GIOVANNI DI CECCA

Programmi in Linguaggio C
per l'esame di

Programmazione Modulo B

Università degli Studi di Napoli
Federico II

http://www.dicecca.net

Indice

- Algoritmi di Ordinamento Ricorsivi
  - Bubble Sort
  - Insertion Sort
  - Selection Sort
  - Merge Sort

- Package
  - Programma per la gestione di una Pila
  - Programma per la gestione di una Lista
  - Programma per la gestione di un DataBase
  - Programma per la gestione di un Albero Binario

Ordinamenti Ricorsivi

Premessa agli algoritmi di ordinamento

Basandomi sulla modularità del **Linguaggio C**, per ciò che concerne il programma MAIN, ne ho creato un modello unico valido per tutti e quattro gli algoritmi di ordinamento di seguito riportati.

```c
/* Questo codice permette l'inserimento dei dati ed il passaggio alle */
/* funzioni di Bubble Sort, Selection Sort e Insertion Sort, Merge    */
/* versioni Ricorsive a mezzo della funzione sort(n, 0, a);           */
/*                                                                    */
/*                       Programma  sviluppato  da                    */
/*                                                                    */
/*                           Giovanni DI CECCA                        */
/*                                                                    */
/*                         http://www.dicecca.net                     */


/* Parte preprocessore */
#include <stdio.h>
#include <stdlib.h>

#include <conio.h>  // Variazione sul tema in C++

/* Carica la libreria esterna che contiene la funzione da utlizzare */
/* per l'ordinamento                                                */
#include "ins-sort-ric.c"

main()
{
   int *a;  /* Array dinamico a run time          */
   int c,n; /* Valore degli elementi e contratore */

   /* Caricamento della funzione che da il nome al programma */
   nomeprogramma ();

   printf("Dimensione dell'array? " );
   scanf("%d",&n);

   /* Funzione di allocazione di mewmoria a run time */
   a = (int*) malloc(sizeof(int)*n);

   /* Procedura di inserimento dati */
   for (c=0;c<n;c++)
   {
      printf("\n\nInserire il valore %d:" ,c);

      scanf("%d",&a[c]);
   }

   /* Stampa l'array inserito */
   printf("\n\nElementi dell'array inserito\n\n" );
   for (c=0; c<n;c++)
   { printf(" %d",a[c]); }


   /* Carica la funzione dell'ordinamento */
   sort(n, 0, a);

   /* Stampa l'array ordinato */
   printf("\n\nElementi dell'array ordinarto\n\n" );
   for (c=0; c<n;c++)
   { printf(" %d",a[c]); }


   getch();        // Variazione sul tema C++
   free(a);
}
```

BUBBLE SORT
Versione Ricorsiva

- **Scopo:** ordinare un insieme di n elementi con il metodo degli interscambi mediante un metodo ricorsivo.

- **Specifiche:**

```
void sort (int n, int k, int *m)
```

dove

n è il numero di elementi dell'array

k è il valore iniziale dell'array

***m** è il puntatore all'array dinamico a run time

- **Descrizione:** utilizzando un approccio di tipo ricorsivo e l'ausilio di una funzione che si occupa di effettuare lo scambio delle variabili, l'algoritmo controlla se due elementi contigui dell'array sono ordinati: se lo sono, la funzione prosegue richiamando se stessa per ordinare gli elementi successivi, altrimenti viene richiamata la funzione swap per effettuare lo scambio di variabili.

- **Complessità computazionale:** la complessità di tempo di questo algoritmo dipende dal valore dei dati forniti, i quali influenzano il numero di confronti e scambi da effettuare. Infatti, nel caso migliore(elementi già ordinati) si effettueranno solo n-1 confronti e nessuno scambio. Se invece l'array è completamente disordinato, allora bisognerà effettuare n-1 passaggi ed un numero di confronti e scambi pari a n(n-1)/2.

- **Funzioni ausiliarie:**

```
void swap (int *m, int i, int j)
```

dove

***m** è l'array

i la posizione successiva

j la posizione precedente

```c
/* Funzione che permette l'ordinamento di un array dinamico           */
/* run time medianto il metodo degli scambi, detto Bubble Sort.       */
/* La versione di seguito riportata è Ricorsiva.                      */
/*                                                                    */
/*                       Programma  sviluppato  da                    */
/*                                                                    */
/*                           Giovanni DI CECCA                        */
/*                                                                    */
/*                         http://www.dicecca.net                     */

/* Funzione per il nome del programma da passare al Main */
void nomeprogramma  ()
{
   printf ("Bubble Sort - Versione Ricorsiva"  );

   printf ("\n\nby Giovanni DI CECCA - http://www.dicecca.net\n\n"   );
}

/* Funzione libreria dello scambio parametri */
void swap  (int *m, int i, int j)
{
   int temp; /* Variabile temporanea */

   /* Scambio dei valore mediante il metodo delle tre variabili */
   temp  = m[i];
   m[i]  = m[j];
   m[j]  = temp;
}

/* Funzione di Bubble Sort */
void sort  (int n, int k, int *m)
{
   /* Indice della ricorsione, indicizzato a 0 */


   /* Variabile di conteggio */
   int i;

   /* Inizio del primo ciclo di scambi */
   for( i=0; i<n-1; i++ )
   {
      /* Funzione di controllo per eventuale scambio */
      if( m[i] > m[i+1] )
      {
         /* Carica la funzione di Swap */
         swap( m,i,i+1 );
         k = i;
      }
   }

   /* Funzione di controllo che carica ricorsivamente la funzione */
   if( k > 0 )
      sort  (n, k+1, m);

}
```

```
Bubble Sort - Versione Ricorsiva

by Giovanni DI CECCA - http://www.dicecca.net

Dimensione dell'array? 5

Inserire il valore 0:243

Inserire il valore 1:234

Inserire il valore 2:2

Inserire il valore 3:245

Inserire il valore 4:1

Elementi dell'array inserito
 243 234 2 245 1
Elementi dell'array ordinarto
 1 2 234 243 245
```

INSERTION SORT
Versione Ricorsiva

- **Scopo:** il presente programma si propone di ordinare in modo un insieme di n elementi disposti in modo casuale utilizzando il metodo di inserzione.

- **Specifiche:**

```
void sort (int n, int k, int *M)
```

dove

n è il numero di elementi dell'array

k è il valore iniziale dell'array

***M** è il puntatore all'array dinamico a run time

- **Descrizione:** Il vettore m di n componenti e' ordinato da 0 a k. Se k=n-1, tutto il vettore e' ordinato. Se k<(n-1), sia j (0<=j<=k) la posizione del primo elemento maggiore di v = M[k+1]. Il vettore ordinato si ottiene spostando gli elementi da j a k di una posizione a destra e copiando v in M[j]. Il resto del vettore si ordina richiamando la funzione con (M, k+1, n).
  La determinazione del valore di j e gli spostamenti si possono fare in una sola scansione della parte di vettore 0,...,k.
  Per non perdere valori, gli spostamenti si devono fare procedendo da k a j.

- **Complessità computazionale:** le operazioni dominanti dell'algoritmo sono confronti e inserimenti. Nel caso peggiore, quando cioè bisogna riordinare successioni ordinate in senso opposto, l'andamento riscontrato della complessità asintotica è O(n^2) [quadratico]. Invece, nel caso migliore, cioè quando l'insieme è già ordinato, la complessità risulta O(n) [lineare].

```c
/* Funzione che permette l'ordinamento di un array dinamico a          */
/* run time mediante il metodo dell'insersione detto Insertion Sort    */
/* La versione di seguito riportata è Ricorsiva.                       */
/*                                                                     */
/*                        Programma  sviluppato  da                    */
/*                                                                     */
/*                            Giovanni DI CECCA                        */
/*                                                                     */
/*                          http://www.dicecca.net                     */

/* Funzione per il nome del programma da passare al Main */
void nomeprogramma  ()
{
   printf ("Insertion Sort - Versione Ricorsiva"  );

   printf ("\n\nby Giovanni DI CECCA - http://www.dicecca.net\n\n"   );
}

/* Funzione di Ordinamento per Inserzione */
void sort  (int n, int k, int *M)
{
   /* Variabili di conteggio */
   int j,v,i;

   /* Cerca l'elemento più piccolo nell'array, e spostalo verso sinistra */
   if( k < n-1 )
   {
      /* Indicizza a 0 la variabile di conteggio dell'array */
      j = 0;

      /* Associa a v il valore del secondo elemento dell'array */
      v = M [k+1];


      while( j <= k && M[j] <= v )
         j++;

      /* Scambia il valore trovato con quello della posizione attuale */
      for(i=k;i>=j;i--)
      {
         M[i+1] = M[i]; /* crea lo spazio */
         M[j]=v;        /* inserisce */
      }

      /*chiamata ricorsiva che ordina la parte restante */
      sort  (n, k+1, M);

   }

}
```

```
Insertion Sort - Versione Ricorsiva

by Giovanni DI CECCA - http://www.dicecca.net

Dimensione dell'array? 5

Inserire il valore 0:10

Inserire il valore 1:33

Inserire il valore 2:2

Inserire il valore 3:132

Inserire il valore 4:1

Elementi dell'array inserito
 10 33 2 132 1
Elementi dell'array ordinarto
 1 2 10 33 132
```

SELECTION SORT
Versione Ricorsiva

- **Scopo:** L'algoitmo ordina in modo crescente un insieme di n elementi disposti casualmente utilizzando il metodo di selezione con un approccio di tipo ricorsivo.

- **Specifiche:**

```
void sort (int n, int k, int *b)
```

dove

n è il numero di elementi dell'array

k è il valore iniziale dell'array

***b** è il puntatore all'array dinamico a run time

- **Descrizione:** la funzione, prima seleziona l'elemento minimo e lo colloca in prima posizione, poi esegue un confronto con l'elemento successivo: se quest'ultimo è minore viene eseguito lo scambio e poi, effettuando una chiamata ricorsiva, si ordina la parte restante dell'insieme.

- **Complessità computazionale:** le operazioni dominanti dell'algoritmo sono CONFRONTI e SCAMBI. Gli scambi eseguiti sono n-1. I confronti sono invece pari a n(n-1)/2.

```c
/* Funzione che permette l'ordinamento di un array dinamico a          */
/* run time medianto il metodo della selezione Selection Sort          */
/* La versione di seguito riportata è Ricorsiva.                       */
/*                                                                     */
/*                        Programma  sviluppato  da                    */
/*                                                                     */
/*                            Giovanni DI CECCA                        */
/*                                                                     */
/*                          http://www.dicecca.net                     */

/* Funzione per il nome del programma da passare al Main */
void nomeprogramma  ()
{
   printf ("Selection Sort - Versione Ricorsiva"  );

   printf ("\n\nby Giovanni DI CECCA - http://www.dicecca.net\n\n"   );
}

/* Funzione di ordinameto per Selezione */
void sort  (int n, int i, int *b)
{

   int j, small; /* Variabile di conteggio */
   int temp; /* Variabile temporanea di swap */

   if (i < n-1)
   {
      /* Passo base: i = n-1. In questo caso la funzione ritorna senza modificare A */
      small = i;

      /* Cerca nell'array glielementi più piccoli */
      for (j = i+1; j < n; j++)
         if (b[j] < b[small])

      small = j;

      /* Funzione di Swap */
      temp = b[small];
      b[small] = b[i];
      b[i] = temp;

      /*ordina il resto per selezione ricorsiva */
      sort (n, i+1, b);

   }

}
```

```
Selection Sort - Versione Ricorsiva

by Giovanni DI CECCA - http://www.dicecca.net

Dimensione dell'array? 5

Inserire il valore 0:2

Inserire il valore 1:9

Inserire il valore 2:25

Inserire il valore 3:8

Inserire il valore 4:1

Elementi dell'array inserito
 2 9 25 8 1
Elementi dell'array ordinarto
 1 2 8 9 25
```

Merge Sort
(ordinamento per Fusione)

- **Scopo:** ordinare un insieme di n elementi utilizzando il metodo di fusione delgli array ordinati.

- **Specifiche:**

```
void sort (int fine, int inizio, int *a);
```

dove

`int fine` indica il valore complessio della grandezza di un array

`int inizio` indica il valore inziale di un array e dalla funzione prinipale è passato con valore 0

`int *a` è l'array dinamico a run time

- **Descrizione:** la funzione merge sort suddivide l'insieme in 2 sequenze, ordina separatamente ciascuna di esse con una chiamata ricorsiva, e dopo, chiamando la funzione merge, le due sequenze ordinate vengono fuse in un unico insieme finale ordinato.

- **Lista dei parametri:**

***a:** puntatore all'array degli elementi.
inizio: primo elemento dell'array.
fine: ultimo elemento dell'array.
medio: variabile locale usata per indicare il punto di confine tra le due sequenze separate da ordinare.

- **Funzioni ausiliarie:**

```
void merge (int *a, int inizio1, int inizio2, int fine);
int *ris;
int i, j, k;
```

- **Complessità computazionale:** l'operazione dominante eseguita è il confronto, ma il numero dipende dal valore dei dati di input.

```
/* Funzione per l'ordinamento di una array mediante l'operazione di    */
/* fusione (detta merge). La libreria comprende sia funzione di        */
/* divisione dell'array che la funzione di ordinamento                 */
/*                                                                     */
/* La versione di seguito riportata è Ricorsiva.                       */
/*                                                                     */
/*                        Programma  sviluppato  da                    */
/*                                                                     */
/*                            Giovanni DI CECCA                        */
/*                                                                     */
/*                          http://www.dicecca.net                     */

/* Funzione per il nome del programma da passare al Main */
void nomeprogramma  ()
{
   printf ("Merge Sort" );

   printf ("\n\nby Giovanni DI CECCA - http://www.dicecca.net\n\n"   );
}


/* Funzione di fusione */
void merge(int *a, int inizio1, int inizio2, int fine)
{
   /*variabili locali */
   int *ris;
   int i, j, k;

   /* Allocazione dinamica a run time dello spazio per inserire gli elementi degli */
   /* array divisi */
   ris = (int *)calloc((inizio1+fine),sizeof(int));

   /* Assegnazione alle altra variabili delle variabili scambiate */
   i=inizio1;
   j=inizio2;
   k=inizio1;

   /* Ciclo di ordinamento */
   while ((i <= inizio2-1) && (j <= fine))
   {
      /* Controllo per l'ordinamento del primo bolocco */
      if (a[i] < a[j])
      {
         ris[k] = a[i];
         i++;
      }

      else
      {
         ris[k] = a[j];
         j++;
      }

      k++;
   }

   /* Controllo per l'ordinamento del secondo bolocco */
   while (i <= inizio2-1)
   {
      ris[k] = a[i];
      i++;
      k++;
   }

   /* Controllo per l'ordinamento del terzo bolocco */
   while (j <= fine)
   {
      ris[k] = a[j];
      j++;
      k++;
   }

   /* Controllo per l'ordinamento del bolocco totale*/
   for (i=inizio1;i<=fine;i++)
      a[i]=ris[i];
```

```
   free (ris);
}

/* Funzione di divisione dell'array */
void sort(int fine, int inizio, int *a)
{

   /*variabile locale */
   int medio;

   /* Verifica se il punto di inizio è più piccolo del punto di inizio */
   if (inizio < fine)
   {
      /* Calcolo del punto mdio dell'array */
      medio = (inizio + fine)/2;

      /* Chiamata ricorsiva 1 */
      sort(medio, inizio, a);

      /* Chiamata ricorsiva 2 */
      sort(fine,medio+1,a);

      /* Chiamata della funzione merge */
      merge (a,inizio,medio+1,fine);
   }
}
```

```
Merge Sort
by Giovanni DI CECCA - http://www.dicecca.net
Dimensione dell'array? 5

Inserire il valore 0:45

Inserire il valore 1:121

Inserire il valore 2:5

Inserire il valore 3:1

Inserire il valore 4:6

Elementi dell'array inserito
 45 121 5 1 6
Elementi dell'array ordinarto
 1 5 6 20 45
```

Progetto per la gestione dello Stack

Lo Stack

- Introduzione

Lo stack (Pila in italiano) è una particolare struttura di dati,nel qule è possibile inserire ed elimianare dati (nel programma che di seguito documenterò utilizzerò solo numeri).

Questo tipo di struttura può essere rappresentato sia con gli array che con la linked list, cioè una catena di caselle (in teoria infinita) contengono in una metà il dato, nell'altra il collegamento (link, appunto) all'elemento successivo.

La versione che di seguito riporto è dinamica a run time, cioè il programma chiede la dimensione massima della pila ed in questa dimensione opera con una linked list.

Il programma l'ho spezzettato in due sorgenti: uno contiene i le chiamate alle funzioni, l'altra le fuzioni, in modo tale da renderlo più leggibile.

Il sorgente dell’inserimento dati

```c
/* Programma Main per la gestione dello Stack.                          */
/*                                                                      */
/*                          Programma sviluppato da                     */
/*                                                                      */
/*                            Giovanni DI CECCA                         */
/*                                                                      */
/*                          http://www.dicecca.net                      */

/* Definizioni delle costanti simboliche, tipi di variabili e di strutture dati */
#define ERR -1
#define EMPTY 1
#define FULL 2
#define OTHER 3

/* Parte Preprocessore */
#include <stdio.h>

/* Include la libreria per la gestione dello stack */
#include "stack.c"



main ()
{
   stack *st;          /* Utilizzo della struttura Stack */
   int n=1;            /* Dimensione massima dello stack, di default uguale a 1 */
   int terminato=0;    /* Variabilelogica per la chiusura del cilo while */
   int j;              /* Variabile che contiene il valore estratto */
   char c;             /* Variabile a carattere per la definizione di una operazione */

   printf("Programma per la gestione di una Pila"  );
   printf("\n\nby Giovanni DI CECCA - http://www.dicecca.net\n\n"   );

   /* Inserimento dati per la definizione dello stack */
   printf ("Dimensione Massima: " );
   scanf ("%d",&n);

   /* Inizializzazione dello stack */
   st=initstack (n);

   /* Ciclo while per l'inserimento e l'estazione degli elementi */
   while (!terminato)
   {
      printf (" ****************************************\n"   );
      printf (" ** PROGRAMMA PER LA GESTIONE DI UNA PILA **\n"   );
      printf (" ****************************************\n"   );
      printf ("\n");
      printf (" I --> Inserimento di un elemento\n"  );
      printf (" E --> Estrazione di un elemento\n"  );
      printf (" V --> Visualizzazione del primo elemento della pila\n"   );
      printf (" F --> Fine\n" );
      scanf ("%1s",&c);

      /* Comando Switch per l'attivazione dell'uso dei comandi */
      switch (c)
      {
         /* Inserimento dei dati nello stack */
         case 'I': case 'i':
            if (stato_stack (st) != FULL)    /* Controllo dello stack, se non è pieno, */
                                             /* inserisci                              */
               push (st,leggi_el ());
            else
               printf ("\nStack pieno\n" ); /* Se lo stack è pieno visualizzalo */
            break;

         /* Estrazione dei dati nello stack */
         case 'E': case 'e':
            if (stato_stack (st) == EMPTY)  /* Se lo stack è vuoto,...     */
               printf ("\nStack vuoto\n" ); /* ... Stampa lo stack è vuoto */
            else
            {
               j=pop (st);                                  /* Estrai l'elemento */
               printf ("\nL'elemento estratto è: %d\n"  ,j);  /* Stampa l'elemto estratto *
/
            }
```

```
            break;

            /* Visualizza l'ultimo elemento inserito nello stack */
            case 'V': case 'v':
               if (stato_stack(st) == EMPTY)  /* Se lo stack è vuoto,... */
                  printf ("\nStack vuoto\n" ); /* Stampa lo stack vuoto   */
               else
                  stampa_el  (top(st));         /* In caso contrario stampa lo stack */
               break;

            /* Uscita dal programma */
            case 'F': case 'f':
               terminato=1;
            break;
         }
      }

      /* Elimina gli elementi dallo stack */
      deletestack  (st);

      /* Libera la memoria occupata dalla variabile dello stack */
      free  (st);
   }
```

Illustrazione della libreria

- Inserimento di un elemento

Function per l'inserimento dati

```
/* Fuznione che inserisce i valori nella testa dello stack */
void push (stack *st,tipo_el el)
{
      /* Definisce la variabile dello stato dello stack */
      int stato;

      /* Carica nella variabile stato la funzione per il controllo dello stack */
      stato=stato_stack (st);

      /* Se lo stato è diverso da ERR (definito nel main) e diverso da FULL */
      /* (anch'esso definito nel main)                                      */
      if (stato != ERR && stato != FULL)
      {
            /* Inserisci st che punta a vett[di st che punta alla testa */
            /* il valore contenuto in el                                */
      st->vett[st->top]=el;

         /* Incrementa la testa dello Stack */
         (st->top)++;
      }
}
```

- Scopo

Effettua l'inserimento di un elemento (o nodo o registrazione) in una pila.

- Specifiche

```
void push (stack *st,tipo_el el);
int stato;
```

dove:

stack *st è la chiamata alla struttura che indica il link al prossimo elemento;

tipo_el el è la chiamata alla struttura che indica il tipo di elemento da inserire;

stato è lo stato dello stack

- Descrizione

Creato un nuovo esemplare della struttura, inserisce l'elemento e assegna al suo puntatore (o link o riferimento) il valore presente del puntatore di testa e a quest'ultimo l'indirizzo del nuovo elemento.

- Complessità computazionale

Complessità di Spazio:

2 variabili passate alla procedura, di cui una struttura + 1 variabile locale

- Esempio d'uso

Per l'esempio d'uso si rimanda all'esecuzione del programma a fine trattazione

- Estrazione di un elemento

Function per l'estrazione di un elemento

```
/* Estrai l'elemento dello stack */
int pop (stack *st)
{
      /* Definisce la variabile dello stato dello stack */
      int stato;

      /* Variabile che contiene l'elemento estratto */
      int k;

      /* Carica nella variabile stato la funzione per il controllo dello stack */
      stato=stato_stack (st);

      /* Se lo stato è diverso da ERR (definito nel main) */
      /* e diverso da EMPTY (anch'esso definito nel main) */
      if (stato != ERR && stato != EMPTY)
      {
            /* Associa a k il valore precedente alla testa della pila */
            k = st->vett[st->top-1];

            /* Decrementa la testa di un elemento */
            (st->top)--;
       }

       /* Ritorna il valore di k */
       return k;
}
```

- Scopo

Effettua l'estrazione di un elemento (o nodo o registrazione) da una pila.

- Specifiche

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

dove

stack *st è la chiamata alla struttura che indica il link al prossimo elemento;

stato è lo stato dello stack

k è la variabile contenente il dato in cima alla pila;

- Descrizione

Se la pila non è vuota, recuperato il dato in cima alla pila, assegna al puntatore di testa l'indirizzo del secondo elemento.

- Esempio d'uso

Per l'esempio d'uso si rimanda all'esecuzione del programma a fine trattazione

Il sorgente della libreria dati

```c
/* Libreria per la gestione dello Stack.                                */
/*                                                                      */
/*                          Programma sviluppato da                     */
/*                                                                      */
/*                            Giovanni DI CECCA                         */
/*                                                                      */
/*                          http://www.dicecca.net                      */

/* Definizione del tipo intero tipo_el */
typedef int tipo_el;


/* Definizione del tipo strutturato stack */
typedef struct stack
{
   /* Utilizzo del tipo usato per definire una variabile */
   tipo_el *vett;

   /* Variabile della dimensione dello stack */
   int dim;

   /* Variabile che memorizza la cima della pila */
   int top;

}stack; /* Istanza che identifica il modello della struttura appena definita */


/* Funzione per valutare lo stato dello stack */
int stato_stack (stack *st)
{
   /* Se la variabile st è uguale a NULL... */
   if (st == NULL)
            /* ...ritorna il valore di ERR (definito nel main) */
      return ERR;

   else
      /* Se st punta che a top, è uguale a 0... */
      if ((st->top) == 0)

         /* ...ritorna il valore di EMPTY (definito nel main) */
         return EMPTY;
   else

      /* Se st punta che a top, è uguale o superiore a st che punta a dim... */
      if ((st->top) >= (st->dim))

         /* ... ritorna il valore FULL (definito nel main) */
         return FULL;
   else

      /* il valore OTHER (anch'esso definito nel main) */
      return OTHER;
}


/* Inizializzazione dello Stack */
stack * initstack (int n)
{

   /* Definizione della variabile a puntatore *tmp che */
   /* usa l'istanza stack definita precedentemente     */
   stack *tmp;


   /* Alloca la memoria per l'inserimento dei dati */
   tmp=(stack *)malloc(sizeof(stack));

   /* Definisce il vettore che contiene gli elementi allocati */
   /* dinamicamente nello stack e ne conserva la posizione    */
   tmp->vett=(tipo_el *)calloc(n,sizeof(tipo_el));

   /* tmp che punta a top inizializza la testa dello stack */
   tmp->top=0;

   /* tmp che punta a dim definisce la grandezza dello stack? */
```

```
    tmp->dim=n;

    /* Ritorna il valore di tmp */
    return tmp;
}


/* Fuznione che inserisce i valori nella testa dello stack */
void push (stack *st,tipo_el el)
{
    /* Definisce la variabile dello stato dello stack */
    int stato;

    /* Carica nella variabile stato la funzione per il controllo dello stack */
    stato=stato_stack (st);

    /* Se lo stato è diverso da ERR (definito nel main) e diverso da FULL */
    /* (anch'esso definito nel main)                                      */
    if (stato != ERR && stato != FULL)
    {
        /* Inserisci st che punta a vett[di st che punta alla testa */
        /* il valore contenuto in el                                */
        st->vett[st->top]=el;

        /* Incrementa la testa dello Stack */
        (st->top)++;
    }
}


/* Estrai l'elemento dello stack */
int pop (stack *st)
{
    /* Definisce la variabile dello stato dello stack */
    int stato;

    /* Variabile che contiene l'elemento estratto */
    int k;

    /* Carica nella variabile stato la funzione per il controllo dello stack */
    stato=stato_stack (st);

    /* Se lo stato è diverso da ERR (definito nel main) */
    /* e diverso da EMPTY (anch'esso definito nel main) */
    if (stato != ERR && stato != EMPTY)
    {
        /* Associa a k il valore precedente alla testa della pila */
        k = st->vett[st->top-1];

        /* Decrementa la testa di un elemento */
        (st->top)--;
    }

    /* Ritorna il valore di k */
    return k;
}


/* Indica l'elemento posto alla cima della pila */
tipo_el top (stack *st)
{
    /* Definisce la variabile dello stato dello stack */
    int stato;

    /* Carica nella variabile stato la funzione per il controllo dello stack */
    stato=stato_stack (st);

    /* Se lo stato è diverso da ERR (definito nel main) */
    /* e diverso da EMPTY (anch'esso definito nel main) */
    if (stato != ERR && stato != EMPTY)

        /* Ritorna il valore che si trova all'estremità dello stack */
        return st->vett[st->top-1];
}
```

```c
/* Liberi gli elementi dello stack */
void deletestack  (stack *st)
{
   free (st->vett);
}


/* Inserimento di un valore dello stack */
tipo_el  leggi_el  (void)
{
   /* Variabile che contiene il valore da inserire nello stack */
   int n;

   /* Stampa a video l'inserimento */
   printf  ("\nInserire l'elemento: " );
   scanf  ("%d",&n);

   /* Ritorna il valore inserito */
   return  n;
}


/* Stampa l'ultimo elemento dello stack */
void stampa_el  (tipo_el  el)
{
   printf  ("L'ultimo elemento della pila Š: %d\n"  ,el);
}
```

```
Programma per la gestione di una Pila

by Giovanni DI CECCA - http://www.dicecca.net

Dimensione Massima: 3
 ****************************************
 ** PROGRAMMA PER LA GESTIONE DI UNA PILA **
 ****************************************

 I --> Inserimento di un elemento
 E --> Estrazione di un elemento
 V --> Visualizzazione del primo elemento della pila
 F --> Fine
i

Inserire l'elemento: 3
 ****************************************
 ** PROGRAMMA PER LA GESTIONE DI UNA PILA **
 ****************************************

 I --> Inserimento di un elemento
 E --> Estrazione di un elemento
 V --> Visualizzazione del primo elemento della pila
 F --> Fine
v
L'ultimo elemento della pila è: 3
 ****************************************
 ** PROGRAMMA PER LA GESTIONE DI UNA PILA **
 ****************************************

 I --> Inserimento di un elemento
 E --> Estrazione di un elemento
 V --> Visualizzazione del primo elemento della pila
 F --> Fine
e

L'elemento estratto Þ: 3
 ****************************************
 ** PROGRAMMA PER LA GESTIONE DI UNA PILA **
 ****************************************

 I --> Inserimento di un elemento
 E --> Estrazione di un elemento
 V --> Visualizzazione del primo elemento della pila
 F --> Fine
```

Progetto per la gestione di una Linked List

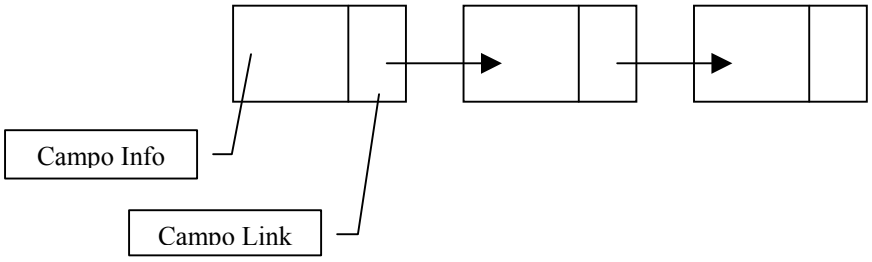
La Linked List

- Introduzione

La Linked List è una lista di contenente dati di tipo dinamico.

Infatti a differenza di un array (che contiene solo *n* dati, allocati a run time) questa, in linea teorica, non ha limiti in quanto viene allocata una locazione di memoria ogni qual volta si inserisce un elemento.

Non avendo un ordine cronologico, la Linekd List è un tipo strutturato formato da due campi: **Campo Link** e **Campo Info**:



Il **Campo Info** contiene il dato della lista, ad esempio un nome, mentre il **Campo Link** contiene il puntatore al blocco di dati successivo.

Illustrazione della libreria

- Inserimento di un elemento

Function per l'inserimento in testa e nel mezzo di una lista

```
/* Funzione che Inserisce il primo elemento in testa */
void inserthead()
{
      /* Variabile che permette l'inserimento dei caratteri */
      char name[20];

      /* Alloca la casella contente il valore da inserire nella lista */
      newelem = (LINK)malloc(sizeof(LISTANOMI));

      /* Il nuovo elemento che punta al prossimo è uguale alla testa */
      newelem -> NEXT = head;

      /* Il valore della testa è uguale al valore del prossimo elemento */
      head = newelem;

      /* Inserisce il valore */
      printf("\nInserire nome da inserire in cima alla lista: ");
      scanf("%s",&name);

      /* Usa la funzione di string.h per inserire il valore nella lista */
      strcpy (newelem -> nome, name);
      printf("\n");
}
```

```
/* Inserisci un valore nella seconda posizione */
void insertmiddle()
{
      /* Variabile che permette l'inserimento dei caratteri */
      char name[20];

      /* Alloca la casella contente il valore da inserire nella lista */
      newelem = (LINK)malloc(sizeof(LISTANOMI));

      /* Il nuovo elemento che punta al prossimo è uguale alla testa */
      /* che punta al prossimo */
      newelem -> NEXT = head -> NEXT;

      /* Il valore della testa che punta la prossimo è uguale al nuovo */
      /* elemento */
      head -> NEXT = newelem;

      printf("\nInserire nome da integrare alla lista: ");
      scanf("%s",name);

      /* Usa la funzione di string.h per inserire il valore nella lista */
      strcpy(newelem -> nome, name);

      printf("\n");
}
```

```
/* Inserisci in coda alla lista */
void insertend()
{
      /* Variabile che permette l'inserimento dei caratteri */
      char name[20];

      /* Considera il valore correte come la testa.                  */
      /* Questo escamotage serve a immettere il valore in coda       */
      /* come se fosse la                                            */
      /* seconda posizione, applicando poi la soluzione insertmiddle */
      current = head;

      /* Scorri la lista finché curent che punta a NEXT è diverso */
      /* da NULL */
      while (current -> NEXT != NULL)
      {
      current = current -> NEXT;
      }

      /* Alloca la casella contente il valore da inserire nella lista */
      newelem=(LINK)malloc(sizeof(LISTANOMI));

      /* Associa newelem alla posizione successiva a quella di current */
      /* che punta a NEXT */
      current -> NEXT = newelem;

      /* Associa NULL alla posizione successiva, cioè a newelem */
      newelem -> NEXT = NULL;

      printf("\nInserire nome da accodare alla lista: ");
      scanf("%s",&name);

      /* Usa la funzione di string.h per inserire il valore nella lista */
      strcpy(newelem -> nome, name);
      printf("\n");
}
```

- Scopo

Effettua l'inserimento di un elemento (o nodo o registrazione) in una lista, in testa, o in coda (in questo caso nel secondo posto)

- Specifiche

```
void inserthead()

void insertmiddle()

void insertend()
```

- Descrizione

Nel menu di scelta si possono inserire i dati **in testa**, **in mezzo** (che per default è la seconda posizione) ed **in coda**.

Per l'inserimento del dato **in testa**, il programma analizza la prima della lista e la sostituisce con quella nuova.

Per l'inserimento dei dati **in mezzo**, il programma analizza la prima voce e sostituisce la seconda voce con la il refer link appena inserito.

Per l'inserimento dei dati **in coda**, il programma analizza quale è l'ultima voce, (quella che punta a NULL) e accoda il referlink del dato, accodando il referlink NULL al dato appena inserito.

- Esempio d'uso

Per l'esempio d'uso si rimanda all'esecuzione del programma a fine trattazione

- Cancellazione di un elemento

Function per l'eliminazione di un nodo in una lista

```
void erase()
{
      /* Variabile logica inizializzata su falso */
      int trovato=0;

      /* Variabile che indica l'elemento da trovare */
      char key;

      /* Torna ad inizio Lista */
      current = head;

      /* Inserisci la chiave di ricerca dell'eliminazione del file */
      printf("\nQuale nome vuoi eliminare?");
      scanf("%s",&key);

      do
      {
            /* Se il la chiave di ricerca positiva */
            if (current->nome == key)
            {
                  printf("Eccolo!");
                  presskey();
                  trovato = 1; /* Indicatore logico su vero */
            }

            current = current -> NEXT;  /* Avanza nella ricerca */

      } while ((current != NULL) && (trovato == 0));
                                    /* Esegui il ciclo finché:   */
                                    /* current è diverso da NULL */
                                    /* o trovato e uguale a 0     */

      /* Se dopo la ricerca non vi è risultato, non esiste */
      /* l'elemento da cancellare */
      if (trovato == 0)
      printf ("\n\nElemento non trovato!!!");
      presskey();
}
```

- Scopo

Effettua la cancellazione di un elemento (o nodo o registrazione) in una lista.

- Specifiche

```
void erase()
```

- Descrizione

Inserendo il nome dell'elemento della lista, viene cancellato spostato il referlink all'elemento del successore dell'elemento da cancellare.

- Esempio d'uso

Per l'esempio d'uso si rimanda all'esecuzione del programma a fine trattazione

Il sorgente del Main e della Liberia delle Function

```c
/* Programma per lagestione di una Linked List.                          */
/* Questo è il programma Main                                            */
/*                                                                       */
/*                          Programma sviluppato da                      */
/*                                                                       */
/*                            Giovanni DI CECCA                          */
/*                                                                       */
/*                          http://www.dicecca.net                       */

/* Parte preprocessore */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Carica le procedure delle funzioni */
#include "liste_menu.c"

main()
{
   int scelta;  /* Variabile che indica la scelta del menu */

   /* Scermata iniziale del programma con l'identificativo di una etichetta      */
   /* che permette il caricamento nel caso si verificasse un errore nella scelta */
   /* del primo valore.                                                          */
   et3:printf("Programma per la gestione liste\n"  );
   printf("by Giovanni DI CECCA - http://www.dicecca.net\n\n\n"   );
   printf("\n\n\n");
   printf("Scegli l'operazione da eseguire:\n\n"  );
   printf(" 0 - Inserisci un elemento in cima alla lista\n"   );
   printf(" 5 - Esci dal programma\n\n" );
   printf("\n\n\n\n\n\n\n\n\n\n\n\n" );
   printf("Scegli: " );
   scanf ("%d",&scelta);

   /* Verifica se la scelta è stata effettuta tra i valori descritti nel menu */
   if (scelta==0 || scelta==5)
   {
      /* Salta allo switch */
      goto et2;
   }
   else
   {
      printf("La lista è ancora vuota!!!\n\n"  );
      printf("Si deve inserire la cima per poter attivare le altre opzioni\n\n\n"    );
      presskey();
      goto et3;
   }

   /* Carica il menu completo di tutte le opzioni.   */
   /* Da questo momento in poi è il menu predefinito */
   et1: printf("Programma per la gestione liste\n"  );
   printf("by Giovanni DI CECCA - http://www.dicecca.net\n\n\n"   );
   printf("\n\n\n");
   printf("Scegli l'operazione da eseguire:\n\n"  );
   printf(" 0 - Inserisci un elemento in cima alla lista\n"   );
   printf(" 1 - Inserisci un elemento nella lista\n"  );
   printf(" 2 - Accoda un elemento nella lista\n"  );
   printf(" 3 - Cancella un elemento\n" );
   printf(" 4 - Visualizza la lista \n" );
   printf(" 5 - Esci dal programma\n\n" );
   printf("\n\n\n\n\n\n\n" );
   printf("Scegli: " );
   scanf ("%d",&scelta);

   /* A secondo del valore di scelta, la procedura smista alle funzioni rispettive */
   et2:  switch(scelta)
   {
      case 0:
      {
         /* Inserisci in testa alla lista */
         inserthead();
         goto et1;
         break;
      }
```

```
        case 1:
        {
           /* Inserisci in mezzo alla lista (secondo posto) */
           insertmiddle ();
           goto et1;
        break;
        }

        case 2:
        {
           /* Inserisci in coda alla lista */
           insertend ();
           goto et1;
           break;
        }

        case 3:
        {
           /* Cancella un elemento dalla lista */
           erase ();
           goto et1;
           break;
        }
        case 4:
        {
           /* Visualizza la lista */
           displaylista ();
           goto et1;
          break;
        }

        case 5:
        {
           /* Esci dal programma */
           exit (0);
        }

        /* Opzione di default */
        default :
        {
           printf ("\nScelta non valida, riprovare\n\n"  );
           presskey ();
           goto et1;
        }
    }
}
```

```c
/* Programma per lagestione di una Linked List.                          */
/* Questo è la parte delle funzioni                                   */
/*                                                                       */
/*                          Programma sviluppato da                      */
/*                                                                       */
/*                            Giovanni DI CECCA                          */
/*                                                                       */
/*                          http://www.dicecca.net                       */

/* Definizione della struttura data */
struct data
{
   char nome[20];
   struct data *NEXT; /* Definizione della nuova struttura il cui puntatore è NEXT */
                      /* che punta al prossimo elemento della lista */
};

/* Definisci nuovi tipi strutturati */
typedef struct data LISTANOMI ;  /* Lista di nomi del tipo  */
typedef struct data *LINK;       /* dichiarato sopra        */
                                 /* LINK gestisce i movimenti */
                                 /* all'interno della lista */

/* Avendo precedentemente definito due nuovi tipi mediante la struttura "data" */
/* uso il puntatore LINK come un nuovo tipo */

/* Inizializzazione dei puntatori della lista a NULL */
LINK head = NULL;     /*    Inizio Lista (Head)                        */
LINK newelem = NULL;  /*    Nuovo Elemento della Lista (Newelem)       */
LINK current = NULL;  /*    Posizione corrente nella Lista (Current)   */


/* Funzione per visualizzare la schermta prima di passare alla successiva */
void presskey ()
{
   char key[20]; /* Variabile che permette l'inserimento dei caratteri */

   printf ("\n\nPremi un tasto e poi invio per continuare\n"   );
   scanf ("%s",&key);
   printf ("\n\n");
}

/* Funzione che Inserisce il primo elemento in testa */
void inserthead ()
{
   char name[20]; /* Variabile che permette l'inserimento dei caratteri */

   /* Alloca la casella contente il valore da inserire nella lista */
   newelem = (LINK)malloc (sizeof (LISTANOMI ));

   /* Il nuovo elemento che punta al prossimo è uguale alla testa */
   newelem -> NEXT = head;

   /* Il valore della testa è uguale al valore del prossimo elemento */
   head = newelem;

   /* Inserisce il valore */
   printf ("\nInserire nome da inserire in cima alla lista: "   );
   scanf ("%s",&name);

   /* Usa la funzione di string.h per inserire il valore nella lista */
   strcpy (newelem -> nome, name);
   printf ("\n");
}

/* Inserisci un valore nella seconda posizione */
void insertmiddle ()
{
   char name[20]; /* Variabile che permette l'inserimento dei caratteri */

   /* Alloca la casella contente il valore da inserire nella lista */
   newelem = (LINK)malloc (sizeof (LISTANOMI ));

   /* Il nuovo elemento che punta al prossimo è uguale alla testa che punta al prossimo *
/
```

```
    newelem -> NEXT = head -> NEXT;

    /* Il valore della testa che punta la prossimo è uguale al nuovo elemento */
    head -> NEXT = newelem;

    printf("\nInserire nome da integrare alla lista: "  );
    scanf("%s",name);

    /* Usa la funzione di string.h per inserire il valore nella lista */
    strcpy(newelem -> nome, name);

    printf("\n");
}

/* Inserisci in coda alla lista */
void insertend()
{
    char name[20]; /* Variabile che permette l'inserimento dei caratteri */

    /* Considera il valore correte come la testa.                          */
    /* Questo escamotage serve a immettere il valore in coda, come se fosse la */
    /* seconda posizione, applicando poi la soluzione insertmiddle         */
    current = head;

    /* Scorri la lista finché curent che punta a NEXT è diverso da NULL */
    while (current -> NEXT != NULL)
    {
        current = current -> NEXT;
    }

    /* Alloca la casella contente il valore da inserire nella lista */
    newelem=(LINK)malloc(sizeof(LISTANOMI));

    /* Associa newelem alla posizione successiva a quella di current che punta a NEXT */
    current -> NEXT = newelem;

    /* Associa NULL alla posizione successiva, cioè a newelem */
    newelem -> NEXT = NULL;

    printf("\nInserire nome da accodare alla lista: "  );
    scanf("%s",&name);

    /* Usa la funzione di string.h per inserire il valore nella lista */
    strcpy(newelem -> nome, name);
    printf("\n");
}

/* Stampa la lista */
void displaylista()
{
    /* Indice contatore inizializzato a 0 */
    int i=0;

    /* Cambia la posizione corrente con la testa, in modo da trovarsi in testa alla lista
*/
    current = head;

    /* Stampa un form a video che contenga la lista */
    printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"  );
    printf("Lista corrente\n\n" );

    /* Ciclo while che stampa gli elementi della lista */
    while (current != NULL)
    {
        /* Contatore che permette la visualizzazione della posizione */
        /* di un elemente nella lista                                */
        i++;

        /* Stampa a video la posizione dell'elemnto nella lista */
        printf("%d ",i);

        /* Stampa gli elementi nella lista */
        printf(">   %s\n", current -> nome);

        /* Avanza nella lista */
```

```
      current = current -> NEXT;
   }

   /* Carica la funzione di Premi un tasto per continuare */
   presskey();
}

void erase()
{
   /* Variabile logica inizializzata su falso */
   int trovato=0;

   /* Variabile che indica l'elemento da trovare */
   char key;

   /* Torna ad inizio Lista */
   current = head;

   /* Inserisci la chiave di ricerca dell'eliminazione del file */
   printf("\nQuale nome vuoi eliminare?" );
   scanf("%s",&key);

   do
   {
      /* Se il la chiave di ricerca positiva */
      if (current->nome == key)
      {
         printf("Eccolo!");
         presskey();
         trovato = 1; /* Indicatore logico su vero */
      }

      current = current -> NEXT;  /* Avanza nella ricerca */

   } while ((current != NULL) && (trovato == 0)); /* Esegui il ciclo finché:   */
                                                  /* current è diverso da NULL */
                                                  /* o trovato e uguale a 0    */

   /* Se dopo la ricerca non vi è risultato, non esiste l'elemento da cancellare */
   if (trovato == 0)
   printf ("\n\nElemento non trovato!!!" );
   presskey();
}
```

```
Programma per la gestione liste
by Giovanni DI CECCA - http://www.dicecca.net



Scegli l'operazione da eseguire:

 0 - Inserisci un elemento in cima alla lista
 5 - Esci dal programma









Scegli: 0

Inserire nome da inserire in cima alla lista: Giovanni

Programma per la gestione liste
by Giovanni DI CECCA - http://www.dicecca.net



Scegli l'operazione da eseguire:

 0 - Inserisci un elemento in cima alla lista
 1 - Inserisci un elemento nella lista
 2 - Accoda un elemento nella lista
 3 - Cancella un elemento
 4 - Visualizza la lista
 5 - Esci dal programma







Scegli: 1

Inserire nome da integrare alla lista: Virginia

Programma per la gestione liste
by Giovanni DI CECCA - http://www.dicecca.net



Scegli l'operazione da eseguire:

 0 - Inserisci un elemento in cima alla lista
 1 - Inserisci un elemento nella lista
 2 - Accoda un elemento nella lista
 3 - Cancella un elemento
 4 - Visualizza la lista
 5 - Esci dal programma
```

```
Scegli: 1

Inserire nome da integrare alla lista: Claudio

Programma per la gestione liste
by Giovanni DI CECCA - http://www.dicecca.net




Scegli l'operazione da eseguire:

 0 - Inserisci un elemento in cima alla lista
 1 - Inserisci un elemento nella lista
 2 - Accoda un elemento nella lista
 3 - Cancella un elemento
 4 - Visualizza la lista
 5 - Esci dal programma




Scegli: 2

Inserire nome da accodare alla lista: Pippo

Programma per la gestione liste
by Giovanni DI CECCA - http://www.dicecca.net




Scegli l'operazione da eseguire:

 0 - Inserisci un elemento in cima alla lista
 1 - Inserisci un elemento nella lista
 2 - Accoda un elemento nella lista
 3 - Cancella un elemento
 4 - Visualizza la lista
 5 - Esci dal programma




Scegli: 4
```

```
Lista corrente

1 >   Giovanni
2 >   Claudio
3 >   Virginia
4 >   Pippo


Premi un tasto e poi invio per continuare
ò


Programma per la gestione liste
by Giovanni DI CECCA - http://www.dicecca.net




Scegli l'operazione da eseguire:

 0 - Inserisci un elemento in cima alla lista
 1 - Inserisci un elemento nella lista
 2 - Accoda un elemento nella lista
 3 - Cancella un elemento
 4 - Visualizza la lista
 5 - Esci dal programma







Scegli: 3

Quale nome vuoi eliminare?pippo


Elemento cancellato!!!

Premi un tasto e poi invio per continuare
h









Lista corrente

1 >   Giovanni
2 >   Claudio
3 >   Virginia


Premi un tasto e poi invio per continuare
```

Progetto per la gestione di un DataBase

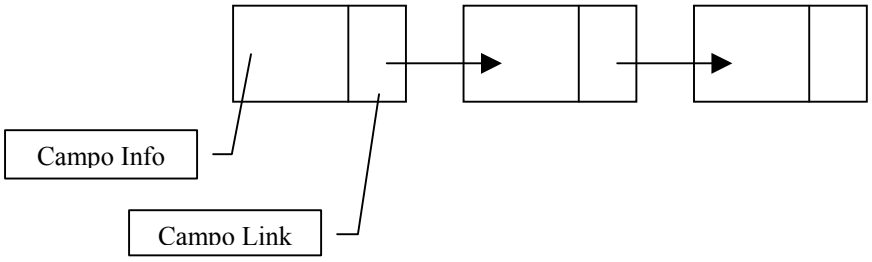
Il DataBase

- Introduzione

Il DataBase è una lista di contenente dati di tipo dinamico.

Infatti a differenza di un array (che contiene solo *n* dati, allocati a run time) questa, in linea teorica, non ha limiti in quanto viene allocata una locazione di memoria ogni qual volta si inserisce un elemento.

Non avendo un ordine cronologico, la Linekd List è un tipo strutturato formato da due campi: **Campo Link** e **Campo Info**:



Il **Campo Info** contiene i dati della lista. A differenza della Liked list il campo info contiene Nome, Cognome, Matricola, Indirizzo e Telefono di uno studente. Il **Campo Link**, come nella Linked List, contiene il puntatore al blocco di dati successivo.

Illustrazione della libreria

- Inserimento di un elemento

Function's per l'inserimento dati nel DataBase

```
/* Funzione di inserimento in testa */
void ins_head(int mat)
{
      /* Definizioni di variabili globali */
      char *temp;

      temp = (char*)malloc(sizeof(char));
      nuovo = (LINK)malloc(sizeof(DATI));
      nuovo -> NEXT = head;
      head = nuovo;
      nuovo -> matricola = mat;

      /* Inserimento dei dati mediante la libreria string.h */
      puts("Inserire il cognome e il nome (rispettivamente): ");
      gets (temp);
      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);
      printf("Inserimento dello studente - %d - effettuato.\n\n",mat);
      presskey();
}



/* Funzione di inserimento dati nel database */
void inserisci()
{
      /* Definizione delle variabile locali */
      char *temp;
      int mat;

      /* Definizione del puntatore contente i dati da inserire */
      /* nel database */
      temp = (char*)malloc(sizeof(char));

      /* Inserimento della matricola */
      printf("\nMatricola da inserire: ");
      scanf("%d",&mat);

      /* Verifica se quello che si sta inserendo è la testa della lista */
      /* del DataBase */
      if (head == NULL)    /* Nel caso la lista sia vuota, l'inserimento */
            ins_head(mat); /* dev'essere effettuato in testa */
      else                 /* In caso contrario... */
      {
            /* Precedente carica la funzione doveinserire */
            prec = doveinserire(mat);
```

```
                /* Se precedente è uguale alla testa, inserisci in testa, */
                /* altrimenti                                              */
                /* inserisici in seconda posizione dopo                    */
                /* head                                                    */
                if (prec == head)
                   ins_head(mat);
                else
                {
                      /* Crea un puntatore "nuovo" contenente il Link */
                      /* al prox elemento e Dati contiente l'informazione */
                      nuovo = (LINK)malloc(sizeof(DATI));
                      nuovo -> NEXT = prec -> NEXT;
                      prec -> NEXT = nuovo;
                      nuovo -> matricola = mat;

                      /* Inserimento dati mediante l'uso la libreria */
                      /* string.h */
                      printf("Inserire il cognome e il nome (rispettivamente): ");
                      temp = NULL;
                      gets (temp);
                      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);
                      printf("Inserimento dello studente - %d - effettuato.\n\n",mat);
                }
          }
    }
```

- Scopo

Effettua l'inserimento di un elemento (o nodo o registrazione) in una lista, in testa, o nel mezzo.

- Specifiche

```
void ins_head(int mat)
```

dove:

`int mat` è la matricola

```
void inserisci()
```

- Descrizione

Nel menu di scelta si possono inserire i dati **in testa**, **in mezzo**.

Per l'inserimento del dato **in testa**, il programma analizza la prima della lista e la sostituisce con quella nuova.

Per l'inserimento dei dati **in mezzo**, il programma analizza la prima voce e sostituisce la seconda voce con la il refer link appena inserito.

- Esempio d'uso

Per l'esempio d'uso si rimanda all'esecuzione del programma a fine trattazione

- Cancellazione di un elemento

Function per la cancellazione di un elemento

```
/* Questa function serve per cancellare l'elemento nella lista */
void cancella()
{
      int mat, trovato=0;
      current = head;
      prec = head;

      if (head == NULL) /* Caso in cui la lista sia vuota */
      {
            printf("\n\nLa lista è vuota!!! Cosa elimino?\n");
            presskey();
      }
      else
      {
            puts("\n\nInserire la matricola dello studente da cancellare: ");
            scanf("%d",&mat);
            while ((current != NULL) && (trovato == 0))
            {

                  if (current -> matricola == mat)
                  {
                        /* Se la matricola da eliminare si trova */
                        /* in testa, assegna direttamente un     */
                        /* valore al puntatore "head"            */
                        if (current == head)
                        {
                              head = current -> NEXT;
                              printf("Lo studente - %d - è stato cancellato\n",mat);
                              presskey();
                              trovato = 1;
                        }
                        /* In caso contrario, salta direttamente al link successivo */
                        else
                        {
                              prec -> NEXT = current -> NEXT;
                              printf("Lo studente - %d - è stato cancellato\n",mat);
                              presskey();
                              trovato = 1;
                        }
                  }

                  prec = current;
                  current = current -> NEXT; /* Avanza nella ricerca */
            }

            if (trovato == 0)
            {
                  printf ("\nNon posso cancellare qualcosa che non esiste!!!\n\n");
                  presskey();
            }
      }
}
```

- Scopo

Effettua la cancellazione di un elemento (o nodo o registrazione) nel database.

- Specifiche

```
void erase()
```

- Descrizione

Inserendo il nome dell'elemento della lista, viene cancellato spostato il referlink all'elemento del successore dell'elemento da cancellare.

- Esempio d'uso

Per l'esempio d'uso si rimanda all'esecuzione del programma a fine trattazione

Il sorgente del Main e della Liberia delle Function

```c
/* Programma per lagestione di un database studente                    */
/* Questo è il programma Main                                          */
/*                                                                     */
/*                          Programma sviluppato da                    */
/*                                                                     */
/*                            Giovanni DI CECCA                        */
/*                                                                     */
/*                          http://www.dicecca.net                     */

/* Parte preprocessore */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Libreria utente */
#include "db.c"

/* Funzione Principale */
main()
{
   int mat, scegli; /* Variabile locale */

   while (1) /* Ciclo while che ripete il menù fin quando non si inserisce "5" */
   {
      /* Comando switch che reindirizza nel menu */
      switch(menu_io())
      {
         /* Caso 1: carica la funzione di inserimento dati nel database */
         case 1:
         {
            inserisci();
            break;
         }

         /* Caso 2: carica la funzione di cancella dati nel database */
         case 2:
         {
            cancella();
            break;
         }


         /* Caso 4: carica la funzione di visualizza dati nel database */
         case 3:
         {
            database_output();
            break;
         }

         /* Caso 5: Esci dal database */
         case 4:
         {
            printf("\nFine del programma...\n\n" );
            exit(0);
         }

         /* Funzione di Default */
         default:
         {
            printf("\nScelta non valida, riprovare\n\n" );
         }
      }
   }

   return 0;
}
```

```
/* Programma per lagestione di un database studente                    */
/* Questo è la libreria delle funzioni                                 */
/*                                                                     */
/*                          Programma sviluppato da                    */
/*                                                                     */
/*                            Giovanni DI CECCA                        */
/*                                                                     */
/*                          http://www.dicecca.net                     */

/* Crea la struttura studente */
struct studente
{
   int matricola;          /* campo matricola   */
   char nome[60];          /* campo nome        */
   char datanascita[10];   /* campo datanascita */
   char indirizzo[80];     /* campo indirizzo   */
   char tel[11];           /* campo telefono    */
   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)   */

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

/* I puntatori, inoltre li abbiamo inizializzati */
/* a NULL; notare che la lista, adesso, esiste   */
/* ed è vuota */

/* Prototipi delle funzioni necessarie al MAIN */
void presskey();
int menu_io();
void inserisci();
LINK doveinserire(int);
void ins_head(int);
void inserimento();
void cancella();
void database_output();

/* Funzione per premere un tasto */
void presskey()
{
   char key[20];
   puts("(Premi un tasto e poi invio per continuare)\n"  );
   scanf("%s",&key);
   printf("Grazie %s\n",key);
}

/* Menu di IO presente nel main */
int menu_io()
{
   int scelta;
   printf("Programma per la gestione di un DataBase\n"  );
   printf("by Giovanni DI CECCA - http://www.dicecca.net\n\n\n"  );
   printf("Scegli l'operazione da eseguire:\n\n"  );
   printf(" 1.  Inserisci un elemento\n" );
   printf(" 2.  Cancella un elemento\n" );
   printf(" 3.  Visualizza l'intero database \n"  );
   printf(" 4.  Esci dal programma\n\n\n\n"  );
   printf("\n\n\n\n\n\n\n\n\n" );
```

```
    printf("Scegli: ");
    scanf ("%d",&scelta);

    /* Ritorna il valore scelto */
    return scelta;
}

/* Funzione di inserimento dati nel database */
void inserisci()
{
    /* Definizione delle variabile locali */
    char *temp;
    int mat;

    /* Definizione del puntatore contente i dati da inserire nel database */
    temp = (char*)malloc(sizeof(char));

    /* Inserimento della matricola */
    printf("\nMatricola da inserire: " );
    scanf("%d",&mat);

    /* Verifica se quello che si sta inserendo è la testa della lista del DataBase */
    if (head == NULL)    /* Nel caso la lista sia vuota, l'inserimento */
        ins_head(mat);   /* dev'essere effettuato in testa */
    else                 /* In caso contrario... */
    {
        /* Precedente carica la funzione doveinserire */
        prec = doveinserire(mat);
        /* Se precedente è uguale alla testa, inserisci in testa, altrimenti */
        /* inserisici in seconda posizione dopo head                         */
        if (prec == head)
            ins_head(mat);
        else
        {
            /* Crea un puntatore "nuovo" contenente il Link al prox elemento */
            /* e Dati contiene l'informazione                                */
            nuovo = (LINK)malloc(sizeof(DATI));
            nuovo -> NEXT = prec -> NEXT;
            prec -> NEXT = nuovo;
            nuovo -> matricola = mat;

            /* Inserimento dati mediante l'uso la libreria string.h */
            printf("Inserire il cognome e il nome (rispettivamente): "   );
            temp = NULL;
            gets (temp);
            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);
            printf("Inserimento dello studente - %d - effettuato.\n\n"   ,mat);
        }
    }
}

/* Funzione di inserimento in testa */
void ins_head(int mat)
{
    /* Definizioni di variabili globali */
    char *temp;

    temp = (char*)malloc(sizeof(char));
    nuovo = (LINK)malloc(sizeof(DATI));
    nuovo -> NEXT = head;
    head = nuovo;
    nuovo -> matricola = mat;

    /* Inserimento dei dati mediante la libreria string.h */
    puts("Inserire il cognome e il nome (rispettivamente): "   );
```

```
    gets (temp);
    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);
    printf("Inserimento dello studente - %d - effettuato.\n\n"   ,mat);
    presskey();
}

/* Stampa a monitor del database */
void database_output ()
{
    /* La variabile corrente è uguale alla testa */
    current = head;

    printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n" );
    printf(" _____________________\n" );
    printf("| Database corrente... |" );
    printf("\n --------------------------------\n" );

    /* Ciclo while che visualizza il DataBase */
    while (current != NULL) /* Fino a quando la lista non finisce */
    {
        printf("> Matricola: %d   Nome: %s\n" , current -> matricola, current -> nome);
        printf("> Nato il: %s\n" , current -> datanascita);
        printf("> Indirizzo: %s\n" , current -> indirizzo);
        printf("> Telefono: %s\n" , current -> tel);

        /* Incrementa il contatore e visualizza gli altri dati */
        current = current -> NEXT;
    }
    printf(" _____________________________________\n\n" );

    presskey();
}

/* Questa funzione serve per trovare il punto in cui inserire */
/* lo studente nella lista, in modo che la lista venga        */
/* ordinata per matricola senza dover necessariamente usare   */
/* una function per l'ordinamento della lista                 */
LINK doveinserire (int mat)
{
    LINK puntatore;
    int max = mat;
    current = head;
    prec = head;

    while (current != NULL)
    {
        if (current -> matricola > max)
        {
            max = current -> matricola;  /* Salviamo la matricola */
            printf("Matricola: %d" , max);
            puntatore = prec;
        }

        prec = current;

        current = current -> NEXT; /* Avanziamo nella ricerca */
    }

    /* Ritorna il valore del puntatore */
    return puntatore;
}

/* Questa function serve per cancellare l'elemento nella lista */
void cancella()
{
```

```
    int mat, trovato=0;
    current = head;
    prec = head;

    if (head == NULL) /* Caso in cui la lista sia vuota */
    {
       printf("\n\nLa lista è vuota!!! Cosa elimino?\n"  );
       presskey();
    }
    else
    {
       puts("\n\nInserire la matricola dello studente da cancellare: "   );
       scanf("%d",&mat);
       while ((current != NULL) && (trovato == 0))
       {

          if (current -> matricola == mat)
          {
             /* Se la matricola da eliminare si trova */
             /* in testa, assegna direttamente un     */
             /* valore al puntatore "head"            */
             if (current == head)
             {
                head = current -> NEXT;
                printf("Lo studente - %d - è stato cancellato\n"  ,mat);
                presskey();
                trovato = 1;
             }
             /* In caso contrario, salta direttamente al link successivo */
             else
             {
                prec -> NEXT = current -> NEXT;
                printf("Lo studente - %d - è stato cancellato\n"  ,mat);
                presskey();
                trovato = 1;
             }
          }

          prec = current;
          current = current -> NEXT; /* Avanza nella ricerca */
       }

       if (trovato == 0)
       {
          printf ("\nNon posso cancellare qualcosa che non esiste!!!\n\n"   );
          presskey();
       }
    }
}
```

```
Programma per la gestione di un DataBase
by Giovanni DI CECCA - http://www.dicecca.net


Scegli l'operazione da eseguire:

 1.  Inserisci un elemento
 2.  Cancella un elemento
 3.  Visualizza l'intero database
 4.  Esci dal programma




Scegli: 1

Matricola da inserire: 50887
Inserire il cognome e il nome (rispettivamente):
DI CECCA Giovanni
Data di nascita (GG/MM/AAAA):
03/11/1977
Indirizzo:
Via Tribunali, 122
Recapito telefonico:
348 80 41 505
Inserimento dello studente - 50887 - effettuato.

(Premi un tasto e poi invio per continuare)

d
Grazie d
Programma per la gestione di un DataBase
by Giovanni DI CECCA - http://www.dicecca.net


Scegli l'operazione da eseguire:

 1.  Inserisci un elemento
 2.  Cancella un elemento
 3.  Visualizza l'intero database
 4.  Esci dal programma




Scegli: 3
```

```
 _____________________
| Database corrente... |
 ------------------------------
> Matricola: 12850   Nome: DI CECCA Giovanni
> Nato il: 03/11/1977Via Tribunali, 122
> Indirizzo: Via Tribunali, 122
> Telefono: 348 80 41 505
Programma per la gestione di un DataBase
by Giovanni DI CECCA - http://www.dicecca.net


Scegli l'operazione da eseguire:

 1.  Inserisci un elemento
 2.  Cancella un elemento
 3.  Visualizza l'intero database
 4.  Esci dal programma




Scegli: 2


Inserire la matricola dello studente da cancellare:
50887
Lo studente - 50887 - Þ stato cancellato
(Premi un tasto e poi invio per continuare)
```

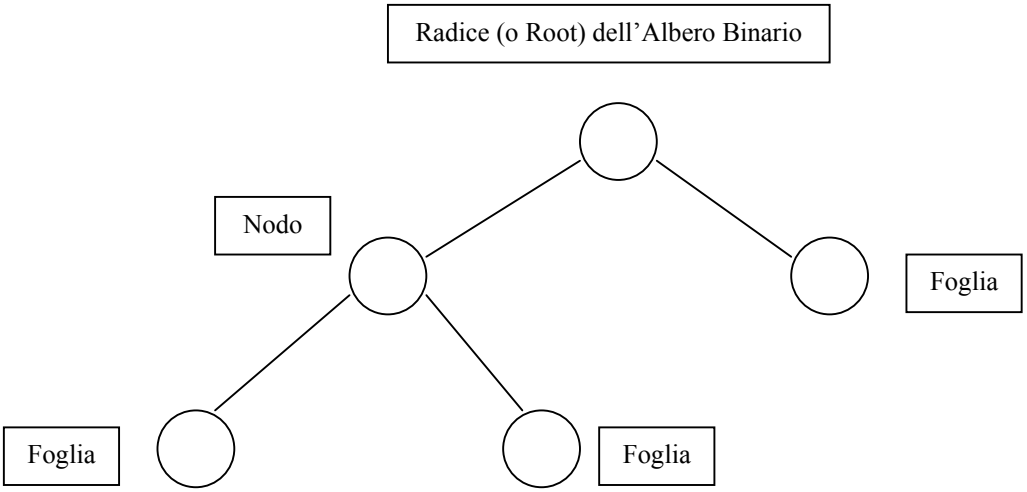
Progetto per la gestione di un Albero Binario

L’Albero Binario

- Introduzione

L’Albero Binario è un particolare tipo di struttura dinamica di tipo gerachico. A differenza delle lista lineare (la precedente Linked List) l’Albero prevede che ogni suo nodo possa avere diversi figli.

Tra i vari tipi di Alberi, l’Albero Binario prevede che per ogni nodo vi siano solo due figli:



Esempio di Albero Binario

La **Root** è la radice dell’Albero.

Il **Nodo** è la congiunzione tra il **Padre** del Nodo ed i **Figli** del Nodo.

La **Foglia** è la terminazione dei figlio del Nodo che Padre che non hanno discendenze.

Illustrazione della libreria

- Inserimento di un elemento

Function dell'inserimento di un nodo o elemento

```
logical insert_node (tree *albero,int k)
{
 tree *predecessore,*a;
 while ((albero != NULL)&&(albero->dato != k))
 {
  predecessore=albero;
  if (k > albero->dato)
   albero=albero->dx;
  else
   albero=albero->sx;
 }
 if (albero != NULL)
  return false;
  printf ("L'elemento è già presente nell'albero \n");
 else
 {
  a=(tree *)malloc(sizeof(tree));
  a->dato=k;
  a->sx=NULL;
  a->dx=NULL;
  if (k > predecessore->dato)
   predecessore->dx=a;
  else
   predecessore->sx=a;
  return true;
 }
}
```

− Scopo

Effettua l'inserimento di un elemento (o nodo) in un albero binario.

− Specifiche

```
logical insert node (tree *albero, int k);
```

dove

albero variabile di tipo strutturato dinamico gerarchico tree definito nella libreria utente "btree.h";

k variabile scalare intera che contiene l'elemento da inserire;

− Descrizione

Creato un nuovo esemplare della struttura, esamina i nodi successivi dell'albero per trovare la posizione del dato da inserire (spostandosi nel sottoalbero destro se maggiore, sinistro se minore) fino a trovare un puntatore il cui valore è NULL se il dato non è presente nell'albero, dopodiché, inserito l'elemento, assegna il suo indirizzo al puntatore destro del nodo che lo precede se il dato di quest'ultimo è minore di tale elemento, al puntatore sinistro se maggiore.

− Complessità computazionale

2 variabili passate alla procedura, di cui una struttura + 2 variabili locali (struttura)

- Ricerca di un elemento

Function della Rcerca

```
logical binary_tree_search (tree *root,int key)
{
        if (root == NULL)
                return false;
         else
                if (root->dato==key)
                        return true;
                else
                        if (root->dato < key)
                                return binary_tree_search (root->dx,key);
                        else
                                return binary_tree_search (root->sx,key);
}
```

– Scopo

Effettua la ricerca di un elemento (o nodo) in un albero binario.

– Specifiche

```
logical binary_tree_search (tree *root,int key);
```

– Descrizione

La function esamina (richiamando se stessa) tutti i nodi successivi dell'albero per trovare il dato da ricercare fino a trovare un puntatore il cui valore è NULL se il dato non è presente nell'albero.

dove:

`root` variabile di tipo strutturato dinamico gerarchico tree definito nella libreria utente "linked_list.h";

`key` varibile scalare intera che contiene l'elemento da ricercare;

– Complessità computazionale

2 variabili passate alla procedura, di cui una struttura

- Metodi per la visualizzazione degli elementi

Function della stampa **In Order**

```
void stampa_in_order (tree *root)
{
        if (root != NULL)
        {
                stampa_in_order (root->sx);
                printf ("%d\n",root->dato);
                stampa_in_order (root->dx);
        }
}
```

Function della stampa **Pre Order**

```
void stampa_pre_order (tree *root)
{
        if (root != NULL)
        {
                printf ("%d\n",root->dato);
                stampa_pre_order (root->sx);
                stampa_pre_order (root->dx);
        }
}
```

Function della stampa **Post Order**

```
void stampa_post_order (tree *root)
{
        if (root != NULL)
        {
                stampa_post_order (root->sx);
                stampa_post_order (root->dx);
                printf ("%d\n",root->dato);
        }
}
```

– Scopo

Effettuano la stampa dei dati dei nodi di un albero binario in ordine naturale, posposto o preposto.

– Specifiche

```
void stampa_in_order (tree *root);

void stampa_pre_order (tree *root);

void stampa_post_order (tree *root);
```

dove:

root variabile di tipo strutturato dinamico gerarchico tree definito nella libreria utente "btree.h";

– Descrizione

Le functions effettuano la stampa (richiamando se stesse) dei dati di tutti i nodi dell'albero. La prima utilizzando il criterio di visita - per visita si intende la generazione di una sequenza che contenga tutti i nodi esattamente una volta - nell'ordine naturale (inorder o SVD, dove V è la rappresentazione del contenuto del nodo, S l'accesso al nodo figlio di sinistra, D l'accesso al nodo figlio di destra), la seconda il criterio di visita posposto (postorder o SDV) e la terza quello preposto (preorder o VSD).

– Complessità computazionale

2 variabili passate alla procedura, di cui una struttura

Il sorgente del Main e della Liberia delle Function

```c
/* Programma per lagestione di un Albero Binario                    */
/* Questo è il programma Main                                       */
/*                                                                  */
/*                          Programma sviluppato da                 */
/*                                                                  */
/*                            Giovanni DI CECCA                     */
/*                                                                  */
/*                          http://www.dicecca.net                  */


/* Parte Preprocessore */
#include <stdio.h>
#include <stdlib.h>

/* Libreria Utente */
#include "btree.h"

/* Programma Main */
main ()
{
 /* Definizione del puntatore alla struttura albero */
 tree *head;

 char c; /* Variabile per la selezione del carattere di scelta */

 /* Variabile per l'inserimento dell'elemento e chiave di ricerca */
 int el, key;

 /* Tipo booleano per la verifica dell'elemento inserito */
 logical x;

 /* Tipo booleano per l'uscita dal programma */
 logical terminato =false;

 /* Carica la libreria che contiene il nome del programma */
 nomeprogramma ();

 /* Inserimento dell'elemento nella radice */
 printf ("Inserire la radice (valori interi): " );
 scanf ("%d",&el);

 /* Carica la funzione di inserimento nall Root dell'albero */
 head=init_albero (el);

 /* Spazi per far scomparire L'inserimento del video */
 printf ("\n\n\n\n\n\n\n\n\n\n" );

 /* Ciclo while di gestione del menu */
 while (!terminato )
 {
  /* Carica il menu dall'apposita libreria */
  menu_io ();

  /* Usa la lettera che contiene l'elemento selezionato */
  printf ("\n\nInserire la scelta: " );
  scanf ("%1s", &c);

  switch (c)
  {

   /* Carica la funzione del caricamento dell'elemento nell'Albero */
   case 'I': case 'i':
    printf ("Elemento da inserire: " );
    scanf ("%d", &el);

    /* Uso del tipo Logical per l'inserimento dell'elemento */
    x=insert_node (head,el);

    /* Verifica dell'elemento inserito */
    if (!x)
     printf ("Il nodo Š gi… presente nell'albero" );
    break;

   /* Ricerca l'elemento nell'albero */
   case 'R': case 'r':
```

```
      printf ("Inserire l'elemento da ricercare: " );
      scanf ("%d",&key);

      /* Carica la funzione di ricerca dell'elemento nell'Albero e verifica */
      if (binary_tree_search (head,key))
       printf ("Elemento presente nell'albero \n" );
      else
       printf ("Elemento non presente nell'albero \n" );
      break;

      /* Stampa la funzione in In Order */
      case 'S': case's':
      stampa_in_order (head);
      break;

     /* Stampa la funzione in Pre Order */
     case 'P': case'p':
      stampa_pre_order (head);
      break;

     /* Stampa la funzione in Post Order */
     case 'O': case'o':
      stampa_post_order (head);
      break;

     /* Carica la funzione di conteggio dei nodi */
     case 'C': case 'c':
      key=conta_nodi (head);
      printf ("L'Albero ha %d nodi\n" ,key);
      break;

     /* Carica la funzione della profondità dell'albero */
     case 'L': case 'l':
      key=profondita (head)-1;
      printf ("L'Albero ha profondit…: %d\n" ,key);
      break;

     /* Esci dal programma */
     case 'Q': case 'q':
      terminato =true;
      break;

     /* Funzione di Default */
     default:
      printf ("Scelta non valida!!!\nRiprovare\n\n" );
      break;
   }
  }
}
```

```
/* Programma per lagestione di un Albero Binario                    */
/* Questo è la libreria utente.                                     */
/*                                                                  */
/*                          Programma sviluppato da                 */
/*                                                                  */
/*                            Giovanni DI CECCA                     */
/*                                                                  */
/*                          http://www.dicecca.net                  */


/* Funzione per il nome del programma da passare al Main */
void nomeprogramma  ()
{
   printf ("Programma per la gestione di un Albero Binario"   );
   printf ("\n\nby Giovanni DI CECCA - http://www.dicecca.net\n\n\n"    );
}

void menu_io ()
{
  /* Carica la libreria che contiene il nome del programma */
  nomeprogramma ();

  /* Lista dei comandi dell'albero */
  printf  ("Menu dei comandi" );
  printf  ("\nI -> Inserimento di un elemento"  );
  printf  ("\nR -> Ricerca di un elemento"  );
  printf  ("\nS -> Stampa In-Order dell'Albero Binario"   );
  printf  ("\nP -> Stampa Pre-Order dell'Albero Binario"   );
  printf  ("\nO -> Stampa Post-Order dell'Albero Binario"   );
  printf  ("\nC -> Conteggio del numero di nodi"  );
  printf  ("\nL -> Profondit… dell'Albero Binario"  );
  printf  ("\nQ -> Uscita dal programma \n"  );
}

/* Definizione del tipo Booleano logico */
typedef enum {false,true} logical;

/* Definizione della struttura nodo_tree */
struct nodo_tree  {
    int dato;
    struct nodo_tree  *sx;
    struct nodo_tree  *dx;
};

/* Dichiarazione del tipo strutturato nodo_tree con tree */
typedef struct nodo_tree  tree;

/* Procedura per l'inizializzazione dell'albero */
tree *init_albero  (int el)
{
 /* Definizione delle variabili mediante la struttura tree delle variabili */
 /* per la definizione della Root dell'albero */
 tree *a;

 /* Allocazione a RunTime del puntatore per la creazione dell'array dinamico */
 a=(tree *)malloc(sizeof(tree));

 /* Inizializzazione dei puntatori della struttura */
 a->dato=el;
 a->sx=NULL;
 a->dx=NULL;

 /* Ritorna il valore di a */
 return a;
}

/* Funzione per l'inserimento di un nodo nell'albero */
logical  insert_node  (tree *albero,int k)
{

 /* Definizione delle variabili mediante la struttura tree delle variabili */
 /* puntatore predecessore e a */
 tree *predecessore,*a;

 /* Ciclo while per la ricerca della foglia per inserire il nodo */
```

```
 while ((albero != NULL)&&(albero->dato != k))
 {

  predecessore =albero;

  /* Ricerca del punto libero nell'albero di destra e di sinistra */
  if (k > albero->dato)
   albero =albero->dx;

  else
   albero =albero->sx;
 }

 /* Se albero è diverso da NULL, allora ritorna il valore Booleano falso, */
 /* alternativamente, alloca lo spazio o a destra o a sinistra del nodo */
 if (albero != NULL)
  return false;
 else
 {
  a=(tree *)malloc(sizeof(tree));
  a->dato=k;
  a->sx=NULL;
  a->dx=NULL;
  if (k > predecessore->dato)
   predecessore->dx=a;
  else
   predecessore->sx=a;
  return true;
 }
}


/* Procedura per la stampa in In Order, prima dela ramo sinistro e poi destro */
void stampa_in_order  (tree *root)
{
 if (root != NULL)
 {
  stampa_in_order  (root->sx);
  printf ("%d\n",root->dato);
  stampa_in_order  (root->dx);
 }
}

/* Procedura per la stampa in Pre Order, prima dela ramo sinistro e poi destro */
void stampa_pre_order  (tree *root)
{
 if (root != NULL)
 {
  printf ("%d\n",root->dato);
  stampa_pre_order  (root->sx);
  stampa_pre_order  (root->dx);
 }
}

/* Procedura per la stampa in Post Order, prima dela ramo sinistro e poi destro */
void stampa_post_order  (tree *root)
{
 if (root != NULL)
 {
  stampa_post_order  (root->sx);
  stampa_post_order  (root->dx);
  printf ("%d\n",root->dato);
 }
}


/* Funzione per la ricerca di un elemento nell'albero, mediante l'uso della */
/* procedura logical */
logical binary_tree_search  (tree *root,int key)
{
 if (root == NULL)
  return false;
 else
  if (root->dato==key)
   return true;
```

```
  else
   if (root->dato < key)
    return binary_tree_search  (root->dx,key);
   else
    return binary_tree_search  (root->sx,key);
}


/* Funzione che conta i nodi presenti nell'albero */
int conta_nodi  (tree *root)
{
 int ns,nd;
 if (root == NULL)
  return 0;
 ns=conta_nodi  (root->sx);
 nd=conta_nodi  (root->dx);
 return (ns+nd+1);
}

/* Funzione che calcola la distanza del livello più profondo nell'albero */
int profondita  (tree *root)
{
 int ps,pd,max;
 if (root == NULL)
  return 0;
 ps=profondita  (root->sx);
 pd=profondita  (root->dx);
 if (ps > pd)
  max=ps;
 else
  max=pd;
 return (max+1);
}
```