Consiglio Nazionale delle ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Strumento programmabile per la valutazione in linea dei tempi di reazione in prove uditive complesse

Progetto software e implementazione

A. Ribolini E. Bozzi

Nota Interna B4-22

Maggio 1990

I N D I C E

Strumento programmabile per la valutazione in linea dei tempi di reazione in prove uditive complesse: Progetto software e implementazione

A. Ribolini , E. Bozzi,

Introduzione

Da una collaborazione con l'Istituto di Neurofisiologia dell'Universita' di Pisa e' sorta la necessita' di studiare e realizzare uno strumento per la misura e la valutazione anche complessa dei tempi di reazione di scelta per stimoli di tipo acustico, secondo sequenze guidate e facilmente definibili. Il tipo di esperimenti che si devono eseguire sono: go no go classico, go no go modificato e choice-RT [1]. Nel primo esperimento il soggetto risponde con la mano indicatagli allo stimolo inviato, in maniera random, in uno dei due auricolari; nel secondo vengono inviati nei due auricolari uno stimolo a frequenza alta o uno a frequenza bassa, il soggetto deve rispondere solamente, con la mano indicatagli, allo stimolo con frequenza piu' alta; nel terzo esperimento, Choice RT, il soggetto deve rispondere con la mano destra o sinistra allo stimolo ricevuto dall'orecchio destro o sinistro, o viceversa. La risposta e' considerata sbagliata quando e' data con l'arto non richiesto, oppure prima di 150 ms o dopo 1 sec dall'invio dello stimolo acustico [2][3]. La variabilita' di esperimenti che possono essere utili, suggerisce l'opportunita' di realizzare uno strumento che possa essere facilmente configurato. Per il progetto del software si e' adottato come criterio principale quello della modularita' che presenta vantaggi nel consentire future espansioni.

Nella presente nota viene riportato l'elenco ed una breve descrizione di tutti i file impiegati e l'elenco, in ordine alfabetico, di tutte le procedure costituenti il programma con a fianco di ciascuna il nome del file che la contiene. Vengono infine allegati i listati del codice

sorgente scritto interamente in linguaggio C opportunamente commentato in modo da renderlo di facile comprensione.

In figura 1 viene riportato uno schema a blocchi del flusso funzionale del programma. Le funzioni contenute in ciascun file vengono raggruppate e evidenziate mediante una riquadratura tratteggiata. In questo modo l'utente ha tutte le indicazioni possibili per identificare il file di interesse, nel quale e' contenuta una determinata procedura, e puo' risalire facilemente al listato relativo.

Elenco dei file contenenti il programma sorgente

Il codice sorgente che costituisce il programma GO.EXE e' suddiviso in 11 file. Di seguito viene riportata, facendo riferimento alla fig.1, una breve descrizione del contenuto di ciascun file e l'elenco delle procedure contenute in ciascuno di essi:

Contenuto del file: GO.C

contiene il programma principale e le definizioni dei vettori di programmazione dello stimolatore. Inizializza lo stimolatore e tutte le variabili globali utilizzate dal programma.

Elenco delle procedure:

```
main(argc,argv)
fasce()
videof(flag)
presenta(ir,ic,ia,is)
exit_close()
led()
apre_video(name)
chiude_video()
```

Contenuto del file: GESFILE.C

contiene le procedure per la gestione dei vari file utilizzati dal programma e le varie procedure di caricamento, modifica e riscrittura dei file per la generazione dei numeri casuali.

Elenco delle procedure:

```
gestisce_file()
search(stringa1,stringa2)
apri_file(file_name,modo)
inizializza_ran()
chiude_file_ran()
carica_casuali(minimo,massimo,array,ncelle)
legge_dati_default(unit,array1,array2,ndata)
messaggio(indice)
carica_archivio(unit,archivio,osservazioni,ncelle)
scrive_archivio(unit,archivio,osservazioni,ncelle)
crea_indice(flag,sesso,eta,stimolatore,arto1,arto2)
aggiorna_buffer(indice,archivio,osservazioni,x,y,w,z)
```

Contenuto del file: GO1.C

contiene tutte le procedure di presentazione video relative alle varie fasi di inizializzazione, introduzione dati anagrafici ed alla scelta della sessione di prove da effettuarsi sul soggetto.

Elenco delle procedure:

```
test_hardware()
tipo_di_prova(flag)
anagrafe_ok()
```

```
anagrafe()
attende_char(iriga,icolo,attrib,nchar_max,buffer,tipo,array)
apre_temporaneo()
stampa_temporaneo()
resp_time()
clear_cursore(posizione)
write_cursore(posizione)
presito(posizione)
messaggio_assenza(posizione)
calcola_libero(posizione,direzione,posm,posn)
determina_stimart(posizione)
riprogramma_stimolatore()
pstree()
```

Contenuto del file: GO2.C

contiene tutte le procedure di presentazione video, di acquisizione ed elaborazione dei dati acquisiti sul soggetto.

Elenco delle procedure:

```
esame_apprendimento()
apprendimento_semplice()
numero_tipo(f)
esegui_apprendi_semplice()
wait(tick)
wait_uno()
esame_semplice(posizione)
presenta_dato_float(iriga,icolo,attributo,valore)
presenta_dato_integer(iriga,icolo,attributo,valore)
archiviazione_fisso()
determina_arti(parola)
data_int(array)
inverte_data(array)
```

```
system_op()
schermo(posizione)
riallinea(sd,sq,pos)
premed(temp,denmedia,mediana1,mediana2,conl1,conl2,pos)
setta_next(risp,pos,next)
via()
```

Contenuto del file: ITBM.C

contiene le procedure di gestione della comunicazione con lo stimolatore uditivo.

Elenco delle procedure:

```
send_addr(indirizzo)
send_data(byte)
reset(time_out)
ini_per()
receive()
sense_bit(ibit,direction,time_out)
trigger(to)
send_data_some(byte)
send_addr_some()
receive_some()
ciclocoa()
```

Contenuto del file: UDITIVO.C

contiene le procedure di programmazione e di controllo dello stimolatore uditivo.

Elenco delle procedure:

```
uditivo(param)
edit_uditivo()
tx_uditivo()
varia_parametro()
intensifica()
normal()
blinka()
rc_put(irig,icol,istri,iattr)
scrivi_attuali()
aggiorna()
aggiorna_val(para)
scrivi_param()
scrivix(par)
iniz_udi()
```

Contenuto del file: SINGOLA.C

contiene le procedure di gestione della singola prova e la gestione del tempo di risposta del soggetto. E' utilizzata sia nella fase di apprendimento che nella fase vera e propria di acquisizione dei tempi di risposta.

Elenco delle procedure:

```
singola_prova(casuale,arto,lim1,lim2,orecch)
lintest()
```

Contenuto del file: TIMER.C

contiene le procedure di gestione del timer/counter presente sulla scheda di interfaccia per l'acquisizione del tempo di risposta del soggetto e dei tempi di stimolazione sia acustica che ottica.

Elenco delle procedure:

```
start_timer()
stop_timer()
load_timer(word)
legge_timer()
trasmette_comandi(str,cod)
```

Contenuto del file: FLOPPY.C

contiene le procedure di gestione dell'archiviazione dati su dischetto, le procedure relative alla eventuale inizializzazione di un nuovo dischetto e le procedure di stampa dei dati intermedi, o finali, per archiviazione su supporto cartaceo.

Elenco delle procedure:

```
scelta_archiviazione(opt)
archivia_floppy()
cambia_formatta()
esegue_inizializzazione()
azzera_righe(flag)
zona_messaggi()
in_corso()
erase_corso()
write_data(ia,iv)
lista_dischetto()
```

```
mediana(array,ncelle,fmedi)
carica_array()
```

Contenuto del file: VISUALIZ.C

contiene le procedure di visualizzazione dei risultati acquisiti nelle prove relative alla sessione prescelta.

Elenco delle procedure:

```
visualizza_dati(opt)
stampall()
```

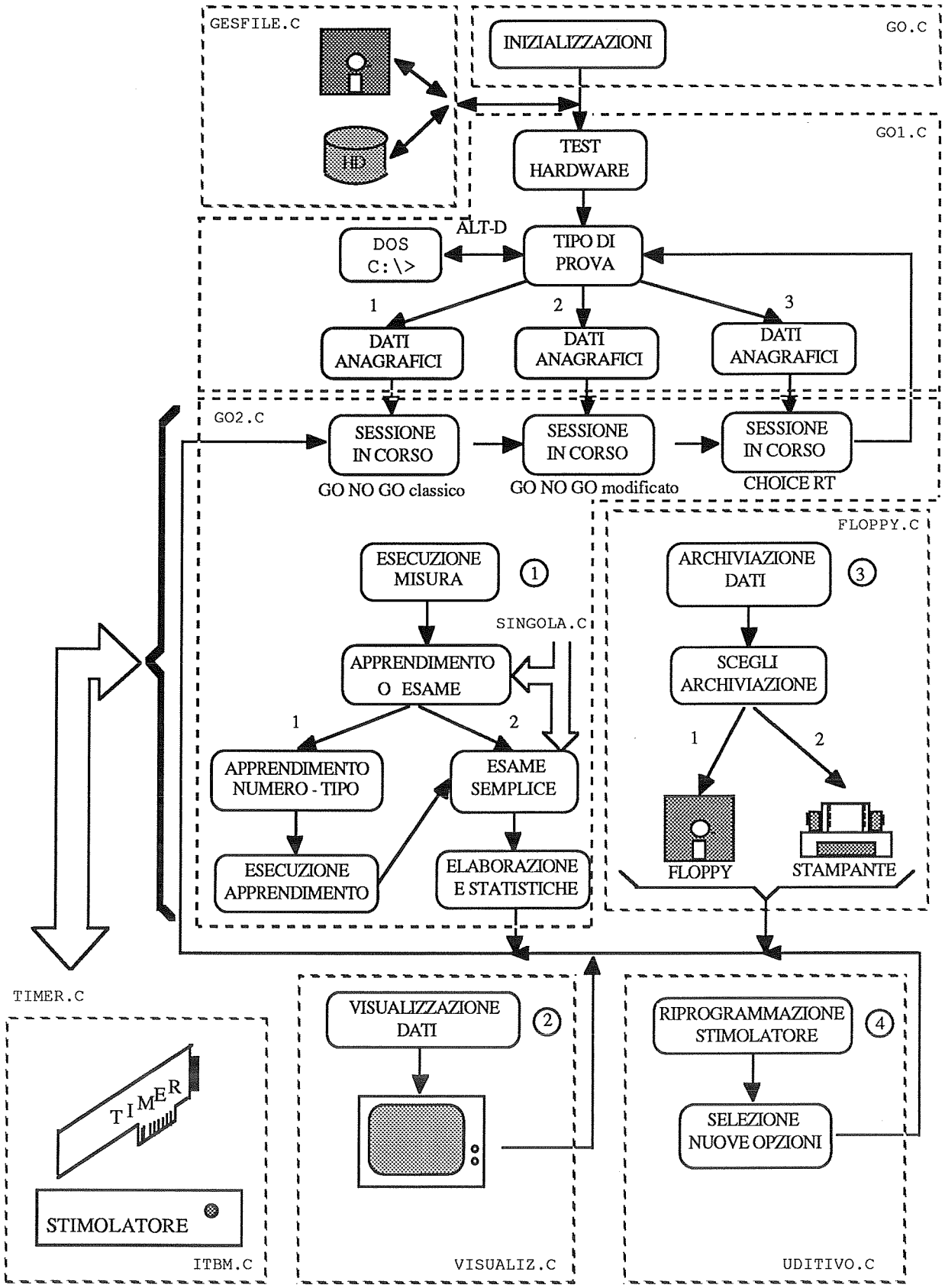
GESFILE.C
HD
GO.C
INIZIALIZZAZIONI
GO1.C
TEST HARDWARE
ALT-D
DOS C:\>
TIPO DI PROVA
1
2
3
DATI ANAGRAFICI
DATI ANAGRAFICI
DATI ANAGRAFICI
GO2.C
SESSIONE IN CORSO
SESSIONE IN CORSO
SESSIONE IN CORSO
GO NO GO classico
GO NO GO modificato
CHOICE RT
FLOPPY.C
ESECUZIONE MISURA
1
ARCHIVIAZIONE DATI
3
SINGOLA.C
APPRENDIMENTO O ESAME
SCEGLI ARCHIVIAZIONE
1
2
1
2
APPRENDIMENTO NUMERO - TIPO
ESAME SEMPLICE
ESECUZIONE APPRENDIMENTO
ELABORAZIONE E STATISTICHE
FLOPPY
STAMPANTE
TIMER.C
VISUALIZZAZIONE DATI
2
RIPROGRAMMAZIONE STIMOLATORE
4
TIMER
SELEZIONE NUOVE OPZIONI
STIMOLATORE
ITBM.C
VISUALIZ.C
UDITIVO.C

Figura 1

Elenco alfabetico delle procedure

Listato dei files in ordine alfabetico

File: FLOPPY.C

```
#include "go.h"

   extern unsigned char tipo_PR;
   extern float    medmed[RPR][CPR];
   extern unsigned char tipo_PR;
   char   type_sem,type_dis,esito_sem,esito_dis;

struct disp17 {
char stringa17[50];
int posizione_riga17;
int posizione_colo17;
int attributo17;
};

struct disp17 screen17[15]={
"                                                ",1,14,23,
"                                                ",2,14,71,
"    S C E L T A   A R C H I V I A Z I O N E     ",3,14,71,
"                                                ",4,14,71,
"                                  ",9,20,23,
"     1:  D I S C H E T T O        ",10,20,23,
"                                  ",11,20,23,
"     2:  S T A M P A N T E        ",12,20,23,
"                                  ",13,20,23,
"     0:  E S C E                  ",14,20,71,
"                                  ",15,20,23,
"   N.B. I DISCHETTI DEVONO ESSERE FORMATTATI    ",18,15,151,
"                                                ",21,15,71,
" <ESC> menu` precedente      ALT-Y Reinizializza ",22,15,71,
"             Introduci: 0 - 1 - 2               ",23,15,71,
};

scelta_archiviazione(opt)
int opt;
{
    extern char nuovo_paziente;
    int  a,iprinter,stat,idisk;
    char risp,*prnt,*mess1,*mess2;
    char flag,bell=7,test;

    prnt="PRN:";
    mess1="     ARCHIVIAZIONE SU FLOPPY DEL SOGGETTO GIA' AVVENUTA       ";
    mess2="     ARCHIVIAZIONE SU FISSO DEL SOGGETTO GIA' AVVENUTA       ";
    flag=0x00;
    while(!flag)
    {
       scr_clr();
       for (a=0; a<15; a++)
          presenta(screen17[a].posizione_riga17,screen17[a].posizione_colo17,
                   screen17[a].attributo17,screen17[a].stringa17);
```

```
        fasce();
        lintest();
        while(((risp=ci())!='1')&&(risp!='2')&&
              (risp!='0')&&(risp!=27)&&(risp!=21));
        switch (risp)
        {
           case '1': test=nuovo_paziente&1;
                     if(test!=0)
                     {
                        presenta(19,10,71,mess1);
                        printf("%c",bell);
                        risp=ci();
                     }
                     else
                     {
                        archivia_floppy();
                        nuovo_paziente=nuovo_paziente|1;
                     }
                     break;
           case '2': iprinter=open(prnt,1);
                     write_data(iprinter,iprinter);
                     stat=close(iprinter);
                     break;
           case 21 : exit();
                     break;
           case 27 : return(27);
                     break;
           case '0': flag=0xFF;
                     break;
        }
     }
     return(0);
}


   int  unitm,unita,unitv,status;
   char filem[15],filea[15],filev[15];
   char code,risp;
   char date[]={0,0,0,0,0,0,0,0,0,0},time[]={0,0,0,0,0,0,0,0,0,0},dummy[80];
   int  nfile1,nfile2;
   float nbytetot;


archivia_floppy()
{
   extern int ftotali;
   long  lval;
   int   a,j,k;
   char  record[132],bell=7;
   char  *mess1,*mess2,*mess3;

   mess1="                                                              ";
   mess2="   A R C H I V I A Z I O N E   S U   D I S C H E T T O  ";
   scr_clr();
   presenta(2,10,71,mess1);
   presenta(3,10,71,mess2);
   presenta(4,10,71,mess1);
```

```
fasce();
lintest();
zona_messaggi();
sprintf(filem,"A:-MASTER-.IDX");
sprintf(record,"ASSICURATI DI AVER INTRODOTTO UN DISCHETTO NELLA UNITA' A:");
presenta(10,13,23,record);
sprintf(record," E DI AVER CHIUSO LO SPORTELLO DEL DRIVER CHE LO CONTIENE ");
presenta(11,13,23,record);
sprintf(record,"       --- BATTI UN TASTO QUALSIASI QUANDO PRONTO ---     ");
presenta(12,13,23,record);
printf("%c",bell);
risp=ci();
zona_messaggi();
unitm=open(filem,2);
if(unitm==-1)
{
   zona_messaggi();
   sprintf(record,
      "DISCHETTO NON INIZIALIZZATO CORRETTAMENTE-MANCA FILE MASTER");
   presenta(10,13,23,record);
   nfile2=cambia_formatta();
}
else
   status=fscanf(unitm,"%d %d %f %s\n",&nfile1,&nfile2,&nbytetot,&dummy);
while (nfile2==0)
{
   status=close(unitm);
   erase_corso();
   zona_messaggi();
   sprintf(record,"IL DISCHETTO INSERITO NON CONTIENE PIU` SPAZIO LIBERO");
   presenta(10,13,23,record);
   nfile2=cambia_formatta();
}
nfile1++;
nfile2--;
nbytetot-=31000.0;
lval=rewind(unitm);
dates(date);
times(time);
status=fprintf(unitm,"%2d %2d %f %s %s\n",
               nfile1,nfile2,nbytetot,date,time);
status=close(unitm);
sprintf(filea,"A:ANAGRAFE.%0*d",3,nfile1);
sprintf(filev,"A:VALORICA.%0*d",3,nfile1);
unita=creat(filea);
if(unita==-1)   exit();
unitv=creat(filev);
if(unitv==-1)   exit();
status=close(unita);
status=close(unitv);
unita=open(filea,1);
if(unita==-1)   exit();
unitv=open(filev,1);
if(unitv==-1)   exit();
write_data(unita,unitv);
status=close(unita);
status=close(unitv);
```

```
}


cambia_formatta()
{
   extern int ftotali;
   int ireturn,status;
   char risp,out,record[80],flag,bell=7;
   erase_corso();
   sprintf(record,"VUOI CAMBIARE DISCHETTO O INIZIALIZZARNE UNO NUOVO (C/I)?");
   presenta(11,13,23,record);
   while(((risp=ci())!='c')&&(risp!='i')&&(risp!='C')&&(risp!='I'));
   if((risp=='i')||(risp=='I'))
   {
      esegue_inizializzazione();
      ireturn=ftotali;
   }
   else
   {
      flag=0;
      while(!flag)
      {
         zona_messaggi();
         sprintf(record,"  BATTI UN TASTO QUALSIASI QUANDO PRONTO.....");
         presenta(10,13,23,record);
         risp=ci();
         unitm=open(filem,2);
         if(unitm==-1)
         {
            printf("%c",bell);
            ireturn=0;
            status=close(unitm);
         }
         else
         {
            status=fscanf(unitm,"%d %d %f %s\n",&nfile1,&nfile2,
                          &nbytetot,&dummy);
            ireturn=nfile2;
         }
      }
   }
   return(ireturn);
}


esegue_inizializzazione()
{
/* extern int printer;  */
   extern int ftotali;
   int status;
   char risposta,bell=7,record[80];
   long lval;
   zona_messaggi();
   erase_corso();
   sprintf(record,"INSERIRE IL DISCHETTO DA INIZIALIZZARE NEL DRIVE A:");
```

```
    presenta(10,15,23,record);
    sprintf(record,"BATTERE UN TASTO QUALSIASI QUANDO PRONTO...");
    presenta(11,15,23,record);
    printf("%c",bell);
    risposta=ci();
    zona_messaggi();
    in_corso();
    azzera_righe(1);
    scr_rowcol(11,1);
    code=exec("C:\\COMMAND.COM","/cFORMAT A:");
    if(code==255)   exit();
    azzera_righe(0);
    fasce();
    lintest();
    zona_messaggi();
    unitm=creat(filem);
    if(unitm==-1)   exit();
    unitm=open(filem,2);
    if(unitm==-1)   exit();
    dates(date);
    times(time);
    status=fprintf(unitm,"0 %2d 360000.000000 %8c %8c\n",ftotali,date,time);
    lval=rewind(unitm);
    erase_corso();
}

azzera_righe(flag)
int flag;
{
    int ir,a,b,c;
    char *riga;

riga="================================================================================
=";
    for(ir=10; ir<21; ir++)
    {
       scr_rowcol(ir,0);
       scr_clrl();
    }
    if(flag==1)
    {
       presenta(10,1,23,riga);
       presenta(20,1,23,riga);
    }
}


zona_messaggi()
{
  char *spazi;
  spazi="                                                                      ";
  int ir,ic=10;
  for(ir=10; ir<15; ir++)
     presenta(ir,ic,23,spazi);
}

in_corso()
```

```
{
   char *mess;
   mess="          O P E R A Z I O N E    I N    C O R S O          ";
   presenta(21,11,71,mess);
}


erase_corso()
{
   char *mess;
   mess="                                                          ";
   presenta(21,11,7,mess);
}

   int possibili[8]={17,20,27,30,81,84,91,94};

   char *identi[12];
   char *id0="ORECCHIO DESTRO    -  PULSANTE DESTRO    ";
   char *id1="ORECCHIO DESTRO    -  PULSANTE SINISTRO  ";
   char *id2="ORECCHIO SINISTRO  -  PULSANTE DESTRO    ";
   char *id3="ORECCHIO SINISTRO  -  PULSANTE SINISTRO  ";
   char *id4="  TONO  ALTO       -  PULSANTE DESTRO    ";
   char *id5="  TONO  ALTO       -  PULSANTE SINISTRO  ";
   char *id6="ORECCHIO DESTRO    -  PULSANTE DESTRO    ";
   char *id7="ORECCHIO DESTRO    -  PULSANTE SINISTRO  ";

write_data(ia,iv)
int ia,iv;
{
   extern int   mediana1[],mediana2[],conl1,conl2;
   extern char  work_uditivo[];
   extern char  cognome[],nome[],giorno[],mese[],anno[];
   extern char  sesso,eta[],residenza[],indirizzo[];
   extern char  documento[],num_doc[],progressivo[],prev,street[CPR][CPR];
   extern float media[VPR][CPR],scarto[RPR][CPR],dimedia[],discarto[];
   extern float varianz[RPR][CPR],assolut[RPR][CPR],divarianz[],diassolut[];
   extern float medmodi[MPR][CPR];
   extern int   anni,numero_prog,gia_effettuate[];
   extern int   validi[RPR][CPR],errate[RPR][CPR],assenti[RPR][CPR];
   extern int   divalidi[],dierrate[],diassenti[];
   extern char possibilita[];
   int a,b,c,d,j,k,status,prove;
   int dato1,dato2,dato3,tipo,medi,jl,lprov,iprov;
   char risposta,ff=0x0C;
   char data_odierna[9];

   in_corso();
   data_int(data_odierna);
status=fprintf(ia,"\n- PROVE DEL: %s ----------------------------",
                     data_odierna);
   status=fprintf(ia,"\nNome soggetto  : %s %s ",cognome,nome);
   status=fprintf(ia,"\nData di nascita: %s/%s/%s",giorno,mese,anno);
   if(sesso=='M')
      status=fprintf(ia,"\nSesso          : Maschile");
   else
      status=fprintf(ia,"\nSesso          : Femminile");
   status=fprintf(ia,"\nEta`          : %d",anni);
```

```
    status=fprintf(ia,"\nIndirizzo      : %s",residenza);
    status=fprintf(ia,"\nProgressivo    : %04d",numero_prog);
    status=fprintf(ia,"\nArto prevalente: %c",prev);
    status=fprintf(ia,"\nNOTA           : %s",indirizzo);
 status=fprintf(ia,"\n-------------------------------------------------\n");
 status=fprintf(iv,"\n-------------------------------------------------\n");
 status=fprintf(iv,"\n -- PARAMETRI DI CALIBRAZIONE STIMOLATORE (HEX) ---\n");
    for(a=0; a<19; a++)
       status=fprintf(iv,"%0*x ",2,work_uditivo[a]);
 status=fprintf(iv,"\n\n");
 status=fprintf(iv,"\n-------------------------------------------------");
 status=fprintf(iv,"\n---    VALORI RELATIVI ALLE PROVE EFFETTUATE    ---");
 status=fprintf(iv,"\n-------------------------------------------------");
 status=fprintf(iv,"\n\n");
    for(tipo=0; tipo<3; tipo++)
    {
       if(tipo == GONOGOC)
       {
          lprov=3;
          iprov=0;
       }
       else
       {
          lprov=2;
          iprov=1;
       }
       for(prove=iprov; prove<=lprov; prove++)
       {
          if(street[prove][tipo]==0x55)
          {
             switch(tipo)
             {
                case GONOGOC:
                     status=fprintf(iv," *** GO NO GO CLASSICO   *** \n");
                     break;
                case GONOGON:
                     status=fprintf(iv," *** GO NO GO MODIFICATO *** \n");
                     break;
                case TIMERES:
                     status=fprintf(iv," ***      CHOICE   RT    *** \n");
                     break;
             }
status=fprintf(iv,"-------------------------------------------------\n");
status=fprintf(iv,"--- DATI RELATIVI ALL'ESAME ---------------------\n");
             status=fprintf(iv,"--- COMBINAZIONE: %s <-----\n",
                            identi[tipo*4+prove]);
             for(j=prove*40; j<(prove*40)+40; j+=10)
             {
                for(k=j; k<j+10; k++)
                   status=fprintf(iv,"%4d ",validi[k][tipo]);
                status=fprintf(iv,"\n");
             }
             if(tipo != GONOGOC)
             {
status=fprintf(iv,"-------------------------------------------------\n");
status=fprintf(iv,"--- VALORI PER MEDIANA ORECCHIO DESTRO ----------\n");
                jl=prove*20+(tipo-1)*40;
```

```
                for(j=jl; j<(jl+20); j+=10)
                {
                   for(k=j; k<j+10; k++)
                      status=fprintf(iv,"%4d ",mediana1[k]);
                   status=fprintf(iv,"\n");
                }
status=fprintf(iv,"--- VALORI PER MEDIANA ORECCHIO SINISTRO ------------\n");
                for(j=jl; j<(jl+20); j+=10)
                {
                   for(k=j; k<j+10; k++)
                      status=fprintf(iv,"%4d ",mediana2[k]);
                   status=fprintf(iv,"\n");
                }
status=fprintf(iv,"-----------------------------------------------------\n");
             }

 status=fprintf(iv,"RISPOSTE ERRATE     = %d\n",errate[prove][tipo]);
 status=fprintf(iv,"RISPOSTE ASSENTI    = %d\n",assenti[prove][tipo]);
 status=fprintf(iv,"VALORE MEDIO        = %6.1f\n",media[prove][tipo]);
 status=fprintf(iv,"DEVIAZIONE STANDARD = %6.1f\n",scarto[prove][tipo]);
             if(tipo == GONOGOC)
 status=fprintf(iv,"MEDIANA             = %6.1f\n",varianz[prove][tipo]);
             else
             {
                medi=prove*2;
 status=fprintf(iv,"MEDIANA  1          = %6.1f\n",medmodi[medi][tipo]);
 status=fprintf(iv,"MEDIANA  2          = %6.1f\n",medmodi[medi+1][tipo]);
 status=fprintf(iv,"MEDIA DELLE MEDIANE = %6.1f\n",medmed[prove][tipo]);
             }
 status=fprintf(iv,"VALORE ASSOLUTO SC. = %6.1f\n",assolut[prove][tipo]);
 status=fprintf(iv,"-----------------------------------------------------");
 status=fprintf(iv,"\n\n\n");
          }
       }
    }
/* fine prove */
}

lista_dischetto()
{
/*   extern int printer;  */
   char *nome,*riga1,*riga2,*cp,risp;
   char *mess1,*mess2,*mess3,*mess4,*mess5,*mess6,*mess7,*mess8,*vuoto;
   char risposta,dummy[80],record[80],bell=7,*file,*file_open;
   int  a,b,c,status,num,unitm,indice,jrig;
   int  nbyte1,nbyte2,leng,lin=18,fr=4,fc=5,tr=22,tc=75;
   float nbytetot;

   nome="A:-MASTER-.IDX";
   riga1="                                                    ";
   riga2="      L I S T A   D A T I   D E L   D I S C H E T T O      ";
   vuoto="                                                  ";
   mess1="     DISCHETTO NON INIZIALIZZATO CORRETTAMENTE     ";
   mess2="     MANCA IL FILE INDICE ...................      ";
   mess3="  BATTI UN TASTO QUALSIASI PER PROSEGUIRE ....     ";
   mess4="  Introduci un dischetto nell'unita` A: ......     ";
   mess5="                    <esc> per uscire                ";
```

```
mess6="    SCEGLI OPZIONE:                                    ";
mess7="        1. : LISTA NOMINATIVI MEMORIZZATI              ";
mess8="        2. : LISTA VALORI MEMORIZZATI                  ";

unitm=-1;
while(unitm==-1)
{
   scr_clr();
   presenta(0,10,71,riga1);
   presenta(1,10,71,riga2);
   presenta(2,10,71,riga1);
   presenta(23,10,71,mess5);
   fasce();
   lintest();
   presenta(10,12,23,vuoto);
   presenta(11,12,23,mess4);
   presenta(12,12,23,mess3);
   presenta(13,12,23,vuoto);
   risposta=ci();
   if(risposta==27)  return(27);
   unitm=open(nome,0);
   if(unitm==-1)
   {
      presenta(11,12,71,mess1);
      presenta(12,12,71,mess2);
      presenta(13,12,71,mess3);
      printf("%c",bell);
      risposta=ci();
      if(risposta==27)  return(27);
   }
}
status=fscanf(unitm,"%d %d %f %s\n",&nfile1,&nfile2,
                    &nbytetot,&record);
status=close(unitm);
sprintf(record,"    IL DISCHETTO CONTIENE %2d FILE(s) ARCHIVIATI    ",
                nfile1);
presenta(7,12,23,vuoto);
presenta(8,12,23,record);
presenta(9,12,23,vuoto);
presenta(10,12,23,mess6);
presenta(11,12,23,vuoto);
presenta(12,12,23,mess7);
presenta(13,12,23,vuoto);
presenta(14,12,23,mess8);
presenta(15,12,23,vuoto);
while(((risposta=ci())!='1')&&(risposta!='2')&&(risposta!=27));
if(risposta=='1')     file="ANAGRAFE";
if(risposta=='2')     file="VALORICA";
if(risposta==27)      return(27);
scr_scrup(lin,fr,fc,tr,tc);
jrig=4;
for(indice=1; indice<=nfile1; indice++)
{
   sprintf(file_open,"A:%8s.%0*d",file,3,indice);
   unitm=open(file_open,0);
   if(unitm==-1)
   {
```

```
            sprintf(record,"   MANCA IL FILE NUMERO %d   ",indice);
            presenta(12,12,71,record);
            printf("%c",bell);
            presenta(14,12,71,mess3);
            risposta=ci();
         }
         else
         {
            if(risposta=='1')
            {
               cp=fgets(dummy,80,unitm);  /* inserisce anche CR e LF */
               leng=strlen(dummy);
               dummy[leng-1]=0;              /* elimina CR e LF         */
               sprintf(record," %0*d -- %s ",3,indice,dummy);
               presenta(indice+3,5,23,record);
            }
            else
            {
               sprintf(record,"   NUMERO DEL FILE IN CORSO : %d  ",indice);
               presenta(3,15,87,record);
               leng=255;
               while(leng!=0)
               {
                  cp=fgets(dummy,80,unitm);
                  if(*cp!=0)
                  {
                     leng=strlen(dummy);
                     dummy[leng-1]=0;
                     presenta(jrig,5,2,dummy);
                     jrig++;
                     if(jrig==23)
                     {
                        presenta(23,12,71,mess3);
                        risp=ci();
                        if(risp==27)   return(27);
                        presenta(23,12,71,mess5);
                        scr_scrup(lin,fr,fc,tr,tc);
                        jrig=4;
                     }
                  }
               }
            }
         }
         status=close(unitm);
      }
      presenta(22,14,23,mess3);
      printf("%c",bell);
      risposta=ci();
   }


mediana(array,ncelle,fmedi)
int array[];
int ncelle;
float *fmedi;
{
```

```
    int temp,j,k,dato1,dato2;
    for(j=0; j<ncelle-1; j++)
    {
       for(k=j+1; k<ncelle; k++)
       {
          if(array[k] < array[j])
          {
             temp=array[j];
             array[j]=array[k];
             array[k]=temp;
          }
       }
    }
    j=ncelle/2;
    *fmedi=array[j];
    if((ncelle%2)==0)          /* --- se il numero di punti e` pari --- */
    {
       dato1=array[j-1];
       dato2=array[j];
       *fmedi=(dato1+dato2)/2.0;
    }
}

carica_array()
{
   identi[0]=id0;
   identi[1]=id1;
   identi[2]=id2;
   identi[3]=id3;
   identi[4]=id4;
   identi[5]=id4;
   identi[6]=id5;
   identi[7]=id5;
   identi[8]=id6;
   identi[9]=id6;
   identi[10]=id7;
   identi[11]=id7;
}
```

File: GESFILE.C

```
   int  macchina;          /* TIPO MACCHINA: 1 COMPLETA - 2 INCOMPLETA     */
   char arch_semplice[60]; /* PATH DEL FILE DI ARCHIVIO PER PROVE SEMPLICI */
   char arch_discrimi[60]; /* PATH DEL FILE DI ARCHIVIO PER PROVE DISCRIM. */
   char file_random[60];   /* PATH DEL FILE SEME RANDOM                    */
   char file_defuditiv[60]; /*FILE CONTENENTE I DEFAULT UDITIVO            */
   int  ftotali;            /*NUMERO FILE TOTALI DU DISCHETTO FORMATTATO   */

   int fp0,fp1,fp2,fp3,fp4,fp5,fp6;

/*
   DEFINIZIONE ARRAY PARAMETRI DI DEFAULT PER I TRE STIMOLATORI
*/

   char  def_uditivo[19];
   char  work_uditivo[19];
   char  diver_toni[4];

/*
   definizione array di memorizzazione dati archivio
*/

   float archivio[2048];
   int   osservazioni[2048];
   int   loop_pc;

gestisce_file()
{
   extern char per_status[];
   char *file_ini,*cp;
   char stringa[80],temporaneo[20];
   char *KEY1,*KEY2,*KEY3,*KEY4,*KEY5,*KEY6,*KEY7,*KEY8,*KEY9;
   char EQUA="=",SLAS="/";
   int  fp,num,result,lung,a,iseme,imacc;

   KEY1="MACCHINA";
   KEY2="ARC_SEMP";
   KEY3="ARC_DISC";
   KEY4="FILE_RAN";
   KEY6="UDIT_DEF";
   KEY8="FILE_FMT";
   KEY9="FREQUENZ";

   iseme=0;
   imacc=3;
   ftotali=18;
   loop_pc=85;                          /* frequenza di default       */
   file_ini="GO.INI";
   fp=open(file_ini,0);
   if (fp==-1)
   {
      messaggio(01);
      exit();
   }
   macchina=0;
```

```
legge_stringa:
   cp=fgets(stringa,80,fp);
   if(cp==0)
   {
      num=close(fp);
      if(iseme==0)
      {
         messaggio(02);
         exit();
      }
      if((macchina!=1)&&(macchina!=2))
      {
         messaggio(03);
         exit();
      }
      if(macchina==1)
      {
         if(imacc!=3)
         {
            messaggio(04);
            exit();
         }
      }
      else
      {
         per_status[2]=-1;    /* definisce assente lo stimolatore some */
         if(imacc<2)
         {
            messaggio(05);
            exit();
          }
       }
       return(num);
   }
   if(stringa[0]==';') goto legge_stringa;
   stringa[strlen(stringa)-1]=0;

/*
     ===== Cerca KEYWORD nel file di inizializzazione =====
*/

   if(search(stringa,KEY1)!=-1)
   {
      result=search(stringa,"=");
      if(result==-1)
         messaggio(06);
      else
      {
         cp=strcpy(&temporaneo,&stringa[result+1]);
         macchina=atoi(&temporaneo);
      }
      goto legge_stringa;
   }
   if(search(stringa,KEY2)!=-1)
   {
      result=search(stringa,"=");
      if(result==-1)
```

```
            messaggio(07);
        else
        {
            cp=strcpy(&arch_semplice,&stringa[result+1]);
            fp0=apri_file(arch_semplice,2);
        }
        goto legge_stringa;
    }
    if(search(stringa,KEY3)!=-1)
    {
        result=search(stringa,"=");
        if(result==-1)
            messaggio(08);
        else
        {
            cp=strcpy(&arch_discrimi,&stringa[result+1]);
            fp1=apri_file(arch_discrimi,2);
        }
        goto legge_stringa;
    }
    if(search(stringa,KEY4)!=-1)
    {
        result=search(stringa,"=");
        if(result==-1)
            messaggio(09);
        else
        {
            cp=strcpy(&file_random,&stringa[result+1]);
            fp2=apri_file(file_random,2);
            iseme=1;
        }
        goto legge_stringa;
    }
    if(search(stringa,KEY6)!=-1)
    {
        result=search(stringa,"=");
        if(result==-1)
            messaggio(12);
        else
        {
            cp=strcpy(&file_defuditiv,&stringa[result+1]);
            fp5=apri_file(file_defuditiv,0);
            legge_dati_default(fp5,def_uditivo,work_uditivo,19);
            legge_dati_default(fp5,diver_toni,diver_toni,4);
        }
        goto legge_stringa;
    }
    if(search(stringa,KEY8)!=-1)
    {
        result=search(stringa,"=");
        if(result==-1)
            messaggio(17);
        else
        {
            cp=strcpy(&temporaneo,&stringa[result+1]);
            ftotali=atoi(&temporaneo);
        }
```

```
            goto legge_stringa;
        }
        if((result=search(stringa,KEY9)) != -1)
        {
            result=search(stringa,"=");
            if(result==-1)
                messaggio(18);
            else
            {
                cp=strcpy(&temporaneo,&stringa[result+1]);
                loop_pc=atoi(&temporaneo);
            }
            goto legge_stringa;
        }
        goto legge_stringa;
    }


    search(stringa1,stringa2)
    char stringa1[],stringa2[];
    /*
        RICERCA LA STRINGA2 NELLA STRINGA1: RITORNA -1 NEL CASO NEGATIVO
        O L'INDICE RELATIVO AL PRIMO CARATTERE NEL CASO POSITIVO
    */
    {
        char first,char1,char2;
        int len1,len2,a,b,iret;
        first=stringa2[0];
        len1=strlen(stringa1);
        len2=strlen(stringa2);
        iret=-1;

        for(a=0; a<len1; a++)
        {
            if(stringa1[a]==first)
            {
                iret=a;
                first=0xFF;
            }
        }

        if(iret==-1) return(iret);
        b=iret;
        iret++;
        for(a=1; a<len2; a++)
        {
            if(stringa2[a]!=stringa1[iret])
            {
                iret=-1;
                return(iret);
            }
            iret++;
        }
        return(b);
    }

    apri_file(file_name,modo)
```

```
char file_name[];
int  modo;
{
   int fp;
   fp=open(file_name,modo);
   if (fp==-1)
   {
      messaggio(14);
      printf(" %s",file_name);
      exit();
   }
   return(fp);
}

inizializza_ran()
{
   int seed,num;
   num=fscanf(fp2,"%d",&seed);
   if(num==-1)
   {
      messaggio(15);
      exit();
   }
   srand(seed);        /*   INIZIALIZZA SEME PER NUMERI CASUALI   */
}


chiude_file_ran()
{
   int numero,num;
   long lval;

   lval=fseek(fp2,0L,0);
   numero=rand();
   num=fprintf(fp2,"%d",numero);
   if(num==-1)
   {
      messaggio(16);
      exit();
   }
   num=close(fp2);
}


carica_casuali(minimo,massimo,array,ncelle)
int minimo,massimo,array[],ncelle;
/*
   CARICA L'ARRAY COSTITUITO DA ncelle LOCAZIONI CON ALTRETTANTI VALORI
   TEMPORALI (in millisecondi) UTILIZZANDO LA ROUTINE DI GENERAZIONE
   NUMERI CASUALI RAND()
*/
{
   float nuovomassimo,valore;
   int indice,ival;

   nuovomassimo=massimo-minimo;
   for(indice=0; indice<ncelle; indice++)
```

```
    {
       ival=rand();
       valore=ival*nuovomassimo/32768.+minimo;
       array[indice]=valore;
    }
}


legge_dati_default(unit,array1,array2,ndata)
int  unit,ndata;
char array1[],array2[];    /* array dati di default e array di lavoro */
{
   int indice,num;
   char dato;
   for (indice=0; indice<ndata; indice++)
   {
      num=fscanf(unit,"%x",&dato);
      array1[indice]=dato;  /* carica array dati di default def_    */
      array2[indice]=dato;  /* carica array di lavoro       work_   */
   }
}

messaggio(indice)
int indice;
{
   printf("\nGESFILE> ");
   switch (indice)
   {
      case (01):  printf("IL FILE GO.INI NON ESISTE");
                  break;
      case (02):  printf("IMPOSSIBILE APRIRE FILE SEME RANDOM");
                  break;
      case (03):  printf("VALORE MACCHINA NON CORRETTO");
                  break;
      case (04):  printf("VALORI DI DEFAULT NON DEFINITI (1)");
                  break;
      case (05):  printf("VALORI DI DEFAULT NON DEFINITI (2)");
                  break;
      case (06):  printf("ERRORE DI SINTASSI CHIAVE MACCHINA");
                  break;
      case (07):  printf("ERRORE DI SINTASSI CHIAVE FILE_SEM");
                  break;
      case (08):  printf("ERRORE DI SINTASSI CHIAVE FILE_DIS");
                  break;
      case (09):  printf("ERRORE DI SINTASSI CHIAVE FILE_RAN");
                  break;
      case (10):  printf("ERRORE DI SINTASSI CHIAVE TEMPORANEO");
                  break;
      case (11):  printf("ERRORE DI SINTASSI CHIAVE DEFAULT VISIVO");
                  break;
      case (12):  printf("ERRORE DI SINTASSI CHIAVE DEFAULT UDITIVO");
                  break;
      case (13):  printf("ERRORE DI SINTASSI CHIAVE DEFAULT SOMESTESICO");
                  break;
      case (14):  printf("FILE NOT FOUND");
                  break;
      case (15):  printf("ERRORE SU LETTURA FILE SEME RANDOM");
```

```
                    break;
        case (16):  printf("ERRORE SU SCRITTURA FILE SEME RANDOM");
                    break;
        case (17):  printf("ERRORE DI SINTASSI CHIAVE FORMAT DISCHETTO");
                    break;
        case (18):  printf("ERRORE SUL VALORE DI FREQUENZA INTRODOTTO");
                    break;
    }
}


carica_archivio(unit,archivio,osservazioni,ncelle)
int   unit,ncelle;
float archivio[];
int   osservazioni[];
{
    int num;
    long lval;
    lval=rewind(unit);
    num=fread(archivio,sizeof(*archivio),ncelle,unit);
    num=fread(osservazioni,sizeof(*osservazioni),ncelle,unit);
}



scrive_archivio(unit,archivio,osservazioni,ncelle)
int   unit,ncelle;
float archivio[];
int   osservazioni[];
{
    int num;
    long lval;
    lval=rewind(unit);
    num=fwrite(archivio,sizeof(*archivio),ncelle,unit);
    num=fwrite(osservazioni,sizeof(*osservazioni),ncelle,unit);
}

crea_indice(flag,sesso,eta,stimolatore,arto1,arto2)
/*
     con FLAG = 0 si intende l'archivio semplice
     con FLAG = 1 si intende l'archivio discriminazione
*/

char flag,sesso;
int  eta,stimolatore,arto1,arto2;
{
    int indice;
/*  printf("\nDATI PASSATI: %c %d %d %d %d",sesso,eta,stimolatore,arto1,arto2); */
    if((eta>=18)&&(eta<=30))   eta=0;
    if((eta>30)&&(eta<=40))    eta=1;
    if((eta>40)&&(eta<=50))    eta=2;
    if((eta>50)&&(eta<=60))    eta=3;
    if(eta>60)                 eta=4;
    if(flag==0)
       indice=arto1*64+stimolatore*16+eta*2;
    else
       indice=arto2*128+arto1*32+stimolatore*16+eta*2;
```

```
    if(sesso=='M'||sesso=='m') indice++;
    indice=indice*4;
/*  printf("\nDATI CREATI: %c %d %d %d %d>%d",sesso,eta,stimolatore,
          arto1,arto2,indice);    ***/
    return(indice);
}

aggiorna_buffer(indice,archivio,osservazioni,x,y,w,z)
int   indice;
float archivio[];
int   osservazioni[];
float x,y,w,z;
{
    float temporaneo;
    int   itemporaneo;

    itemporaneo=osservazioni[indice];
    itemporaneo++;
    temporaneo=archivio[indice]*itemporaneo;
    temporaneo=temporaneo+x;
    temporaneo=temporaneo/itemporaneo;
    archivio[indice]=x;
    temporaneo=archivio[indice+1]*itemporaneo;
    temporaneo=temporaneo+y;
    temporaneo=temporaneo/itemporaneo;
    archivio[indice+1]=y;
    temporaneo=archivio[indice+2]*itemporaneo;
    temporaneo=temporaneo+w;
    temporaneo=temporaneo/itemporaneo;
    archivio[indice+2]=w;
    temporaneo=archivio[indice+3]*itemporaneo;
    temporaneo=temporaneo+z;
    temporaneo=temporaneo/itemporaneo;
    archivio[indice+3]=z;
    osservazioni[indice]=itemporaneo;
}
```

File: GO.C

```
/*-----------------------------------------------------------------

      PROGRAMMA DI GESTIONE DEL MODULO STIMOLATORE MULTISENSORIALE

  -----------------------------------------------------------------

   ---------------------------------------------------------
   ---          Descrizione array di utilizzo generale      -----
   ---------------------------------------------------------

   per_status[1] = byte di stato stimolatore uditivo.
                   Il byte di stato puo' avere i seguenti valori:
                   [0] - stato di power-up:  nessuna operazione
                   [1] - riconosciuto
                   [2] - programmato
                 [255] - errore durante il riconoscimento
                 [254] - errore durante la programmazione.

                   ---------------------------------------

    def_uditivo = contiene i parametri di default dello stimolatore
                  uditivo. (external GESFILE).
   work_uditivo = contiene i parametri da inviare allo stimolatore
                  uditivo per la relativa programmazione. Utilizzato /
                  modificato dalle procedure tx_uditivo()/edit_uditivo().
                  (external GESFILE).
     def_attr[] = contiene le indicazioni di stringa reverse per le
                  rappresentazioni su video dei valori di default.
         attr[] = contiene le indicazioni necessarie alla procedura
                  edit_visivo() per la visualizzazione dei valori
                  di default (modificati o no).

   ---------------------------------------------------------   */


   char    per_status[3]={0x00,0x00,0x00};

   char         attr[17]={0x0F,0x70,0x0F,0x70,0x0F,0x0F,0x0F,0x0F,0x70,
                          0x0F,0x0F,0x0F,0x0F,0x0F,0x70,0x0F,0x0F};

   char     def_attr[17]={0x0F,0x70,0x0F,0x70,0x0F,0x0F,0x0F,0x0F,0x70,
                          0x0F,0x0F,0x0F,0x0F,0x0F,0x70,0x0F,0x0F};

   char nuovo_paziente; /*** permette l'archiviazione dei dati
                             anagrafici e dei valori risultanti del
                             paziente di cui si sono introdotte le
                             informazioni anagrafiche              ***/
int  printer,printes,on_off;
char *printen;
int  numero_prog;

/* ============================================================ */
/* ===== PROGRAMMA PRINCIPALE ================================= */
/* ============================================================ */
```

```
main(argc,argv)
int  argc;
char *argv[];
{
    extern char def_uditivo[],work_uditivo[];
    extern char scr_mode;
/*  extern char type_dis,type_sem,esito_dis,esito_sem;     */
    char param,cara,flag,risposta,flagg;
    int buffer[20];
float dummy1[12],dummy2[10];
int status;

    scr_setup();          /* setta le globals interne relative al tipo di */
    carica_array();       /* inizializza array di puntatori a stringhe    */
    flag=0x00;            /* adattore grafico che e' presente sul sistema */
    nuovo_paziente=0;
/*  type_sem=esito_sem=0;    */
/*  type_dis=esito_dis=0;    */
    numero_prog=0;
    flagg=0;
    on_off=0;
    if(argc>1)
       if ((*argv[1]=='p')||(*argv[1]=='P'))    on_off=1;

    scr_clr();
    gestisce_file();
    ini_per();
    reset(1000);
    inizializza_ran();
    printen="PRN:";
    printer=open(printen,1);
    while(!flag)
    {
primo_menu:
       risposta=test_hardware();
       if(risposta==21)  goto fine_multi;
       risposta=tipo_di_prova(&flagg);
       if(risposta==21)  goto fine_multi;
       if(risposta==27)  goto primo_menu;
    }
fine_multi:
   scr_clr();
   exit_close();
}


   unsigned char schchar[25][81];
   int      unif;


fasce()
{
   videof(0);
   videof(1);
}
videof(flag)
```

```
int flag;
{
   int riga,colo;
   char barra[3]={0x20,0x20,0x00};

   if (flag)
      colo=1;
   else
      colo=77;
   for (riga=1; riga<24; riga++)
   {
      presenta(riga,colo,23,barra);
      strcpy(&schchar[riga][colo],"**");
      if(colo==1)
         schchar[riga][colo+2]=0x20;
   }
}

presenta(ir,ic,ia,is)
   int ir,ic,ia;
   char is[];
{
   int slen;
   extern char scr_mode;
scr_mode=3;
   scr_rowcol(ir,ic);
   if (scr_mode!=0x03)    /* testa se colori o monocromatico       */
   {                      /* se in monocromatico:                  */
      if (ia>0x0F)        /* controlla se e' settato il sottofondo */
         ia=0x70;         /* se si' setta il modo reverse (fondo   */
      else                /* bianco e carattere nero)              */
      {
         ia=0x07;         /* se no  setta il carattere bianco su   */
      }                   /* sfondo nero                           */
   }
   scr_aputs(is,ia);
   strcpy(&schchar[ir][ic],is);
   slen=strlen(is);
   schchar[ir][ic+slen]=0x20;
}



exit_close()
{
   chiude_file_ran();
   exit();
}

led()
{
   extern int loop_pc;
   extern portim;
   int casuali[10];
   int i,tempo;
   stop_timer();
   _outb(0x00,portim-3);
```

```
    load_timer(100);
    for(i=0; i<loop_pc; i++);
    while(legge_timer() > 10);
    stop_timer();
    _outb(0x01,portim-3);
    carica_casuali(1000,3000,casuali,10);
    load_timer(casuali[5]);
    for(i=0; i<loop_pc; i++);
    while((tempo=legge_timer()) > 50);
    stop_timer();
}


apre_video(name)
char *name;
{
    int a,b;
    unif=creat(name);
    a=close(unif);
    unif=open(name,1);
    for(a=0; a<25; a++)
    {
        for(b=0; b<80; b++)
            schchar[a][b]=0x20;
        schchar[a][80]=0;
    }
}


chiude_video()
{
    int stat,a;
    for(a=0; a<25; a++)
        stat=fprintf(unif,"\n%s",schchar[a]);
    stat=close(unif);
}
```

File: GO1.C

```
#include "go.h"


/*** valori globali per esame_semplice ***/
    int riga[40]={8,9,10,11,12,13,14,15,16,17,
                  8,9,10,11,12,13,14,15,16,17,
                  8,9,10,11,12,13,14,15,16,17,
                  8,9,10,11,12,13,14,15,16,17};
    int colo1[40]={8,8,8,8,8,8,8,8,8,8,
                  26,26,26,26,26,26,26,26,26,26,
                  44,44,44,44,44,44,44,44,44,44,
                  62,62,62,62,62,62,62,62,62,62};
    int colo2[40]={9,9,9,9,9,9,9,9,9,9,
                  27,27,27,27,27,27,27,27,27,27,
                  45,45,45,45,45,45,45,45,45,45,
                  63,63,63,63,63,63,63,63,63,63};
    int colo3[40]={14,14,14,14,14,14,14,14,14,14,
                  32,32,32,32,32,32,32,32,32,32,
                  50,50,50,50,50,50,50,50,50,50,
                  68,68,68,68,68,68,68,68,68,68};

struct disp01 {
   char stringa01[58];
   int posizione_riga01;
   int posizione_colo01;
   int attributo01;
};

struct disp01 screen01[08]={
"                                                    ",2,10,71,
"            T E S T    H A R D W A R E              ",3,10,71,
"                                                    ",4,10,71,
"STIMOLATORE UDITIVO  . . . . . .",11,15,7,
"PULSANTIERA  . . . . . .",14,15,7,
"                                                    ",20,10,71,
"                 <ENTER> PER CONTINUARE             ",21,10,71,
"                  ALT-Y  REINIZIALIZZA PROGRAMMA    ",22,10,71,
};
    char rispostaok[9]="PRESENTE";
    char rispostano[9]="ASSENTE ";

test_hardware()
{
    extern char per_status[];
    extern int macchina;
    extern unsigned portim;
  int irrisp[4]={9,11,14,15};
  int icrisp=51,attribok=32,attribno=71;
  char indirizzi[3]={0x50,0x70,0x90};
  char key,pulsanti,out,cic;
    int  a,b,c,d,istimolo;
    unsigned word,word1,word2,word3,word4;

    scr_clr();
```

```
    for (a=0; a<8; a++)
    presenta(screen01[a].posizione_riga01,screen01[a].posizione_colo01,
           screen01[a].attributo01,screen01[a].stringa01);
    fasce();

    /* inizio test su periferiche collegate ------
                                esegue cicli di riconoscimento ------------*/
    reset(1000);
    istimolo=1;
    trasmette_comandi('U','1');          /* ciclo di riconoscimento uditivo */
    if (per_status[istimolo]==1)
       presenta(irrisp[istimolo],icrisp,attribok,rispostaok);
    else
       presenta(irrisp[istimolo],icrisp,attribno,rispostano);

/*--- Testa presenza quattro pulsanti ---------------------------- */
    istimolo+=1;
    pulsanti=_inb(portim-2)&0x0F;
    cic=_inb(portim-2)&0x10;
    if ((pulsanti==0x0F)&&(cic==0))
       presenta(irrisp[istimolo],icrisp,attribok,rispostaok);
    else
       presenta(irrisp[istimolo],icrisp,attribno,rispostano);

/* --- Test sul funzionamento dei timer ----------------------------*/

    word=0x1234;
    load_timer(word);
    start_timer();
    for(b=0; b<500; b++);    /*      WAIT      */
    word4=legge_timer();
    for(b=0; b<500; b++);    /*      WAIT      */
    word1=legge_timer();
    for(b=0; b<500; b++);    /*      WAIT      */
    word2=legge_timer();
    for(b=0; b<500; b++);    /*      WAIT      */
    word3=legge_timer();
    stop_timer();
/*  printf("IMPOSTATO %u -- 1-2-3 %u %u %u",word,word1,word2,word3); */
    if(word3==word2) printf("  --- FAIL");
    if(word2==word1) printf("  --- FAIL");
    if(word1==word4) printf("  --- FAIL");

/* --- ATTESA COMANDO ------------------------------------------- */

    while (((key=ci())!=21)&&(key!=13));
    return(key);
}
/*
=============================================================================
*/

   int flop;
   unsigned char tipo_PR;

/*
=============================================================================
```

```
*/


struct disp02 {
char stringa02[55];
int posizione_riga02;
int posizione_colo02;
int attributo02;
};

struct disp02 screen02[9]={
"                                                    ",2,10,71,
"              T I P O   D I   P R O V A             ",3,10,71,
"                                                    ",4,10,71,
"1. : GO no GO classico                ",10,24,2,
"2. : GO no GO modificato              ",12,24,2,
"3. : Choice RT                        ",14,24,2,
"                 <ESC> menu' precedente             ",18,10,71,
"ALT-D = DOS              ALT-Y Reinizializza programma",19,10,71,
"                 Introduci: 1 - 2 - 3               ",20,10,71,
};

tipo_di_prova(flag)
char *flag;
{
    extern int macchina,printer,on_off;
    char risposta,key;
    char op1=0x31,op2=0x32,op3=0x33;
    int  a,status;
    risposta=0x00;
    while (!risposta)    /* ---- loop generale -------*/
    {
       scr_clr();
       for (a=0; a<9; a++)
       presenta(screen02[a].posizione_riga02,screen02[a].posizione_colo02,
              screen02[a].attributo02,screen02[a].stringa02);
       fasce();
       while (((key=ci())!=21)&&(key!=27)&&(key!=32)&& /* ALT-Y o ESCape */
               (key!=op1)&&(key!=op2)&&(key!=op3));    /* Numerico       */
       apre_temporaneo();
       switch (key)
       {
          case 32:   system_op();
                     break;
          case 21:   exit_close();
                     break;
          case 27:   risposta=0x0FF;
                     return(27);
                     break;
          case '1':  uditivo('2');
                     if(anagrafe_ok()==27)
                        risposta=0;
                     else
                     {
                        tipo_PR=GONOGOC;
                        resp_time();
                     }
```

```
                    break;
            case '2':  uditivo('2');
                       if(anagrafe_ok()==27)
                          risposta=0;
                       else
                       {
                          tipo_PR=GONOGON;
                          resp_time();
                       }
                       break;
            case '3':  uditivo('2');
                       if(anagrafe_ok()==27)
                          risposta=0;
                       else
                       {
                          tipo_PR=TIMERES;
                          resp_time();
                       }
                       break;
         }
      }
/*  return(key);    */
}

/*************************************************************************/
   float media[RPR][CPR],scarto[RPR][CPR],dimedia[2],discarto[2];
   float varianz[RPR][CPR],assolut[RPR][CPR],divarianz[2],diassolut[2];
   float medmodi[MPR][CPR];
   int validi[VPR][CPR],errate[RPR][CPR],assenti[RPR][CPR];
   int divalidi[2],dierrate[2],diassenti[2];          /* MODIFICATI */
   int voto[RPR][CPR],divoto[2],gia_effettuate[CPR],incurs;
   char effettuate[8],possibilita[8];
   char scelta[14],attr[14],inside[14],street[CPR][CPR];
/*************************************************************************/

anagrafe_ok()
{
    extern int  numero_prog;
    extern char nuovo_paziente;
    extern int  printer,on_off;
    extern char cognome[],nome[],giorno[],mese[],anno[];
    extern char sesso,eta[],residenza[],indirizzo[];
    extern char documento[],num_doc[],progressivo[];
    extern char type_dis,type_sem,esito_dis,esito_sem;
    int a,b,status;
    char *mess0,*mess1,*vuoi_anagrafe,risposta,bell=7,test;
    char data[10],tempo[10],setta[3],rese[3],risparc,ff=12;
    vuoi_anagrafe="  VUOI INTRODURRE I DATI ANAGRAFICI (S/N)?  ";
    mess0="          DATI NON COMPLETAMENTE ARCHIVIATI          ";
    mess1="      VUOI ARCHIVIARE PRIMA DI PROSEGUIRE (S/N)?     ";
    setta[0]=0x1B;
    setta[1]='E';
    setta[2]=0;
    rese[0]=0x1B;
    rese[1]='F';
    rese[2]=0;
    for(a=0; a<14; a++)
```

```
 {
    scelta[a]=0;
    attr[a]=2;
    inside[a]=0;
 }
 if(numero_prog>0)
 {
    presenta(16,15,23,vuoi_anagrafe);
    printf("%c",bell);
    risposta=ci();
    if((risposta=='s')||(risposta=='S'))
    {
       test = nuovo_paziente&3;
       if(test != 3)
       {
          printf("%c",bell);
          presenta(15,10,87,mess0);
          presenta(16,10,87,mess1);
          while(((risparc=ci())!='s')&&(risparc!='S')&&
                (risparc!='n')&&(risparc!='N'));
          if((risparc=='s')||(risparc=='S'))
             scelta_archiviazione(0);
       }
    }
 }
 else
    risposta='s';

 if(risposta=='s'||risposta=='S')
 {
    if(anagrafe()==27) return(27);
    for(a=0; a<CPR; a++)
    {
        for(b=0; b<3; b++)
        {
           street  [a] [b] = 0;
           errate  [a] [b] = 0;
           assenti [a] [b] = 0;
           media   [a] [b] = 0.0;
           scarto  [a] [b] = 0.0;
           varianz [a] [b] = 0.0;
           assolut [a] [b] = 0.0;
        }
    }
    type_sem=esito_sem=0;
    for(a=0; a<2; a++)
       divalidi[a]=0.0;
    for(a=0; a<VPR; a++)
    {
        for(b=0; b<3; b++)
          validi[a][b]=0;
    }
    for(a=0; a<8; a++)
    {
       effettuate[a]=0;
       possibilita[a]=0;
       dierrate[a]=0;
```

```
            diassenti[a]=0;
            dimedia[a]=0.0;
            discarto[a]=0.0;
            divarianz[a]=0.0;
            diassolut[a]=0.0;
          }
          type_dis=esito_dis=0;
          nuovo_paziente=0;
          gia_effettuate[tipo_PR]=0;
          data_int(data);
          times(tempo);
          if(on_off!=0)
          {
            status=fprintf(flop,"%c\n ---------- %s --- %s--------------------",ff,
                           data,tempo);
            status=fprintf(flop,"\nNUMERO PROGRESSIVO: %s%d%s",setta,
                           numero_prog,rese);
            status=fprintf(flop,"\nCOGNOME: %s%s%s     NOME: %s%s%s",
                           setta,cognome,rese,setta,nome,rese);
            status=fprintf(flop,
                     "\nSESSO: %s%c%s       DATA DI NASCITA: %s%s-%s-%s%s",
                           setta,sesso,rese,setta,giorno,mese,anno,rese);
            status=fprintf(flop,"\nINDIRIZZO: %s%s%s",
                           setta,indirizzo,rese);
            status=fprintf(flop,"\nRESIDENTE A: %s%s%s",
                           setta,residenza,rese);
            status=fprintf(flop,"\nESTREMI DI RICONOSCIMENTO: %s%s %s%s",
                           setta,documento,num_doc,rese);
            status=fprintf(flop,
                   "\n ------------------------------------------------------\n");
            stampa_temporaneo();
          }
      }
      apre_temporaneo();
}

/*************************************************************************/

struct disp04 {
char stringa04[59];
int posizione_riga04;
int posizione_colo04;
int attributo04;
};

struct disp04 screen04[22]={
"                                                          ",2,10,71,
"           D A T I   A N A G R A F I C I                  ",3,10,71,
"                                                          ",4,10,71,
"COGNOME: ",7,4,7,
"                        ",7,13,23,
" NOME: ",7,40,7,
"                          ",7,47,23,
"DATA DI NASCITA: ",9,4,7,
"  /  /  ",9,21,23,
" SESSO: ",9,31,7,
"       ",9,39,23,
```

```
"ETA` ANNI: ",9,48,7,
"                ",9,59,23,
"INDIRIZZO:    ",11,4,7,
"                                                          ",11,17,23,
"NOTA : ",13,4,7,
"                                                                  ",13,11,23,
"ARTO PREVALENTE ",17,4,7,
"    ",17,20,23,
"                                                           ",20,10,71,
"              <ENTER> Abilitazione/Continua                ",21,10,71,
"                                                           ",22,10,71,
};

    char cognome[26],nome[28],giorno[3],mese[3],anno[3];
    char sesso,eta[16],residenza[48],indirizzo[58];
    char documento[20],num_doc[25],progressivo[14],prev[2],trixer;
    int  lunghezze[11]={26,28,2,2,2,1,16,48,58,1,14};
    int  anni;
    unsigned indirizzi[11]={&cognome,&nome,&giorno,&mese,&anno,
                  &sesso,&eta,&residenza,&indirizzo,
                  &prev,&progressivo};
    int  stipo[11]={0,0,1,1,1,3,0,2,2,3,0};
    char idents[2]={'M','F'};
    char identa[2]={'D','S'};

anagrafe()
{
    extern int numero_prog;
    extern char per_status[];
    int  iriga[11]={7,7,9,9,9,9,9,11,13,17,17};
    int  icolo[11]={13,47,21,24,27,39,59,17,11,20,52};
    char buffer[20],today[9],bell=0x07,fanni;
    char day[3],mou[3],yea[3];
    char *messaggio,*abblenka,risposta,record[80];
    int  a,b,igiorno,imese,ianno,idata,ioggi,nonvalidi;

    scr_clr();
    messaggio=" PREGO CONFERMARE LA CORRETTA INTRODUZIONE DATI (S/N)  ";
    abblenka="                                                        ";
    today[8]=0;
    giorno[2]=0;
    mese[2]=0;
    anno[2]=0;
    day[2]=0;
    mou[2]=0;
    yea[2]=0;
    prev[2]=0;
    giorno[0]=giorno[1]='0';
    mese[0]=mese[1]='0';
    anno[0]=anno[1]='0';
    for (a=0; a<22; a++)
    presenta(screen04[a].posizione_riga04,screen04[a].posizione_colo04,
           screen04[a].attributo04,screen04[a].stringa04);
    fasce();
    b=0;
    nonvalidi=0;
 while ((nonvalidi==0)||(nonvalidi==1))
```

```
{
  for (a=b; a<6; a++)
  {
  attesa:
     if(attende_char(iriga[a],icolo[a],151,lunghezze[a],indirizzi[a],
                  stipo[a],idents)==0x1B)
                  return(27);
     presenta(iriga[a],icolo[a],23,indirizzi[a]);
     if (((igiorno=atoi(&giorno)) < 0) || (igiorno > 31))
     {
        presenta(iriga[a],icolo[a],148,indirizzi[a]);
        printf("%c",bell);
        goto attesa;
     }
     if (((imese=atoi(&mese)) < 0) || (imese > 12))
     {
        presenta(iriga[a],icolo[a],148,indirizzi[a]);
        printf("%c",bell);
        goto attesa;
     }
  }
  ianno=atoi(&anno);
  idata=ianno*365+imese*30+igiorno;
  data_int(today);
  day[0]=today[0];
  day[1]=today[1];
  mou[0]=today[3];
  mou[1]=today[4];
  yea[0]=today[6];
  yea[1]=today[7];
  igiorno=atoi(&day);
  imese=atoi(&mou);
  ianno=atoi(&yea);
  ioggi=ianno*365+imese*30+igiorno;
  anni=(ioggi-idata)/365.0;
  fanni=0;
  if(anni>35)
  {
     sprintf(record,"      %d      ",anni);
     presenta(iriga[6],icolo[6],23+128,record);
     printf("%c",bell);
     fanni=255;
  }
  else
  {
     sprintf(record,"      %d      ",anni);
     presenta(iriga[6],icolo[6],23,record);
     nonvalidi=0;
     fanni=0;
     b=0;
  }
  if(fanni!=0)
  {
     nonvalidi=1;
     b=2;
  }
  if(nonvalidi==0)
```

```
    {
       for (a=7; a<10; a++)
       {
          if(attende_char(iriga[a],icolo[a],151,lunghezze[a],indirizzi[a],
                          stipo[a],identa)==0x1B)
                          return(27);
          presenta(iriga[a],icolo[a],23,indirizzi[a]);
       }
       presenta(19,10,199,messaggio);
       printf("%c",bell);
       while(((risposta=ci())!=83)&&(risposta!=115)&&
             (risposta!=78)&&(risposta!=110));
       presenta(19,10,7,abblenka);
       if((risposta==78)||(risposta==110))
          b=0;
       else
          nonvalidi=2;
    }
  }
  numero_prog++;
}

/*
   GESTISCE I CAMPI NELLA VIDEATA ANAGRAFE
*/

int  attende_char(iriga,icolo,attrib,nchar_max,buffer,tipo,array)
int  iriga,icolo,nchar_max;
char attrib,*buffer;
int  tipo;
char array[];
{
   char carattere,flag,oktoload;
   char string2[2]={0X00,0X00};
   char bell=7,*spazi;
   int indice,posizione,finest;
   posizione=icolo;
   indice=0;
   flag=0;
   spazi="                                                                    ";
   attrib=attrib&0xF4;       /* sfondo come richiesto,caratteri rossi */
   attrib=attrib|0x07;
   scr_rowcol(iriga,icolo);
   while(!flag)
   {
      carattere=ci();
      if((carattere>96)&&(carattere<123))
          carattere-=32;
      switch (carattere)
      {
         case 8:  if(indice!=0)
                  {
                     posizione--;
                     indice--;
                     scr_rowcol(iriga,posizione);
                     string2[0]=0X20;
                     presenta(iriga,posizione,attrib,string2);
```

```
                    scr_rowcol(iriga,posizione);
                }
                break;

        case 13: if(indice!=0)
                {
                   buffer[indice]=0;
                   finest=nchar_max-indice;
                   spazi[finest]=0x00;
                   scr_aputs(spazi,attrib);
                   spazi[finest]=0x20;
                   flag=0xFF;
                }
                else
                   if((tipo==3)||(tipo==1))
                      flag=0;
                   else
                      flag=0xFF;
                break;
        case 21: exit_close();
                break;
        case 27: return(27);
                break;
        default: oktoload=0;
                switch (tipo)
                {
                   case 0: if ((isalpha(carattere)) || (carattere==0x20))
                              oktoload=1;
                           break;
                   case 1: if (isdigit(carattere))
                              oktoload=1;
                           break;
                   case 2: if ((isalnum(carattere)) || (carattere==0x20))
                              oktoload=1;
                           break;
                   case 3: if (isalpha(carattere))
                              if ((carattere==array[0])||
                                  (carattere==array[1]))  oktoload=1;
                           break;
                }
                if (oktoload!=0)
                {
                   string2[0]=carattere;
                   presenta(iriga,posizione,attrib,string2);
                   buffer[indice]=carattere;
                   indice++;
                   posizione++;
                   if(indice>nchar_max-1)    flag=0xFF;
                }
                else
                   printf("%c",bell);
                break;
      }
   }
}

/*
```

```
==========================================================================
*/

apre_temporaneo()
{
   extern int on_off,printer;
   char *floptemp="FILETEMP.DAT";
   int status;
   if(on_off!=0)
   {
      flop=creat(floptemp);
      status=close(flop);
      flop=open(floptemp,1);
   }
}

/*
==========================================================================
*/

stampa_temporaneo()
{
   extern int flop,printer,on_off;
   char *floptemp="FILETEMP.DAT";
   char record[132],sta;
   int status;
   if(on_off!=0)
   {
      status=close(flop);
      flop=open(floptemp,0);
      while((sta=fgets(record,132,flop))!=0)
          fprintf(printer,"%s",record);
      status=close(flop);
   }
}

/*
==========================================================================
*/

struct disp05 {
char stringa05[62];
int posizione_riga05;
int posizione_colo05;
int attributo05;
};

struct disp05 screen05[16]={
"                                                          ",1,10,23,
"                                                          ",2,10,71,
"           S E S S I O N E   I N   C O R S O              ",3,10,71,
"                                                          ",4,10,71,
"      USA LE FRECCE <>    PER SELEZIONARE LA MISURA       ",5,10,23,
"--------------------------",9,25,7,
"!ORECCHIO DS !ORECCHIO DS !",10,25,7,
"!PULSANTE DS !PULSANTE SN !",11,25,7,
"--------------------------",12,25,7,
```

```
"!           !           !",13,25,7,
"--------------------------",14,25,7,
"1.: ESECUZIONE MISURA      2.: VISUALIZZAZIONE DATI ACQUISITI",16,8,2,
"3.: ARCHIVIAZIONE DATI     4.: RIPROGRAMMAZIONE STIMOLATORE",18,8,2,
"       ALT-Y  reinizializza programma                    ",20,10,71,
"       <ESC>  menu` precedente                           ",21,10,71,
"              Introduci: 1 - 2 - 3 - 4                   ",22,10,71,
};

   char curriga[4]={13,13,13,13};
   char curcolo[4]={13,26,39,52};
   char cursore[13]={32,32,32,32,32,32,32,32,32,32,32,32,0};
   char attributo_on=32,attributo_off=7;
   char stim_selezionato; /*           1:UDITIVO                          */
   char arto_selezionato; /* contiene gia' la maschera corrispondente  */
   char oto_sel;          /* 0:destro  1:sinistro                       */
   char apprendarray[3];  /* [1]:UDITIVO                                */
                          /* 00:DA EFFETUARSI APPRENDIMENTO             */
                          /* FF:APPRENDIMENTO GIA` EFFETTUATO           */
   char arto_numerico;    /* 0:mds  1:msn                               */
   char arto_num1,arto_num2;

resp_time()
{
   extern int macchina;
   extern char per_status[];
   extern printer;
   int  a,b,iflag,istep,status,nliberi,posm,posn;
   char ch,out,risposta,*finesec;
   char flag=0x00;
   char *messag,bell=0x07;
   char *str1,*str2,nflag,somok;   /* soglia stim. somestesico */
   somok=0;
   apprendarray[1]=0x00;
   messag="   ERRORE SULLO STIMOLATORE SELEZIONATO   ";
   finesec="1.: ESECUZIONE MISURA    ";

/***** status=fprintf(flop,"STATI> %d %d %d\n",
              per_status[0],per_status[1],per_status[2]); ***************/

/* NEL CASO DI ASSENZA DI TUTTI E TRE GLI STIMOLATORI ESCE DALLA FUNZIONE */
   if((per_status[1]!=1)&&(per_status[1]!=2))
   {
       scr_clr();
       printf("\n*** IMPOSSIBILE ATTIVARE LA FUNZIONE RICHIESTA ***");
       printf("\n***   Batti un tasto qualsiasi per proseguire  ***");
       ch=ci();
       return;
   }
      /* per usare i simulato metti la fine di commento su questa riga */
      /* e toglilo dalla fine della riga NE CASO DI ASSENZA ...        */
      /* togli i commenti dalla riga successiva                        */
      /*   per_status[1]=2;                                            */

   flag=0x00;
   switch(tipo_PR)
   {
```

```
 case GONOGOC:
    posm=3;
    posn=0;
    strcpy(screen05[5].stringa05,
          "-------------------------------------------------------");
    strcpy(screen05[6].stringa05,
          "!ORECCHIO DS !ORECCHIO DS !ORECCHIO SN !ORECCHIO SN !");
    strcpy(screen05[7].stringa05,
          "!PULSANTE DS !PULSANTE SN !PULSANTE DS !PULSANTE SN !");
    strcpy(screen05[8].stringa05,
          "-------------------------------------------------------");
    strcpy(screen05[9].stringa05,
          "!              !            !            !            !");
    strcpy(screen05[10].stringa05,
          "-------------------------------------------------------");
    screen05[5].posizione_colo05=12;
    screen05[6].posizione_colo05=12;
    screen05[7].posizione_colo05=12;
    screen05[8].posizione_colo05=12;
    screen05[9].posizione_colo05=12;
    screen05[10].posizione_colo05=12;
    break;

 case GONOGON:
    posm=2;
    posn=1;
    strcpy(screen05[5].stringa05,
          "---------------------------");
    strcpy(screen05[6].stringa05,
          "! TONO ALTO  ! TONO ALTO  !");
    strcpy(screen05[7].stringa05,
          "!PULSANTE DS !PULSANTE SN !");
    strcpy(screen05[8].stringa05,
          "---------------------------");
    strcpy(screen05[9].stringa05,
          "!             !            !");!
    strcpy(screen05[10].stringa05,
          "---------------------------");
    screen05[5].posizione_colo05=25;
    screen05[6].posizione_colo05=25;
    screen05[7].posizione_colo05=25;
    screen05[8].posizione_colo05=25;
    screen05[9].posizione_colo05=25;
    screen05[10].posizione_colo05=25;
    break;

 case TIMERES:
    posm=2;
    posn=1;
    strcpy(screen05[5].stringa05,
          "---------------------------");
    strcpy(screen05[6].stringa05,
          "!ORECCHIO DS !ORECCHIO DS !");
    strcpy(screen05[7].stringa05,
          "!PULSANTE DS !PULSANTE SN !");
    strcpy(screen05[8].stringa05,
          "---------------------------");
```

```
            strcpy(screen05[9].stringa05,
                  "!              !              !");!
            strcpy(screen05[10].stringa05,
                  "--------------------------");
            screen05[5].posizione_colo05=25;
            screen05[6].posizione_colo05=25;
            screen05[7].posizione_colo05=25;
            screen05[8].posizione_colo05=25;
            screen05[9].posizione_colo05=25;
            screen05[10].posizione_colo05=25;
            break;
      }

nuovo_display:

      scr_clr();
      for (a=0; a<16; a++)
      presenta(screen05[a].posizione_riga05,screen05[a].posizione_colo05,
             screen05[a].attributo05,screen05[a].stringa05);
      fasce();
      lintest();
      if(tipo_PR==GONOGOC)
         nliberi=4;
      else
         nliberi=2;
      incurs=posn;
/*    incurs=-1;    */
      for(a=posn; a<=posm; a++)
      {
         if(street[a][tipo_PR]==0x55)
         {
             presito(a);          /* scrive ESEGUITA in corrispondenza */
             nliberi--;
         }
         if(street[a][tipo_PR]==0xFF)
             nliberi--;
      }
      if(nliberi==0)
         presenta(16,8,71+128,finesec);
      if(incurs!=-1)
      {
         flag=0x00;
         if(street[incurs][tipo_PR] != 0)
         {
            istep=calcola_libero(incurs,1,posm,posn);
            if(istep!=-1)
            {
               iflag=street[incurs+istep][tipo_PR];
               if(iflag==0)
               {
                   incurs+=istep;
                   write_cursore(incurs);
               }
            }
         }
         else
         {
```

```
      write_cursore(incurs);
   }
}
else
{
   flag=0xFF;
}
while(!flag)
{
   while(((ch=ci())!='1')&&(ch!='2')&&(ch!='3')&&(ch!='4')&&(ch!=0)
                         &&(ch!=21)&&(ch!=27));
   switch (ch)
   {
      case 0:   while(((ch=ci())!=75)&&(ch!=77)&&(ch!=21));
                switch (ch)
                {
                   case 75:
                            if(incurs>posn)
                            {
                               istep=calcola_libero(incurs,0,posm,posn);
                               if(istep!=-1)
                               {
                                  iflag=street[incurs-istep][tipo_PR];
                                  if(iflag==0)
                                  {
                                     clear_cursore(incurs);
                                     incurs-=istep;
                                     write_cursore(incurs);
                                  }
                               }
                            }
                            break;

                   case 77:
                            if(incurs<posm)
                            {
                               istep=calcola_libero(incurs,1,posm,posn);
                               if(istep!=-1)
                               {
                                  iflag=street[incurs+istep][tipo_PR];
                                  if(iflag==0)
                                  {
                                     clear_cursore(incurs);
                                     incurs+=istep;
                                     write_cursore(incurs);
                                  }
                               }
                            }
                            break;
                   case 21: flag=0xFF;
                            break;
                }
                break;
     case '1': /*          pstree();              */
                if(street[incurs][tipo_PR] != 0x55)
                {
                   determina_stimart(incurs);
```

```
                        reset(500);
                        trasmette_comandi('U','5');
                          if(per_status[stim_selezionato]!=2)
                        {
                           presenta(19,10,23,messag);
                           printf("%c",bell);
                           risposta=ci();
                           goto nuovo_display;
                        }
                        out=esame_apprendimento();
                        if(out==21)   exit_close();
                        if(out!=27)
                        {
                            if(out==1)                          /* esame          */
                           {
                              if(esame_semplice(incurs)!=27)
                                   street[incurs][tipo_PR]=0x55;
                              flag=0x55;
                           }
                           else
                           {                                    /* apprendimento */
                              if(apprendimento_semplice()!=27)
                              {
                                 apprendarray[stim_selezionato]=0xFF;
                                 if(esame_semplice(incurs)!=27)
                                     street[incurs][tipo_PR]=0x55;
                              }
                              flag=0x55;
                           }
                        }
                     }
      /*             pstree();        */
                     break;
         case '2':   visualizza_dati('S');
                     flag=0x55;
                     break;
         case '3':   if(scelta_archiviazione(0)==27)
                        flag=0x55;
                     else
                        flag=0xFF;
                     break;
         case '4':   determina_stimart(incurs);
                     riprogramma_stimolatore();
                     flag=0x55;
                     break;
         case 21:    exit_close();
                     break;
         case 27:    return(27);
                     break;
      }
   }
   if(flag==0x55) goto nuovo_display;
}


clear_cursore(posizione)
int posizione;
```

```
{
   presenta(curriga[posizione],curcolo[posizione],
            attributo_off,cursore);
}


write_cursore(posizione)
int posizione;
{
   presenta(curriga[posizione],curcolo[posizione],
            attributo_on,cursore);
}

presito(posizione)
int posizione;
{
   char *esito;
   esito=" ESEGUITA  ";
   presenta(curriga[posizione],curcolo[posizione],71,esito);
}


messaggio_assenza(posizione)
int posizione;
{
    char *assente;
    assente=" ASSENTE ";
    assente[9]=0;
    presenta(curriga[posizione],curcolo[posizione],71,assente);
    street[posizione][tipo_PR]=0xFF;
}


calcola_libero(posizione,direzione,posm,posn)
int posizione,direzione;
int posm,posn;
{
   int iret,istep,indice;
   istep=256;
   if(direzione==0)
   {
      for(indice=posizione-1; indice>=posn; indice--)
      {
         if(street[indice][tipo_PR]==0)
         {
            istep=indice;
            iret=posizione-istep;
            indice=-1;
         }
      }
   }
   else
   {
      for(indice=posizione+1; indice<=posm; indice++)
      {
         if(street[indice][tipo_PR]==0)
         {
```

```
            istep=indice;
            iret=istep-posizione;
            indice=posm;
          }
        }
    }
    if(istep==256)
       iret=-1;
    return(iret);
}

determina_stimart(posizione)
int posizione;
{
   int risp,lpos;
   char oto[4]           = {0,0,1,1};
   char arto_risposta[4] = {1,2,1,2};
   char artnum[4]        = {0,1,0,1};

   lpos=posizione;
   if(tipo_PR != GONOGOC)
       lpos--;

   stim_selezionato = 1;
   arto_selezionato = arto_risposta[lpos];
   oto_sel          = oto           [lpos];
   arto_numerico    = artnum        [lpos];
}



riprogramma_stimolatore()
{
   extern char per_status[];
   char *mess1,*mess2,bell=0x07,risposta;
   mess1="  STIMOLATORE NON INIZIALIZZATO  ";
   mess2="   STIMOLATORE NON PROGRAMMATO   ";
   if((per_status[1]==1)||(per_status[1]==2))
   {
      uditivo('4');
      trasmette_comandi('U','5');
      if(per_status[1]!=2)
      {
         presenta(19,10,23,mess2);
         printf("%c",bell);
         risposta=ci();
      }
   }
   else
   {
      presenta(19,10,23,mess1);
      printf("%c",bell);
      risposta=ci();
   }
}

/* ================================================================== */
```

```
pstree()
{
   scr_rowcol(2,1);
   printf("%d %d %d %d",street[0][tipo_PR],
                        street[1][tipo_PR],
                        street[2][tipo_PR],
                        street[3][tipo_PR]);
}
```

File: GO2.C

```
#include "go.h"

/*************************************************************************/
   int mediana1[120],mediana2[120];
   float medmed[RPR][CPR];
   int casuali[100],casual0[100],tcasual[100];
/*************************************************************************/
   extern float media[RPR][CPR],scarto[RPR][CPR],dimedia[],discarto[];
   extern float varianz[RPR][CPR],assolut[RPR][CPR],divarianz[],diassolut[];
   extern float medmodi[MPR][CPR];
   extern int   validi[VPR][CPR],errate[RPR][CPR],assenti[RPR][CPR];
   extern int   divalidi[],dierrate[],diassenti[];
   extern int   voto[RPR][CPR],divoto[],gia_effettuate[],incurs;
   extern char  effettuate[],possibilita[];
   extern char  scelta[],attr[],inside[],street[CPR][CPR];
   extern int   riga[],colo1[],colo2[],colo3[];
   extern int   numero_prog;
   extern char  nuovo_paziente;
   extern int   printer,on_off;
   extern char  cognome[],nome[],giorno[],mese[],anno[];
   extern char  sesso,eta[],residenza[],indirizzo[];
   extern char  documento[],num_doc[],progressivo[];
   extern char  type_dis,type_sem,esito_dis,esito_sem;
   extern unsigned char tipo_PR;
   extern int   flop,anni;
   extern char  stim_selezionato,arto_selezionato,oto_sel;
   extern char  arto_numerico;
   extern char  arto_num1,arto_num2;

/*************************************************************************/

unsigned char gellerman [12][20] = {
     0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
     0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
     1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1,
     1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1,
     1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
     1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
     0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
     0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0,
     0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0,
     0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
     1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1,
     1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1
};

/*************************************************************************/

struct disp07 {
char stringa07[57];
int posizione_riga07;
int posizione_colo07;
int attributo07;
};
```

```
struct disp07 screen07[8]={
"                                                    ",2,10,71,
"              T I P O   D I   P R O V A             ",3,10,71,
"                                                    ",4,10,71,
"1. : PROVE DI APPRENDIMENTO",10,23,2,
"2. : PROVE DI ESAME",13,23,2,
"                                                   ",20,10,71,
"   <ESC> menu` precedente      ALT-Y reinizializza  ",21,10,71,
"              Introduci: 1 - 2                     ",22,10,71,
};


esame_apprendimento()           /* Ritorna 0:ESAME o 1:APPRENDIMENTO     */
{
    int  a;
    char risposta,key;
    scr_clr();
    for (a=0; a<8; a++)
    presenta(screen07[a].posizione_riga07,screen07[a].posizione_colo07,
          screen07[a].attributo07,screen07[a].stringa07);
    fasce();
    lintest();
    risposta=0x00;
    while (!risposta)                        /* ---- loop generale -------*/
    {
       while (((key=ci())!=21)&&(key!=27)&&          /* ALT-Y o ESCape */
              (key!='1')&&(key!='2'));               /* Numerico       */
       switch (key)
       {
          case 21:   exit_close();
                     break;
          case 27:   risposta=0x0FF;
                     break;
          case '1':  key=0;
                     risposta=0x0FF;
                     break;
          case '2':  key=1;
                     risposta=0x0FF;
                     break;
       }
    }
    return(key);
}
/*
=============================================================================
*/



struct disp08 {
char stringa08[59];
int posizione_riga08;
int posizione_colo08;
int attributo08;
};

struct disp08 screen08[18]={
```

```
"                                                                ",2,10,71,
"        P R O V E   D I   A P P R E N D I M E N T O             ",3,10,71,
"                                                                ",4,10,71,
" NOME SOGGETTO =                                  DATA=         ",5,10,23,
"COMBINAZIONE = ",6,11,2,
"            ",6,25,23,
"ARTO = ",6,40,2,
"                  ",6,47,23,
" ",7,11,2,
"NUMERO DELLE PROVE = ",10,18,2,
"        ",10,39,23,
"TIPO = ",12,17,2,
" AUTOMATICO ",12,24,23,
"  /  ",12,36,7,
" MANUALE    ",12,41,7,
"                                                                ",19,10,71,
"            <ESC>  menu` precedente                             ",20,10,71,
"           <ENTER> abilitazione/continua                        ",21,10,71,
};

    int nprove,auto_man;

apprendimento_semplice()
{
    char out;
    out=numero_tipo(0);
    if(out==27)  return(27);
    if(esegui_apprendi_semplice()==27)    return(27);
}


numero_tipo(f)
char f;
{
    extern char possibilita[];
    extern int gia_effettuate[];
    extern int  possibili[];
    char *nmaxprove;
    int  a,idd,len;
    int  cpaziente=27,cdata=59;
    char *cursore,*spazi,*fonte,*arisp,*orec,data_odierna[9];
    char buffer[9],key,risposta,out,flag,dum[2],dato;
    char record[80],c_modo[7]={0,0,0,0,0,0,0};
    cpaziente=27;
    buffer[2]=0;
    nmaxprove="            NUMERO MASSIMO DELLE PROVE = 8              ";
    spazi="                                                  ";
    cursore=" <----- ";
    spazi="        ";
    cursore[8]=0;
    spazi[8]=0;
    flag=0x00;
    scr_clr();
    for (a=0; a<6; a++)
    presenta(screen08[a].posizione_riga08,screen08[a].posizione_colo08,
           screen08[a].attributo08,screen08[a].stringa08);
    for (a=8; a<17; a++)
```

```
      presenta(screen08[a].posizione_riga08,screen08[a].posizione_colo08,
            screen08[a].attributo08,screen08[a].stringa08);
      if(f==0)
      {
         for (a=6; a<8; a++)
         presenta(screen08[a].posizione_riga08,screen08[a].posizione_colo08,
               screen08[a].attributo08,screen08[a].stringa08);
         switch (arto_selezionato)
         {
            case 1: arisp="PULSANTE DESTRO   ";
                    break;
            case 2: arisp="PULSANTE SINISTRO ";
                    break;
         }
         if(tipo_PR==GONOGON)
            orec =" TONO ALTO          ";
         else
         {
            if(oto_sel == 0)
               orec =" ORECCHIO DESTRO   ";
            else
               orec =" ORECCHIO SINISTRO ";
         }
       /*  presenta(6,25,23,fonte);*/
         presenta(6,47,23,arisp);
         presenta(6,25,23,orec);
      }
      else
      {
         sprintf(record," COMBINAZIONE=                                  ");
         presenta(6,11,23,record);
         dato=possibilita[gia_effettuate[tipo_PR]-1];
         idd=possibilita[gia_effettuate[tipo_PR]-1];
         dato=possibili[idd];
/***     crea_stringa(dato,c_modo);
         presenta(6,26,23,c_modo);    ***/
      }
      fasce();
      lintest();
      presenta(8,10,71+128,nmaxprove);
      data_int(data_odierna);
      len=strlen(cognome)+1;
      presenta(screen08[3].posizione_riga08,cpaziente,23,cognome);
      cpaziente+=len;
      presenta(screen08[3].posizione_riga08,cpaziente,23,nome);
      presenta(screen08[3].posizione_riga08,cdata,23,data_odierna);
      presenta(10,58,23,cursore);
      if(attende_char(10,39,151,2,buffer,1,dum)==0x1B)
            return(27);
      presenta(10,58,7,spazi);
      presenta(12,58,23,cursore);
      presenta(10,39,23,buffer);
      nprove=atoi(buffer);
      auto_man=0;
      while(!flag)
      {
         while(((key=ci())!=27)&&(key!=21)&&(key!=0)&&(key!=13));
```

```
        switch (key)
        {
            case 21: exit_close();
                     break;
            case 13: key=0;
                     flag=0xFF;
                     break;
            case 27: flag=0xFF;
                     break;
            case 00: while(((key=ci())!=75)&&(key!=77));
                     switch (key)
                     {
                        case 77: auto_man=1;
                                 presenta(12,24,7,screen08[12].stringa08);
                                 presenta(12,41,23,screen08[14].stringa08);
                                 break;
                        case 75: auto_man=0;
                                 presenta(12,41,7,screen08[14].stringa08);
                                 presenta(12,24,23,screen08[12].stringa08);
                                 break;
                     }
                     break;
        }
    }
    return(key);
}

/*
=============================================================================
*/

struct disp09 {
char stringa09[58];
int posizione_riga09;
int posizione_colo09;
int attributo09;
};

struct disp09 screen09[13]={
"                                                          ",2,10,71,
"       P R O V E   D I   A P P R E N D I M E N T O        ",3,10,71,
"                                                          ",4,10,71,
" NOME SOGGETTO =                              DATA        ",5,10,23,
"COMBINAZIONE = ",6,11,2,
"            ",6,25,23,
"ARTO = ",6,40,2,
"                  ",6,47,23,
"MODALITA` =      ",7,25,2,
"NUMERO PROVE =      ",8,22,2,
"                                                        ",20,10,71,
"                                                        ",21,10,71,
"                                                        ",22,10,71,
};


esegui_apprendi_semplice()
{
```

```
    extern char work_uditivo[];
    unsigned b,c,d;
    static int casled[20];
    int  casor[10],riga,colo1,colo2,ireturn,len,a;
    int  cpaziente=27,cdata=59,next_prov;
    char *automatico,*manuale,*premi_barra,*gok,record[80];
    char *errata,*assente,*in_corso,*start,*spazi,*enter,*out_corso;
    char bell=0x07,risposta,uscita;
    char *fonte,*arisp,data_odierna[9];
    char *nuova_prova,*rdx,*rsn;
    char *orec,oto,arto;
    unsigned char joto1,joto2;
    cpaziente=27;
    nuova_prova=" VUOI RIPETERE LA PROVA DI APPRENDIMENTO (S/N) ";
    automatico=" AUTOMATICO ";
    manuale="   MANUALE  ";
    premi_barra="  PREMI LA BARRA PER INIZIALIZZARE LO STIMOLO  ";
    spazi="                                                    ";
    errata="  ERRATA          ";
    assente="  ASSENTE          ";
    gok="  --- OK ---      ";
    in_corso=" ---> ";
    if(tipo_PR==GONOGON)
    {
       rdx=" TA ";
       rsn=" TB ";
       work_uditivo[5]=ATTHMOD;
       work_uditivo[6]=ATTLMOD;
    }
    else
    {
       rdx=" DX ";
       rsn=" SN ";
       work_uditivo[5]=ATTHNOR;
       work_uditivo[6]=ATTLNOR;
    }
    out_corso="     ";
    start="  PREMI UN TASTO PER PARTIRE ";
    enter="PREMI UN TASTO PER CONTINUARE";
    next_prov=0;
    joto1=joto2=0;
  while(next_prov==0)
  {
    scr_clr();
    for (a=0; a<13; a++)
    presenta(screen09[a].posizione_riga09,screen09[a].posizione_colo09,
           screen09[a].attributo09,screen09[a].stringa09);
    data_int(data_odierna);
    len=strlen(cognome)+1;
    presenta(screen09[3].posizione_riga09,cpaziente,23,cognome);
    cpaziente+=len;
    presenta(screen09[3].posizione_riga09,cpaziente,23,nome);
    presenta(screen09[3].posizione_riga09,cdata,23,data_odierna);

    if(tipo_PR==GONOGON)
       orec =" TONO ALTO          ";
    else
```

```
    {
       if(oto_sel == 0)
          orec =" ORECCHIO DESTRO   ";
       else
          orec =" ORECCHIO SINISTRO ";
    }
    switch (arto_selezionato)
    {
       case 1: arisp="PULSANTE DESTRO   ";
               break;
       case 2: arisp="PULSANTE SINISTRO ";
               break;
    }
/*    presenta(6,25,23,fonte);*/
    presenta(6,47,23,arisp);
    presenta(6,25,23,orec);
    if(auto_man==0)
       presenta(7,43,23,automatico);
    else
       presenta(7,43,23,manuale);
    if(nprove>8)   nprove=8;
    sprintf(record,"      %d     ",nprove);
    presenta(8,43,23,record);
    scr_rowcol(9,1);
    for(a=0; a<nprove; a++)
       printf("\n                %d:               RISPOSTA:",a+1);
    fasce();
    lintest();
    if(auto_man==0)               /* --- E' STATO SCELTO IL MODO AUTOMATICO */
    {
       carica_casuali(2000,5000,casuali,nprove);   /* TRA 2 E 5 SECONDI     */
       carica_casuali(1000,10000,casor,nprove);
       carica_casuali(1000,10000,tcasual,nprove);
       riga=10;
       colo1=20;
       colo2=44;
       presenta(21,10,71,start);
       printf("%c",bell);
       risposta=ci();
       presenta(21,10,71,spazi);
       for(a=0; a<nprove; a++)
       {
          presenta(riga,colo1,23,in_corso);
          if(casor[a]>5500)
          {
             oto=0;
             joto1++;
             if(joto1 >= 3)
             {
                joto1=0;
                oto=1;
                joto2++;
             }
          }
          else
          {
             oto=1;
```

```
      joto2++;
      if(joto2 >= 3)
      {
         joto2=0;
         oto=0;
         joto1++;
      }
   }
   if(oto==0)
      presenta(riga,colo1+5,71,rdx);
   else
      presenta(riga,colo1+5,71,rsn);
   if(oto==oto_sel)
      arto=arto_selezionato;
   else
   {
      if(arto_selezionato==1)
         arto=2;
      else
         arto=1;
   }
   led();
   if(tipo_PR==GONOGON)
   {
      if(oto==0)                          /* $$$ */
         work_uditivo[11]=FREQH;
      else
         work_uditivo[11]=FREQL;
      if(tcasual[a]>5500)
         work_uditivo[4]=ODESTRO;
      else
         work_uditivo[4]=OSINIST;
   }
   else
   {
      work_uditivo[4]=oto;
      work_uditivo[11]=FREQC;
   }
   trasmette_comandi('U','5');
   load_timer(casuali[a]);

   /* per il simulato togli tutti i commenti all'istruzione    */
   /* successiva e linka con PUBBLI.BAT                        */
   /*   via();   */

   trigger(10);
   ireturn=singola_prova(casuali[a],arto,100,5000,oto);
   if(ireturn==0)
      presenta(riga,colo2,71,assente);
   if(ireturn==-2)
   {
      presenta(riga,colo2,71,gok);
      wait_uno();
   }
   if(ireturn==-1)
   {
      presenta(riga,colo2,71,errata);
```

```
            wait_uno();
         }
         if((ireturn!=0)&&(ireturn!=-1)&&(ireturn!=-2))
         {
            sprintf(record,"      %4d mS     ",ireturn);
            presenta(riga,colo2,23,record);
            wait_uno();
         }
         presenta(riga,colo1,7,out_corso);
         riga++;
         uscita=csts();
         if(uscita!=0)
         {
            uscita=ci();
            if(uscita==21) exit_close();
            if(uscita==27) return(27);
         }
      }
   }
   else                              /* --- E' STATO SCELTO IL MODO MANUALE    */
   {
      carica_casuali(2000,5000,casuali,nprove);    /* TRA 2 E 5 SECONDI     */
      carica_casuali(1000,10000,casor,nprove);
      carica_casuali(1000,10000,tcasual,nprove);
      riga=10;
      colo1=20;
      colo2=44;
      for(a=0; a<nprove; a++)
      {
         if(casor[a]>5500)
         {
            oto=0;
            joto1++;
            if(joto1 >= 3)
            {
               joto1=0;
               oto=1;
               joto2++;
            }
         }
         else
         {
            oto=1;
            joto2++;
            if(joto2 >= 3)
            {
               joto2=0;
               oto=0;
               joto1++;
            }
         }
         if(oto==oto_sel)
            arto=arto_selezionato;
         else
         {
            if(arto_selezionato==1)
               arto=2;
```

```
            else
               arto=1;
         }
         presenta(21,10,71,premi_barra);
         while((risposta=ci())!=0x20);
         presenta(21,10,71,spazi);
         presenta(riga,colo1,23,in_corso);
         if(oto==0)
            presenta(riga,colo1+5,71,rdx);
         else
            presenta(riga,colo1+5,71,rsn);

/* ---------------------- Ritardo casuale tra tasto e trigger ------------ */
         for(d=0; d<casuali[a]*3; d++)   /* per ritardare in modo casuale */
         {
            b = d*b;
            c = b/(d+1);
            b = b+c;
         }
 /* ------------------------------------------------------------------------ */
         led();
         if(tipo_PR==GONOGON)
         {
            if(oto==0)                          /* $$$ */
               work_uditivo[11]=FREQH;
            else
               work_uditivo[11]=FREQL;
            if(tcasual[a]>5500)
               work_uditivo[4]=ODESTRO;
            else
               work_uditivo[4]=OSINIST;
         }
         else
         {
            work_uditivo[4]=oto;
            work_uditivo[11]=FREQC;
         }
         trasmette_comandi('U','5');
         load_timer(casuali[a]);
         /*   via();   */
         trigger(10);
         ireturn=singola_prova(casuali[a],arto,100,5000,oto);
         if(ireturn==0)  presenta(riga,colo2,71,assente);
         if(ireturn==-2)
         {
            presenta(riga,colo2,71,gok);
            wait_uno();
         }
         if(ireturn==-1)
         {
            presenta(riga,colo2,71,errata);
            wait_uno();
         }
         if((ireturn!=0)&&(ireturn!=-1)&&(ireturn!=-2))
         {
            sprintf(record,"      %4d mS     ",ireturn);
            presenta(riga,colo2,23,record);
```

```
            wait_uno();
         }
         presenta(riga,colo1,7,out_corso);
         riga++;
         uscita=csts();
         if(uscita!=0)
         {
            uscita=ci();
            if(uscita==21) exit_close();
            if(uscita==27) return(27);
         }
      }
   }
   presenta(21,10,23,nuova_prova);
   printf("%c",bell);
   risposta=ci();
   presenta(21,10,71,spazi);
   if((risposta=='n')||(risposta=='N'))   next_prov=1;
  }
}


wait(tick)
unsigned tick;
{
   unsigned letto;
   stop_timer();
   load_timer(tick);
   start_timer();
   letto=0xFFFF;
   while(letto!=1)
   {
      letto=legge_timer();
   }
   stop_timer();
}


wait_uno()                    /* aspetta che venga rilasciato il tasto  */
{
   extern portim;
   char byte;
   while((byte=_inb(portim-2)&0x0F)!=0x0F);
   wait(100);
}


/*
================================================================================
*/

struct disp10 {
char stringa10[58];
int posizione_riga10;
int posizione_colo10;
int attributo10;
```

```
};

struct disp10 screen10[24]={
"                                                          ",2,10,71,
"              P R O V E   D I   E S A M E                 ",3,10,71,
"                                                          ",4,10,71,
" NOME SOGGETTO =                                DATA      ",5,10,23,
" COMBINAZIONE = ",6,10,23,
"              ",6,26,23,
"         = ",6,37,23,
"                    ",6,47,23,
"RISPOSTE ERRATE = ",19,10,2,
"           ",19,28,23,
"RISPOSTE ASSENTI  = ",19,39,2,
"            ",19,59,23,
"VALORE MEDIO ATTUALE = ",20,5,2,
"           ",20,28,23,
"DEVIAZ. STANDARD  = ",20,39,2,
"            ",20,59,23,
"                                                          ",21,10,71,
"     <ESC>  per terminare                                 ",22,10,71,
"    <ENTER> per proseguire       <PrtSc> stampa schermo   ",23,10,71,
"      VALORE MEDIANA = ",18,5,2,
"          ",18,28,23,
" VALORE MEDIANA 1  = ",18,38,2,
"            ",18,59,23,
"         ",18,69,23
};

    char data_odierna[9],buffer[8],dum[2];
    char *fonte,*arisp,*mess0,*mess1,*mess2;
    char *mess3,*mess4,*curson,*cursoff;
/*  int koto1,koto2,joto1,joto2,loto1,loto2,conl1,conl2,jdex,nvalide;   */
    int koto1,koto2,joto1,joto2,conl1,conl2,jdex,nvalide;
    int cstimolo=25,carto=47,cpaziente,cdata=59,prove_massime;
    int gell1,gell2,gindex1,gindex2,gcasual[10];



esame_semplice(posizione)
int  posizione;
{
    #include "math.h"
    extern char work_uditivo[];
    extern char cognome[],nome[],stim_selezionato,arto_selezionato;
    extern int printer,on_off;
    extern char per_status[];
    int dati[2],riga1=19,riga2=20,cerrate=33,cassenti=65;
    int ir,a,b,len,prime_due,nrecord,arto;
    static char record[80],ff=0x0C;
    static char risposta,bell=0x07,uscita,nuova_prova;
    static int temporaneo[40];
    int ireturn,ncasual,status,denmedia;;
    int singola_prova(),meda,c;
    float  somma_dati,somma_quadrati,quadrato;
    double somma_varia,tempor;
```

```
    double x;
    char *curDX,*curSN,*gok;
    char   oto,orec;

    mess0="   BATTI UN TASTO PER INIZIARE LE PROVE   ";
    mess1="                                          ";
    mess2="      BATTI UN TASTO PER PROSEGUIRE       ";
    mess3="        Introduci numero di prove:        ";
    mess4="  VUOI RIPETERE LA PROVA COMPLETA  (S/N)  ";
    apre_temporaneo();
    curson=">";
    cursoff=" ";
    if(tipo_PR==GONOGON)
    {
       curDX=" TA ";
       curSN=" TB ";
       work_uditivo[5]=ATTHMOD;
       work_uditivo[6]=ATTLMOD;
    }
    else
    {
       curDX=" DX ";
       curSN=" SN ";
       work_uditivo[5]=ATTHNOR;
       work_uditivo[6]=ATTLNOR;
    }
    gok=" -OK-";
    nuova_prova='S';
    nrecord=1;
    for(a=0; a<8; a++)
       buffer[a]=0;

  while (nuova_prova=='S')
  {
    schermo(posizione);
    nvalide=posizione*40;
    meda=posizione*20+(tipo_PR-1)*40;  /* indice per mediana1 e mediana2 */
    somma_dati=0.0;
    somma_quadrati=0.0;
    jdex=0;
    gell1=gell2=0;
    carica_casuali(1,12,gcasual,2);
    gindex1=gcasual[0]-1;
    gindex2=gcasual[1]-1;
    if(gindex1==gindex2)
    {
       if(gindex2==11)
          gindex2=0;
       else  if(gindex2==0)
                gindex2=1;
             else
                gindex2++;
    }
    ncasual=0;
    prime_due=0;
    while((koto1 < prove_massime) || (koto2 < prove_massime))
    {
```

```
        presenta(riga[jdex],colo1[jdex],23,curson);
        if(casual0[jdex]>2500)
        {
           oto=0;
           joto1++;
           if(joto1 >= 3)
           {
              presenta(riga[jdex],colo3[jdex],71,curSN);
              if(tipo_PR!=GONOGOC)
                 presenta(riga[jdex],colo2[jdex]+9,2,"    ");
              joto1=0;
              oto=1;
              joto2++;
           }
           else
           {
              presenta(riga[jdex],colo3[jdex],71,curDX);
              joto2=0;
           }
        }
        else
        {
           oto=1;
           joto2++;
           if(joto2 >= 3)
           {
              presenta(riga[jdex],colo3[jdex],71,curDX);
              joto2=0;
              oto=0;
              joto1++;
           }
           else
           {
              presenta(riga[jdex],colo3[jdex],71,curSN);
              if(tipo_PR!=GONOGOC)
                 presenta(riga[jdex],colo2[jdex]+9,2,"    ");
              joto1=0;
           }
        }
        if(oto==oto_sel)
           arto=arto_selezionato;
        else
        {
           if(arto_selezionato==1)
              arto=2;
           else
              arto=1;
        }
        led();
        if(tipo_PR==GONOGON)
        {
           if(oto==0)                          /* $$$ */
              work_uditivo[11]=FREQH;
           else
              work_uditivo[11]=FREQL;
/*******    if(tcasual[jdex]>5500)
           {
```

```
            if(conl2 < (prove_massime/2))
            {
               orec=1;
               loto1=0;
               loto2++;
            }
            else
            {
               orec=0;
               loto1++;
               loto2=0;
            }
         }
         else
         {
            if(conl1 < (prove_massime/2))
            {
               loto2=0;
               orec=0;
               loto1++;
            }
            else
            {
               orec=1;
               loto1=0;
               loto2++;
            }
         } ************************************/
         if(oto==0)
            orec=gellerman[gindex1][gell1];
         else
            orec=gellerman[gindex2][gell2];
         work_uditivo[4]=orec;
      }
      else
      {
         work_uditivo[4]=oto;
         work_uditivo[11]=FREQC;
         orec=oto;
      }
      trasmette_comandi('U','5');
      load_timer(casuali[ncasual]);
      /*   via();   */
      trigger(10);
      start_timer();
      ireturn=singola_prova(casuali[ncasual],arto,100,1000,oto);
      if(ireturn==0)
      {
/*       if(on_off!=0)
            status=fprintf(flop,"\n%2d --- PROVA ASSENTE ---",nrecord); */
         assenti[posizione][tipo_PR]++;
         presenta_dato_integer(screen10[11].posizione_riga10,
          screen10[11].posizione_colo10,23,assenti[posizione][tipo_PR]);
         presenta(riga[jdex],colo2[jdex]+9,2,"    ");
         ncasual++;
      }
      if(ireturn==-1)
```

```
        {
/*         if(on_off!=0)
              status=fprintf(flop,"\n%2d --- PROVA ERRATA ----",nrecord); */
           errate[posizione][tipo_PR]++;
           presenta_dato_integer(screen10[9].posizione_riga10,
             screen10[9].posizione_colo10,23,errate[posizione][tipo_PR]);
           presenta(riga[jdex],colo2[jdex]+9,2,"    ");
           ncasual++;
        }
        if(ireturn==-2)
        {
           if(((oto==0)&&(koto1<prove_massime))||
              ((oto==1)&&(koto2<prove_massime)))
           {
/*            if(on_off!=0)
                 status=fprintf(flop,"\n%2d ----- PROVA OK ------",nrecord); */
/*            errate[posizione][tipo_PR]++;                                  */
              presenta(riga[jdex],colo2[jdex],2,gok);
              presenta(riga[jdex],colo1[jdex],7,cursoff);
              if(tipo_PR==GONOGON)
                 gell2++;
              jdex++;
              ncasual++;
              if(oto==0)
                 koto1++;
              else
                 koto2++;
           }
        }
        if((ireturn!=0)&&(ireturn!=-1)&&(ireturn!=-2))
        {
           if(((oto==0)&&(koto1<prove_massime))||
              ((oto==1)&&(koto2<prove_massime)))
           {
              sprintf(record,"%4d ",ireturn);
              presenta(riga[jdex],colo2[jdex],2,record);
/*            if(on_off!=0)
                 if(prime_due==0)
                    status=fprintf(flop,
                    "\n%2d  TEMPO DI RISPOSTA =  %5d mS -NO-",nrecord,ireturn);
                 else
                    status=fprintf(flop,
                     "\n%2d  TEMPO DI RISPOSTA =  %5d mS ****",nrecord,ireturn); */

              validi[nvalide][tipo_PR]=ireturn;
              presenta(riga[jdex],colo1[jdex],7,cursoff);
              jdex++;
              nvalide++;
              switch (posizione)
              {
                 case 0:
                         denmedia=nvalide;
                         break;
                 case 1:
                         denmedia=nvalide-40;
                         break;
                 case 2:
```

```
                denmedia=nvalide-80;
                break;
            case 3:
                denmedia=nvalide-120;
                break;
        }
        somma_dati+=ireturn;
        media[posizione][tipo_PR]=somma_dati/denmedia;
        quadrato=ireturn;
        quadrato=quadrato*quadrato;
        somma_quadrati+=quadrato;
        presenta_dato_float(screen10[13].posizione_riga10,
          screen10[13].posizione_colo10,23,media[posizione][tipo_PR]);
        x=somma_quadrati/denmedia-
          media[posizione][tipo_PR]*media[posizione][tipo_PR];
        x=sqrt(x);
        scarto[posizione][tipo_PR]=x;
        presenta_dato_float(screen10[15].posizione_riga10,
         screen10[15].posizione_colo10,23,scarto[posizione][tipo_PR]);
        ncasual++;
        if(oto==0)
           koto1++;
        else
           koto2++;
        if(tipo_PR != GONOGOC)
        {
           if(orec==0)
           {
              if(tipo_PR==GONOGON)
              {
                 presenta(riga[jdex-1],colo2[jdex-1]+9,23," DX ");
                 gell1++;
              }
              mediana1[conl1+meda]=validi[nvalide-1][tipo_PR];
              conl1++;
           }
           else
           {
              if(tipo_PR==GONOGON)
              {
                 presenta(riga[jdex-1],colo2[jdex-1]+9,71," SN ");
                 gell1++;
              }
              mediana2[conl2+meda]=validi[nvalide-1][tipo_PR];
              conl2++;
           }
        }
     }
  }
  if((koto1+koto2==2)&&(prime_due==0))
  {
     riallinea(&somma_dati,&somma_quadrati,posizione);
     prime_due=1;
  }
  uscita=csts();
  if(uscita!=0)
  {
```

```
            uscita=ci();
            if(uscita==21) exit_close();
            if(uscita==27) return(27);
         }
         nrecord++;
      }
      b=posizione*40;
      somma_varia=0.0;
      for (a=b,c=0; a<b+denmedia; a++,c++)
      {
         tempor=validi[a][tipo_PR]-media[posizione][tipo_PR];
         somma_varia+=fabs(tempor);
         temporaneo[c]=validi[a][tipo_PR];
      }
      somma_varia=somma_varia/denmedia;
      assolut[posizione][tipo_PR]=somma_varia;
      premed(temporaneo,denmedia,mediana1,mediana2,conl1,conl2,posizione);
      presenta(21,14,23,mess4);
      printf("%c",bell);
      risposta=ci();
      presenta(21,14,71,mess1);
      setta_next(risposta,posizione,&nuova_prova);
   }
/*if(on_off!=0)
   {
      status=fprintf(flop,"\n ------------------------------------");
      sprintf(record,"\nVALORI: [%.1f] [%.1f] [%.1f] [%.1f]",
              media[posizione][tipo_PR],
              scarto[posizione][tipo_PR],
              varianz[posizione][tipo_PR],
              assolut[posizione][tipo_PR]);
      status=fprintf(flop,"%s",record);
      status=fprintf(flop,"\n ------------------------------------");
   }  */
   stampa_temporaneo();
}



presenta_dato_float(iriga,icolo,attributo,valore)
int   iriga,icolo;
float valore;
char  attributo;
{
   char record[20];
   sprintf(record,"  %3.1f  ",valore);
   presenta(iriga,icolo,attributo,record);
}

presenta_dato_integer(iriga,icolo,attributo,valore)
int   iriga,icolo,valore;
char  attributo;
{
   char record[20];
   sprintf(record,"  %5d  ",valore);
   presenta(iriga,icolo,attributo,record);
}
```

```
archiviazione_fisso()
{
   extern char def_uditivo[],work_uditivo[];
   extern float archivio[];
   extern int  osservazioni[];
   extern int  fp0,fp1;
   extern int  possibili[];
   int ieta,indice,index,num,a,idd;
   char risposta,flag,record[80];
   char *riga1,*riga2;
   char *mess1,*mess2,*mess3,*mess4,*mess5,*erro1,bell=7;
   riga1="                                                                    ";
   riga2="              A R C H I V I A Z I O N E   D A T I                    ";
   mess1="    STO ARCHIVIANDO I DATI RELATIVI ALLE PROVE SEMPLICI             ";
   mess2=" STO ARCHIVIANDO I DATI RELATIVI ALLE PROVE DI DISCRIMINAZIONE ";
   mess3="    ARCHIVIO PROVE SEMPLICI AGGIORNATO.....                         ";
   mess4="    ARCHIVIO PROVE DI DISCRIMINAZIONE AGGIORNATO.....               ";
   mess5="  Batti un tasto qualsiasi per continuare.....                      ";
   erro1="    IMPOSSIBILE AGGIORNARE ARCHIVIO: DATI NON STANDARD.....         ";
   scr_clr();
   presenta(0,10,71,riga1);
   presenta(1,10,71,riga2);
   presenta(2,10,71,riga1);
   presenta(22,10,71,riga1);
   fasce();
   flag=0;
   presenta(10,10,23,mess1);
   work_uditivo[4]=def_uditivo[4];
   work_uditivo[2]=def_uditivo[2];
   work_uditivo[5]=def_uditivo[5];
   work_uditivo[6]=def_uditivo[6];
   work_uditivo[11]=def_uditivo[11];
   work_uditivo[14]=def_uditivo[14];
   for(a=0; a<21; a++)
   {
      if(def_uditivo[a]!=work_uditivo[a])
      {
         flag=255;
         printf("\nDATO DIVERSO --- UDITIVO #%d",a);
      }
   }
   if(flag!=0)
   {
      presenta(22,10,71,erro1);
      printf("%c",bell);
      risposta=ci();
      return(27);
   }
   carica_archivio(fp0,archivio,osservazioni,1024);
   ieta=anni;
   for(a=0; a<12; a++)
   {
      if(street[a][tipo_PR]==0x55)
      {
         determina_stimart(a);
```

```
         indice=crea_indice(0,sesso,ieta,stim_selezionato,
                                 arto_numerico,arto_numerico);
/**      printf("\nSTO AGGIORNANDO L'INDICE: %d",indice);  **/
         sprintf(record,"  PROVE SEMPLICI INDICE #%d  ",indice);
         presenta(11,10,23,record);
         aggiorna_buffer(indice,archivio,osservazioni,
              media[a][tipo_PR],scarto[a][tipo_PR],
              varianz[a][tipo_PR],assolut[a][tipo_PR]);
                                 /* x,y,w,z */
      }
   }
   scrive_archivio(fp0,archivio,osservazioni,1024);
   presenta(12,10,23,mess3);
   presenta(22,10,71,mess5);
   printf("%c",bell);
   risposta=ci();
   scr_clr();
   presenta(0,10,71,riga1);
   presenta(1,10,71,riga2);
   presenta(2,10,71,riga1);
   presenta(22,10,71,riga1);
   fasce();
   presenta(10,10,23,mess2);
   if(flag!=0)
   {
      presenta(22,10,71,erro1);
      printf("%c",bell);
      risposta=ci();
      return(27);
   }
   carica_archivio(fp1,archivio,osservazioni,2048);
   ieta=anni;
   for(a=0; a<gia_effettuate[tipo_PR]; a++)
   {
      idd=possibilita[a];
      index=possibili[idd];
      determina_arti(index);
      indice=crea_indice(1,sesso,ieta,stim_selezionato,arto_num1,arto_num2);
/**   printf("\nSTO AGGIORNANDO L'INDICE: %d",indice);   **/
      sprintf(record,"  PROVE DI DISCRIMINAZIONE INDICE #%d  ",indice);
      presenta(11,10,23,record);
      aggiorna_buffer(indice,archivio,osservazioni,dimedia[idd],
            discarto[idd],divarianz[idd],diassolut[idd]);  /* x,y,w,z */
   }
   scrive_archivio(fp1,archivio,osservazioni,2048);
   presenta(12,10,23,mess4);
   presenta(22,10,71,mess5);
   printf("%c",bell);
   risposta=ci();
}

/*********************************************************************/

determina_arti(parola)
int parola;
{
   stim_selezionato=(parola&64)/64;
```

```
    arto_num1=(parola&12)/4;
    arto_num2=parola&3;
}


data_int(array)
char array[];
{
    dates(array);
    inverte_data(array);
}


inverte_data(array)
char array[];
{
    char temp;
    int  a;
    for (a=0; a<2; a++)
    {
        temp = array[a];
        if(temp==0x20) temp=0x30;
        array[a] = array[a+3];
        array[a+3] = temp;
    }
}

system_op()
{
    char *mess0,*mess1,*mess2;
    char code;

 mess0="Consiglio Nazionale delle Ricerche IEI -- Rev. 1.0 -- Febbraio 1990";
 mess1="------------------------------------------------------------------";
 mess2="            Digita 'exit' per tornare al programma                ";
 scr_clr();
 presenta(1,1,23,mess0);
 presenta(2,1,23,mess1);
 presenta(3,1,23,mess2);
 scr_rowcol(4,1);
 code=exec("C:\\COMMAND.COM","");
}


schermo(posizione)
int posizione;
{
    int a,ir,len;
    char record[80],bell=7,risposta;

    cpaziente=27;
    scr_clr();
    for (a=0; a<21; a++)
    presenta(screen10[a].posizione_riga10,screen10[a].posizione_colo10,
           screen10[a].attributo10,screen10[a].stringa10);
```

```
    if(tipo_PR!=GONOGOC)
    {
       for (a=21; a<24; a++)
       presenta(screen10[a].posizione_riga10,screen10[a].posizione_colo10,
           screen10[a].attributo10,screen10[a].stringa10);
    }
    ir=8;
    for(a=0; a<10; a++)
    {
       sprintf(record,"%2d.          ",a+1);
       presenta(ir,5,2,record);
       sprintf(record,"%2d.          ",a+11);
       presenta(ir,23,2,record);
       sprintf(record,"%2d.          ",a+21);
       presenta(ir,41,2,record);
       sprintf(record,"%2d.          ",a+31);
       presenta(ir,59,2,record);
       ir++;
    }
    fasce();
    lintest();
    data_int(data_odierna);
    len=strlen(cognome)+1;
    presenta(screen10[3].posizione_riga10,cpaziente,23,cognome);
    cpaziente+=len;
    presenta(screen10[3].posizione_riga10,cpaziente,23,nome);
    presenta(screen10[3].posizione_riga10,cdata,23,data_odierna);
    if(tipo_PR==GONOGON)
       fonte =" TONO ALTO         ";
    else
    {
       if(oto_sel == 0)
          fonte =" ORECCHIO DESTRO   ";
       else
          fonte =" ORECCHIO SINISTRO ";
    }
    koto1=koto2=0;
    joto1=joto2=0;
/*  loto1=loto2=0;   */
    conl1=conl2=0;
    switch (arto_selezionato)
    {
       case 1: arisp="  PULSANTE DESTRO    ";
               break;
       case 2: arisp="  PULSANTE SINISTRO  ";
               break;
    }
    presenta(screen10[5].posizione_riga10,screen10[5].posizione_colo10,
             23,fonte);
    presenta(screen10[5].posizione_riga10,screen10[5].posizione_colo10+20,
             23,arisp);
    presenta_dato_integer(screen10[9].posizione_riga10,
             screen10[9].posizione_colo10,23,errate[posizione][tipo_PR]);
    presenta_dato_integer(screen10[11].posizione_riga10,
             screen10[11].posizione_colo10,23,assenti[posizione][tipo_PR]);
    presenta_dato_float(screen10[13].posizione_riga10,
             screen10[13].posizione_colo10,23,media[posizione][tipo_PR]);
```

```
    presenta_dato_float(screen10[15].posizione_riga10,
             screen10[15].posizione_colo10,23,scarto[posizione][tipo_PR]);
    prove_massime=1;
    while((prove_massime&0x01 != 0)||(prove_massime==0))
    {
       presenta(21,14,23,mess3);
       printf("%c",bell);
       if(attende_char(21,50,151,2,buffer,1,dum)==27)
          return(27);
       else
          prove_massime=atoi(buffer);
    }
    if(prove_massime>20) prove_massime=20;
    sprintf(record,
            " ---------------  NUMERO PROVE = %3d +%3d -------------- ",
            prove_massime,prove_massime);
    presenta(7,10,87,record);
    presenta(21,14,23,mess2);
    risposta=ci();
    presenta(21,14,71,mess1);

/*** INIZIALIZZA VARIABILI DI UTILIZZO **********************************/

    carica_casuali(2000,5000,casuali,100);
    carica_casuali(1000,4000,casual0,100);
    carica_casuali(1000,10000,tcasual,100);
}



riallinea(sd,sq,pos)
float *sd,*sq;
int pos;
{
   *sd=0.0;
   *sq=0.0;
   media[pos][tipo_PR]=0.0;
   errate[pos][tipo_PR]=0;
   assenti[pos][tipo_PR]=0;
   scarto[pos][tipo_PR]=0.0;
   assolut[pos][tipo_PR]=0.0;
   varianz[pos][tipo_PR]=0.0;
   presenta_dato_float(screen10[13].posizione_riga10,
    screen10[13].posizione_colo10,23,media[pos][tipo_PR]);
   presenta_dato_float(screen10[15].posizione_riga10,
    screen10[15].posizione_colo10,23,scarto[pos][tipo_PR]);
   presenta_dato_integer(screen10[9].posizione_riga10,
    screen10[9].posizione_colo10,23,errate[pos][tipo_PR]);
   presenta_dato_integer(screen10[11].posizione_riga10,
    screen10[11].posizione_colo10,23,assenti[pos][tipo_PR]);
   jdex=0;
   koto1=koto2=0;
   joto1=joto2=0;
   gell1=gell2=0;
/* loto1=loto2=0;    */
   conl1=conl2=0;
```

```
    nvalide=pos*40;
}


premed(temporaneo,denmedia,mediana1,mediana2,conl1,conl2,pos)
int temporaneo[],mediana1[],mediana2[];
int denmedia,conl1,conl2,pos;
{
    int meda,meda0;
    char rec[30];
    meda=pos*2;
    meda0=pos*20+(tipo_PR-1)*40;
    if(tipo_PR == GONOGOC)
    {
        mediana(temporaneo,denmedia,&varianz[pos][tipo_PR]);
        presenta_dato_float(screen10[20].posizione_riga10,
            screen10[20].posizione_colo10,23,varianz[pos][tipo_PR]);
    }
    else
    {
        mediana(&mediana1[meda0],conl1,&medmodi[meda][tipo_PR]);
        mediana(&mediana2[meda0],conl2,&medmodi[meda+1][tipo_PR]);
        presenta_dato_float(screen10[20].posizione_riga10,
              screen10[20].posizione_colo10,23,medmodi[meda][tipo_PR]);
        presenta_dato_float(screen10[22].posizione_riga10,
              screen10[22].posizione_colo10,23,medmodi[meda+1][tipo_PR]);
        medmed[pos][tipo_PR]=
              (medmodi[meda][tipo_PR]+medmodi[meda+1][tipo_PR])/2.0;
        sprintf(rec,"MM=%5.1f",medmed[pos][tipo_PR]);
        presenta(screen10[23].posizione_riga10,screen10[23].posizione_colo10,
              71,rec);
    }
}


setta_next(risp,pos,next)
char risp,*next;
int  pos;
{
    if ((risp=='s')||(risp=='S'))
    {
        media[pos][tipo_PR]=0.0;
        errate[pos][tipo_PR]=0;
        assenti[pos][tipo_PR]=0;
        scarto[pos][tipo_PR]=0.0;
        *next='S';
    }
    else
        *next='N';
}



via()
{
   int  i;
   char *pippo;
```

```
    pippo="     ";
    presenta(1,1,71,pippo);
    for(i=0; i<10; i++);
    presenta(1,1,2,pippo);
}
```

File: ITBM.C

```
/*
   DEFINIZIONE INDIRIZZI ASSOLUTI UTILIZZATI DALLE PORTE PARALLELE
   E DAL TIMER PRESENTI SULLA SCHEDA HARDWARE
*/
   unsigned port=0x01F7;
   unsigned portim=0x01F3;
   unsigned timer=0x01FC;
/*
================================================================================

    FUNCTION: send_addr(indirizzo)

    Funzione che invia l`indirizzo della periferica secondo il
    protocollo ITBM.  Necessita del parametro "indirizzo" relativo
    all'indirizzo dello strumento da interrogare.

================================================================================
*/

send_addr(indirizzo)
char  indirizzo;
{
      int    val;
      char   out;
/* ------------------------------------------------------------------- */
/*    out=sense_bit(0x10,1,500);
      if (out==2)  printf ("\nERROR> Trovato COA a zero\n");             */

      _outb(0x07,port-1);         /* si pone DIR basso                  */
      _outb(indirizzo,port-3);    /* invio ind. della per. sulla porta A */
      _outb(0x06,port-1);         /* si pone ADV basso                  */
      out=sense_bit(0x10,0,500); /* si controlla l`arrivo del COA       */
      if(out==2)
      {
          _outb(0x07,port-1);     /* si pone ADV alto                   */
/*        printf("\n COLLEGAMENTO INTERROTTO a");                       */
          out=2;
          return(out);
       }
       _outb(0x07,port-1);          /* si pone ADV alto                    */
       _outb(0x0F,port-1);          /* si pone DIR alto                    */
       out=sense_bit(0x10,1,500);  /* si controlla il ritorno alto di COA */
       if(out==2)
       {
/*        printf("\n COLLEGAMENTO INTERROTTO d");                        */
          out=2;
       }
       return(out);
}
/*
================================================================================

    FUNCTION: send_data(byte)
```

```
     Funzione che invia il byte "byte" sul bus comune secondo
     protocollo ITBM.

 ================================================================================
 */

 send_data(byte)
 char byte;
 {
      char    out;
 /* ------------------------------------------------------------- */
 /*   out=sense_bit(0x10,1,500);
      if (out==2)  printf ("\nERROR> Trovato COA a zero\n");       */

      _outb(0x07,port-1);          /* si pone DIR basso             */
      _outb(byte,port-3);          /* invio dato sulla porta A      */
      _outb(0x03,port-1);          /* si pone DTV basso             */
      out=sense_bit(0x10,0,500);   /* si controlla l`arrivo del COA */
      if(out==2)
      {
           _outb(0x07,port-1);     /* si pone DTV alto              */
 /*         printf("\n COLLEGAMENTO INTERROTTO");                   */
           out=2;
           return(out);
      }
      _outb(0x07,port-1);          /* si pone DTV alto */
      _outb(0x0F,port-1);          /* si pone DIR alto */
      return(out);
 }

 /*
 ================================================================================

     FUNCTION: reset(time_out)

     Esegue il reset di tutte le apparecchiature collegate al bus
     secondo protocollo ITBM. Il valore time_out viene utilizzato
     per introdurre un certo ritardo prima di rilasciare lo stato
     di reset.

 ================================================================================
 */

 reset(time_out)
 int   time_out;
 {
       int     a;
 /* ------------------------------------------------------------- */
       _outb(0x0A,port-1);
       for(a=0; a<=time_out; a++);      /*  CICLO DI ATTESA PER t.res */
       _outb(0x0F,port-1);
 }
 /*
 ================================================================================

     FUNCTION: ini_per()
```

```
    Inizializza la scheda contenente le porte parallele 8255 come:
    PORTA RELATIVA AL BUS ITBM
              porta A in OUT
              porta B in IN
              porta C bit 0,1,2,3 in OUT
                      bit 4,5,6,7 in IN
              C0=ADV, C1=TROUT, C2=DTV, C3=DIR
              C4=COA, C5=TRIN
              parola di controllo 8A
    PORTA RELATIVA AI PULSANTI
              porta B in IN
              porta C in OUT

================================================================================
*/

ini_per()
{
      extern unsigned portim;  /* seconda porta                 */
      extern unsigned timer;   /* timer/counter di conteggio    */
      char out,datalow,datahig;
      int  parola;

      parola=250;
      datalow=parola&0x00FF;
      datahig=parola/256;
/* ------------------------------------------------------------ */
      _outb(0x8A,port);        /* Configurazione delle porte     */
      _outb(0x0F,port-1);      /* Tutte le linee interessate a 1 */
      _outb(0x82,portim);      /* Pulsantiera                    */
      _outb(0x00,portim-1);    /* Gate timer a zero-timer fermo  */
      _outb(0x01,portim-3);    /* Spenge i led                   */
/* ------------------------------------------------------------ */
      _outb(0x36,timer-1);     /* contatore 0 modo 3             */
      _outb(0x74,timer-1);     /* contatore 1 modo 2             */
      _outb(datalow,timer-4);  /* forma clock da 1 millisec.     */
      _outb(datahig,timer-4);
      _outb(0xFF,timer-3);     /* tempo in millisecondi          */
      _outb(0xFF,timer-3);
}


/*
================================================================================

    FUNCTION: receive()

    Riceve l'indirizzo di ritorno dalla periferica selezionata

================================================================================
*/

receive()
{
      extern   unsigned port;
      char     val,addr,out;
/* ------------------------------------------------------------ */
```

```
      out=sense_bit(0x10,0,500); /* Testa la linea COA per zero  */
/*    if(out==2) printf("\nNON E'ARRIVATO IL COA\n");            */
      val =_inb(port-2);         /* Prelevo valore dalla porta B */
      out=0;
      addr=val;
      return(addr);
}

/*
=============================================================================

    FUNCTION: sense_bit (ibit,direction,time_out)

    Controlla il bit della porta C indicato da ibit se resettato o
    settato a seconda del valore di direction. La condizione deve
    essere trovata nel periodo di tempo indicato da time_out.
    In caso di time_out spirato la funziona ritorna il valore 2 al
    programma chiamante.

=============================================================================
*/

sense_bit(ibit,direction,time_out)
char  ibit,direction;
int   time_out;

{
      char val,out;
      int  a,c;
/* ------------------------------------------------------------------ */
      if(direction==0)
      {
         for(a=0; a<=time_out*2; a++)     /* Ciclo di attesa     */
         {
            val=_inb(port-1)&ibit;
            if(val==direction)          /* Controllo del bit   */
            {
               out=0;
               return(out);
            }
            else
               for(c=0; c<=1; c++);
         }
      }
      else
      {
      for(a=0; a<=time_out*2; a++)     /* Ciclo di attesa     */
         {
            val=_inb(port-1)&ibit;
            if(val!=0)                /* Controllo del bit   */
            {
               out=0;
               return(out);
            }
            else
               for(c=0; c<=1; c++);
```

```
        }
      }
      out=2;
      return(out);
}

/*
=============================================================================

    FUNCTION: trigger(to)

    Genera l'impulso di trigger per la strumentazione collegata
    con protocollo I.T.B.M. L'impulso ha durata "to" cicli.

=============================================================================
*/

trigger(to)             /* Genera l'impulso di trigger    */
      int to;
{
      int a,c;
      for(a=0; a<=to*3; a++)     /* ciclo di attesa*/
         for(c=0; c<=10; c++);
      _outb(0x05,port-1);      /* attivo TROUT */
      for(c=0; c<=20; c++);
      _outb(0x07,port-1);       /* disattivo TROUT */
}
/*
=============================================================================

    FUNCTION: send_data_some()

    Funzione di trasmissione dato allo stimolatore somestesico

=============================================================================
*/
send_data_some(byte)
char byte;
{
     extern  unsigned port;
     int out;
/* ------------------------------------------------------------- */

     _outb(0x07,port-1);         /* si pone DIR basso              */
     _outb(byte,port-3);         /* invio dato sulla porta A       */
     _outb(0x03,port-1);         /* si pone DTV basso              */
     out=ciclocoa();             /* aspetta la transizione del COA */
     _outb(0x0F,port-1);         /* si pone DIR alto               */
     return(out);
}
/*
=============================================================================

    FUNCTION: send_addr_some()

    Funzione di trasmissione indirizzo allo stimolatore somestesico
```

```
==========================================================================
*/
send_addr_some()
{
      char  indirizzo=0x90;
      int    val,out;
/* ------------------------------------------------------------------- */
      _outb(0x07,port-1);         /* si pone DIR basso                  */
      _outb(indirizzo,port-3);    /* invio ind. della per. sulla porta A */
      _outb(0x06,port-1);         /* si pone ADV basso                  */
      out=ciclocoa();  ,          /* aspetta la transizione del COA     */
      _outb(0x0F,port-1);         /* si pone DIR alto                   */
      return(out);
}

/*
==========================================================================

    FUNCTION: receive_some()

    Riceve l'indirizzo di ritorno dallo stimolatore somestesico

==========================================================================
*/

receive_some()
{
      extern   unsigned port;
      char     val,out;
      int      a;
/* ------------------------------------------------------------------- */

      for (a=0; a<10; a++);       /* Attende                        */
      val =_inb(port-2);          /* Prelevo valore dalla porta B */
      return(val);
}
/* ------------------------------------------------------------------- */
/* ------------------------------------------------------------------- */
ciclocoa()
{
   extern int errori,printer;
   int a,c;
   char val;

   for(a=0;a<=300;a++)
   {
      val=_inb(port-1)&0x10;
      if(val==0)                    /* SEGNALE RICEVUTO*/
      {
      _outb(0x07,port-1);          /* si pone DTV o ADV alto */
         while((val==0))
            val=_inb(port-1)&0x10;
         return(0);
      }
      else
         for(c=0;c<=10;c++);
   }
```

```
/*   printf("ERRORE DI COLLEGAMENTO routine CICLOCOA-SOME");     */
   return(-1);
}
/*
================================================================================
================================================================================
*/
```

File:SINGOLA.C

```
#include "go.h"

singola_prova(casuale,arto,lim1,lim2,orecch)
unsigned casuale;
char arto,orecch;
int  lim1,lim2;
/* ------------------------------------------------------------------------

  Esegue il calcolo del tempo di risposta relativamente alla singola prova
  ------------------------------------------------------------------------

  Parametri di ingresso:
  ----------------------
    casuale : e' l'intervallo temporale settato dalla routine chiamante
              nel quale l'operatore deve effettuare la risposta
       arto : e' il numero (un unico bit settato a 1) che identifica
              (mascheratura) l'arto con cui deve avvenire la risposta
  lim1,lim2 : determinano il campo di convalida della risposta;

     orecch : indica l'orecchio emittente

  Parametro di uscita:   iret
  --------------------
      cosi' configurato:

      iret =  n  --> n e` il tempo di risposta in millisecondi
      iret =  0  --> risposta assente
      iret = -1  --> risposta errata
      iret - -2  --> risposta assente ma corretta per GoNoGo

------------------------------------------------------------------------ */

{
   extern unsigned char tipo_PR;
   extern portim;
   extern char oto_sel;
   unsigned iret;
   unsigned legge_timer();
   unsigned tempo;
   int  a;
   char byte;

   arto=~arto&0x0F;
   iret=0;
   while(((tempo=legge_timer())!=1)&&(iret==0))
   {
      byte=_inb(portim-2)&0x0F;
      if(byte!=0x0F)
      {
         if(byte==arto)
         {
            iret=legge_timer();
            for(a=0; a<300; a++);
            iret=casuale-iret;
```

```
            if((iret<lim1)||(iret>lim2))
               iret=-1;
         }
         else
            iret=-1;
      }
   }
   while (((tempo=legge_timer())>50)&&(tempo < casuale));
   stop_timer();

   switch(tipo_PR)
   {
      case GONOGOC:
      case GONOGON:
         if(orecch!=oto_sel)
         {
            if(iret==0)
               iret=-2;
            else
               iret=-1;
         }
         break;

      case TIMERES:
         break;
   }
   return(iret);
}


lintest()
{
   extern unsigned char tipo_PR;
   char *record;

   switch(tipo_PR)
   {
      case GONOGOC:
         record="           G O   N O   G O   C L A S S I C O          ";
         break;
      case GONOGON:
         record="         G O   N O   G O   M O D I F I C A T O        ";
         break;
      case TIMERES:
         record="                  C H O I C E   R T                   ";
         break;
   }
   presenta(1,10,23,record);
}
```

File: TIMER.C

```
start_timer()
{
   extern unsigned portim;
   unsigned tempo_letto;
   _outb(0x01,portim-1);
   legge_timer();
}

stop_timer()
{
   extern unsigned portim;
   _outb(0x00,portim-1);
}


load_timer(word)
unsigned  word;
{
   extern unsigned timer;
   int i;
   char lsb,msb;
   msb=word/256;
   lsb=word&0xFF;
   _outb(lsb,timer-3);
   _outb(msb,timer-3);
   start_timer();
}


legge_timer()
{
   extern unsigned timer;
   char lsb,msb;
   unsigned word;

   _outb(0x40,timer-1);         /* richiede il latch del counter 0 */
   lsb=_inb(timer-3);
   msb=_inb(timer-3);
   word=msb*256+lsb;
   return(word);
}

/*****************************************************************/


trasmette_comandi(str,cod)
{
}



/*** trasmette_comandi(str,cod)    ***/
trasmette_comandi_OK(str,cod)
char str,cod;
```

```
{
   extern char per_status[];
extern printer;
   int prove,a,indice,status;
   char bell=7,flag;
   indice=0;
   flag=0;
   prove=5;
/***** status=fprintf(printer,"str = %c cod = %c\n",str,cod);    *****/
   switch (str)
   {
      case 'V': while((indice<prove)&&(flag==0))
                {
                  /* visivo(cod);*/
/*** status=fprintf(printer,"(%d)\n",per_status[0]);    *****/
                  if((per_status[0]==1)||(per_status[0]==2))
                     flag=255;
/*** status=fprintf(printer,"|%d|\n",indice);           *****/
                  indice++;
                }
                break;
      case 'U': while((indice<prove)&&(flag==0))
                {
                  uditivo(cod);
/*** status=fprintf(printer,"(%d)\n",per_status[1]);    ***/
                  if((per_status[1]==1)||(per_status[1]==2))
                     flag=255;
/*** status=fprintf(printer,"|%d|\n",indice);    ***/
                  indice++;
                }
                break;
      case 'S': while((indice<prove)&&(flag==0))
                {
/*                   somestesico(cod);*/
/*** status=fprintf(printer,"(%d)\n",per_status[2]);  ***/
                  if((per_status[2]==1)||(per_status[2]==2))
                     flag=255;
/*** status=fprintf(printer,"|%d|\n",indice);  ***/
                  indice++;
                }
                break;
   }
   if(flag==0)
   {
/*    scr_clr();
      printf("\n%cERRORE SULLO STIMOLATORE %c - CODICE %c",bell,str,cod);
      exit();                                                          */
      printf("%c",bell);
   }
/***** status=fprintf(printer,"STATI> %d %d %d\n",
               per_status[0],per_status[1],per_status[2]);  *****/
}
```

File: UDITIVO.C

```
/*
   ---------------------------------------------------------------------
         MODULO UDITIVO: contiene tutte le routines di gestione dello
         stimolatore uditivo.
   ---------------------------------------------------------------------

   --- Definizione strutture parametri e stringhe per rappresentazione
   --- video.
*/

struct disp {
char stringa[74];
int posizione_riga;
int posizione_colo;
int attributo;
};

/*
 ----- Contiene tutti messaggi alfabetici da presentare, le relative
 ----- posizioni riga/colonna su video e l'attributo colore di ogni
 ----- stringa.
*/

struct disp screen[39]={
"                                                                    ",1,6,71,
"                PROGRAMMAZIONE STIMOLATORE UDITIVO                  ",2,6,71,
"                                                                    ",3,6,71,
"ATTUALE",5,21,32,
"ATTUALE",5,64,32,
" 1 ",6,1,32,
"Indice stimolo.",6,5,7,
" 8 ",6,39,32,
" Frequenza.....",6,42,7,
" 2 ",8,1,32,
"Stato masking..",8,5,7,
" 9 ",8,39,32,
"Sviluppo rise/fall..",8,43,7,
" 3 ",10,1,32,
"Indice output..",10,5,7,
" 10",10,39,32,
"Durata   rise/fall..",10,43,7,
" 4 ",12,1,32,
"Att. stimolo...",12,5,7,
" 11",12,39,32,
"Durata plateau Btono",12,43,7,
" 5 ",14,1,32,
"Att. masking...",14,5,7,
" 12",14,39,32,
"Durata plateau noise",14,43,7,
" 6 ",16,1,32,
" Polarita`.....",16,4,7,
" 7 ",18,1,32,
"Durata click...",18,5,7,
" 13",16,39,32,
```

```
"Numero stimoli......",16,43,7,
" 14",18,39,32,
"Ritardo stimolo.....",18,43,7,
"-------------------------------------------------------------------------------",20,3,23,
"   UP    ^  Seleziona parametro precedente  <ESC> Predispone variazione  ",21,3,23,
"   DOWN V   Seleziona parametro successivo                               ",22,3,23,
"   LEFT <   Seleziona nuovo default         <ENTER> FINE CAMPO/FINE EDIT ",23,3,23
};


/*
-- All'accensione vengono presentati i campi workd e pard: i primi relativi
   ai parametri attuali e i secondi relkativi ai parametri possibili tra i
   quali scegliere quelli di lavoro. Il comando 2 (settaggio parametri di
   default) ricopia defd in workd e pard defpar in pard.
*/


struct dispd {
char defd[10];
char pard[10];
char defpar[10];
char workd[10];
};

struct dispd videod[14]={
" CLICK "," CLICK "," CLICK "," CLICK ",
"  ON   ","  OFF  ","  OFF  ","  ON   ",
"Rph/Lph","Rph/Lph","Rph/Lph","Rph/Lph",
"  64   ","  60   ","  60   ","  64   ",
"  24   ","  20   ","  20   ","  24   ",
" POSIT "," POSIT "," POSIT "," POSIT ",
" 100uS "," 100uS "," 100uS "," 100uS ",
"   250 ","   250 ","   250 ","   250 ",
"LINEARE","LINEARE","LINEARE","LINEARE",
"   1mS ","   1mS ","   1mS ","   1mS ",
"   1mS ","   1mS ","   1mS ","   1mS ",
"   1mS ","   1mS ","   1mS ","   1mS ",
"    0  ","    0  ","    0  ","    0  ",
"   1mS ","   1mS ","   1mS ","   1mS "
};


struct dispv {
char string[10];
int posr;
int posc;
char cod;
int appart;
};

/*
-- Videox contiene le nomenclature di tutti i parametri possibili
*/
```

```
struct dispv videox[83]={
" CLICK ",6,21,0x00,1,
" LOGON ",6,21,0x01,1,
" BTONO ",6,21,0x02,1,
" BNOISE",6,21,0x03,1,
/* ------------------------------------ */
"  OFF  ",8,21,0x00,2,
"  ON   ",8,21,0x01,2,
/* ------------------------------------ */
"Rph/Lph",10,21,0x00,3,
"Lph/Rph",10,21,0x01,3,
"Rph/Rph",10,21,0x02,3,
"Lph/Lph",10,21,0x03,3,
"RLp/off",10,21,0x04,3,
"off/RLp",10,21,0x05,3,
"Rff/Lff",10,21,0x06,3,
"Lff/Rff",10,21,0x07,3,
/* ------------------------------------ */
"  60   ",12,21,60,4,
"  64   ",12,21,64,4,
"  68   ",12,21,68,4,
"  72   ",12,21,72,4,
"  76   ",12,21,76,4,
"  80   ",12,21,80,4,
"  84   ",12,21,84,4,
"  88   ",12,21,88,4,
/* ------------------------------------ */
"  20   ",14,21,20,5,
"  24   ",14,21,24,5,
"  28   ",14,21,28,5,
"  32   ",14,21,32,5,
"  36   ",14,21,36,5,
"  40   ",14,21,40,5,
"  44   ",14,21,44,5,
"  48   ",14,21,48,5,
/* ------------------------------------ */
" POSIT ",16,21,0x00,6,
" NEGAT ",16,21,0x01,6,
"ALTERN ",16,21,0x02,6,
/* ------------------------------------ */
" 100uS ",18,21,0x00,7,
" 200uS ",18,21,0x01,7,
" 500uS ",18,21,0x02,7,
"1000uS ",18,21,0x03,7,
/* ------------------------------------ */
"   250 ",6,64,0x00,8,
"   500 ",6,64,0x01,8,
"  1000 ",6,64,0x02,8,
"  2000 ",6,64,0x03,8,
"  3000 ",6,64,0x04,8,
"  4000 ",6,64,0x05,8,
"  6000 ",6,64,0x06,8,
"  8000 ",6,64,0x07,8,
/* ------------------------------------ */
"LINEARE",8,64,0x00,9,
"COSENIC",8,64,0x01,9,
/* ------------------------------------ */
```

```
"   1mS ",10,64,0x00,10,
"   2mS ",10,64,0x01,10,
"   3mS ",10,64,0x02,10,
"   4mS ",10,64,0x03,10,
"   5mS ",10,64,0x04,10,
"   6mS ",10,64,0x05,10,
"   7mS ",10,64,0x06,10,
"   8mS ",10,64,0x07,10,
"   9mS ",10,64,0x08,10,
"  10mS ",10,64,0x09,10,
"  20mS ",10,64,0x0A,10,
"  30mS ",10,64,0x0B,10,
"  40mS ",10,64,0x0C,10,
"  50mS ",10,64,0x0D,10,
"  60mS ",10,64,0x0E,10,
"  70mS ",10,64,0x0F,10,
"  80mS ",10,64,0x10,10,
"  90mS ",10,64,0x11,10,
" 100mS ",10,64,0x12,10,
" 200mS ",10,64,0x13,10,
" 300mS ",10,64,0x14,10,
" 400mS ",10,64,0x15,10,
" 500mS ",10,64,0x16,10,
" 600mS ",10,64,0x17,10,
" 700mS ",10,64,0x18,10,
" 800mS ",10,64,0x19,10,
" 900mS ",10,64,0x1A,10,
"1000mS ",10,64,0x1B,10,
/* ------------------------------------ */
"    0  ",16,72,0,11,
"    2  ",16,72,2,11,
"    4  ",16,72,4,11,
"    6  ",16,72,6,11,
"    8  ",16,72,8,11,
"   10  ",16,72,10,11,
"   12  ",16,72,12,11,
"   14  ",16,72,14,11
};


/*
----- DESCRIZIONE ARRAY INTERNI ----------------------------------

   limiti  = contiene per ognuno dei quattordici campi da presentare
             il limite inferiore ed il limite superiore tra i quali
             puo` essere variato il parametro relativo al campo
             in corso. Detta coppia di dati e' riferita all'indice
             numerico della struttura videox.

   parametro = contiene per ogni campo l'indice riferito alla struttura
               videox di rappresentazione stringa. Detti valori sono
               aggiornati dalla funzione aggiorna.

   defparame = contiene i valori da ricare in parametro ogni qualvolta
               si riselezionano i parametri di default.

   prig - pcol = contengono le coordinate riga/colonna relative alla
```

```
                rappresentazione su video dei parametri.

   prig - pcol = contengono le coordinate riga/colonna relative alla
                rappresentazione su video dei valori attuali di lavoro.

   corrisponde = contiene per ogni campo la corrispondenza tra l'array
                parametro e l'array work_uditivo/def_uditivo (vedi:
                alcune informazioni da trasmettere allo stimolatore
                sono costituite da due byte).

   ------------------------------------------------------------------------
*/

   int  voce;
   int  corrisponde[14]={2,3,4,5,7,9,10,11,12,13,14,15,16,18};
   int  prig[14]={6,8,10,12,14,16,18,6,8,10,12,14,16,18};
   int  dcol[14]={21,21,21,21,21,21,21,64,64,64,64,64,64,64};
   int  pcol[14]={29,29,29,29,29,29,29,72,72,72,72,72,72,72};
   int  limiti[28]={0,3,4,5,6,13,14,21,22,29,30,32,33,36,37,44,
                    45,46,47,74,47,74,47,74,75,83,47,74};
   int  defparame[14]={0,4,6,14,22,30,33,37,45,47,47,47,75,47};
   int  parametro[14]={0,4,6,14,22,30,33,37,45,47,47,47,75,47};

/*
 ------------------------------------------------------------------
 --- Entry point globale modulo uditivo ---------------------------
 ------------------------------------------------------------------
*/


uditivo(param)
   int param;
{
   int k,j;
   extern char per_status[];
   extern char work_uditivo[];
   extern char def_uditivo[];
   switch (param)
   {
      case '1':
              per_status[1]=1;
              iniz_udi();
              break;
      case '2':
              for (k=0; k<14; k++)
              {
                 for (j=0; j<10; j++)
                    videod[k].workd[j] = videod[k].defd[j];
                 for (j=0; j<10; j++)
                    videod[k].pard[j] = videod[k].defpar[j];
                 work_uditivo[k] = def_uditivo[k];
                 parametro[k]=defparame[k];
              }
              break;
      case '3':
              break;
      case '4':
```

```
                edit_uditivo();
                break;
        case '5':
                per_status[1]=2;
                tx_uditivo();
                break;
        default:
                break;
    }
}



edit_uditivo()
{
    extern def_uditivo[];
    extern work_uditivo[];
    int  a,b,index;
    char ch,flag;
/* ------------------------------------------------------------------ */
    scr_clr();
    for (a=0; a<39; a++)
       scrivix(&screen[a]);
    scrivi_attuali();
    scrivi_param();
    index=0;
    voce=0;
    intensifica();
    flag=0x00;
    while(!flag)
    {
       while(((ch=ci())!=0)&&(ch!=13)&&(ch!=27));
       switch (ch)
       {
          case 0:
                  while(((ch=ci())!=80)&&(ch!=72)&&(ch!=75)&&(ch!=77));
                     switch (ch)
                     {
                         case 72:
                                 if(voce!=0)
                                 {
                                    normal();
                                    voce--;
                                    intensifica();
                                 }
                                 break;
                         case 75:
                                 if(voce>=7)
                                 {
                                     normal();
                                     voce -= 7;
                                     intensifica();
                                 }
                                 break;
                         case 77:
                                 if(voce<=6)
```

```
                                {
                                    normal();
                                    voce += 7;
                                    intensifica();
                                }
                                break;
                        case 80:
                                if(voce!=13)
                                {
                                   normal();
                                   voce++;
                                   intensifica();
                                }
                                break;
                     }
                  break;
          case 27:
                  blinka();
                  varia_parametro();
                  break;
          case 13:
                  flag=0xFF;
/*************     scr_clr(); ***************/
                  break;
       }
   }
}



/* ================================================================ */
/* ===== TRASMETTE I PARAMETRI ALLO STIMOLATORE ================== */
/* ================================================================ */

tx_uditivo()
{
      char ind,out,check,dato;
      int j;

      extern char def_uditivo[];
      extern char work_uditivo[];
      extern char per_status[];
/* ---------------------------------------------------------------- */
/*----- Inizio trasmissione vera e propria -----*/

      ind=work_uditivo[0];
      out=send_addr(ind);
      if(out==2)                    /* non si e` ricevuto il COA*/
           per_status[1]=-2;
      check=work_uditivo[0];
      for(j=1;j<19;j++)
      {
         out=send_data(work_uditivo[j]);
/*       printf("\n%d --- %x",j,work_uditivo[j]);                */
         check=check+work_uditivo[j];
         if(out==2)                     /*non si e' ricevuto il COA */
              per_status[1]=-2;
```

```
        }

        dato=~check+1;
        out=send_data(dato);
        if(out==2)                  /*non si e` ricevuto il COA del cs*/
             per_status[1]=-2;

/**** PER UN ERRORE NEL FIRMWARE DELLO STIMOLATORE IL PRIMO TRIGGER
      NON VIENE RICONSCIUTO. IL TRIGGER POSTO SUBITO DI SEGUITO
      RIALLINEA LA SEQUENZA DEGLI STIMOLI ****************************/
      trigger(100);

}




varia_parametro()
{
   char ch,flag,temporanea[10];
   int liminf,limsup,indice,j;

   flag=0;
   while (!flag)
   {
      while(((ch=ci())!=0)&&(ch!=27));
      switch (ch)
      {
         case 0:
             while(((ch=ci())!=80)&&(ch!=72));
             liminf=limiti[voce*2];
             limsup=limiti[voce*2+1];
             switch (ch)
             {
                case 72:
                        if (parametro[voce]!=limsup)
                        {
                           parametro[voce]=parametro[voce]+1;
                           indice=parametro[voce];
                           rc_put(prig[voce],pcol[voce],
                                  videox[indice].string,0xCF);
                        }
                        break;
                case 80:
                        if (parametro[voce]!=liminf)
                        {
                           parametro[voce]=parametro[voce]-1;
                           indice=parametro[voce];
                           rc_put(prig[voce],pcol[voce],
                                  videox[indice].string,0xCF);
                        }
                        break;
             }
             break;
        case 27:
```

```
                flag=0xFF;
                indice=parametro[voce];
                for (j=0; j<10; j++)
                {
                   videod[voce].workd[j] = videox[indice].string[j];
                   videod[voce].pard[j] = videox[indice].string[j];
                }
                rc_put(prig[voce],dcol[voce],
                       videod[voce].workd,0x70);
                intensifica();
                aggiorna();
                break;
        }
    }
}



intensifica()
{
   rc_put(prig[voce],pcol[voce],videod[voce].pard,71);
}


normal()
{
   rc_put(prig[voce],pcol[voce],videod[voce].pard,0x07);
}


blinka()
{
   rc_put(prig[voce],pcol[voce],videod[voce].pard,0xCF);
}

rc_put(irig,icol,istri,iattr)
    int  irig,icol;
    char istri[],iattr;
{
    scr_rowcol(irig,icol);
    scr_aputs(istri,iattr);
}


scrivi_attuali()
{
    int  a;
    for (a=0; a<14; a++)
       rc_put(prig[a],dcol[a],videod[a].workd,0x70);
}


aggiorna()
{
   switch (voce)
   {
```

```
        case  3:
        case  4:
                aggiorna_val(0);
                break;
        case 12:
                aggiorna_val(1);
                break;
        default:
                aggiorna_val(2);
                break;
    }
}



aggiorna_val(para)
    int para;
{
    extern char work_uditivo[];
    int valore,indice,index;

    indice=parametro[voce];
    valore=videox[indice].cod;
    index=corrisponde[voce];
       work_uditivo[index]=valore;
       switch (para)
       {
          case 0:
                 work_uditivo[index+1]=~valore;
                 break;
          case 1:
                 work_uditivo[index+1]=0;
                 break;
       }
}



scrivi_param()
{
     int  a,indice,j;

     for (a=0; a<14; a++)
     {
        indice=parametro[a];
        for (j=0; j<10; j++)
           videod[a].pard[j]=videox[indice].string[j];
        rc_put(prig[a],pcol[a],videod[a].pard,0x07);
     }
}


scrivix(par)
struct disp *par;
{
   scr_rowcol(par->posizione_riga,par->posizione_colo);
```

```
   scr_aputs(par->stringa,par->attributo);
}


/* ================================================================ */
/* ===== ROUTINE DI RICONOSCIMENTO ================================ */
/* ================================================================ */


iniz_udi()
{
     extern char per_status[];
     char ind,out;

     ind=0x70;
     out=send_addr(ind);
     if(out==2)                    /* non si e` ricevuto il COA */
          per_status[1]=-1;
     out=send_data(0x01);
     if(out==2)                    /* non si e` ricevuto il COA */
          per_status[1]=-1;
     out=receive();
     if(out==2)                    /* non si e` ricevuto il COA */
          per_status[1]=-1;
/*   if(ind==out)
        printf("-------------------- CORRETTO");
     else                                                        */
     if(ind!=out)
     {
/*      printf("INDIRIZZO DI RITORNO NON CORRETTO");             */
        per_status[1]=-1;
     }
}
```

File: VISUALIZ.C

```
#include "go.h"


   extern unsigned char tipo_PR;
   extern float    medmed[RPR][CPR];
   extern char     oto_sel;

visualizza_dati(opt)
char opt;
{
   extern float archivio[];
   extern int  osservazioni[];
   extern int  fp0,fp1;
   extern int  possibili[];
   extern int  numero_prog;
   extern char nuovo_paziente;
   extern int  printer,on_off;
   extern char cognome[],nome[],giorno[],mese[],anno[];
   extern char sesso,eta[],residenza[],indirizzo[];
   extern char documento[],num_doc[],progressivo[];
   extern int  validi[VPR][CPR],errate[RPR][CPR];
   extern int  assenti[RPR][CPR];
   extern char effettuate[],possibilita[];
   extern float media[RPR][CPR],scarto[RPR][CPR];
   extern float varianz[RPR][CPR],assolut[RPR][CPR];
   extern float medmodi[MPR][CPR];
   extern char  arto_numerico,arto_num1,arto_num2;
   extern char  arto_selezionato,stim_selezionato,street[CPR][CPR];
   extern int   anni,gia_effettuate[];
   char type_dis,type_sem,esito_dis,esito_sem;
   int ieta,indice,index,num,a,meda;
   int index,jdex,adex,a,b,c,ir,ic,aaa,ioff,istar;
   char risposta,flag,record[80],*fonte,*resti,*resti1,data_odierna[9];
   char *mess2,*mess3,*mess4,bell=7,chmodo[7]={0,0,0,0,0,0,0};
   mess2="                                                    ";
   mess3="       V I S U A L I Z Z A Z I O N E    D A T I      ";
   mess4=" <cr> successiva                       <esc> termina ";
   scr_clr();
   data_int(data_odierna);
   presenta(2,10,71,mess2);
   presenta(3,10,71,mess3);
   presenta(4,10,71,mess2);
   sprintf(record," %s %s %s %d",cognome,nome,data_odierna,anni);
   presenta(21,12,23,record);
   fasce();
   lintest();
/* stampall();  */
   ieta=anni;
   ir=4;
   ic=5;
   if(tipo_PR==GONOGOC)
   {
      ioff=3;
      istar=0;
```

```
}
else
{
   ioff=2;
   istar=1;
}
for(a=istar; a<=ioff; a++)
{
   ir+=2;
   if(street[a][tipo_PR]==0x55)
   {
      determina_stimart(a);
      if(tipo_PR==GONOGON)
         fonte =" TONO ALTO          ";
      else
      {
         if(oto_sel==0)
            fonte =" ORECCHIO DESTRO    ";
         else
            fonte =" ORECCHIO SINISTRO ";
      }
      switch (arto_selezionato)
      {
         case 1: resti="PULSANTE DESTRO    ";
                 break;
         case 2: resti="PULSANTE SINISTRO ";
                 break;
      }
      sprintf(record,"COMBINAZIONE = %s - %s",fonte,resti);
      presenta(ir,10,23,record);
      jdex=a*40;
      meda=a*2;
      for(b=jdex; b<jdex+10; b++)
      {
         sprintf(record," %4d ",validi[b][tipo_PR]);
         presenta(ir+1,ic,23,record);
         ic+=7;
      }
      ic=5;
      for(b=jdex+10; b<jdex+20; b++)
      {
         sprintf(record," %4d ",validi[b][tipo_PR]);
         presenta(ir+2,ic,23,record);
         ic+=7;
      }
      ic=5;
      for(b=jdex+20; b<jdex+30; b++)
      {
         sprintf(record," %4d ",validi[b][tipo_PR]);
         presenta(ir+3,ic,23,record);
         ic+=7;
      }
      ic=5;
      for(b=jdex+30; b<jdex+40; b++)
      {
         sprintf(record," %4d ",validi[b][tipo_PR]);
         presenta(ir+4,ic,23,record);
```

```
            ic+=7;
         }
         if(tipo_PR == GONOGOC)
            sprintf(record,
     "DATI RISULTANTI = [M%6.1f V%6.1f MD%6.1f E%2d ]",
            media[a][tipo_PR],scarto[a][tipo_PR],
            varianz[a][tipo_PR],errate[a][tipo_PR]);
         else
            sprintf(record,
     "DATI RISULTANTI = [ M%6.1f V%6.1f MD%6.1f MD1%6.1f MM%6.1f E%2d ]",
      media[a][tipo_PR],scarto[a][tipo_PR],
      medmodi[meda][tipo_PR],medmodi[meda+1][tipo_PR],
      medmed[a][tipo_PR],errate[a][tipo_PR]);
         presenta(ir+5,5,23,record);
         ir+=6;
         if(ir >= 16)
         {
            presenta(23,10,71,mess4);
            printf("%c",bell);
            while(((risposta=ci())!=0x1B)&&(risposta!=0x0d));
            switch(risposta)
            {
               case 0x0d:
                    for(c=4; c<21; c++)
                    {
                       scr_rowcol(c,1);
                       scr_clrl();
                    }
                    fasce();
                    ir=4;
                    break;

               case 0x1b:
                    return(27);
                    break;
            }
         }
      }
      ic=5;
   }
   presenta(23,10,71,mess4);
   printf("%c",bell);
   while(((risposta=ci())!=0x1B));
}

stampall()
{
   extern int  validi[VPR][CPR],errate[RPR][CPR];
   int tipo,ipx;
   int a,b,c;

   ipx=open("PRN:",1);
   for(tipo=0; tipo<3; tipo++)
   {
      fprintf(ipx,"\n--- tipo = %d ---",tipo);
      for(a=0; a<160; a+=10)
      {
```

```
            fprintf(ipx,"\n%4d--",a);
            for(b=a; b<a+10; b++)
               fprintf(ipx,"%4d ",validi[b][tipo]);
         }
      }
      close(ipx);
}
```

Bibliografia

[1] J.Richard Simon, James V. Hinrichs
"Auditory S-R Compatibility:Reaction time as a function of ear-hand corrispondence and ear-response-location corrispondence." Journal of Esperimental Psychology 1980 vol 86 No. 1, 97-102.

[2] L. Bedini, E. Bozzi, A.Ribolini
"Strumento programmabile per la valutazione in linea dei tempi di reazione in prove uditive complesse: specifiche funzionali"
Nota Interna B4-14 Marzo 1990

[3] E. Bozzi, A. Ribolini
"Progetto di un sistema integrato HW/SW per l'analisi in tempo reale dei tempi di reazione in prove uditive complesse".
Nota Interna B4-18 Maggio 1990