        } /* end while */

                if (trovato == 0)  /* Caso in cui l'elemento non viene trovato */
                {
                        printf ("\n Non posso cancellare qualcosa che non esiste!!! \n \n");
                }
        } /*end 1° ciclo if */
} /* end function */
```

FUNCTION DISPLAYLISTA

Documentazione esterna

- **Scopo:** visualizzare il contenuto di una linked-list.

- **Specifiche:**

  void displaylista (LINK head)

- **Descrizione:** utilizzando un ciclo di tipo while, la funzione stampa tutto il contenuto della lista.

- **Lista dei parametri:**

  head: puntatore alla cima della lista.

- **Funzioni ausiliarie:**

  void inserthead (LINK head);
  void insertmiddle (LINK head);
  void insertend (LINK head);

- **Esempio d'uso**

esempio di programma chiamante

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

/* prototipi */
    void inserthead ( LINK head );
    void insertmiddle ( LINK head );
    void insertend ( LINK head );
    void displaylista ( LINK head );

/*definizione della struttura che costituisce ciascun elemento della lista */
        struct data
        {
        char nome[40];
        struct data *NEXT;  /* NEXT punta al prossimo elemento della lista */
        };

typedef struct data LINK; /* definizione del puntatore alla struttura */
typedef struct data LISTANOMI;
```

```
 /* Puntatori:                                   */
 /*    Inizio Lista ( Head )                     */
 /*    Nuovo Elemento della Lista (Newelem)      */
 /*    Posizione corrente nella Lista (Current)  */
 /*    Elemento precedente della Lista ( Prec )   */

/*creazione della lista vuota */
 LINK head  = NULL; /* ricordo che LINK e' un tipo puntatore */
 LINK newelem  =  NULL;
 LINK current  = NULL;
 LINK  prec = NULL;

main(  )
{
/* variabili */
        int terminato = 0;
        int c;

while (!terminato)
{
system("cls");
printf ("\n");
printf ("Scegliere l'operazione da effettuare: \n \n");
printf ("1.   Inserimento in testa \n");
printf ("2.   Inserimento nella lista \n");
printf ("3.   Inserimento in coda \n");
printf ("4.   Visualizzazione \n");
printf ("5.   Esci \n");

printf ("\n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch (c)
{

        case 1 :
        inserthead (head);
        displaylista (head);
        system ("pause");
        break;

        case 2:
        insertmiddle( head);
        displaylista ( head);
        system ("pause");
        break;

        case 3:
        insertend( head );
        displaylista ( head);
        system ("pause");
```

```
        break;

        case 4:
        displaylista(head);
        system ("pause");
        break;

        case 5:
        terminato=1;
        printf(" \n Fine del programma...\n \n ");
        break;

        default:
        printf(" \n Funzione non associata , riprovare \n \n");
        system ("pause");
        break;

}/* end switch */

}/* end while */


}/* end main */
```

esempio di esecuzione

```
Scegliere l'operazione da effettuare:

1. Inserimento in testa
2. Inserimento nella lista
3. Inserimento in coda
4. Visualizzazione
5. Esci

 Eseguire una scelta:
4


 _____________________
| Lista corrente... |
 -------------------------------
>  Alex
>  Dana
>  Peter
>  Harry
 ____________________________________

Premere un tasto per continuare...
```

FUNCTION DISPLAYLISTA

In C:

```
void displaylista()
{
/* la posizione current è inizialmente la testa della lista */
        current = head;
        printf("\n \n");

printf(" _____\n");
printf("| Lista corrente... |\n ---------------------------------\n");

/* stampa di tutto il contenuto della lista */
while (current != NULL)
{
        printf(">   %s\n", current -> nome);
        current = current -> NEXT;
}
printf(" _____\n\n");

}
```

PACKAGE PER LA GESTIONE DI UN ALBERO BINARIO

Premessa

L'**ALBERO** appartiene alla categoria delle strutture dati di tipo gerarchico, in cui ciascun elemento risulta logicamente collegato anche a più di due elementi, e i collegamenti sono di tipo gerarchico (relazione padre - figlio).

La componente iniziale dell'albero viene definita *radice*, e ciascun elemento prende il nome di *nodo*.
Ogni nodo può avere dei *figli*, cioè altri nodi gerarchicamente a lui collegati, e se ciò non si verifica esso viene denominato *foglia*.
L'insieme di tutti i nodi gerarchicamente collegati ad un nodo padre prende il nome di *sottoalbero* di tale nodo.

L'**ALBERO BINARIO** è un tipo particolare di albero in cui ogni nodo può avere al massimo due figli: viene definito ordinato perché tutti i valori del sottoalbero di sinistra risultano sempre minori della radice, mentre i valori del sottoalbero di destra sono sempre maggiori della medesima.

ESEMPIO
DI ALBERO
BINARIO

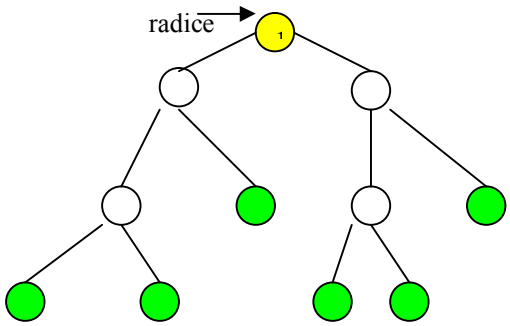


ILLUSTRAZIONE DELLA LIBRERIA DI FUNZIONI

DEFINIZIONE DELLA STRUTTURA DATI CHE RAPPRESENTA CIASCUN NODO DELL'ALBERO

(essa viene utilizzata da tutte le function della libreria)

```
/* puntatore alla struttura nodo_tree */
        typedef struct nodo_tree  *tree

        typedef struct nodo_tree
        {

                int dato;
                tree left;     /*puntatore al figlio di sinistra */
                tree right;   /*puntatore al figlio di sinistra */

        }nodo;
```

FUNCTION INIT_ALBERO

Documentazione esterna

- **Scopo:** creazione di un albero binario.

- **Specifiche:**

  tree init_albero (int el);

- **Descrizione:** la funzione crea la struttura vuota in cui inserire il valore della radice dell'albero.

- **Lista dei parametri:**

  el: intero. E' la radice dell'albero.

- **Esempio d'uso:**

esempio di programma chiamante
(vedi main allegato alla function insertnode)

Function init_albero
In C:

```
tree init_albero (int el)
{
/*puntatore al nuovo nodo che accoglierà il valore della radice */
        tree a;

/*creazione del nuovo nodo */
        a=(tree) malloc (sizeof(nodo));
        a -> dato = el;
        a -> left = NULL;
        a -> right = NULL;

        return a;

}
```

FUNCTION INSERTNODE

Documentazione esterna

- **Scopo:** inserimento di un elemento in un albero binario.

- **Specifiche:**

  int insertnode (tree albero, int k)

- **Descrizione:**

- **Lista dei parametri:**

  albero: puntatore alla radice dell'albero binario.
  k: intero. Elemento da inserire.

- **Esempio d'uso:**

esempio di programma chiamante

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

/* prototipi */
        int insertnode ( tree albero, int k );
        void stampa_in_order (tree root);
main ( )
{
/* variabili */
        tree head;
        int c;
        int el, terminato=0;
        int x=0;
printf ("\n Inserire la radice: ");
scanf ("%d", &el);

/* creazione dell'albero vuoto */
head = init_albero(el);

while(!terminato)
{
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf (" ° ° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf ("1. Inserimento di un elemento \n");
printf ("3. Stampa In - Order dell'Albero Binario \n");
printf ("8. Uscita dal programma \n");

printf (" \n Eseguire una scelta: \n");
```

```
scanf  ("%d", &c);

switch(c)
{
          case 1:
          system("cls");
          printf (" Elemento da inserire: ");
          scanf("%d", &el);
 /* Inserimento del nodo */
          x = insertnode (head, el);
          if (!x)
          printf (" Il nodo e' gia' presente nell'albero");

          system(" pause ");
          break;

          case 3:
          system("cls");

          stampa_in_order (head);
          system("pause");
          break;

          case 8:
          terminato=1;
          printf ("\n Fine del programma \n");
          break;

          default:
          printf(" Funzione non associata: riprovare! \n \n");
          break;

   }/*end switch */
  } /*end while */
} /*end main */
```

esempio di esecuzione

```
Inserire la radice: 5
°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
°°° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO°°°
°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
1. Inserimento di un elemento
3. Stampa In-Order dell'Albero Binario
8. Uscita dal programma

Eseguire una scelta: 1
Inserire l'elemento: 4

Output con il criterio inorder:
4   5

Eseguire una scelta: 1
Inserire l'elemento: 7

Output con il criterio inorder:
4 5 7
```

FUNCTION INSERTNODE

In C:

```
int insertnode (tree albero, int k)
{
/* variabili */
        tree prec, a;

/*se la radice è stata inserita e il nodo non è già presente nell'albero… */
        while ((albero != NULL)&&(albero->dato != k))
        {
                prec = albero;
/*…stabiliamo in quale sottoalbero spostarsi  */
                        if (k > albero ->dato)
                                albero = albero ->right;
                        else
                                albero = albero ->left;
        }


        if (albero = NULL)
        return 0;

        else
        {
/* creazione del nuovo nodo inserendovi il dato */
                a=(tree)malloc(sizeof(nodo));
                a->dato=k;
                a->left=NULL;
                a->right=NULL;
/* stabilisce dove inserire il nuovo nodo creato */
                        if (k > prec ->dato)
                                prec->right = a;
                        else
                                prec->left = a;

        return 1;
        }

}
```

FUNCTION RICERCA

Documentazione esterna

- **Scopo:** ricercare un elemento in un albero binario.

- **Specifiche:**

  int ricerca (tree root, int key)

- **Descrizione:** utilizzando un approccio di tipo ricorsivo la function esegue una serie di confronti per stabilire se la chiave di ricerca appartiene all'albero.

- **Lista dei parametri:**

  root: puntatore alla radice dell'albero.
  key: intero. Elemento da ricercare.

- **Funzioni ausiliarie:**

  int insertnode (tree albero, int k);
  void stampa_in_order (tree root);

- **Esempio d'uso:**

esempio di programma chiamante

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

/* prototipi */
        int ricerca (tree root, int key);
        int insertnode (tree albero, int k);
        void stampa_in_order (tree root );

main ( )
{
/* variabili */
        tree head;
        int c;
        int el, key, terminato=0;
        int x=0;

        printf ("\nInserire la radice: ");
        scanf ("%d",&el);

/* creazione dell'albero binario */
        head = init_albero(el);

while(!terminato)
```

```
{
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf (" °° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf ("1. Inserimento di un elemento \n");
printf ("2. Ricerca di un elemento \n");
printf ("8. Uscita dal programma \n");

        printf (" \n Eseguire una scelta: \n ");
        scanf  ("%d", &c);

switch(c)
{
        case 1:
        system(" cls ");
        printf("Elemento da inserire: ");
        scanf("%d", &el);
        x = insertnode (head, el);
        if (!x)
        printf ("Il nodo e' gia' presente nell'albero");
        system("pause");
        break;

        case 2:
        system("cls");
        printf ("Inserire l'elemento da ricercare: ");
        scanf ("%d", &key);

        if ( ricerca (head, key))
                printf ("Elemento presente nell'albero \n");
        else
                printf ("Elemento non presente nell'albero \n");

        system("pause");
        break;

        case 8:
        terminato=1;
        printf (" \n Fine del programma \n");
        break;


        default:
        printf(" Funzione non associata: riprovare! \n \n");
        break;

   } /*end switch */
  } /*end while */
} /*end main */
```

esempio di esecuzione

```
Inserire la radice: 5
°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
°°° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO°°°
°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
1.Inserimento di un elemento
2. Ricerca di un elemento
3. Stampa In-Order dell'Albero Binario
8. Uscita dal programma

Eseguire una scelta: 1
Inserire l'elemento: 4

Output con il criterio inorder:
4   5

Eseguire una scelta: 1
Inserire l'elemento: 7

Output con il criterio inorder:
4 5 7

Eseguire una scelta: 2
Inserire l'elemento da ricercare: 7
Elemento trovato

Eseguire una scelta: 2
Inserire l'elemento da ricercare: 6
Elemento non trovato
```

Function ricerca

In c:

```
int ricerca (tree root, int key)
{

/*caso in cui non vi sono elementi */
        if (root == NULL)
                return 0;

/* caso in cui la chiave di ricerca coincide con la radice */
        else if (root->dato==key)
                return 1;

/* se la radice è minore della chiave cercata la ricerca si sposta a destra */
        else if (root ->dato < key)
                return ricerca (root ->right, key);

/* se la radice è minore della chiave cercata la ricerca si sposta a sinistra */
                else
                return ricerca (root ->left, key);

}
```

FUNCTION STAMPA IN ORDER

Documentazione esterna

- **Scopo:** visita degli elementi di un albero binario secondo il criterio dell'ordine naturale.

- **Specifiche:**

  void stampa_in_order (tree root);

- **Descrizione:** utilizzando un approccio di tipo ricorsivo, la funzione genera la sequenza risultante della visita seguendo il criterio dell'ordine naturale: con tale metodo, il valore della radice viene stampato tra i valori dei due sottoalberi.

- **Lista dei parametri:**

  root: puntatore alla radice.

- **Funzioni ausiliarie:**

  int insertnode (tree albero, int k);

- **Esempio d'uso:**

esempio di programma chiamante

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

/* prototipi */
        int insertnode ( tree albero, int k );
        void stampa_in_order (tree root );

main ( )
{
/* variabili */
        tree head;
        int c;
        int el,  key,  terminato=0;
        int x=0;

printf ("\n Inserire la radice: ");
scanf ("%d", &el);

/* creazione dell'albero vuoto */
head = init_albero(el);

while(!terminato)
{
```

```
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf (" ° ° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf ("1. Inserimento di un elemento \n");
printf ("3. Stampa In - Order dell'Albero Binario \n");
printf ("8. Uscita dal programma \n");

printf (" \n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch(c)
{
        case 1:
        system("cls");
        printf (" Elemento da inserire: ");
        scanf("%d", &el);
 /* Inserimento del nodo */
        x = insertnode (head, el);
        if (!x)
        printf (" Il nodo e' gia' presente nell'albero");

        system(" pause ");
        break;


        case 3:
        system("cls");

        stampa_in_order (head);
        system("pause");
        break;

        case 8:
        terminato=1;
        printf ("\n Fine del programma \n");
        break;


        default:
        printf(" Funzione non associata: riprovare! \n \n");
        break;

   }/*end switch */

  } /*end while */

} /*end main */
```

esempio di esecuzione
Inserire la radice: 5
°°
°°° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO°°°
°°

1.Inserimento di un elemento
2. Ricerca di un elemento
3. Stampa In-Order dell'Albero Binario
8. Uscita dal programma

Eseguire una scelta: 1
Inserire l'elemento: 4

Output con il criterio inorder:
4   5

Eseguire una scelta: 1
Inserire l'elemento: 7

Output con il criterio inorder:
4 5 7

Funzione stampa_in_order

In C:

```
void stampa_in_order (tree root)
{

        if (root != NULL)
        {
/* stampa il valore della radice tra quello dei due sottoalberi */
                stampa_in_order ( root ->left);
                printf ("%d \n", root ->dato);
                stampa_in_order (root ->right);
        }

}
```

FUNCTION STAMPA PRE ORDER

Documentazione esterna

- **Scopo:** visita degli elementi di un albero binario secondo il criterio dell'ordine anticipato.

- **Specifiche:**

  void stampa_pre_order (tree root);

- **Descrizione:** utilizzando un approccio di tipo ricorsivo, la funzione genera la sequenza risultante della visita seguendo il criterio dell'ordine anticipato: con tale metodo, il valore della radice viene stampato prima di entrambi i valori dei due sottoalberi.

- **Lista dei parametri:**

  root: puntatore alla radice.

- **Funzioni ausiliarie:**

  int insertnode (tree albero, int k);

- **Esempio d'uso:**

esempio di programma chiamante

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

/* prototipi */
        int insertnode ( tree albero, int k );
        void stampa_in_order (tree root );

main ( )
{
/* variabili */
        tree head;
        int c;
        int el,  key,  terminato=0;
        int x=0;

printf ("\n Inserire la radice: ");
scanf ("%d", &el);

/* creazione dell'albero vuoto */
head = init_albero(el);

while(!terminato)
{
```

```
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf (" ° ° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf ("1. Inserimento di un elemento \n");
printf ("4. Stampa Pre - Order dell'Albero Binario \n");
printf ("8. Uscita dal programma \n");

printf (" \n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch(c)
{
        case 1:
        system("cls");
        printf (" Elemento da inserire: ");
        scanf("%d", &el);
 /* Inserimento del nodo */
        x = insertnode (head, el);
        if (!x)
        printf (" Il nodo e' gia' presente nell'albero");

        system(" pause ");
        break;

        case 4:
        system("cls");

        stampa_pre_order (head);
        system("pause");
        break;

        case 8:
        terminato=1;
        printf ("\n Fine del programma \n");
        break;


        default:
        printf(" Funzione non associata: riprovare! \n \n");
        break;

   }/*end switch */

  } /*end while */

} /*end main */
```

esempio di esecuzione
Inserire la radice: 5
°°
°°° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO°°°
°°

1.Inserimento di un elemento
4. Stampa Pre-Order dell'Albero Binario
8. Uscita dal programma

Eseguire una scelta: 1
Inserire l'elemento: 4

Output con il criterio pre order:
5 4

Eseguire una scelta: 1
Inserire l'elemento: 7

Output con il criterio pre order:
5 4 7

Funzione stampa_pre_order

In C:

```
void stampa_pre_order (tree root)
{

        if (root != NULL)
        {
/* stampa il valore della radice tra quello dei due sottoalberi */
                printf ("%d \n", root -> dato);
                stampa_in_order (root ->left );
                stampa_in_order (root ->right);
        }

}
```

FUNCTION POST ORDER

Documentazione esterna

- **Scopo:** visita degli elementi di un albero binario secondo il criterio dell'ordine differito.

- **Specifiche:**

  void stampa_post_order (tree root);

- **Descrizione:** utilizzando un approccio di tipo ricorsivo, la funzione genera la sequenza risultante della visita seguendo il criterio dell'ordine differito: con tale metodo, il valore della radice viene stampato dopo entrambi i valori dei due sottoalberi.

- **Lista dei parametri:**

  root: puntatore alla radice.

- **Funzioni ausiliarie:**

  int insertnode (tree albero, int k);

- **Esempio d'uso:**

esempio di programma chiamante

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

/* prototipi */
        int insertnode ( tree albero, int k );
        void stampa_post_order (tree root );

main ( )
{
/* variabili */
        tree head;
        int c;
        int el,  key,  terminato=0;
        int x=0;

printf ("\n Inserire la radice: ");
scanf ("%d", &el);

/* creazione dell'albero vuoto */
head = init_albero(el);

while(!terminato)
```

```
{
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf (" ° ° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf ("1. Inserimento di un elemento \n");
printf ("5. Stampa Post - Order dell'Albero Binario \n");
printf ("8. Uscita dal programma \n");

printf (" \n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch(c)
{
        case 1:
        system("cls");
        printf (" Elemento da inserire: ");
        scanf("%d", &el);
 /* Inserimento del nodo */
        x = insertnode (head, el);
        if (!x)
        printf (" Il nodo e' gia' presente nell'albero");

        system(" pause ");
        break;

        case 5:
        system("cls");

        stampa_post_order (head);
        system("pause");
        break;

        case 8:
        terminato=1;
        printf ("\n Fine del programma \n");
        break;


        default:
        printf(" Funzione non associata: riprovare! \n \n");
        break;

   }/*end switch */

  } /*end while */

} /*end main */
```

esempio di esecuzione
Inserire la radice: 5
°°°
°°° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO°°°
°°°

1.Inserimento di un elemento
5. Stampa Post-Order dell'Albero Binario
8. Uscita dal programma

Eseguire una scelta: 1
Inserire l'elemento: 4

Output con il criterio post order:
4   5

Eseguire una scelta: 1
Inserire l'elemento: 7

Output con il criterio post order:
4 7 5

Funzione stampa_post_order
In C:

```
void stampa_post_order (tree root)
{

        if (root != NULL)
        {
/* stampa il valore della radice tra quello dei due sottoalberi */
                stampa_in_order (root ->left );
                stampa_in_order (root ->right);
                printf ("%d \n", root -> dato);
        }

}
```

FUNCTION CONTA NODI

Documentazione esterna

- **Scopo:** conteggiare il numero dei nodi presenti in un albero.

- **Specifiche:**

  int contanodi (tree root);

- **Descrizione:** utilizzando un approccio di tipo ricorsivo, la funzione conta il totale degli elementi presenti nell'albero.

- **Lista dei parametri:**

  root: puntatore alla radice.

- **Funzioni ausiliarie:**

  int insertnode (tree albero, int k);
  tree init_albero (int el);

- **Esempio d'uso:**

esempio di programma chiamante

```
        #include < stdio.h >
        #include < stdlib.h >
/* prototipi */
        int insertnode ( tree albero, int k );
        int contanodi (tree root );
        tree init_albero (int el);

main ( )
{
/* variabili */
        tree head;
        int c;
        int el,  key,  terminato=0;
        int x=0;

printf ("\n Inserire la radice: ");
scanf ("%d", &el);

/* creazione dell'albero vuoto */
head = init_albero(el);

while(!terminato)
{
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
```

```
printf (" ° ° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf ("1. Inserimento di un elemento \n");
printf ("6. Conteggio del numero di nodi \n");
printf ("8. Uscita dal programma \n");

printf (" \n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch(c)
{
        case 1:
        system("cls");
        printf (" Elemento da inserire: ");
        scanf("%d", &el);
 /* Inserimento del nodo */
        x = insertnode (head, el);
        if (!x)
        printf (" Il nodo e' gia' presente nell'albero");

        system(" pause ");
        break;

        case 6:
        system("cls");
        key = contanodi (head
        printf (" L'albero ha %d nodi \n", key);
        system("pause");
        break;

        case 8:
        terminato=1;
        printf ("\n Fine del programma \n");
        break;


        default:
        printf(" Funzione non associata: riprovare! \n \n");
        break;

   }/*end switch */

  } /*end while */

} /*end main */
```

esempio di esecuzione

Inserire la radice: 5
°°
°°° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO°°°
°°

1.Inserimento di un elemento
6. Conteggio del numero dei nodi
8. Uscita dal programma

Eseguire una scelta: 1
Inserire l'elemento: 4

Eseguire una scelta: 6
L'albero ha 2 nodi.

Eseguire una scelta: 1
Inserire l'elemento: 7

Eseguire una scelta: 6
L'albero ha 3 nodi.

Function contanodi
In C:

```
int contanodi (tree root)
{

        int ns, nd;

/*caso in cui non ci sono elementi */
        if (root == NULL)
        return 0;

        else
/*calcola il numero totale dei nodi */
        ns=contanodi (root->left);
        nd=contanodi (root->right);

/*restituzione del valore al programma chiamante */
return (ns+nd+1);

}
```

FUNCTION PROFONDITA'

Documentazione esterna

- **Scopo:** calcolare la profondità (numero di livelli) di un albero binario.

- **Specifiche:**

  int profondità (tree root)

- **Descrizione:** utilizzando un approccio di tipo ricorsivo la function calcola il numero massimo di livelli presenti nell'albero, e poi restituisce tale valore al programma chiamante.

- **Lista dei parametri:**

  root: puntatore alla radice dell'albero.

- **Funzioni ausiliarie:**

  int insertnode (tree albero, int k);
  tree init_albero (int el);

- **Esempio d'uso:**

esempio di programma chiamante

```
        #include < stdio.h >
        #include < stdlib.h >
/* prototipi */
        int insertnode ( tree albero, int k );
        int profondità (tree root );
        tree init_albero (int el);

main ( )
{
/* variabili */
        tree head;
        int c;
        int el,  key,  terminato=0;
        int x=0;

printf ("\n Inserire la radice: ");
scanf ("%d", &el);

/* creazione dell'albero vuoto */
head = init_albero(el);

while(!terminato)
{
```

```
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf (" ° ° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO °°\n");
printf (" °°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf ("\n");
printf ("1. Inserimento di un elemento \n");
printf ("7. Profondità dell'albero binario \n");
printf ("8. Uscita dal programma \n");

printf (" \n Eseguire una scelta: \n");
scanf  ("%d", &c);

switch(c)
{
        case 1:
        system("cls");
        printf (" Elemento da inserire: ");
        scanf("%d", &el);
 /* Inserimento del nodo */
        x = insertnode (head, el);
        if (!x)
        printf (" Il nodo e' gia' presente nell'albero");

        system(" pause ");
        break;

        case 7:
        system("cls");
        key = profondità (head);
        printf (" L'albero ha profondità %d \n", key);
        system("pause");
        break;

        case 8:
        terminato=1;
        printf ("\n Fine del programma \n");
        break;


        default:
        printf(" Funzione non associata: riprovare! \n \n");
        break;

   }/*end switch */

  } /*end while */

} /*end main */
```

esempio di esecuzione

Inserire la radice: 5
°°
°°° PROGRAMMA PER LA GESTIONE DI UN ALBERO BINARIO°°°
°°

1.Inserimento di un elemento
6. Conteggio del numero dei nodi
8. Uscita dal programma

Eseguire una scelta: 1
Inserire l'elemento: 4

Eseguire una scelta: 7
L'albero ha profondità 2

Eseguire una scelta: 1
Inserire l'elemento: 1

Eseguire una scelta: 7
L'albero ha profondità 3

FUNCTION PROFONDITA'

In C:

```
int profondità (tree root)
{
/* variabili locali */
        int ps, pd, max;

/* se non ci sono elementi, allora la profondità è 0 */
        if (root == NULL)
        return 0;
/* se ci sono elementi, allora si calcola ricorsivamente la profondità di ciascun   */
/* sottoalbero sinistro e destro */
        ps = profondità ( root ->left);
        pd = profondità ( root ->right);

/* restituzione del valore al programma chiamante */
        if ( ps > pd )
                max = ps;
        else
                max = pd;


        return (max+1);
}
```

PACKAGE PER LA GESTIONE DI UN DATABASE STUDENTI

ILLUSTRAZIONE DELLA LIBRERIA DI FUNZIONI

FUNCTION INS_HEAD

Documentazione esterna

- **Scopo:** inserire un elemento in testa al database.

- **Specifiche:**

  void ins_head (char *mat);

- **Descrizione:** dopo aver allocato dinamicamente lo spazio di memoria necessario per creare il nuovo elemento da inserire nella lista, la funzione crea tale elemento e lo inserisce in cima alla lista.

- **Lista dei parametri:**

  ***mat:** puntatore alla matricola da inserire.

- **Esempio d'uso:**

(vedi esempio per la funzione di inserimento)

Funzione ins_head

In C:

```
void ins_head(char *mat)
{
        char *temp;
/*allocazione dell'array temporaneo che serve per inserire gli elementi */
        temp = (char*)malloc(sizeof(char));
        gets (temp);

/*creazione e inserimento in testa del nuovo nodo */
        nuovo = (LINK)malloc(sizeof(DATI));
        nuovo -> NEXT = head;
        head = nuovo;

/* copia dei valori inseriti nel nuovo nodo creato */
        strcpy(nuovo -> matricola, mat);
        puts(" Inserire il cognome e il nome (rispettivamente): ");
        gets (temp);
        strcpy(nuovo -> nome, temp);
        puts(" Data di nascita (GG/MM/AAAA): ");
        gets (temp);
        strcpy(nuovo -> data nascita, temp);
        puts("Indirizzo: ");
        gets (temp);
        strcpy(nuovo -> indirizzo, temp);
        puts(" Recapito telefonico: ");
        gets(temp);
        strcpy(nuovo -> tel, temp);

}
```

FUNCTION INSERISCI

Documentazione esterna

- **Scopo:** inserire il nominativo di uno studente nel database.

- **Specifiche:**

  void inserisci( );

- **Descrizione:** dopo aver effettuato una ricerca per stabilire la posizione adatta, la funzione inserisce il nominativo nella lista.

- **Funzioni ausiliarie:**

  LINK doveinserire (char *);
  void ins_head(char *);

- **Esempio d'uso:**

esempio di programma chiamante

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

/*struttura che rappresenta ciascun elemento della lista */
struct studente
{
        char matricola[10];
        char nome[60];
        char datanascita[15];
        char indirizzo[80];
        char tel[12];
        struct studente *NEXT; /* punta al prossimo elemento della lista */
};

/* Definizione della struttura come tipo puntatore */
/* Per muoversi all'interno della lista */
typedef struct studente *LINK;

/* Definizione della struttura come tipo per dichiarare  */
/* una generica struttura. Utile ad esempio per allocare */
/* memoria quando si crea un nuovo elemento della lista  */
typedef struct studente DATI;

/* Puntatori:                          */
/*    Inizio Lista (Head)              */
/*    Nuovo Elemento della Lista (New) */
/*    Posizione corrente nella Lista (Current) */
/*    Elemento precedente della Lista (Prec)   */
```

```
/* creazione della lista vuota  */

        LINK head = NULL;
        LINK nuovo = NULL;
        LINK current = NULL;
        LINK prec = NULL;

/* Prototipi delle funzioni */

        void inserisci();
        LINK doveinserire(char *);
        void ins_head(char *);

main( )
{
/*variabili */
        int terminato=0;
        int c;


while(!terminato)
{
        system("cls");
        printf("\n ********************************** ***\n");
        printf("\n***** DATABASE STUDENTI *********\n");
        printf("\n ************************************\n");
        printf("\n Scegliere l'operazione da effettuare: \n");
        printf("1. Inserimento di uno studente. \n");
        printf("5. Uscita dal programma. \n");

        printf(" \n Eseguire una scelta: \n");
        scanf("%d", &c);

switch (c)
{
        case 1:
        inserisci( );
        system(" pause ");
        break;


        case 5:
        terminato=1;
        printf(" \n Fine del programma...\n \n");
        break;

        default:
        printf(" \n  Scelta non valida, riprovare \n \n");
        break;

 } /*end switch */
```

```
} /* end while */

} /*end main */
```

esempio di esecuzione

```
*******************************
*****DATABASE STUDENTI******
*******************************

Scegliere l'operazione da effettuare:
1. Inserimento di uno studente.
5. Uscita dal programma.

Eseguire una scelta:
1

Matricola da inserire: 408466
Inserire il cognome e il nome (rispettivamente
Bellino Virginia
Data di nascita (GG/MM/AAAA):
01/08/1977
Indirizzo:
Via Camaldoli 27
Recapito telefonico:
3403185376
Premere un tasto per continuare...
```

Function inserisci

In C:

```
void inserisci( )
{
        char *temp, *mat;
/* allocazione degli array temporanei che permettono di inserire gli elementi */
        temp = (char*)malloc(sizeof(char));
        mat = (char*)malloc(sizeof(char));
        temp = NULL;

        printf(" \n Matricola da inserire: ");
        gets(mat);

        if (head == NULL)  /* Nel caso la lista sia vuota, l'inserimento */
                ins_head(mat);   /* dev'essere effettuato in testa */

        else
        {
                prec = doveinserire(mat); /*cerca la posizione dove inserire */
                        if (prec == head)
                                ins_head (mat);
                        else
                        { /*crea il nuovo nodo e vi inserisce i valori */
                        nuovo = (LINK)malloc(sizeof(DATI));
                        nuovo -> NEXT = prec -> NEXT;
                        prec -> NEXT = nuovo;
                        strcpy(nuovo -> matricola, mat);
                        printf(" Inserire il cognome e il nome (rispettivamente): ");
                        temp = NULL;
                        gets (temp);
                        strcpy(nuovo -> nome,temp);
                        puts(" Data di nascita (GG/MM/AAAA): ");
                        gets (temp);
                        strcpy(nuovo -> datanascita, temp);
                        puts(" Indirizzo: ");
                        gets (temp);
                        strcpy(nuovo -> indirizzo, temp);
                        puts(" Recapito telefonico: ");
                        gets(temp);
                        strcpy(nuovo -> tel, temp);
                        }
        }

}
```

FUNCTION DOVEINSERIRE

Documentazione esterna

- **Scopo:** trovare la posizione in cui inserire lo studente nella lista, in modo da mantenere l'ordinamento della lista stessa.

- **Specifiche:**

  LINK doveinserire (char *mat);

- **Descrizione:** effettuando una serie di confronti tra la matricola da inserire e quelle già presenti nel database, la funzione cerca la posizione dove inserire, restituendo poi al programma chiamante un puntatore ad essa.

- **Lista dei parametri:**

  ***mat:** puntatore alla matricola da inserire.

- **Esempio d'uso:**

(vedi esempio per la funzione di inserimento)

Function doveinserire

In C:

```
LINK doveinserire(char *mat)
{
        LINK puntatore;
        char max = mat; /* variabile usata per il confronto */
        current = head;
        prec = head;

        while (current != NULL)
        {
                if (strcmp(current -> matricola, max) > 0)
                {
                strcpy (max, current -> matricola);  /* Salviamo la matricola */
                printf("Matricola: %s", max);
                puntatore = prec; /* valore da restituire al programma chiamante una volta …*/
                }                       /* …usciti dal ciclo */

        prec = current;
        current = current -> NEXT; /* Avanziamo nella ricerca */
        }
 return puntatore;
}
```

FUNCTION CANCELLA

Documentazione esterna

- **Scopo:** eliminare il nominativo di uno studente dal database.

- **Specifiche:**

  void cancella ( );

- **Descrizione:** dopo un'accurata ricerca, se questa ha avuto esito positivo, viene eliminato l'elemento in questione. Se invece la lista è vuota viene stampato un apposito messaggio.

- **Funzioni ausiliarie:**

  LINK doveinserire (char *);
  void ins_head(char *);
  void inserisci ( );

- **Esempio d'uso:**

esempio di programma chiamante

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

/*struttura che rappresenta ciascun elemento della lista */
struct studente
{
        char matricola[10];
        char nome[60];
        char datanascita[15];
        char indirizzo[80];
        char tel[12];
        struct studente *NEXT; /* punta al prossimo elemento della lista */
};

/* Definizione della struttura come tipo puntatore */
/* Per muoversi all'interno della lista */
typedef struct studente *LINK;

/* Definizione della struttura come tipo per dichiarare  */
/* una generica struttura. Utile ad esempio per allocare */
/* memoria quando si crea un nuovo elemento della lista  */
typedef struct studente DATI;

/* Puntatori:                    */
/*    Inizio Lista (Head)        */
```

```
/*    Nuovo Elemento della Lista (New)        */
/*    Posizione corrente nella Lista (Current) */
/*    Elemento precedente della Lista (Prec)   */
/* creazione della lista vuota  */

        LINK head = NULL;
        LINK nuovo = NULL;
        LINK current = NULL;
        LINK prec = NULL;

/* Prototipi delle funzioni */
        void inserisci( );
        LINK doveinserire(char *);
        void ins_head(char *);
        void cancella ( );

main( )
{
/*variabili */
        int terminato=0;
        int c;


while(!terminato)
{
        system("cls");
        printf("\n ********************************** ***\n");
        printf("\n***** DATABASE STUDENTI *********\n");
        printf("\n ************************************\n");
        printf("\n Scegliere l'operazione da effettuare: \n");
        printf("1. Inserimento di uno studente. \n");
        printf("2. Eliminazione di uno studente. \n ");
        printf("5. Uscita dal programma. \n");

        printf(" \n Eseguire una scelta: \n");
        scanf("%d", &c);

switch (c)
{
        case 1:
        inserisci( );
        system(" pause ");
        break;

        case 2:
        cancella( );
        system(" pause ");
        break;

        case 5:
        terminato=1;
```

```
        printf(" \n Fine del programma...\n \n");
        break;

        default:
        printf(" \n  Scelta non valida, riprovare \n \n");
        break;

 } /*end switch */

} /* end while */

} /*end main */
```

esempio di esecuzione
Avendo inserito in input gli studenti aventi le seguenti matricole:
408466 - 50887 - 408489 - 50910
si ottiene in output..

Inserire la matricola dello studente da cancellare: 408466
Lo studente - 408466 - è stato cancellato.

Inserire la matricola dello studente da cancellare: 408551
Non posso cancellare qualcosa che non esiste!!!

Function cancella

In C:

```
void cancella( )
{
        int trovato=0;
        char *mat;

        current = head;
        prec = head;

        if (head == NULL) /* Caso in cui la lista sia vuota */
        {
                printf(" \n \n La lista è vuota!!! Cosa elimino? \n");

        }

        else
        {
                puts(" \n \n Inserire la matricola dello studente da cancellare: ");
                scanf("%s",&mat);

                while ((current != NULL) && (trovato == 0))
                {
                if (strcmp(current -> matricola, mat) == 0)
                {
  /* Se la matricola da eliminare si trova */
  /* in testa, assegno direttamente un  */
  /* valore al puntatore "head" */
                        if (current == head)
                        {
                                head = current -> NEXT;
                                printf(" Lo studente - %s - è stato cancellato\n",mat);
                                trovato = 1;
                        }
                        else
                        {
                        prec -> NEXT = current -> NEXT;
                        printf(" \n Lo studente - %s - è stato cancellato \n", mat);
                        trovato = 1;
                        }
                }/* end 2° ciclo if */
                prec = current;
                current = current -> NEXT; /* Avanza nella ricerca */
                } /*end while */

                if (trovato == 0)
                {
                        printf (" \n Non posso cancellare qualcosa che non esiste!!!\n\n");
                }
        } /* end 1° ciclo if */
} /* end function */
```

FUNCTION RICERCA

Documentazione esterna

- **Scopo:** ricercare un nominativo nel database.

- **Specifiche:**

  void ricerca (int tipo);

- **Descrizione:** dopo aver ricevuto in input il tipo di ricerca da effettuare, la function offre due opzioni : ricerca per matricola e ricerca per nome.

- **Lista dei parametri:**

  tipo: intero. Tipologia della ricerca da effettuare.

- **Funzioni ausiliarie:**

  LINK doveinserire (char *);
  void ins_head(char *);
  void inserisci ( );

- **Esempio d'uso:**

esempio di programma chiamante

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

/*struttura che rappresenta ciascun elemento della lista */
struct studente
{
        char matricola[10];
        char nome[60];
        char datanascita[15];
        char indirizzo[80];
        char tel[12];
        struct studente *NEXT; /* punta al prossimo elemento della lista */
};

/* Definizione della struttura come tipo puntatore */
/* Per muoversi all'interno della lista */
typedef struct studente *LINK;

/* Definizione della struttura come tipo per dichiarare  */
/* una generica struttura. Utile ad esempio per allocare */
/* memoria quando si crea un nuovo elemento della lista  */
```

```
typedef struct studente DATI;

/* Puntatori:                          */
/*    Inizio Lista (Head)              */
/*    Nuovo Elemento della Lista (New) */
/*    Posizione corrente nella Lista (Current) */
/*    Elemento precedente della Lista (Prec)   */
/* creazione della lista vuota  */

        LINK head = NULL;
        LINK nuovo = NULL;
        LINK current = NULL;
        LINK prec = NULL;

/* Prototipi delle funzioni */

        void inserisci();
        LINK doveinserire(char *);
        void ins_head(char *);
        void ricerca (int tipo );
main( )
{
/*variabili */
        int terminato=0;
        int c, tipo_ricerca;

while(!terminato)
{
        system("cls");
        printf("\n ********************************* ***\n");
        printf("\n***** DATABASE STUDENTI *********\n");
        printf("\n ************************************\n");
        printf("\n Scegliere l'operazione da effettuare: \n");
        printf("1. Inserimento di uno studente. \n");
        printf("3. Ricerca di uno studente. \n");
        printf("5. Uscita dal programma. \n");

        printf(" \n Eseguire una scelta: \n");
        scanf("%d", &c);

switch (c)
{
        case 1:
        inserisci( );
        system(" pause ");
        break;

        case 3:
        printf(" \n \n Ricerca per:\n1. Matricola \n2. Nome \n \n Scegli [1/2]: ");
        scanf("%d", &tipo_ricerca);
```

```
        if (tipo_ricerca = = 1)
        ricerca (1);
        else
        ricerca(2);
        system(" pause ");
        break;

        case 5:
        terminato=1;
        printf(" \n Fine del programma...\n \n");
        break;

        default:
        printf(" \n  Scelta non valida, riprovare \n \n");
        break;

 } /*end switch */

} /* end while */

} /*end main */
```

esempio di esecuzione

Input: …… database vuoto

Output:
Database vuoto, impossibile effettuare la ricerca.
****
Eseguire una scelta: 3

Ricerca per:
1. Matricola
2. Nome

Scegli [1/2]: 1
Inserire la matricola dello studente da ricercare: ……..

(se la ricerca dà esito positivo ……..segue l'elenco dei dati dello studente trovato).
****
Scegli [1/2]: 2
Inserire il cognome e il nome dello studente: ……..

(se la ricerca dà esito positivo ……..segue l'elenco dei dati dello studente trovato).

FUNCTION DATABASE_OUTPUT

Documentazione esterna

- **Scopo:** visualizzare il contenuto dell'intero database.

- **Specifiche:**

  void database_output ( );

- **Descrizione:** utilizzando un ciclo di tipo while, la funzione stampa tutti gli elementi del database.

- **Funzioni ausiliarie:**

  LINK doveinserire (char *);
  void ins_head(char *);
  void inserisci ( );
  void cancella ( );

- **Esempio d'uso:**

esempio di programma chiamante

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

/*struttura che rappresenta ciascun elemento della lista */
struct studente
{
        char matricola[10];
        char nome[60];
        char datanascita[15];
        char indirizzo[80];
        char tel[12];
        struct studente *NEXT; /* punta al prossimo elemento della lista */
};

/* Definizione della struttura come tipo puntatore */
/* Per muoversi all'interno della lista */
typedef struct studente *LINK;

/* Definizione della struttura come tipo per dichiarare  */
/* una generica struttura. Utile ad esempio per allocare */
/* memoria quando si crea un nuovo elemento della lista  */
typedef struct studente DATI;

/* Puntatori:                    */
/*    Inizio Lista (Head)        */
```

```
/*    Nuovo Elemento della Lista (New)        */
/*    Posizione corrente nella Lista (Current) */
/*    Elemento precedente della Lista (Prec)   */
/* creazione della lista vuota  */

        LINK head = NULL;
        LINK nuovo = NULL;
        LINK current = NULL;
        LINK prec = NULL;

/* Prototipi delle funzioni */
        void inserisci( );
        LINK doveinserire(char *);
        void ins_head(char *);
        void cancella ( );
        void database_output ( );

main( )
{
/*variabili */
        int terminato=0;
        int c;


while(!terminato)
{
        system("cls");
        printf("\n ********************************* ***\n");
        printf("\n***** DATABASE STUDENTI *********\n");
        printf("\n ***********************************\n");
        printf("\n Scegliere l'operazione da effettuare: \n");
        printf("1. Inserimento di uno studente. \n");
        printf("2. Eliminazione di uno studente. \n ");
        printf("4. Visualizzazione dell'intero database. \n ");
        printf("5. Uscita dal programma. \n");

        printf(" \n Eseguire una scelta: \n");
        scanf("%d", &c);

switch (c)
{
        case 1:
        inserisci( );
        system(" pause ");
        break;

        case 2:
        cancella( );
        system(" pause ");
        break;
```

```
        case 4:
        database_output ( );
        system(" pause ");
        break;

        case 5:
        terminato=1;
        printf(" \n Fine del programma...\n \n");
        break;

        default:
        printf(" \n  Scelta non valida, riprovare \n \n");
        break;

 } /*end switch */

} /* end while */

} /*end main */
```

esempio di esecuzione

(si ottiene la lista di tutti i nominativi inseriti nel database.)

Function database output

In C:

```
void database_output( )
{
        current = head;
        printf("\n \n \n \n \n \n \n \n \n \n \n \n \n \n \n");
        printf(" _____\n");
        printf(" | Database corrente... |\n ----------\n");

while (current != NULL) /* Fino a quando la lista non finisce */
 {
        printf("> Matricola: %s   Nome: %s \n", current -> matricola, current -> nome);
        printf("> Nato il: %s \n", current -> data nascita);
        printf("> Indirizzo: %s \n", current -> indirizzo);
        printf("> Telefono: %s \n", current -> tel );
        current = current -> NEXT;
 }
        printf(" _____\n \n");

}
```