ALGORITMI E STRUTTURE DATI

Progetto sessione estiva 2015/2016

Angelo Sasso

Matricola 263143

Specifica del problema

Si supponga di dover elaborare delle informazioni di input che rappresentano il personale che lavora in una determinata azienda. Scrivere un programma ANSI C che esegue le seguenti elaborazioni:

Acquisisce un file di testo, il cui formato prevede un certo numero di righe (record) ognuna delle quali contiene: cognome e nome della persona, codice identificativo numerico, data di assunzione, stipendio, categoria (impiegato, dirigente, operaio). I vari campi di ogni riga sono separati da tabulazione oppure da spazio.

- *Inserire i dati in un'opportuna struttura dati.*
- *Permettere all'utente di inserire (da tastiera) un nuovo record relativo a nuovo personale.*
- *Permettere all'utente di cancellare un record, selezionandolo opportunamente da tastiera.*
- *Permettere all'utente di ricercare il record relativo ad un determinato lavoratore, selezionandolo opportunamente da tastiera.*
- *Dato un numero interno i inserito dall'utente, restituire il record relativo all'i-esimo elemento più piccolo in base allo stipendio.*

Per quanto riguarda l'analisi teorica si deve fornire la complessità corrispondente ad ognuna delle seguenti operazioni: inserimento di un nuovo record, cancellazione di un record, ricerca di un record, selezione i-esimo record stipendiale. Oltre all'analisi teorica della complessità si deve effettuare uno studio sperimentale della stessa. In particolare, si deve operare generando casualmente un numero N di record da fornire in input al programma. L'analisi sperimentale deve quindi valutare la complessità al variare del parametro N per le fasi di: inserimento, cancellazione, ricerca, selezione i-esimo record.

Analisi del problema

Input:

L’insieme degli input del programma è contenuto nel file “persone.txt”, il quale, dovrebbe essere contenuto all’interno della cartella del file eseguibile.

Gli input sono:

-Il cognome,

-il nome,

-il codice identificativo,

-la data di assunzione,

-lo stipendio,

-la categoria,

-il codice identificativo da cercare,

-un intero per la ricerca dell’elemento i-esimo.

-il numero degli elementi da generare casualmente

Ognuno di questi campi è poi inserito nel record al fine di procedere con le manipolazioni dei dati.

Output:

L’insieme degli output è dato invece dai vari tipi di stampa a video e su file:

-La stampa su file di nuovi record.

-La rimozione da file di selezionati record.

-La stampa a video dei dati presenti sul file “persone.txt”.

-La stampa a video dei dati considerando l’aggiunta o la rimozione di almeno uno di essi.

-La stampa a video del record contenente il codice identificativo cercato.

-La stampa a video dell’i-esimo elemento in base allo stipendio.

Progettazione dell'algoritmo

Ho inserito una validazione iniziale in modo che il file da cui si leggono i dati sia esclusivamente del tipo specificato “.txt”. Deve presentare 6 colonne di dati, una per ogni campo di informazione del record divisi tra loro mediante spazi o tabulazioni: per ogni riga si andranno a leggere le informazioni di ogni persona fino alla fine del file.

Struttura dati dinamica utilizzata:

{

Cognome (stringa di caratteri);

Nome (stringa di caratteri);

Codice identificativo (numero intero);

Data di assunzione (stringa di caratteri);

Stipendio (numero intero);

Categoria (stringa di caratteri);

}

I dati vengono acquisiti dal file (se presenti) oppure generati automaticamente con stringhe di caratteri e interi casuali a discrezione dell’utente.

Si tratta di un array di strutture allocato dinamicamente scelto per la semplicità di implementazione, per un utilizzo accurato della memoria e per una più semplice indicizzazione.

L’interfaccia dell’utente si presenta in modo intuitivo, viene ciclata finché l’utente non decide di uscire (e quindi salvare automaticamente i dati su file) e presenta la possibilità di compiere più azioni ad ogni avvio del programma.

Main()

Nella funzione main ho optato per uno switch e controllato che permettesse all'utente di scegliere tra le funzioni disponibili senza uscire dal programma quando si inseriscono per sbaglio valori non corrispondenti alle scelte possibili.

Prima di dare la possibilità all'utente di scegliere si procede all'acquisizione da file, se presente mediante "acquisire_dati": nel caso in cui il file non fosse presente nella cartella verrebbe segnalato un errore e si uscirebbe dal programma, nel caso di un file vuoto invece si procederebbe ad una stampa a video vuota, nel caso di un file contenente dati invece si stamperebbero tutti i dati istantaneamente.

Il controllo per il tipo di input dato dall'utente viene effettuato svuotando il buffer nel caso di un return da parte di scanf = 0 e stampando la stringa "Azione non valida." .

Il tutto viene ciclato fino a quando l'utente non inserisce un valore accettabile.

Ora che ci siamo accertati che l'utente abbia inserito un valore accettabile si potrà procedere al riconoscimento di tale valore e al suo utilizzo all'interno dello switch.

Case 1: "inserimento_dati"

Caso in cui l'utente sceglie di aggiungere dei dati al file vuoto oppure ad un file che ne contiene già.

Il primo passo da fare è aprire il file in append mode ("a"), successivamente si chiede all'utente il numero di record da registrare e si apre un ciclo for che termina al raggiungimento del numero inserito dall'utente;

Si alloca spazio per ogni campo della struttura e si procede all'acquisizione per ogni campo con una successiva realloc che provvederà a utilizzare unicamente lo spazio occupato dalla stringa o dall'intero inseriti.

Concluso il ciclo si procede ad una nuova acquisizione con i dati completi.

Case 2: "cancellazione_dati"

In questo caso avviene automaticamente una stampa a video con gli stessi dati, però indicizzati in modo da facilitare e velocizzare la scelta dell'utente.

Al momento della scelta dell'utente si procede alla liberazione dello spazio allocato per ogni campo della struttura scelta e alla sua inizializzazione come puntatore a NULL per poi andare a riscrivere sul fine tutti i dati presenti nell'array di strutture meno quello scelto dall'utente.

Case 3: "trova_elemento"

In questo caso viene chiesto all'utente di inserire la chiave da ricercare e vengono scansionati i campi dei codici identificativi ; La funzione sfrutta un contatore per restituire il record relativo alla chiave scelta o per restituire un avviso se quella chiave non è presente nell'array.

Case 4: “ordinamento”

Quando l’utente sceglie questa funzione si avvia in automatico uno scorrimento del file per avere il numero totale di dati (i).

Si chiede all’utente di inserire un numero intero.

Ho scelto di implementare l’algoritmo Insertion Sort che non è l’algoritmo più efficiente per numeri elevati data la complessità asintotica O(n^2), ma gode di una buona semplicità di implementazione e risulta molto più vantaggioso per insiemi di partenza quasi ordinati e per la salvaguardia degli stessi.

L’algoritmo confronta gli interi per l’ordinamento degli stipendi.

Durante l’ordinamento quando avviene uno scambio si procede con tutti i campi della struttura e non solo quello da ordinare.

Infine la funzione restituisce una stampa a video dell’i-esimo record più piccolo relativo allo stipendio mentre se l’intero inserito non corrisponde a nessun record verrà stampato un avviso.

Insertsort:

```
{
        < Per ogni j = 1, 2, 3, ... i – 1 ripeti: >
                < tmp prende il valore di *persona[ i ] >;
                < Poni k = j – 1 >
                < Mentre k >= 0 e *persona[ k ] > tmp >
                        < *persona[ k+1 ] prende il valore di *persona[ k ] >;
                        < k si riduce di 1 >;
                <*persona[ k+1 ] prende il valore di tmp >;
}
```

Case 5: “random”

Per l’analisi sperimentale degli algoritmi ho implementato una funzione che crea tanti dati quanti vengono decisi dall’utente con stringhe e numeri casuali.

Il funzionamento è garantito dalla funzione rand() implementata in 5 cicli for: il primo termina con il numero di elementi scelti dall’utente decidendo il numero di righe da stampare sul file; il ciclo più interno termina al raggiungimento delle iterazioni, una per ogni campo di stringhe casuali, l'altro invece consente di generare numeri interi casuali.

La funzione “pausa” invece è stata implementata per far sì che l’utente confermi premendo invio prima di proseguire con stampe o con funzioni.

Strutture utilizzate

```
typedef struct record_t
{
        char    *cognome;
        char    *nome;
        int     *codice;
        char    *data;
        int     *stipendio;
        char    *categoria;
}
```

Implementazione delle funzioni:

“acquisire_dati”

In questo caso si leggono i dati da file e si salvano nell’array di strutture.

Un accorgimento importante riguarda l’allocazione di memoria: viene inizializzata per un valore abbastanza alto da poter contenere i campi per poi essere riallocata durante il ciclo.

Per le stringhe si usa una riallocazione per *strlen()*, ovvero leggendo la lunghezza esatta, mentre per gli interi si alloca sempre lo spazio necessario a memorizzare un solo intero.

```
for(i = 0; (!feof(file)) ; i++)
{
    persona                 = realloc(persona, sizeof(record_t)*(i+1));
    persona[i].cognome      = (char *)malloc(30 * (sizeof(char)));
    persona[i].nome         = (char *)malloc(30 * (sizeof(char)));
    persona[i].codice       = (int  *)malloc( 1 * (sizeof(int )));
    persona[i].data         = (char *)malloc(30 * (sizeof(char)));
    persona[i].stipendio    = (int  *)malloc( 1 * (sizeof(int )));
    persona[i].categoria    = (char *)malloc(30 * (sizeof(char)));

    if (fscanf(file, "%s", persona[i].cognome) != EOF)
    {
        persona[i].cognome =
        realloc(persona[i].cognome, sizeof((strlen(persona[i].cognome))));

        printf("%-15s", persona[i].cognome);
    }

    if (fscanf(file, "%s", persona[i].nome) != EOF)
    {
        persona[i].nome =
        realloc(persona[i].nome, sizeof((strlen(persona[i].nome))));

        printf("%-10s\t", persona[i].nome);
    }

    if (fscanf(file, "%d", persona[i].codice) != EOF)
    {

        printf("%d\t", *persona[i].codice);
    }

    if (fscanf(file, "%s", persona[i].data) != EOF)
    {
        persona[i].data =
        realloc(persona[i].data, sizeof((strlen(persona[i].data))));

        printf("%-10s\t", persona[i].data);
    }

    if (fscanf(file, "%d", persona[i].stipendio) != EOF)
    {
        printf("%d\t", *persona[i].stipendio);
    }

    if (fscanf(file, "%s", persona[i].categoria) != EOF)
    {
        persona[i].categoria =
        realloc(persona[i].categoria, sizeof((strlen(persona[i].data))));

        printf("%-10s\n", persona[i].categoria);
    }
```

“Main”

Per quanto riguarda invece la funzione main(), è stata organizzata con uno switch per un buon impatto visivo e principalmente per comodità.

Alla fine di ogni funzione è sempre presente l’acquisizione dei dati al fine di memorizzare il tutto automaticamente senza il bisogno di interpellare l’utente.

```
switch(azione)
{
    case 1: inserimento_dati(persona);
            acquisire_dati(persona);

            break;

    case 2: cancellazione_dati(persona);
            acquisire_dati(persona);

            break;

    case 3: trova_elemento(persona, elem);
            acquisire_dati(persona);

            break;

    case 4: ordinamento();
            acquisire_dati(persona);

            break;

    case 5: random();
            acquisire_dati(persona);

            break;


    case 0:
            break;

    default:
            break;
}
```

“inserimento_dati”

Questa funzione è molto simile alla funzione di acquisizione dati, solo che in questo caso si chiede all’utente quanti record desidera aggiungere e si salvano sul file (precedentemente aperto in append mode) per poi riacquisire il tutto una volta usciti dalla funzione.

La complessità di questa funzione è lineare Θ(n):

Θ((n-1)+Θ(9+(30+1)+Θ(30+1)+Θ(30+1)+Θ(1+1)+Θ(30+1)+Θ(1+1)+Θ(30+1)+Θ(1+1+1))=

Θ(n+139)

```
for(i = 0; (i < k) ; i++)
{
    persona             = realloc(persona, sizeof(record_t)*(i+1));
    persona[i].cognome      = (char *)malloc(30 * (sizeof(char)));
    persona[i].nome         = (char *)malloc(30 * (sizeof(char)));
    persona[i].codice           = (int  *)malloc( 1 * (sizeof(int )));
    persona[i].data         = (char *)malloc(30 * (sizeof(char)));
    persona[i].stipendio        = (int  *)malloc( 1 * (sizeof(int )));
    persona[i].categoria        = (char *)malloc(30 * (sizeof(char)));

    printf("Digitare COGNOME, NOME, CODICE IDENTIFICATIVO, DATA ASSUNZIONE, STIPENDIO e CATEGORIA\n");
    printf("della nuova persona separati da tabulazione o da spazio: \n\n");

    if (scanf("%s", persona[i].cognome) != EOF)
    {
        persona[i].cognome =
        realloc(persona[i].cognome, sizeof((strlen(persona[i].cognome))));

        fprintf(file, "%s\t", persona[i].cognome);
    }

    if (scanf("%s", persona[i].nome) != EOF)
    {
        persona[i].nome =
        realloc(persona[i].nome, sizeof((strlen(persona[i].nome))));

        fprintf(file, "%s\t", persona[i].nome);
    }

    if (scanf("%d", persona[i].codice) != EOF)
    {

        fprintf(file, "%d\t", *persona[i].codice);
    }

    if (scanf("%s", persona[i].data) != EOF)
    {
        persona[i].data =
        realloc(persona[i].data, sizeof((strlen(persona[i].data))));

        fprintf(file, "%s\t", persona[i].data);
    }

    if (scanf("%d", persona[i].stipendio) != EOF)
    {
        fprintf(file, "%d\t", *persona[i].stipendio);
    }

    if (scanf("%s", persona[i].categoria) != EOF)
    {
        persona[i].categoria =
        realloc(persona[i].categoria, sizeof((strlen(persona[i].categoria))));
```

“cancellazione_dati”

In questa funzione è importante liberare lo spazio riservato all’elemento che si desidera eliminare dalla memoria per poi “saltarlo” durante la scrittura su file.

La complessità di questa funzione è quadratica Θ(n²):

Θ(1+1+1)+Θ(n*(5+7))+Θ(n*15)+Θ(n+4)+Θ(1)+Θ(n-1)+Θ(30+30+30+1+30+1+30+20)+Θ(10)=

Θ(n*13)+Θ(n*15)+Θ(3)+Θ(n+5)+Θ(n-1)+Θ(181)+Θ(10)=

Θ(194)+Θ(n*195)+Θ(n+4)= Θ(n*195+198)

```
k = k - 1;

free(persona[k].cognome);
persona[k].cognome = NULL;

free(persona[k].nome);
persona[k].nome = NULL;

free(persona[k].codice);
persona[k].codice = NULL;

free(persona[k].data);
persona[k].data = NULL;

free(persona[k].stipendio);
persona[k].stipendio = NULL;

free(persona[k].categoria);
persona[k].categoria = NULL;


   file = fopen("persone.txt", "w");

   for (j = 0 ; (j < (i - 1)); j++)
   {
       if (j == k);

       else
       {
           fprintf(file, "%-15s",      persona[j].cognome);
           fprintf(file, "%-10s\t",    persona[j].nome);
           fprintf(file, "%d\t",       *persona[j].codice);
           fprintf(file, "%-10s\t",    persona[j].data);
           fprintf(file, "%d\t",       *persona[j].stipendio);
           fprintf(file, "%-10s\n",    persona[j].categoria);
       }
   }
```

“trova_elemento”

Dopo aver acquisito i campi dell’array si procede a confrontare i campi relativi al codice identificativo con la chiave “elem” inserita dall’utente.

Se la chiave corrisponde ad un intero contenuto in campo “codice”, il contatore viene incrementato e si provvede a stampare a video i valori del record corrispondente, altrimenti viene restituito un avviso.

La complessità di questa funzione è lineare Θ(n):

Θ(1)+Θ(1)+Θ(1)+Θ(1)+Θ(n*(30+30+1+30+1+30+1)+Θ(1)+Θ(1)+Θ(5)+Θ(1)+Θ(1+2)+Θ(1)+Θ(n-1)+Θ(1+9)+Θ(1)+Θ(1)+Θ(1)+Θ(1)+Θ(4))=

Θ(34)+Θ(n*(153)+Θ(n-1))=

Θ(n-33)+Θ(n*(153))

```
    k = i-1;

    printf("\nInserire il CODICE IDENTIFICATIVO da cercare\n");
    if (scanf("%d",&elem)==1);


    for(k=0; k<i; k++)

    if(*persona[k].codice == elem)
    {
        trovato++;
        printf("\nIl CODICE IDENTIFICATIVO cercato '%d' si trova in posizione %d\n",elem,k+1);
        printf("\nQuesto è il record relativo:\n\n");
        printf("%s\t", persona[k].cognome);
        printf("%s\t", persona[k].nome);
        printf("%d\t", *persona[k].codice);
        printf("%s\t", persona[k].data);
        printf("%d\t", *persona[k].stipendio);
        printf("%s\n", persona[k].categoria);


     }

    if(trovato==0)

    printf("\nl'elemento '%d' non esiste nell'array\n",elem);

    pausa();
```

“ordinamento”

In questo caso si utilizza l’algoritmo insertsort con gli argomenti stipendio.

Questo algoritmo è fortemente sconsigliato per l'utilizzo di un numero molto grande di dati, ma finché essi sono nell'ordine delle migliaia non si avranno problemi di tempo o risorse.

La complessità di questa funzione è quadratica $\Theta(n^2)$:

$\Theta(1)+ \Theta(1)+ \Theta(n-1)*(\Theta(2)+ \Theta(7)+ \Theta(8)+ \Theta(4)+ \Theta(n*7)+ \Theta(3))=$

$\Theta(2)+ \Theta(n-1)* (\Theta(24)+ \Theta(n*7))= \Theta(2)+ \Theta(n-1)*(\Theta(22)+ \Theta(n*7))$

```
j = 0;

for (j = 1; j < i; j++)
{
    tmp1 = persona[j].cognome;
    tmp2 = persona[j].nome;
    tmp3 = *persona[j].codice;
    tmp4 = persona[j].data;
    tmp5 = *persona[j].stipendio;
    tmp6 = persona[j].categoria;

    k = j - 1;

    while(k >= 0 && (*persona[k].stipendio > tmp5) )
    {
        persona[k+1].cognome        = persona[k].cognome;
        persona[k+1].nome       = persona[k].nome;
        *persona[k+1].codice        = *persona[k].codice;
        persona[k+1].data       = persona[k].data;
        *persona[k+1].stipendio     = *persona[k].stipendio;
        persona[k+1].categoria      = persona[k].categoria;

        k--;
    }

    persona[k+1].cognome        = tmp1;
    persona[k+1].nome       = tmp2;
    *persona[k+1].codice        = tmp3;
    persona[k+1].data       = tmp4;
    *persona[k+1].stipendio     = tmp5;
    persona[k+1].categoria      = tmp6;
}
```

“random”

Per la generazione casuale di dati vengono generate delle serie di stringhe separate da tabulazione e un intero che verranno scritti su file e successivamente quindi acquisiti.

La funzione ha complessità quadratica: $\Theta(n^2)$.

```
for(k = 0; k < elem; k++)
{

    for(j = 0; j < 2; j++)
    {

        for(i = 0; i < 8 ; i++)
        {
            fprintf(file, "%c", ('a'+rand()%('z'-'a')));

        }
        fprintf(file,"\t");

    }

        num = rand()%1000000;
        fprintf(file, "%d\t", ('1'+rand()%( num)));

        for(l = 0; l < 8 ; l++)
        {
            fprintf(file, "%c", ('a'+rand()%('z'-'a')));

        }
        fprintf(file,"\t");

        num = rand()%1000000;
        fprintf(file, "%d\t", ('1'+rand()%( num)));

        for(t = 0; t < 8 ; t++)
        {
            fprintf(file, "%c", ('a'+rand()%('z'-'a')));

        }
        fprintf(file,"\t");



    fprintf(file,"\n");
```

Testing del programma

Segue la schermata iniziale nel caso in cui il file contiene gli esempi come da direttiva.

```
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 *#*#*#*#*#*#*#                 #*#*#*#*#*#*#*
 #*             LISTA PERSONALE              *#
 #*#*#*#*#*#*#*                 *#*#*#*#*#*#*#
 *#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*

Cognome1       Nome1            192     20/07/2011      1700    impiegato
Cognome2       Nome2            15      01/10/1998      2560    dirigente
Cognome3       Nome3            140     02/01/2014      1400    operaio


 1) AGGIUNGI PERSONE

 2) RIMUOVI PERSONA

 3) RICERCA PERSONE

 4) RICERCA i-esimo STIPENDIO

 5) GENERA DATI

 0) ESCI

- Selezionare la funzione da eseguire > 
```

Nel caso in cui l'utente non inserisce un intero valido viene riproposta la "scanf" e si continua fino all'acquisizione di un'opzione valida.

```
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 *#*#*#*#*#*#                 #*#*#*#*#*#*
 #*            LISTA PERSONALE            *#
 #*#*#*#*#*#*                 *#*#*#*#*#*#
 *#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*

Cognome1       Nome1           192     20/07/2011      1700    impiegato
Cognome2       Nome2           15      01/10/1998      2560    dirigente
Cognome3       Nome3           140     02/01/2014      1400    operaio

 1) AGGIUNGI PERSONE

 2) RIMUOVI PERSONA

 3) RICERCA PERSONE

 4) RICERCA i-esimo STIPENDIO

 5) GENERA DATI

 0) ESCI


- Selezionare la funzione da eseguire > gh
Azione non valida.

- Selezionare la funzione da eseguire > █
```

Nel caso in cui si voglia procedere con l'aggiunta di nuovi record si presenterà questa schermata:

```
- Selezionare la funzione da eseguire > 1
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*              LISTA PERSONE            *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*             AGGIUNGI PERSONE          *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#


- Inserire il numero delle persone che si desidera registrare > 2
Digitare COGNOME, NOME, CODICE IDENTIFICATIVO, DATA ASSUNZIONE, STIPENDIO e CATEGORIA
della nuova persona separati da tabulazione o da spazio:

■
```

Inserendo tutti i campi per ogni record richiesto avverrà il salvataggio su file e successivamente il refresh dell'elenco.

```
#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
#*                  RIMUOVI PERSONE                  *#
#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#

1       Cognome1        Nome1           192     20/07/2011      1700    impiegato
2       Cognome2        Nome2           15      01/10/1998      2560    dirigente
3       Cognome3        Nome3           140     02/01/2014      1400    operaio
4       Cognome4        Nome4           179     12/12/1993      300     studente
5       Cognome5        Nome5           13      08/78/200       1400    operaio


- Selezionare il numero della persona per eliminare il relativo record >        5


Refresh elenco

Cognome1        Nome1           192     20/07/2011      1700    impiegato
Cognome2        Nome2           15      01/10/1998      2560    dirigente
Cognome3        Nome3           140     02/01/2014      1400    operaio
Cognome4        Nome4           179     12/12/1993      300     studente


 1) AGGIUNGI PERSONE

 2) RIMUOVI PERSONA

 3) RICERCA PERSONE

 4) RICERCA i-esimo STIPENDIO

 5) GENERA DATI

 0) ESCI


- Selezionare la funzione da eseguire > 
```

Nel caso in cui l'utente voglia rimuovere un record:

```
- Selezionare la funzione da eseguire > 2
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                  LISTA PERSONE                    *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                  RIMUOVI PERSONE                  *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#

1       Cognome1        Nome1           192     20/07/2011      1700    impiegato
2       Cognome2        Nome2           15      01/10/1998      2560    dirigente
3       Cognome3        Nome3           140     02/01/2014      1400    operaio
4       Cognome4        Nome4           179     12/12/1993      300     studente
5       Cognome5        Nome5           13      08/78/200       1400    operaio


- Selezionare il numero della persona per eliminare il relativo record >
```

Se l’utente non inserisce un numero valido:

```
- Selezionare la funzione da eseguire > 2
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                  LISTA PERSONE                                          *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                 RIMUOVI PERSONE                                         *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#

1       Cognome1        Nome1           192     20/07/2011      1700    impiegato
2       Cognome2        Nome2           15      01/10/1998      2560    dirigente
3       Cognome3        Nome3           140     02/01/2014      1400    operaio


- Selezionare il numero del record da eliminare > ghjjk
Azione non valida.


- Selezionare il numero del record da eliminare > 
```

Una volta selezionata un’opzione valida si procede con la rimozione dell’elemento, il salvataggio e l’acquisizione dei record.

```
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                 RIMUOVI PERSONE                                         *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#

1       Cognome1        Nome1           192     20/07/2011      1700    impiegato
2       Cognome2        Nome2           15      01/10/1998      2560    dirigente
3       Cognome3        Nome3           140     02/01/2014      1400    operaio
4       Cognome4        Nome4           179     12/12/1993      300     studente
5       Cognome5        Nome5           13      08/78/200       1400    operaio


- Selezionare il numero della persona per eliminare il relativo record >        5


Refresh elenco

Cognome1        Nome1           192     20/07/2011      1700    impiegato
Cognome2        Nome2           15      01/10/1998      2560    dirigente
Cognome3        Nome3           140     02/01/2014      1400    operaio
Cognome4        Nome4           179     12/12/1993      300     studente


 1) AGGIUNGI PERSONE

 2) RIMUOVI PERSONA

 3) RICERCA PERSONE

 4) RICERCA i-esimo STIPENDIO

 5) GENERA DATI

 0) ESCI


- Selezionare la funzione da eseguire > 
```

Se l’utente desidera ricercare un record relativo ad un determinato codice identificativo ed esso viene trovato sarà stampato a video l’intero record.

```
- Selezionare la funzione da eseguire > 3
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                  LISTA PERSONE           *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                RICERCA PERSONE           *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#


Inserire il CODICE IDENTIFICATIVO da cercare
15

Il CODICE IDENTIFICATIVO cercato '15' si trova in posizione 2

Questo è il record relativo:

Cognome2        Nome2   15      01/10/1998      2560    dirigente


Premere <INVIO> per continuare...
```

Nel caso in cui il codice identificativo inserito non è stato trovato in alcun record:

```
- Selezionare la funzione da eseguire > 3
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                  LISTA PERSONE           *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                RICERCA PERSONE           *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#


Inserire il CODICE IDENTIFICATIVO da cercare
567

Il CODICE IDENTIFICATIVO '567' non esiste nell'array


Premere <INVIO> per continuare...
```

La funzione relativa allo stipendio i-esimo più basso si presenta così nel caso in cui è stato inserito un intero valido:

```
- Selezionare la funzione da eseguire > 4
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*                  LISTA PERSONE            *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*          RICERCA i-esimo STIPENDIO      *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#


Inserire un intero 'i' >        3

Tempo necessario a riordinare la struttura per STIPENDIO : 0.000002


Premere <INVIO> per continuare...


Il record relativo al '3' elemento per stipendio è:

3)      Cognome2                Nome2           15              01/10/1998              2560            dirigente

Cognome1        Nome1           192     20/07/2011      1700    impiegato
Cognome2        Nome2           15      01/10/1998      2560    dirigente
Cognome3        Nome3           140     02/01/2014      1400    operaio


 1) AGGIUNGI PERSONE

 2) RIMUOVI PERSONA

 3) RICERCA PERSONE
```

Se l’intero inserito supera la dimensione dell’array viene stampato un avviso.

```
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
 #*           RICERCA i-esimo STIPENDIO       *#
 #*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#


Inserire un intero 'i' >        6

Tempo necessario a riordinare la struttura per STIPENDIO : 0.000003


Premere <INVIO> per continuare...


Non esiste il record numero '6'.
```

Se invece l’utente desidera generare dati casualmente, verrà mostrata questa schermata:

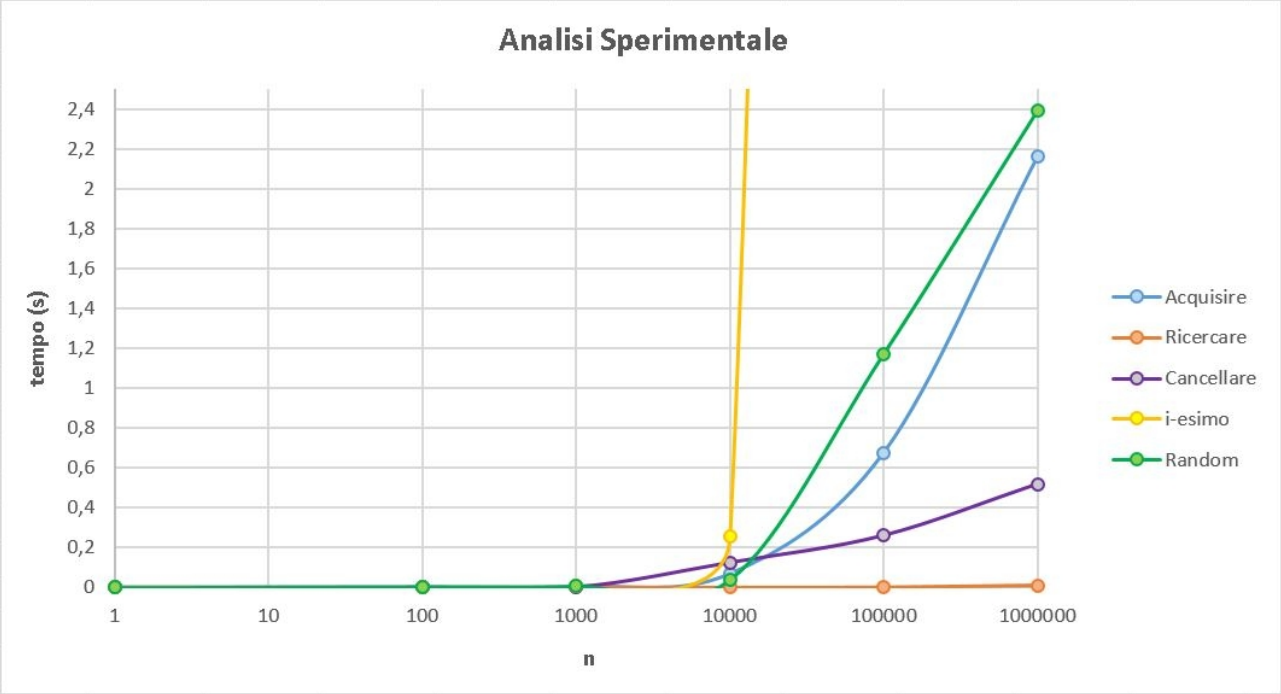
```
- Selezionare la funzione da eseguire > 5

- Digitare il numero di elementi che si desidera generare >     5

Tempo necessario per generare i RECORD : 0.000088


Premere <INVIO> per continuare...


ilcpsklr        yvmcpjnb        16195   wllsrehf        180746  rkecwitr
sglrexvt        jmxypunb        142044  gxmuvgfa        104656  lfvenhyu
huorjosa        mibdnjdb        213932  hkbsombl        9308    uujdrbwc
rrcgbflq        pottpegr        281980  gajcrgwd        208464  gitydvhe
dtusippy        vxsuvbvf        73051   odqasajo        184043  mgsqcpjl

Tempo necessario per l'acquisizione dei record: 0.000237

        Premere INVIO per contniuare >
```

Infine si procede alla chiusura del programma con le informazioni del programmatore.

```
 1) AGGIUNGI PERSONE

 2) RIMUOVI PERSONA

 3) RICERCA PERSONE

 4) RICERCA i-esimo STIPENDIO

 5) GENERA DATI

 0) ESCI


- Selezionare la funzione da eseguire > 0

Grazie per aver utilizzato questo programma!

--------------------------------------
--------------------------------------
 Programma creato da
 Angelo Sasso
 a.sasso4@campus.uniurb.it
 PROGETTO ALGORITMI E STRUTTURE DATI
--------------------------------------
--------------------------------------
```

VALUTAZIONE DELLA COMPLESSITÀ

Per l'analisi sperimentale sono state poste sotto osservazione (al variare dei dati 'n') le funzioni di generazione di dati, di ricerca tramite codice identificativo, di selezione i-esimo stipendio, la funzione di cancellazione di dati a l'acquisizione degli stessi;



ne risultano i seguenti valori che confermano l'analisi teorica del codice:

*n=**1**:* *Acquisizione(**0.000042**) - Ricerca(**0.000038**) - Cancellazione(**0.000144**) - i-esimo(**0.000002**) - rand(**0.000029**)*

*n=**100**:* *Acquisizione(**0.001195**) - Ricerca(**0.000041**) - Cancellazione(**0.000506**) - i-esimo(**0.000032**) - rand(**0.003115**)*

*n=**1000**:* *Acquisizione(**0.005454**) - Ricerca(**0.000058**) - Cancellazione(**0.002198**) - i-esimo(**0.005227**) - rand(**0.003170**)*

*n=**10.000**:* *Acquisizione(**0.067612**) - Ricerca(**0.000126**) - Cancellazione(**0.005962**) - i-esimo(**0.254083**) - rand(**0.067716**)*

*n=**100.000**:* *Acquisizione(**0.675378**) - Ricerca(**0.011060**) - Cancellazione(**0.260170**) - i-esimo(**44.810073**) - rand(**1.170169**)*

*n=**1.000.000**:* *Acquisizione(**2.161475**) - Ricerca(**0.049895**) - Cancellazione(**2.644409**) – i-esimo(**ND**) – rand(**11.362071**)*