PROBLEMA: Caricamento e visualizzazione di un record di tipo "Dipendente" il cui tracciato è assegnato secondo la seguente rappresentazione tabellare (CON O SENZA L'USO DI SOTTOPROGRAMMI):

Sia assegnata la seguente struttura dati relativamente ad un **dipendente** di una determinata azienda, dettagliata utilizzando la seguente forma tabellare:

Numero	Nome Campo	Tipo Campo	Lunghezza[1]	Descrizione	FlagSubRec[2]
1	Codice	INT	4	Codice identificativo del dipendente (OBBLIGATORIO e POSITIVO)	X
2	Cognome	ARRAY DI CHAR	50	Cognome del dipendente (OBBLIGATORIO)	X
3	Nome	ARRAY DI CHAR	50	Nome del dipendente (OBBLIGATORIO)	X
4	Stipendio	REAL	6,2	Stipendio del dipendente (OBBLIGATORIO e compreso tra 800 e 1200 euro)	
5	Livello	INT	1	Livello del dipendente (OBBLIGATORIO e compreso tra 5 e 9)	

1. In caso di valori numerici decimali scrivere "6,2" vuol dire 6 cifre totali di cui 2 decimali (max 9999.99)
In caso di valori numerici interi scrivere 4 vuol dire prevedere un massimo di 4 cifre significative (max 9999)
2. Se nel campo FlagSubRec viene posta una X vuol dire che quel campo deve essere considerato parte di un altro record (vedi SOTTORECORD)

Scrivere la PROGETTAZIONE (tabelle dei dati, pseudocodifica e flowchart) dell'ALGORITMO che, **UTILIZZANDO L'ALLOCAZIONE STATICA**, effettui le seguenti azioni NELL'ORDINE IN CUI ESSE VENGONO DESCRITTE:

a) Legga i dati relativi ad **UN SOLO dipendente** effettuando tutti i dovuti controlli ;

b) Mostri a video i dati relativi a quel **dipendente** (nel formato dettagliato nell'esempio mostrato sotto):

*Esempio: Immaginiamo che dopo l'esecuzione dell'azione descritta **al punto a)** siano stati letti i dati del seguente dipendente:*
Codice: 101 Cognome: ROSSI Nome: MARIO Stipendio: 1100.75 Livello: 6

*Allora l'esecuzione dell'azione descritta **al punto b)** causerà la seguente visualizzazione:*

```
Codice immesso:      104
Cognome immesso:     NERI
Nome immesso:        FILIPPO
Stipendio immesso:   1100.75
Livello immesso:     6
```

E' possibile implementare le azioni ai punti **a)** e **b)** con i seguenti **SOTTOPROGRAMMI**:

a) FUNZIONE Leggi_Dip (……) : ……..

b) PROCEDURA Visualizza_Dip (……)

TABELLE DEI DATI

DATI DI INPUT DEL PROBLEMA PRINCIPALE PROCEDURA main()				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

DATI DI OUTPUT DEL PROBLEMA PRINCIPALE PROCEDURA main()				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

DATI DI ELABORAZIONE o DI LAVORO DEL PROBLEMA PRINCIPALE PROCEDURA main()				
Nome variabile oppure **nome costante**	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
d	**Dipendente**	**STATICA**	**Vedi TRACCIATO record**	Istanza generica del tipo Dipendente
MAXNUMCHAR	**INT**	**STATICA**	**50**	Massimo numero di caratteri per Nome e Cognome del Dipendente

```
ALGORITMO Dipendente_SubRec
MAXNUMCHAR     50
TIPO DettaglioDipendente = RECORD
                              Codice : INT
                              Cognome : ARRAY[MAXNUMCHAR] DI CHAR
                              Nome : ARRAY[MAXNUMCHAR] DI CHAR
                           FINE RECORD
TIPO Dipendente = RECORD
                     DetDip : DettaglioDipendente
                     Sipendio : REAL
                     Livello : INT
                  FINE RECORD
PROCEDURA main()
d : Dipendente
INIZIO
//Utilizziamo la funzione a)
d ← Leggi_Dip ()
//Utilizziamo la procedura b)
Visualizza_Dip (d)
RITORNA
FINE
```

TABELLE DEI DATI

DATI DI INPUT DEL SOTTOPROBLEMA: FUNZIONE Leggi_Dip()				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

DATI DI OUTPUT DEL SOTTOPROBLEMA: FUNZIONE Leggi_Dip()				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

DATI DI ELABORAZIONE o DI LAVORO DEL SOTTOPROBLEMA: FUNZIONE Leggi_Dip()				
Nome variabile oppure **nome costante**	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
d	**Dipendente**	**STATICA**	**Vedi TRACCIATO record**	Dipendente generico restituito nel nome della funzione

```
FUNZIONE Leggi_Dip ( ) : Dipendente
d : Dipendente
INIZIO
RIPETI
   Scrivi("Codice Dipendente = ")
    Leggi (d.DetDip.Codice)
FINCHE' (d.DetDip.Codice > 0) AND (d.DetDip.Codice ≤ 9999)
RIPETI
   Scrivi("Cognome Dipendente = ")
    Leggi (d.DetDip.Cognome)
FINCHE' (Lunghezza(d.DetDip.Cognome) ≠ 0) AND
          (Lunghezza(d.DetDip.Cognome) ≤ MAXNUMCHAR)
RIPETI
   Scrivi("Nome Dipendente =  ")
    Leggi (d.DetDip.Nome)
FINCHE' (Lunghezza(d.DetDip.Nome) ≠ 0) AND
          (Lunghezza(d.DetDip.Nome) ≤ MAXNUMCHAR)
RIPETI
   Scrivi("Stipendio Dipendente = ")
    Leggi (d.Stipendio)
FINCHE' (d.Stipendio ≥ 800.00) AND (d.Stipendio ≤ 1200.00)
RIPETI
   Scrivi("Livello Dipendente = ")
    Leggi (d.Livello)
FINCHE' (d.Livello ≥ 5) AND (d.Livello ≤ 9)
RITORNA (d)
FINE
```

TABELLE DEI DATI

DATI DI INPUT DEL SOTTOPROBLEMA: PROCEDURA Visualizza_Dip()				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
d	**Dipendente**	**STATICA**	**Vedi TRACCIATO record**	Parametro passato per VALORE

DATI DI OUTPUT DEL SOTTOPROBLEMA: PROCEDURA Visualizza_Dip ()				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
d	**Dipendente**	**STATICA**	**Vedi TRACCIATO record**	Parametro passato per VALORE

DATI DI ELABORAZIONE o DI LAVORO DEL SOTTOPROBLEMA: PROCEDURA Visualizza_Dip ()				
Nome variabile oppure **nome costante**	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

```
PROCEDURA Visualizza_Dip (VAL d : Dipendente)
INIZIO
Scrivi("Codice immesso: ")
Scrivi (d.DetDip.Codice)
Scrivi("Cognome immesso: ")
Scrivi (d.DetDip.Cognome)
Scrivi("Nome immesso: ")
Scrivi (d.DetDip.Nome)
Scrivi("Stipendio immesso: ")
Scrivi (d.Stipendio)
Scrivi("Livello immesso: ")
Scrivi (d.Livello)
RITORNA
FINE
```

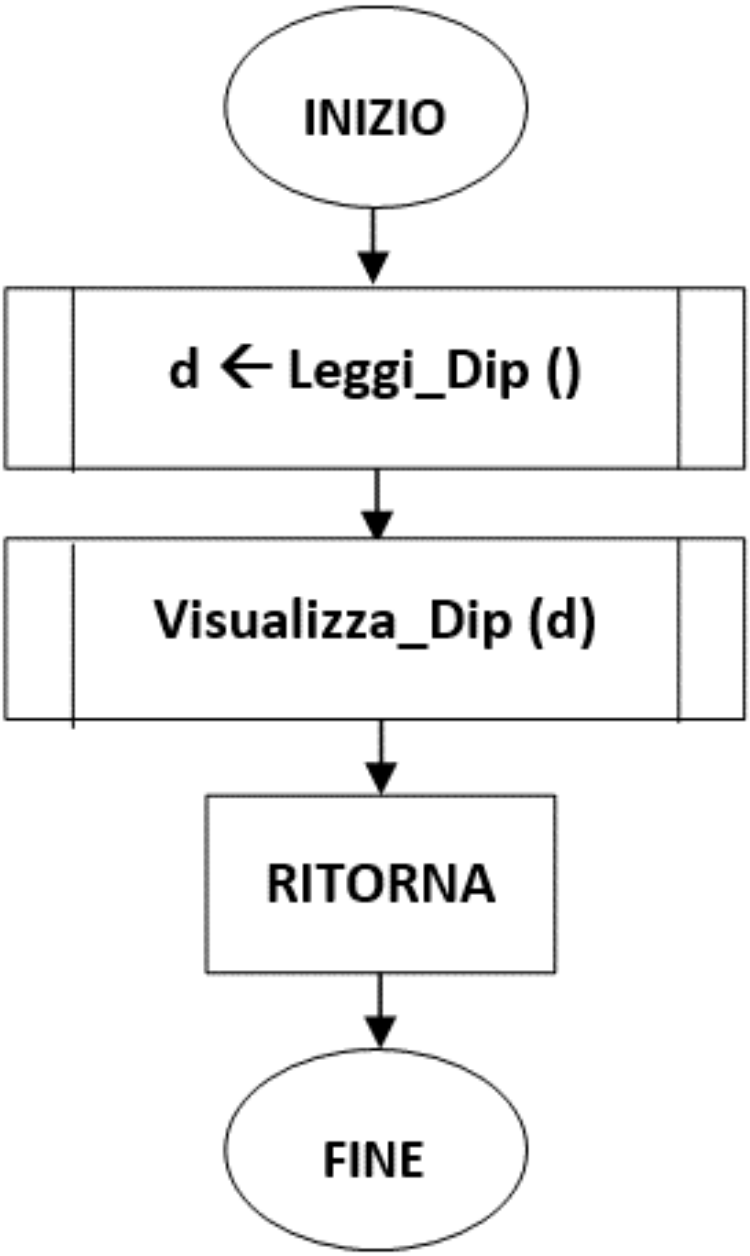
PSEUDODODIFICA

```
ALGORITMO Dipendente_SubRec
MAXNUMCHAR     50
TIPO DettagliDipendente = RECORD
                              Codice : INT
                              Cognome : ARRAY[MAXNUMCHAR] DI CHAR
                              Nome : ARRAY[MAXNUMCHAR] DI CHAR
                          FINE RECORD
TIPO Dipendente = RECORD
                      DetDip : DettagliDipendente
                      Sipendio : REAL
                      Livello : INT
                  FINE RECORD
PROCEDURA main()
d : Dipendente
INIZIO
//Utilizziamo la funzione a)
d ← Leggi_Dip ()
//Utilizziamo la procedura b)
Visualizza_Dip (d)
RITORNA
FINE
FUNZIONE Leggi_Dip ( ) : Dipendente
d : Dipendente
INIZIO
RIPETI
   Scrivi("Codice Dipendente = ")
   Leggi (d.DetDip.Codice)
FINCHE' (d.DetDip.Codice > 0) AND (d.DetDip.Codice ≤ 9999)
RIPETI
   Scrivi("Cognome Dipendente = ")
   Leggi (d.DetDip.Cognome)
FINCHE' (Lunghezza(d.DetDip.Cognome) ≠ 0) AND
        (Lunghezza(d.DetDip.Cognome) ≤ MAXNUMCHAR)
RIPETI
   Scrivi("Nome Dipendente =  ")
   Leggi (d.DetDip.Nome)
FINCHE' (Lunghezza(d.DetDip.Nome) ≠ 0) AND
        (Lunghezza(d.DetDip.Nome) ≤ MAXNUMCHAR)
RIPETI
   Scrivi("Stipendio Dipendente = ")
   Leggi (d.Stipendio)
FINCHE' (d.Stipendio ≥ 800.00) AND (d.Stipendio ≤ 1200.00)
RIPETI
   Scrivi("Livello Dipendente = ")
   Leggi (d.Livello)
FINCHE' (d.Livello ≥ 5) AND (d.Livello ≤ 9)
RITORNA (d)
FINE
```

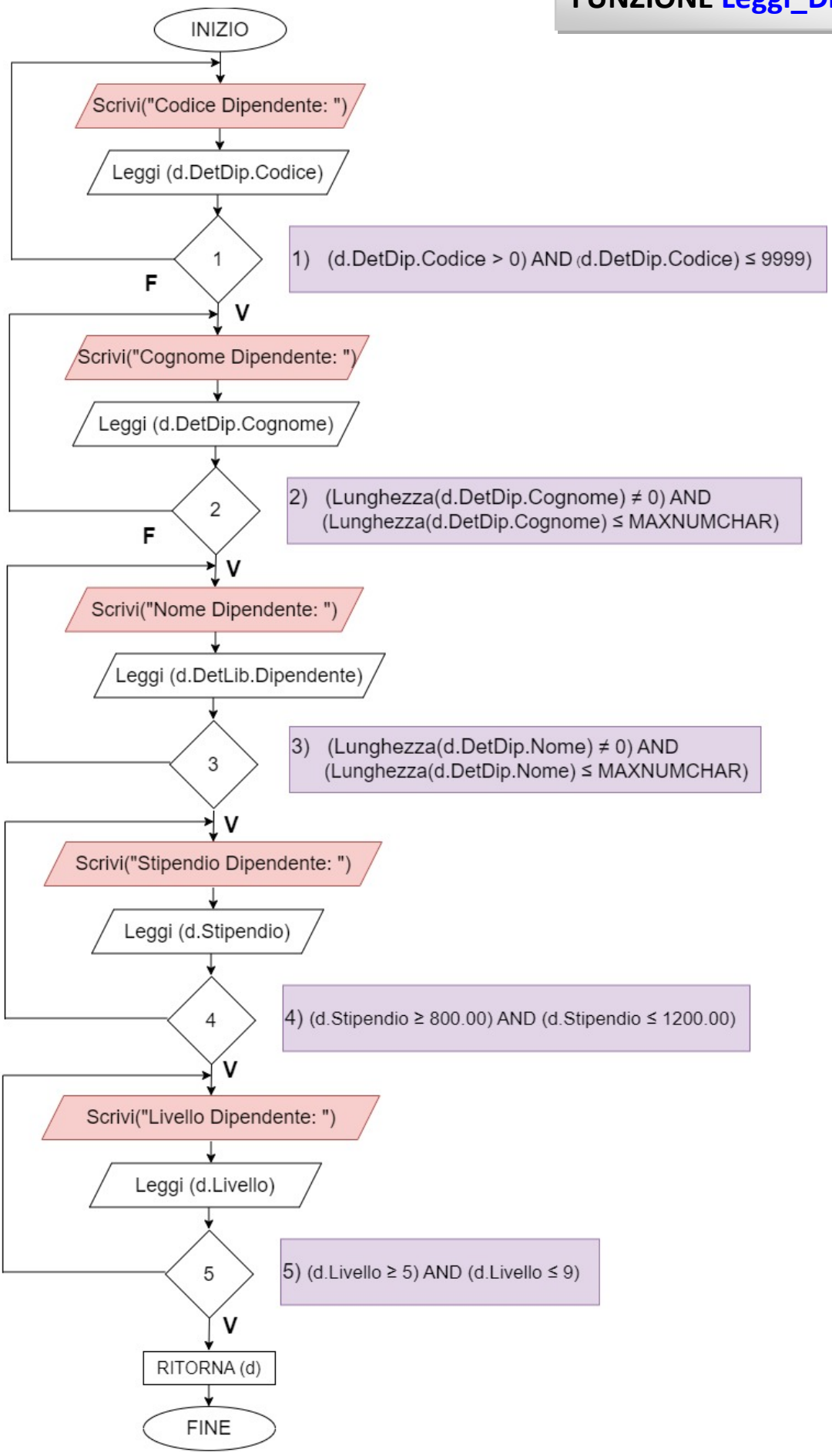
```
PROCEDURA Visualizza_Dip (VAL d : Dipendente)
INIZIO
Scrivi("Codice immesso: ")
Scrivi (d.DetDip.Codice)
Scrivi("Cognome immesso: ")
Scrivi (d.DetDip.Cognome)
Scrivi("Nome immesso: ")
Scrivi (d.DetDip.Nome)
Scrivi("Stipendio immesso: ")
Scrivi (d.Stipendio)
Scrivi("Livello immesso: ")
Scrivi (d.Livello)
RITORNA
FINE
```

FLOWCHART

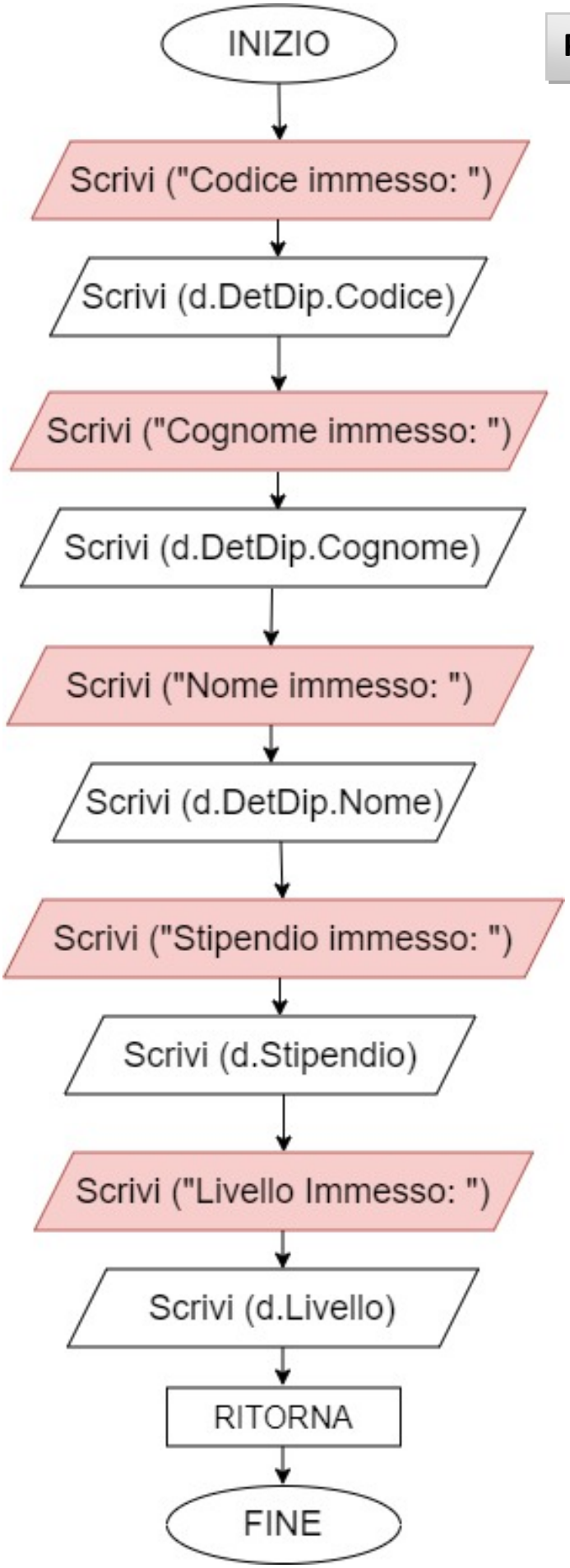
PROCEDURA main ( )



FUNZIONE Leggi_Dip ( )



PROCEDURA Visualizza_Dip ( )



CODIFICA C (MONOFILE)

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

#define MAXNUMCHAR 50

typedef struct
   {
   int Codice;
   char Cognome [MAXNUMCHAR + 1];
   char Nome [MAXNUMCHAR + 1];
   } DettagliDipendente;

typedef struct
   {
   DettagliDipendente DetDip;
   float Stipendio;
   int Livello;
   } Dipendente;

// prototipi funzioni C necessarie
Dipendente Leggi_Dip ();
void Visualizza_Dip (Dipendente d);

// PROCEDURA main()
int main(int argc, char*argv[])
{
/* dati di lavoro */
Dipendente d;

/* CALL alla FUNZIONE Leggi_Dip() */
d = Leggi_Dip ();

/* CALL alla PROCEDURA Visualizza_Dip() */
Visualizza_Dip (d);

return 0;
}


// FUNZIONE Leggi_Dip()()
Dipendente Leggi_Dip ()
{
Dipendente d;

// caricamento e controllo campi del record
do
   {
   printf ("Codice Dipendente: ");
   scanf("%d", &(d.DetDip.Codice));
   }
while ( (d.DetDip.Codice < 0) || (d.DetDip.Codice > 9999) );

do
   {
   printf ("Cognome Dipendente: ");
   fflush (stdin);
   gets (d.DetDip.Cognome);
   }
while ( (strlen(d.DetDip.Cognome) == 0) || (strlen(d.DetDip.Cognome) > MAXNUMCHAR) );
```

```
do
   {
   printf ("Nome Dipendente: ");
   gets (d.DetDip.Nome);
   }
while ( (strlen(d.DetDip.Nome) == 0) || (strlen(d.DetDip.Nome) > MAXNUMCHAR) );

do
   {
   printf ("Stipendio Dipendente: ");
   scanf("%f", &(d.Stipendio));
   }
while ( (d.Stipendio < 800.00) || (d.Stipendio > 1200.00) );

do
   {
   printf ("Livello Dipendente: ");
   scanf("%d", &(d.Livello));
   }
while ( (d.Livello < 5) || (d.Livello > 9) );

return (d);
}


// PROCEDURA Visualizza_Dip)
void Visualizza_Dip (Dipendente d)
{

printf ("\n******** STAMPA **************");
// Visualizzazione campi del record
// Codice dipendente
printf ("\nCodice immesso: ");
printf ("%d", d.DetDip.Codice);

// Cognome dipendente
printf ("\nCognome immesso: ");
puts (d.DetDip.Cognome);

// Nome dipendente
printf ("Nome immesso: ");
puts (d.DetDip.Nome);

// Stipendio dipendente
printf ("Stipendio immesso: ");
printf ("%.2f",d.Stipendio);

// Livello immesso
printf ("\nLivello immesso: ");
printf ("%d",d.Livello);

return;
}
```

CODIFICA C (MULTIFILE)

FILE **costanti.h**

```
#define MAXNUMCHAR 50
```

FILE **tipi.h**

```
# include "costanti.h"

typedef struct
   {
   int Codice;
   char Cognome [MAXNUMCHAR + 1];
   char Nome [MAXNUMCHAR + 1];
   } DettagliDipendente;

typedef struct
   {
   DettagliDipendente DetDip;
   float Stipendio;
   int Livello;
   } Dipendente;
```

FILE **prototipi.h**

```
// prototipi funzioni C necessarie
Dipendente Leggi_Dip ();
void Visualizza_Dip (Dipendente d);
```

FILE **main.c**

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

//INCLUDE header file utente
#include "costanti.h"
#include "tipi.h"
#include "prototipi.h"

// PROCEDURA main()
int main(int argc, char*argv[])
{
/* dati di lavoro */
Dipendente d;

/* CALL alla FUNZIONE Leggi_Dip() */
d = Leggi_Dip ();

/* CALL alla PROCEDURA Visualizza_Dip() */
Visualizza_Dip (d);

return 0;
}
```

FILE **Leggi_Dip.c**

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

# include "tipi.h"

// FUNZIONE Leggi_Dip()()
Dipendente Leggi_Dip ()
{
Dipendente d;

// caricamento e controllo campi del record
do
   {
   printf ("Codice Dipendente: ");
   scanf("%d", &(d.DetDip.Codice));
   }
while ( (d.DetDip.Codice < 0) || (d.DetDip.Codice > 9999) );

do
   {
   printf ("Cognome Dipendente: ");
   fflush (stdin);
   gets (d.DetDip.Cognome);
   }
while ( (strlen(d.DetDip.Cognome) == 0) || (strlen(d.DetDip.Cognome) > MAXNUMCHAR) );

do
   {
   printf ("Nome Dipendente: ");
   gets (d.DetDip.Nome);
   }
while ( (strlen(d.DetDip.Nome) == 0) || (strlen(d.DetDip.Nome) > MAXNUMCHAR) );

do
   {
   printf ("Stipendio Dipendente: ");
   scanf("%f", &(d.Stipendio));
   }
while ( (d.Stipendio < 800.00) || (d.Stipendio > 1200.00) );

do
   {
   printf ("Livello Dipendente: ");
   scanf("%d", &(d.Livello));
   }
while ( (d.Livello < 5) || (d.Livello > 9) );

return (d);
}
```

FILE **Visualizza_Dip.c**

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

# include "tipi.h"

// PROCEDURA Visualizza_Dip)
void Visualizza_Dip (Dipendente d)
{

printf ("\n******** STAMPA **************");
// Visualizzazione campi del record
// Codice dipendente
printf ("\nCodice immesso: ");
printf ("%d", d.DetDip.Codice);

// Cognome dipendente
printf ("\nCognome immesso: ");
puts (d.DetDip.Cognome);

// Nome dipendente
printf ("Nome immesso: ");
puts (d.DetDip.Nome);

// Stipendio dipendente
printf ("Stipendio immesso: ");
printf ("%.2f",d.Stipendio);

// Livello immesso
printf ("\nLivello immesso: ");
printf ("%d",d.Livello);

return;
}
```