1. Esercizi risolti

Esempio 1

Scrivere un programma per la registrazione dei voti finali di studenti, tenendo conto che alla valutazione viene associato un giudizio nel seguente modo: insufficiente (<6), sufficiente (6), discreto (7), buono (8), ottimo (>=9).
Il lettore verificherà facilmente che una soluzione può essere quella indicata nel programma seguente.

ValutazioneStudenti.c

```
#include <iostream>
using namespace std;
#define NSTUD 5
typedef enum {insufficiente, sufficiente, discreto, buono, ottimo} esito;
typedef struct                          /* struttura dati... */
{                                       /* ...che rappresenta uno... */
  string cognome,                       /* ... studente */
         nome;
         float punti;                   /* voto */
  esito  valutazione;                   /* trasforma voto in giudizio */
} studente;
void leggi_dati(studente classe[ ]);    /* legge array classe[ ] */
void leggi_dati_alunno(studente &stud); /* legge dati singolo alunno */
void elenco_promossi(studente classe[ ]); /* stampa elenco dei promossi */

int main(int argc, char *argv[])
{
   studente classe[NSTUD];              /* array di studenti */
   leggi_dati(classe);                  /* acquisisce i dati */
   elenco_promossi(classe);             /* valuta i promossi */
   system("pause");
   return EXIT_SUCCESS;
}
void leggi_dati(studente classe[])
{  int i;
  for(i=0; i<NSTUD; i++)                /* per ogni studente, legge i...*/
    leggi_dati_alunno(classe[i]);       /* ...dati in una struct passata...*/
}                                       /* ...per indirizzo */
void leggi_dati_alunno(studente &stud)  /* stud e' passata per indirizzo...*/
{   cout<<"Cognome..: ";                /* ...poichè deve essere ...*/
    cin>>stud.cognome;                  /* ...restituita modificata */
    cout<<"Nome.....: ";
    cin>>stud.nome;
    cout<<"Voto.....: ";
    cin>>stud.punti;
    if (stud.punti < 6)
      stud.valutazione=insufficiente;
    else
      if (stud.punti < 7)
        stud.valutazione=sufficiente;
      else
        if (stud.punti < 8)
          stud.valutazione=discreto;
        else
          if (stud.punti < 9)
            stud.valutazione=buono;
          else
            stud.valutazione=ottimo;
```

```
}
void elenco_promossi(studente classe[])
{
   int i;
  cout<<"*** Elenco promossi ***"<<endl;
  for (i=0;i<NSTUD;i++)
    if (classe[i].valutazione!=insufficiente)
       cout<<classe[i].cognome<<"  "<<classe[i].nome<<endl;
}
```

OSSERVAZIONI:

- la funzione *leggi_dati()* legge i dati dell'intera classe con un ciclo che istanzia *leggi_dati_alunno()* per ogni alunno, passando la **struct** dell'alunno per indirizzo;

Esempio 2

Scrivere un programma che visualizzi la data e l'ora del sistema.
Il programma necessita dell'uso della seguente **struct** di sistema

```
struct tm
{       int    tm_sec;          /* secondi */
        int    tm_min;          /* minuti */
        int    tm_hour;         /* ore */
        int    tm_mday;         /* giorno del mese */
        int    tm_mon;          /* mese */
        int    tm_year;         /* anno */
        int    tm_wday;         /* giorno della settimana */
        int    tm_yday;         /* giorno dell'anno */
        int    tm_isdst;        /* ora legale */
};
```

utilizzata dalle seguenti funzioni di gestione della data e dell'ora:

```
char *asctime(struct tm *timeptr);
```

che restituisce la data e l'ora in formato stringa e la seguente:

```
struct_tm *localtime(time_t *time);
```

che restituisce il puntatore ad una **struct** tm vista sopra, da cui è possibile estrarre i valori dei vari attributi. Il listato richiesto è riportato di seguito.

FunzioniTime.c

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{   time_t t;
   struct tm *lt;
   char *at;
   time(&t);    /* time() restituisce il calendar time (numero di secondi dal 01/01/1970  */
   cout<<"Dal 01/01/1970 sono trascorsi "<<t<<" secondi"<<endl;
   lt=localtime(&t);
   cout<<lt->tm_hour<<":"<<lt->tm_min<<":"<<lt->tm_sec<<endl;
   at=asctime(lt);
   cout<<at;
   system("pause");
```

```
   return 0;
}
```

Esempio 3

Scrivere un programma che legge una sequenza di interi distinti (MAXVAL = 1000), ciascuno seguito immediatamente da una lettera. Si legge un altro intero e, sulla base della sequenza introdotta, si determini se esiste una lettera associata a quell'intero. Per esempio per i dati in ingresso (MAXVAL = 3):
123B
492M
7621C
4V
7B
e per il valore letto:
492
si ha in uscita:
La lettera associata a 492 è M

/ Struttura-Ricerca.cpp */*

```
#include <iostream>
#include <cstdlib>
using namespace std;
#define MAXVAL 1000
char lettera_associata(int num, int maxind);
struct
{
  int  num;
  char ch;
} associato [MAXVAL];

int main(int argc, char *argv[])
{
int  i, num;
  cout<<"Immetti sequenza"<<endl;
  for (i=0; i<MAXVAL;i++)
  {
     cout<<"Immetti la stringa: "<<endl;
     cin>>associato[i].num;
     cin>>associato[i].ch;
  }
  cout<<"Immetti il numero: ";
  cin>>num;
  cout<<"La lettera associata a "<<num<<" e' "<<lettera_associata(num,i-1)<<endl;
  system("Pause");
  return 0;
}
char lettera_associata(int num, int maxind)
{
int  i;
  for (i=0; (i<=maxind) && (associato[i].num!=num); i++);
  return((i>maxind) ? '-' : associato[i].ch);                    /* operatore ternario */
}
```

2. Esercizi applicativi

1. Date le seguenti dichiarazioni:

```
typedef struct                          struct persona
{     int gg;                           {       char cognome[20];
      int mm;                                   char nome[20];
      int aa;                                   char codice_fiscale[15];
} data;                                         data nascita;
                                                data assunzione;
                                                int coniugato;
                                        }       anagrafe [100];
```

   a. Scrivere una funzione per acquisire i dati di una persona;
   b. Scrivere una funzione per ricercare una persona in base al codice fiscale
   c. Scrivere una funzione per stampare l'elenco dei nati in un dato anno.

2. Individuare gli errori sintattici e/o logici nel seguente frammento di codice e, dopo averli corretti, descriverne l'effetto.

```
#include <iostream>
#define COSTO_A_SCATTO 20
{
     struct
      int codice;
              int scatti_precedenti;
              int scatti_attuali;
              int scatti;
}     utenti [1000]
….
for (i=0; i<1000; i++)
{
     utenti.scatti = utenti[i].scatti_attuali – utenti[i].scatti_precedenti;
              utenti.importo = utenti[i].scatti + COSTO_A_SCATTO;
}
…..
```

3. Con riferimento alle strutture dati dichiarate in esercizio 1, individuare gli errori sintattici e/o logici nei seguenti frammenti di codice e, dopo averli corretti, descriverne l'effetto:
   a. **cout**<<anagrafe.data.gg<<**endl**;
   b. **cout** <<anagrafe[i].cognome<<**endl**;
   c. **cin >>**anagrafe[i].assunzione.data.mm;
   d. anagrafe[i].nascita.gg="31";
   e. anagrafe[i].coniugato=1;
   f. anagrafe[i].persona.coniugato=1;
   g. anagrafe[i]=assunzione;
   h. anagrafe[i]=anagrafe[j];