Programmare in C su Amiga (37)

di Dario de Judicibus (MC2120 su MC-link)

Si conclude con la gestione dei campi, sia numerici che di tipo stringa, questa seconda serie di puntate dedicate ad i controlli di Intuition. Ritorna alla grande la scheda tecnica con la versione 2.0 dell'Amiga OS. A partire da questa puntata, infatti, incominceremo a vedere in dettaglio l'ultima versione del sistema operativo dell'Amiga

Terza ed ultima puntata dedicata ai campi. Abbiamo visto quali sono le strutture che servono a definire sia i campi numerici che quelli di tipo stringa.

Abbiamo inoltre visto come si crea un controllo di tipo *campo* e come si fa ad attaccarlo ad una finestra o ad un quadro.

Adesso non ci resta che vedere come si gestiscono gli eventi emessi da un campo e come si fa a leggere il valore immenso dall'utente nel campo stesso.

La gestione dei campi

Per la gestione di un campo ci rifaremo al solito schema già utilizzato in precedenza. Per ovvi motivi di spazio non riporteremo tutto il codice, ma solo le funzioni relative ai campi. In pratica abbiamo il solito programma scheletro con un menu dedicato ai campi che contiene una voce il cui testo è *Campi singoli*. Abbiamo scelto questo termine dato che si può creare un controllo per campi multipli con una tecnica analoga

```
/*************************************************************************
** Fields: crea o cancella i campi alfanumerici o numerici singoli      **
*************************************************************************/
void Fields (onoff)
  BOOL onoff;
{
  if (onoff)
  {
    if (mask & MSK_SGL)
      ShowMsgReq(w,"Campi già visibili");
    else
    {
     /*
     **  Crea i campi alfanumerici
     */
      fields[AAFIELD] = CreateField(AALFFID,&c,AALFTXT,AALFDEF,AALFLEN,
         AALFBOX,AALFCOL,AALFROW,AATXCOL,AATXROW,ALPHAFIELDCLASS);
      if (fields[AAFIELD] == NULL) CloseAll(EMSG_NOMEMORY) ;

     /*
     **  Attenzione: basta un solo campo per attivare la maschera
     */
      mask |= MSK_SGL;

      fields[ANFIELD] = CreateField(ANMRFID,&c,ANMRTXT,ANMRDEF,ANMRLEN,
         ANMRBOX,ANMRCOL,ANMRROW,ANTXCOL,ANTXROW,NUMERICFIELDCLASS);
      if (fields[ANFIELD] == NULL) CloseAll(EMSG_NOMEMORY) ;

     /*
     **  Visualizza i campi
     */
      DisplayFields(fields[AAFIELD],&c);
      DisplayFields(fields[ANFIELD],&c);
    }
  }
  else
  {
    if (!(mask & MSK_SGL))
      ShowMsgReq(w,"Campi già cancellati");
    else
    {
      if (fields[AAFIELD]) DeleteField(fields[AAFIELD],&c);
      if (fields[ANFIELD]) DeleteField(fields[ANFIELD],&c);
      mask &= ~MSK_SGL;
    }
  }
}
```

Figura 1 - Fields().

a quella utilizzata per i pulsanti a rilascio incrociato (e che lasciamo al lettore sviluppare). A questa voce è associato un sotto-menu con due voci: *Crea* e *Cancella*. Quando l'utente seleziona la prima, vengono creati due campi nella finestra a cui il menu si riferisce (vedi la funzione **Fields()** in figura 1, uno di tipo stringa ed uno numerico. Quando viene invece selezionata la seconda sotto-voce, i campi precedentemente visualizzati vengono rimossi. Ulteriore codice è stato aggiunto per gestire le condizioni di errore. Come si può vedere in figura, la procedura **Fields()** richiama le tre funzioni mostrate nella scorsa puntata. Vediamo ora le macro cui abbiamo accennato nella scorsa puntata e che servono ad ottenere il valore immesso nel campo dell'utente.

Le macro di lettura

Nella scorsa puntata visto le tre funzioni base di controlli di tipo *campo*: una per la creazione dinamica delle strutture atte a definire un campo (**CreateField()**), una per la visualizzazione del campo nel contenitore (**DisplayField()**), ed una per la rimozione del campo dal contenitore e dalla memoria (**DeleteField()**).

A queste aggiungiamo ora due macro che verranno in seguito utilizzate per ottenere da un campo il valore immesso dall'utente, una volta che sia stato intercettato l'evento emesso dal campo stesso. Le chiameremo **StringFieldValue()** e **NumericFieldValue()**.

Le due macro sono simili, solo che una ritorna la stringa immessa dall'utente nel campo, e quindi disponibile nell'area dati associata al campo stesso e mantenuta da Intuition, mentre l'altra ritorna direttamente l'intero corrispondente alla stringa presente nell'area dati, e che Intuition ha già convertito per noi nel caso di campi *numerici*.

La **StringFieldValue()** è riportata in figura 2, mentre la **NumericFieldValue()** è riportata in figura 3. La prima ritorna il puntatore all'area dati, la seconda una variabile di tipo **LONG.** Entrambe possono essere sostituite da una funzione equivalente che garantisca il controllo del tipo, e la validità del contenuto dell'area dati. Chi di voi ha incominciato a programmare in C++ infatti, sa che in questo linguaggio le macro sono fortemente sconsigliate, date le scarse garanzie di controllo che danno sul risultato che producono. Anche se è vero che questo è fondamentale prevalentemente con i linguaggi orientati agli oggetti come appunto il C++, potrebbe essere interessante provare a ridurre il numero di macro anche in programmi scritti in C, a favore di procedure piccole ma più sicure. Analogamente provate a sostituire le costanti definite al pre-processore con **#define**, con costanti vere e tipi **enum.** Potrebbe essere il primo passo per arrivare ad affrontare linguaggi più moderni come appunto il C++.

```
/*****************************************************************************
** StringFieldValue()        MACRO                 Versione 1.00           **
**                                                                         **
** Funzione fornita:  restituisce la stringa contenuta in un campo di      **
**                    tipo alfa                                            **
**                                                                         **
** Dati in ingresso:  field       puntatore al campo                       **
**                                                                         **
** Dati in uscita:    alfa        stringa del campo                        **
**                                                                         **
*****************************************************************************
**                                                                         **
** Definita in:  gdgusrh.c                                                 **
**                                                                         **
** #define StringFieldValue(f) ((STRI *)(((f)->SpecialInfo)->Buffer)       **
**                                                                         **
*****************************************************************************/
```

Figura 2 - StringFieldValue().

```
/*****************************************************************************
** NumericFieldValue()       MACRO                 Versione 1.00           **
**                                                                         **
** Funzione fornita:  restituisce l'intero contenuto in un campo di        **
**                    tipo numerico                                        **
**                                                                         **
** Dati in ingresso:  field       puntatore al campo                       **
**                                                                         **
** Dati in uscita:    longint     valore intero del campo                  **
**                                                                         **
*****************************************************************************
**                                                                         **
** Definita in:  gdgusrh.c                                                 **
**                                                                         **
** #define NumericFieldValue(f) ((STRI *)(((f)->SpecialInfo)->LongInt)     **
**                                                                         **
*****************************************************************************/
```

Figura 3 - NumericFieldValue().

```
/*****************************************************************************
** HandleEvent: gestione dei messaggi da Intuition                         **
*****************************************************************************/
 int HandleEvent(msg)
   IMSG *msg;
 {
   IMSG localmsg;     /* Questa è una fotocopia del messaggio ricevuto */
   int  result;       /* Questo è il valore da restituire al chiamante */
                      /* Fotocopiamo il messaggio originale            */
   CopyMem((char *)msg,(char *)&localmsg,sizeof(IMSG));

  /* ------------------------------------------------------------------ *
   * ATTENZIONE: i controlli di tipo VERIFY vanno messi PRIMA di       *
   *             rispondere al messaggio, altrimenti non servono a     *
   *             niente! Potevamo anche usare msg->Class, ovviamente.  *
   * ------------------------------------------------------------------ */
   switch (localmsg.Class)
   {
     case MENUVERIFY  : result = H_MenuVerify   (&localmsg);   break;
   }

   ReplyMsg((struct Message *)msg); /* OK. Adesso possiamo rispondere. */

  /* ------------------------------------------------------------------ *
   * ATTENZIONE: da questo punto in poi il messaggio puntato da "msg"  *
   *             non è più disponibile a questo task.                  *
   *             Useremo la copia locale salvata in "localmsg".        *
   *             MENUVERIFY è ripetuto per evitare che l'assegnazione  *
   *             di "result" in "default:" si sovrapponga a quella     *
   *             precedente. Questo è uno dei tanto modi per evitarlo. *
   * ------------------------------------------------------------------ */

   switch (localmsg.Class)
   {
     case CLOSEWINDOW  : result = H_CloseWindow   (&localmsg);   break;
     case MENUPICK     : result = H_MenuPick      (&localmsg);   break;
     case MOUSEBUTTONS : result = H_MouseButtons  (&localmsg);   break;
     case REFRESHWINDOW: result = H_RefreshWindow (&localmsg);   break;
     case GADGETDOWN   : result = H_GadgetDown    (&localmsg);   break;
     case GADGETUP     : result = H_GadgetUp      (&localmsg);   break;
     case MENUVERIFY   : /* # già trattato in precedenza # */    break;
     default           : result = GOAHEAD                   ;   break;
   };
   return (result);
 }
```

Figura 4 - HandleEvent().

L'immissione del dato

Vediamo adesso come si intercetta l'immissione da parte dell'utente di un valore nel campo, e come si ricava tale valore.

Innanzi tutto l'evento che ci segnala l'avvenuta immissione di un dato, o l'accettazione di un eventuale valore assegnato da programma *[default]*, è **GADGETUP**, come riportato in figura 4. Questo è dovuto appunto al fatto che abbiamo associato al campo solo la richiesta di segnalazione **RELVERIFY**, e non la **GADGETIMMEDIATE**. Come già accennato nella scorsa puntata, infatti, la deselezione del campo *non* viene attivata quando il pulsante del mouse viene rilasciato, bensì quando viene premuto *invio* oppure l'utente seleziona un altro controllo. La selezione invece è attivata, come al solito, quando l'utente fa click con il mouse sul campo, ma il programma, in genere, non ha bisogno di essere informato della cosa dato che ci pensa Intuition a gestire il campo, in tale fase. In pratica quindi, un campo rimane selezionato per tutto il tempo che l'utente vi rimane in edizione. A questo riguardo c'è da precisare una cosa: l'avvenuta immissione di un valore nel

La scheda tecnica

Con questa puntata, iniziamo a vedere in dettaglio la nuovissima versione del sistema operativo dell'Amiga: la 2.0. In realtà proprio nuova questa versione non è, dato che, chi ha la fortuna di possedere un Amiga 3000 è già in grado di utilizzarla tranquillamente. Tuttavia, non avendo la Commodore messo a disposizione questa versione in contemporanea anche agli utenti degli Amiga 500 e 2000 (per non parlar del 1000), che rappresentano tuttora la maggioranza degli utenti Amiga, essa è rimasta per parecchio tempo dominio di un gruppo ristretto di utenti e dei vari addetti ai lavori. Quest'ultimi hanno ovviamente usufruito anche di tutta la documentazione necessaria a sviluppare con l'Amiga OS 2.0 e di opportuni rilasci non definitivi con la possibilità di caricare il sistema anche su macchine differenti, come l'Amiga 2000.

Recentemente la Commodore ha annunciato un prossimo rilascio della 2.0 anche per l'Amiga 2000 ed il 500. Al momento di scrivere questo articolo questo rilascio non è ancora avvenuto, tuttavia l'ultima versione *beta* è ormai in distribuzione da un paio di mesi, e mi è stato confermato dalla Commodore stessa che può essere ritenuta l'ultima prima del rilascio ufficiale.

Così, dato che questo articolo uscirà dopo le vacanze estive, e dato l'approssimarsi della data agognata da molti utenti Amiga, ho chiesto ed ottenuto il permesso dalla Commodore di iniziare a presentare in questa rubrica la nuova versione, dal punto di vista dello sviluppo dei programmi (ricordo che, dell'Amiga OS 2.0 dal punto di vista dell'utente finale se ne era già parlato in dettaglio in questa rivista all'uscita dell'Amiga 3000).

Ho quindi il piacere di annunciare una nuova serie della *Scheda Tecnica*, interamente (ed in anteprima) dedicata all'**Amiga OS 2.8**. Parallelamente continueremo comunque la trattazione di vari argomenti relativi alla versione attuale (la 1.3), sia perché la maggior parte degli utenti Amiga ha modo di esercitarsi solo con questa al momento, sia perché comunque tutti gli argomenti trattati continuano a valere anche per la nuova versione. Il ritardo del rilascio della 2.0 anche sulle macchine più vecchie del 3000 era dovuto ad una certa incompatibilità della nuova versione rispetto alla precedente. Ora questa incompatibilità si è ridotta, mi hanno assicurato in Commodore, a meno del 5%, per la maggior parte limitata a programmi molto particolari (alcuni giochi, ad esempio), o che non interfacciano il sistema seguendo rigidamente le regole fornite dalla casa madre.

Tutti gli sviluppatori con cui ho parlato mi hanno assicurato che chi sviluppa con la 2.0 non torna indietor, dato la quantità e la qualità delle nuove funzioni. Per quello che mi riguarda, conto di verificarlo di persona al più presto, avendo ordinato un A3000.

Ed ora...

Inside 2.0

Facciamo una breve panoramica di ciò che la nuova versione ci mette a disposizione, tenendo presente che per il momento ci limiteremo a considerare solo le funzioni, macro o comandi nel loro complesso, non in dettaglio. Nelle tabelle che seguiranno, quindi, la colonna *Stesse* si riferisce a quelle funzioni che erano presenti con lo stesso nome anche nella versione precedente, senza entrare in merito a cambiamenti interni, eventuali parametri differenti, od a specifiche d'utilizzo cambiate.

Vediamo innanzitutto le librerie statiche, cioè quelle che vanno usate quando si compila e si lega il programma, dette appunto *link libraries*.

La versione 2.0 ha una nuova libreria statica, la *cx.lib.*, che include tutta una serie di funzioni complementari a quelle della **commidities. library.** Le altre due sono rimaste le stesse, esternamente, della versione precedente. In tutto quattordici nuove funzioni su quarantotto (29%).

LIBRERIE STATICHE	1.3 --> 2.0	Aggiunte	Tolte	Stesse
amiga.lib	Vecchia	--	--	27
cx.lib	*Nuova*	14	--	--
debug.lib	Vecchia	--	--	7
Tutte lel librerie statiche	2 --> 3	14	--	34

Le librerie dinamiche, utilizzabili quando il programma viene eseguito, o *dynamic shared libraries*, sono quelle che hanno subito i maggiori cambiamenti: ben dieci nuove librerie su ventitre. Le nuove funzioni, sia quelle appartenenti alle nuove librerie, sia quelle aggiunte e quelle vecchie, sono addirittura trecentosessantaquattro su settecentoottanta (47%).

La parte del leone la fa la **dos.library**, che da sole trentun funzioni passa a centocinquanta! Estremamente interessante l'aggiunta del blocco dinamico di singoli *record* di un file.

Da segnalare la **iffparse.library** e la **gadtools.library**.

La **graphics.library** contiene inoltre quattro nuove funzioni per la lettura e scrittura di segmenti orizzontali ed aree rettangolari di pixel.

LIBRERIE DINAMICHE	1.3 --> 2.0	Aggiunte	Tolte	Stesse
asl.library	*Nuova*	6	--	--
commodities.library	*Nuova*	25	--	--
diskfont.library	Vecchia	1	--	4
dos.library	Vecchia	119	--	31
exec.library	Vecchia	11	--	92
expansion.library	Vecchia	1	3	17
gadtools.library	*Nuova*	18	--	--
graphics.library	Vecchia	24	--	108
icon.library	Vecchia	4	--	8
intuition.library	Vecchia	38	--	72
iffparse.library	*Nuova*	40	--	--
keymap.library	*Nuova*	4	--	--
layers.library	Vecchia	4	--	25
mathffp.library	Vecchia	--	--	12
mathieeedoubbas.library	Vecchia	--	--	12
mathieeedoubtrans.library	Vecchia	--	--	17
mathieeesingbas.library	*Nuova*	12	--	--
mathieeesingtrans.library	*Nuova*	17	--	--
mathtrans.library	Vecchia	--	--	17
rexxsyslib.library	*Nuova*	10	--	--
translator.library	Vecchia	--	--	1
utility.library	*Nuova*	24	--	--
workbench.library	*Nuova*	6	--	--
Tutte le librerie dinamiche	13 --> 23	364	3	416

campo avviene *solamente* quando l'utente preme *invio*, non quando è selezionato un altro controllo, anche se questo comporta comunque la deselezione del campo. Se quindi un utente seleziona un campo, vi edita un qualunque valore e poi, senza premere *Invio* seleziona un altro controllo, il programma non viene avvertito da Intuition del fatto che l'area dati è stata modificata, se non per il fatto che riceve l'evento che segnala la nuova selezione. Se lo sviluppatore è interessato a *qualunque* modifica, anche a quelle non confermate dalla pressione del tasto *Invio*, è necessario che tenga traccia dei campi selezionati aggiungendo la richiesta **GADGETIMMEDIATE** alla **RELVERIFY**.

Un'altra possibilità è la seguente. Supponiamo di far apparire sullo schermo un quadro, con lo scopo di acquisire a programma alcune informazioni. Diciamo che il quadro contiene un certo numero di campi e due pulsanti, uno di accettazione dei valori immessi *[OK]*, ed uno di cancellazione richiesta *[Cancel]*. A questo punto, indipendentemente dal fatto che l'utente prema *Invio* per *tutti* i campi, o che lo faccia solo per alcuni lavorando tuttavia anche con gli altri, il programma andrà a verificare tutte le aree dati se l'immissione è stata confermata con il pulsante **OK**, altrimenti si limiterà ad ignorare qualunque operazione fatta.

Chiusa questa parentesi, torniamo ora al nostro esempio, come riportato in Figura 5. Abbiamo detto che l'immissione di un valore nel campo viene segnalato tramite **GADGETUP**. In realtà non è detto che l'utente abbia effettivamente immesso qualcosa nel campo. Basta che esso abbia selezionato il campo e poi abbia premuto subito *Invio*. In tal caso nulla sarà cambiato nell'area dati associata al campo, e quindi essa sarà vuota e conterrà il valore precedente-

Per quello che riguarda la gestione delle periferiche, non abbiamo nuovi *device*. Tuttavia anche qui i cambiamenti ci sono, anche se per la maggior parte dei casi sono invisibili ad una prima analisi esterna. In questo caso gli sviluppatori del sistema operativo hanno rafforzato le capacità dei *device* precedenti, aumentandone le possibili. In ogni caso abbiamo anche qui qualche funzione e qualche comando nuovo (sette in tutto).

DEVICE (*)	1.3 --> 2.0	Aggiunte	Tolte	Stesse
audio.device	Vecchia	--	--	7
clipboard.device	Vecchia	2	--	2
console.device	Vecchia	--	--	6
gameport.device	Vecchia	--	--	5
input.device	Vecchia	1	--	8
keyboard.device	Vecchia	--	--	5
narrator.device	Vecchia	--	--	--
parallel.device	Vecchia	--	--	2
printer.device	Vecchia	--	--	5
serial.device	Vecchia	--	--	3
timer.device	Vecchia	2	--	6
trackdisk.device	Vecchia	2	--	12
Tutti i device	12 --> 12	7	--	61

(*) NOTA: nel computo delle funzioni/comandi aggiunti, tolti, o rimasti invariati, non sono stati inclusi i comandi di tipo CMD_supportati. Ad esempio, dato che la «narrator.device» supporta solo funzioni e comandi standard (OpenDevice, CloseDevice, AbortIO e 6 verbi CMD_), non avendo subito modifiche esterne, tutte le colonne riportano «».*

E per finire, le risorse del sistema, o *resources*. Queste sono passate da quattro a sei, per un totale di dieci nuove funzioni su ventidue.

RISORSE	1.3 --> 2.0	Aggiunte	Tolte	Stesse
battclock.resource	*Nuova*	3	--	--
battmem.resource	*Nuova*	4	--	--
cia.resource	Vecchia	--	--	4
disk.resource	Vecchia	1	--	5
misc.resource	Vecchia	2	2	--
potgo.resource	Vecchia	--	--	3
Tutte le risorse	4 --> 6	10	2	12

Per quello che riguarda i file di inclusione *[header or include files]*, chi possiede la versione 5.x del *SAS/C*, troverà in uno dei dischetti anche tutti i file di tipo **.h** ed **.i** relativi alla versione 2.0, sia in formato leggibile, che in quello compresso.

Nelle prossime puntate incominceremo a vedere in dettaglio le nuove funzioni delle librerie dinamiche. Mi raccomando, non perdetevi i prossimi numeri di *MCmicrocomputer!*

```
/*****************************************************************************
** H_GadgetUp: gestisce l'evento GADGETUP                                   **
*****************************************************************************/
int H_GadgetUp(msg)
  IMSG *msg;
{
  IGDG *gdg;
  USHORT gid;
  char *temp ;

  /*
  **  E' stata fatta effettivamente una selezione? Se si, allora
  **  determiniamo a quale controllo si riferisce.
  */
  gdg = (IGDG *)msg->IAddress;       /* Puntatore al controllo          */
  gid = gdg->GadgetID;               /* Identificativo selezione        */

  /*
  **  Alcune definizioni per la stampa
  */
  :
  #define PRT_ALF(g,v) printf("Campo [%s] con valore [%s]\n",(g),(v))
  #define PRT_NMR(g,v) printf("Campo [%s] con valore [%ld]\n",(g),(v))

  /*
  **  BLOCCO PER LA GESTIONE DEI CODICI
  */
  switch (gid)
    {
  :
      case AALFFID: temp = (gdg->GadgetText)->IText ;
                    PRT_ALF(temp, StringFieldValue(gdg));
                    break;
      case ANMRFID: temp = (gdg->GadgetText)->IText ;
                    PRT_NMR(temp, NumericFieldValue(gdg));
                    break;
      defaults    : break;
    }

  return(GOAHEAD);
}
```

Figura 5 - H_GadgeUp().

```
#define SetFieldValue(f,v) \
        sprintf((char *)((STRI *)((f)->SpecialInfo)->Buffer),"%s\0",(v))
```

Figura 6 - SetField Value().

Novità nel mondo Amiga

Questo mese, riporto qui di seguito un estratto dell'annuncio ufficiale USA della nuova versione di Uniux per Amiga 3000UX, come riportato in *Compuserve* (riferimento *944@amix.commodore-.com).*

Unix Amiga ed A3000UX

La versione 1.1 dello *UNIX System V Release 4* per Amiga, è stata rilasciata per l'inclusione nelle macchine della serie A3000UX.

Hardware

- Amiga 3000 dotato di 68030 da 25 MHz., tastiera, mouse, drive da 3"1/2, porta seriale e parallela, come nell'A3000 base
- 4M o 8M di memoria RAM do tipo *FAST* da 32 bit sulla scheda madre.
- Un disco fisso SCSI da 100M o 200M.
- Scheda *Ethernnet.*

Due configurazioni saranno rese disponibili al più presto:

- 4M di RAM, HD da 100M, senza scheda Ethernet;
- 8M di RAM, HD da 200M, con scheda Ethernet;

Il sistema verrà rilasciato con l'Unix già installato sul disco fisso. Esso conterrà inoltre anche l'ultima versione dell'AmigaDOS. Inoltre, una unità nastro opzionale (A3070) sarà necessaria per reinstallare il software, future versioni del sistema operativo, od alcuni *pacchetti* piuttosto consistenti (che ovviamente non risulterebbe pratico installare da dischetti). L'unità A3070 è esterna e rimovibile, così da poter essere utilizzata da più unità, non contemporaneamente.

L'Amiga Unix non sarà per il momento disponibile per altre macchine, anche se in linea di massima non esistono motivi tecnici per cui non possa girare anche su di un A2000 opportunamente «rinforzato».

Software

Lo UNIX per Amiga è conforme alla versione *AT«MDSU»&T Unix System «MDNM»V release 4*, cioè la più recente. Si tratta di una versione che include caratteristiche dei precedenti rilasci dell' AT & T System V, dello Unix BSD, e di alcune versioni del SunOS e dello Xenix, oltre ad alcune novità.

Per i patiti di Unix, ecco uno stralcio di alcune caratteristiche del nuovo Amiga Unix:

- Compatibilità a livello comandi e codice C con lo Unix BSD.
- Compatibilità completa con lo Unix AT&T System V.
- Shell Bourne, Korn, e C, tutti con il controllo BSD.
- Comando man e manuale completo in linea.
- Compilatore ANSI C ed un completo sistema di sviluppo delle applicazioni.
- Compatibilità POSIX 1003.1 ed X/Open XPG3.
- Interfaccia ABI (Application Binary Interface) AT&T/Motorola m68k.
- Sistema completo di preparazione dei documenti nella tradizione Unix (**nroff, troff -ms, -mm, -mv, eqn, tbl, pic, grap,** filtri postscript, e via dicendo).
- Posta elettronica *(E-mail)* ed UUCP.
- Sistema X-Windows (X11R3, prossimamente X11R4).
- Interfaccia grafica Open Look.
- TCP/IP su Ethernet.
- Programmi *Berkeley internet* (**telnet, ftp, rlogin, rcp,** e via dicendo).
- Funzioni *Sun NFS file sharing* (per clienti e serventi).
- Funzioni *Sun Remote Procedure Call.*
- Interfaccia di rete AT&T TLI.
- *Virtual File System*, compreso il *Berkeley Fast File System.*
- Librerie dinamiche condivise.

A cui si vanno ad aggiungere, propri dell'Amiga Unix,

- Installazione e configurazione del sistema immediata e di facile utilizzo.
- Schermi virtuali (fino a 10 sessioni di **login**) con coesistenza di grafica e testo simultaneamente.
- Device Driver dell'Amiga con il codice sorgente (dischetti, porta seriale, interfacce SCSI A3000 ed A2091, A2065 Ethernet, scheda seriale A2232, e via dicendo).
- Programmi pubblici completi di sorgente, come *GNU Emacs*, il compilatore *GNU C*, l'interfaccia *elm mail*, i programmi *USENET* (**bnews, rn, nntp ed rrn**).
- Vari programmi di pubblico dominio come **less, shar, sc**.
- Vari giochi (perché no?).
- Coesistenza con AmigaDOS e supporto dei dischi AmigaDOS.

La prossima versione

Sempre dalla stessa fonte (Commodore-Amiga Unix Development Group), riporto alcune indicazioni relative ad alcune caratteristiche della prossima versione, la 2.0.

- X-Windows X11R4.
- Supporto dischetti MS_DOS 1.44M.
- Maggiore compatibilità con l'AmigaDOS.
- Un nuovo compilatore GCC.
- Migliori prestazioni.
- Migliori *driver* per l'unità nastro, dischi, e la porta seriale.

mente immesso dall'utente o dal programma stesso.

In genere è buona norma fornire ogni campo di un valore di *default*, se possibile, in modo da rendere l'interfaccia più semplice da usare, e ridurre di fatto la quantità di dati che l'utente deve battere sulla tastiera. Sarebbe anche opportuno che il programma ricordasse i cambiamenti effettuati dall'utente in un campo, in modo da evitargli di reintrodurre più volte gli stessi valori. In effetti, finché il campo esiste, ci pensa Intuition a mantenere l'area dati associata a quel campo, e quindi a ricordarsene il contenuto. Molti campi, tuttavia, fanno parte di quadri dinamici *[dialog box]* che vengono visualizzati quando occorrono, e poi cancellati dallo schermo una volta terminata l'immissione dei dati. In questo caso, quando il quadro scompare, i campi ad esso associati vengono anch'essi cancellati, in modo da liberare memoria (ad esempio con la **DeleteField()**). A questo punto, quando il quadro deve nuovamente essere visualizzato, i campi in esso contenuti vengono di fatto creati *ex novo*. E questo può rappresentare un problema di *usabilità* per l'interfaccia.

Facciamo un esempio. Supponiamo di aver scritto un elaboratore di testi, ed di avere sviluppato una funzione che permetta all'untente di personalizzare i passi di tabulazione.

Supponiamo inoltre che il quadro di impostazione della tabulazione venga richiamato premendo **Ctrl-Tab**, e che esso contenga un campo per impostare il passo di tabulazione (che supponiamo costante per semplicità), e tre pulsanti, uno per accettare il nuovo valore impostato e memorizzare il punto del documento dal quale esso è valido a tutti gli effetti, uno per cancellare la richiesta di impostazione, ed uno di guida *[help].*

Quando l'utente richiama il quadro la prima volta il campo di ingresso dovrà contenere un valore di *default*, magari ricavato dal sistema operativo, se esiste. Ad esempio, **«4»**. Supponiamo ora che il passo venga cambiato ad **«8»**. Quando l'utente preme *Invio*, il programma acquisisce il dato, ma *non mo-*

difica ancora il passo. Se a questo punto l'utente seleziona *Va bene*, il dato viene accettato ed il passo modificato. Se invece preme *Cancella*, il dato ignorato ed il passo rimane lo stesso.

Quando il quadro sparisce dallo schermo, tuttavia, tutta la memoria allocata per i pulsanti, il campo, ed il quadro stesso viene liberata. Sarebbe infatti troppo oneroso per il sistema mantenere le strutture necessarie ad ogni quadro e finestra che viene aperta per tutta la vita del programma. In questo modo, però, si perdono anche le aree dati dei campi. Naturalmente il passo di tabulazione è stato comunque salvato da qualche parte. Così, quando il quadro è richiamato una seconda volta, sarà sempre possibile pre-impostare il valore del campo con il passo corrente, e cioè **«8»**, e non **«4»** come era all'inizio. Questa tecnica si chiama del *default dinamico*.

Per impostare il valore dell'area dati basta utilizzare la stessa tecnica utilizzata nella scorsa puntata quando abbiamo descritto la **CreatedField()**. In pratica basta copiare il valore in questione, sempre sotto forma di stringa di caratteri anche nel caso di campi numerici, come già detto nella 36ª puntata, nell'area dati associata al campo, aggiungendovi in fondo un ulteriore byte nullo. In figura 6 è mostrata una possibile soluzione che utilizza una macro. In realtà, sarebbe opportuno scrivere una funzione al posto della macro, che verifichi anche che la stringa da copiare nell'area dati non sia lunga quanto o più di quest'ultima, per evitare di sporcare aree di memoria contigue. Lo stesso controllo andrebbe fatto nella **CreateField()**. Lascio a voi, come esercizio, quello di scrivere la nuova funzione, e di aggiungere detta verifica ed eventuali altre isturzioni alle funzioni proposte in queste ultime puntate, al fine di renderle più sicure e perché no, anche un po' più intelligenti. Non dimenticate inoltre che quando si modifica un controllo, è prassi toglierlo temporanemente dalla lista dei controlli, per poi riaggiungervelo.

La logica della **GadgetUp()**, tanto per tornare al codice in figura, è quella solita. Una volta ricevuto il messaggio da Intuition relativo all'immissione del dato nel campo, se ne ricava il puntatore alla struttura principale del controllo e da qui l'identificativo del campo. A questo punto c'è un blocco per la gestione del controllo, nel nostro caso, si limita a stampare il contenuto dell'area dati utilizzando le due macro viste in precedenza, distinguendo tra campi stringa e campi numerici.

Conclusione

Anche con i campi abbiamo finito. Nella prossima puntata affronteremo il terzo ed ultimo controllo base di Intuition: il *controllo proporzionale*. Nel frattempo esercitatevi lavorando sulle funzioni presentate in queste puntate, migliorandole, eliminando eventuali errori, ed aggiungendo nuove possibilità. Un'altra serie di esercizi molto utili potrebbe essere quella di creare nuovi controlli misti, mettendo insieme campi e pulsanti. Ad esempio, potreste creare un campo che è di tipo stringa o numerico a seconda del valore di un pulsante a rotazione ad esso collegato, trattando il tutto come un unico controllo. Buon divertimento! MC

Il software MS-DOS, Amiga e Macintosh di Pubblico Dominio e Shareware distribuito da

microcomputer®
HARDWARE & SOFTWARE DEI SISTEMI PERSONALI

in collaborazione con Microforum

Questo software non può essere venduto a scopo di lucro ma solo distribuito dietro pagamento delle spese vive di supporto, confezionamento, spedizione e gestione del servizio. I programmi classificati Shareware comportano da parte dell'utente l'obbligo morale di corrispondere all'autore un contributo indicato al lancio del programma.

MSDOS

COMUNICAZIONE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
COM/01	ONE TO ONE	mc104
COM/02	PROCOMM Noto programma di comunicazione	Hard disk
COM/03	OMEGA LINK	mc106
COM/04	BACKCOMM	mc103
COM/05	ZIP	mc110
COM/06	FOSSIL DRIVER & .TPU	mc110
COM/07	MAXIHOST	mc110

DATABASE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
DBS/01	EASY LABELS Per creare etichette	
DBS/02	VIDEO DATABASE	mc105 Hard disk
DBS/03	HOME MANAGER DataBase, calcolatrice e calendario	Hard disk
DBS/04	MAIL-MONSTER	mc103
DBS/05	MAKE MY DAY Per organizzare il lavoro	
DBS/06	PC-FILE+	mc106
DBS/07	TASK MASTER Projet Plaining	
DBS/08	RELIANCE MAILING LIST Mailing per associazioni culturali	
DBS/09	DMS	mc107
DBS/10	ARCHIVIO PARROCCHIALE	mc109

EDUCATIVO

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
EDU/01	ABC FUN KEYS	mc103
EDU/02	COMPUTER TUTOR Auto-apprendimento del computer	
EDU/03	PC-FASTYPE Imparare professionalmente ad usare la tastiera	CGA
EDU/04	GEOBASE ARCH. GEOGRAFICO	mc109

GIOCO

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
GIO/02	2BIT POKER Poker Canadian	EGA/VGA
GIO/03	ASTRO BLASTER Clone di Space Invaders	PC-AT/286
GIO/04	ALDO'S ADVENTURE	mc103 EGA/VGA
GIO/05	CAESAR Strategia	BASIC+EGA/VGA
GIO/07	CLONE INVADERS Clone di Space Invaders	
GIO/08	EGAINT	mc104 EGA/VGA
GIO/09	PC-JIGSAW Puzzle	
GIO/10	MAHJONG Solitario orientale	EGA/VGA
GIO/11	SUPER PINBALL Super Flipper	
GIO/12	ARK Clone di Arkanoid	EGA/VGA
GIO/13	BANYON WARS Strategia	EGA/VGA
GIO/14	CAPTAIN COSMIC Gioco grafico	EGA/VGA
GIO/16	EGA GOLF Gioco del Golf	EGA/VGA
GIO/17	EGA TREK Star Trek	EGA/VGA
GIO/18	JOUST VGA Gioco da bar	VGA
GIO/19	MINER VGA	mc104 VGA
GIO/21	MOSAIX Puzzle	VGA
GIO/22	OTHELLO EGA	mc103 EGA/VGA
GIO/23	POKER SOLITAIRE Poker da soli	EGA/VGA
GIO/24	QUATRIS Tetris con Bombe ecc.	EGA/VGA
GIO/25	SHARKS Giocate ai sommozzatori	EGA/VGA
GIO/26	SLOT EGA Slot Machine	EGA/VGA
GIO/27	BASSTOUR Pesca d'altura	EGA/VGA
GIO/28	BLACKJACK Gioco da Casinò	EGA/VGA
GIO/29	GALACTIC BATTLE Clone di Invaders con sonoro	EGA/VGA
GIO/30	HOUSE OF HORRORS Casa degli orrori	EGA/VGA
GIO/31	NOID Consegnate la pizza all'ultimo piano	EGA/VGA
GIO/32	PINBALL EGA Super Flipper	EGA/VGA
GIO/33	STARDEF Missili distruggono la terra...	
GIO/34	MAHJONG EGA Gioco di società orientale	EGA/VGA
GIO/35	MR.SPOCK	mc105 EGA/VGA
GIO/36	MONUMENTS OF MARS	mc106
GIO/37	PHARAOH'S TOMB	mc106
GIO/38	POKER	mc107EGA/VGA
GIO/39	NIM	mc108 CGA
GIO/40	TESORI	mc108 CGA
GIO/41	TOMBOLA	mc108 CGA
GIO/42	SMILE	mc109 VGA
GIO/43	CHINESE SOLITARIE	mc111 VGA
GIO/44	TRETRIX	mc111 VGA

GRAFICA

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
GRF/01	FINGER PAINT Programma di disegno	
GRF/02	PC-KEY-DRAW	mc107 CGA
GRF/03	H&P CALENDAR	mc103
GRF/04	PC-DEMO SYSTEM	mc105
GRF/05	GRAPHICWORKSHOP	mc106

SPREADSHEET

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
SPD/01	AS-EASY-AS	mc103
SPD/02	EXPRESS-CALC	mc104
SPD/03	EZ-SPREADSHEET Calcoli di budget	
SPD/04	INSTACALC	mc107
SPD/05	QUEBECALC Spreadsheet 3D	

UTILITY

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
UTI/01	PC-DESK-TEAM	mc107
UTI/02	HARD DISK UTILITIES Per gestire l'Hard Disk	Hard Disk
UTI/03	DOS HELP	mc104
UTI/04	DISK SPOOL II	mc103
UTI/05	LOCKTITE Protegge i file con password	
UTI/06	VIRUS SCAN Cerca i virus	
UTI/07	LHARC	mc105
UTI/08	ARJ	mc105
UTI/09	LZEXE	mc105
UTI/10	DIET	mc105
UTI/11	PKLITE	mc105
UTI/12	NEWSPACE	mc105
UTI/13	CATDISK	mc105
UTI/14	POINT&SHOOT	mc105
UTI/15	SHEZ	mc106
UTI/16	ZZAP	mc106
UTI/17	GUARDIAN ANGEL	mc107
UTI/18	STORE	mc107
UTI/19	TXT	mc107
UTI/20	xSET	mc108
UTI/21	ZAPDIR	mc108
UTI/22	UTILITY COLLECTION	mc109
UTI/23	DiR	mc109
UTI/24	CLEANUP	mc111
UTI/25	SAB DISKETTE UTILITY	mc111
UTI/26	TIF2GRAY	mc111
UTI/27	FILLDISK	mc111
UTI/26	ORASCO	mc111
UTI/27	XDIR	mc111

VARIE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
VAR/01	COMPOSER Per suonare al computer e stampare lo spartito	
VAR/02	CHECK-MATE Controllo delle finanze personali	
VAR/03	PIANO-MAN	mc104
VAR/04	BARTENDER Tutti i cocktail	mc103
VAR/05	DIET DISK La dieta al computer	
VAR/06	ELEMENTARY C Per programmatori in C	
VAR/07	RECIPES	mc104

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE	
VAR/08	PERSONAL C COMPILER	mc105	
VAR/09	MOUSE.TPU & NEWEXEC	mc106	
VAR/10	TSR, PRINT & GESTECC	mc106	
VAR/11	ARIANNA	mc106	
VAR/12	TOTOPROJET	mc108	CGA
VAR/13	COVER	mc108	
VAR/14	CODICE FISCALE	mc109	Hard disk
VAR/15	FLIGHT	mc109	
VAR/16	DIZIONARIO INFORMATICO	mc109	
VAR/17	ITALIA90	mc110	
VAR/18	TATA-BIGNOMIX UTILITY	mc110	
VAR/19	QUICK BASIC ROUTINES	mc110	

WORDPROCESSOR

WPR/01	W.P.FOR CHILDREN Per insegnare ai bambini il WP	
WPR/02	FREEWORD	mc103
WPR/03	PC-WRITE	mc106
WPR/04	THESAURUS PLUS Sinonimi in inglese (TSR)	
WPR/05	GALAXY	mc104
WPR/06	EDITOR	mc110

AMIGA

COMUNICAZIONE

AMCO/01	AMIPAC	mc110

GIOCO

AMGI/02	WELLTRIX	mc105
AMGI/03	SYS	mc105
AMGI/04	SCOPONE SCIENTIFICO	mc108
AMGI/05	LA FINE DI UN TIRANNO	mc109
AMGI/06	LA PANTERA SIAMO NOI	mc109
AMGI/07	MEGABALL	mc110

GRAFICA

AMGR/01	PRINTSTUDIO	mc104
AMGR/02	TEXTPAINT	mc105
AMGR/03	SCREENX	mc105
AMGR/04	SETPAL	mc105

SPREADSHEET

AMSP/01	SPREAD	mc104
AMSP/02	EQUATIONWRITER	mc110

UTILITY

AMUT/01	MACH III	mc104
AMUT/02	RULER	mc104
AMUT/03	HEX	mc104
AMUT/04	MOM	mc104
AMUT/05	CB	mc104
AMUT/06	ZETAVIRUS	mc104
AMUT/07	DIRMASTER	mc105
AMUT/08	KDC	mc105
AMUT/09	XCOPYIII	mc105
AMUT/10	CD2TAPE	mc105
AMUT/11	BBS & LOG	mc106
AMUT/12	UTILITIES	mc106
AMUT/13	VIEW80 II	mc106
AMUT/14	MATCALC	mc106
AMUT/15	ICONMASTER	mc106
AMUT/16	HERMIT	mc106
AMUT/17	TURBO IMPLODER	mc106
AMUT/18	FONTS2PRINTER	mc107
AMUT/19	SVD	mc107
AMUT/20	MC-PROGRAMS	mc107
AMUT/21	CHP&SAVE-PREFS	mc107
AMUT/22	CIDITEIP	mc108
AMUT/23	DISKEDITOR	mc108
AMUT/24	5 UTILITY	mc108
AMUT/25	OROLOGIO PARLANTE	mc108
AMUT/26	LSLAB	mc110
AMUT/27	DIRWORK	mc111
AMUT/28	SCREENMOD	mc111
AMUT/29	SYSINFO	mc111
AMUT/30	SUPERDUPER	mc111

VARIE

AMVR/01	FRACTUS	mc108
AMVR/02	RUBRICA, DACIA & GESTFATT	mc109
AMVR/03	FUNZ3D	mc109
AMVR/04	PLAYSMUS	mc110
AMVR/05	MULTI PLAYER	mc111
AMVR/06	DRAWMAP	mc111

MACINTOSH

COMUNICAZIONE

MICO/01	RED RYDER	mc110

EDUCATIVO

MIED/01	KID PIX	mc107
MIED/02	NUMBER TALK	mc107
MIED/03	ALPHA TALK	mc107

GIOCO

MIGI/01	STELLA OBSCURA	mc106
MIGI/02	PARARENA	mc106
MIGI/03	VIDEO POKER FOR FUN	mc106
MIGI/04	SPACE STATION PHETA	mc106
MIGI/05	STRATEGO	mc106
MIGI/06	THE LAWNZAPPER	mc107
MIGI/07	MACTRIS	mc107
MIGI/08	CANFIELD	mc107
MIGI/09	YAHTZEE	mc108
MIGI/10	GLIDER	mc108
MIGI/11	MACNINJA	mc108
MIGI/12	GLIPHA	mc108
MIGI/13	MONOPOLY	mc109
MIGI/14	GOLF	mc109
MIGI/15	WHEEL	mc109
MIGI/16	GUNSHY	mc109
MIGI/17	MEGAROIDS	mc110
MIGI/18	SHUFFLEPUCK	mc110
MIGI/19	CRIMINALS	mc111

GRAFICA

MIGR/01	CALENDAR MAKER	mc106

STACK

MISK/01	FOOD 1	mc111
MISK/02	BUSINESS 1	mc111
MISK/03	SOUND 1	mc111

UTILITY

MIUT/01	OLIVER'S BUTTONS	mc107
MIUT/02	POPCHAR	mc107
MIUT/03	RAMDISK	mc108
MIUT/04	SCROLL2	mc109
MIUT/05	DECK EDITOR	mc109
MIUT/06	BANNER MAKER	mc110
MIUT/07	SPEEDOMETER	mc110

VARIE

MIVR/01	RIDICOLO	mc108
MIVR/02	ELIZA	mc109

Compilare e spedire a: ***MCmicrocomputer***

Desidero acquistare il software di seguito elencato al prezzo di **L. 8.000 a titolo (ordine minimo: tre titoli).** Per l'ordinazione inviare l'importo (a mezzo assegno, c/c o vaglia postale) alla: Technimedia srl, Via Carlo Perrier 9, 00157 Roma.

dischetti da ☐ 3.5" ☐ 5.25"

Codici: _____

Totale dischi ☐ x 8.000=Lire _____

Manuali in italiano:

☐ TSPD/01 AS EASY AS ☐ TUTI/01 HARD DISK UTILITIES

☐ TVAR/02 CHEKMATE ☐ TWPR/05 GALAXY

Totale manuali ☐ x 8.000=Lire _____

Nome e Cognome _____

Indirizzo _____

CAP/Città _____

Telefono _____

MCmicrocomputer non offre alcuna garanzia e non si assume alcuna responsabilità sugli eventuali danni diretti o indiretti derivanti dall'utilizzo del software distribuito

NOTEBOOK LAPTOP

PORTATILE ZENITH 8088

1 Mega ram - hd 20 mega - drive 720 batt.ricaricabile aut. **1.490.000**

Portatile zenith 80286

1 Mega ram - Hard disk 20 mega - drive 1.44 mega - Batt. ricaricabile. **L.2.280.000**

NOTEBOOK ZENITH 80286

Notebook con processore 80286 a 20 mhz display vga 640x480 retroilluminato 9" - attacco per monitor esterno a colori - drive 1,44 mega - hard disk 31 mega con possib. 60 mega - 6,5 ore di autonomia. ***L.3650.000***

NOTEBOOK ZENITH 386 SX

Notebook con processore 80386 sx - display vga 640x480 retroilluminato con 16 toni di grigio - attacco per monitor esterno a colori - 2 mega di memoria ram espandibile a 8 mega - hard disk da 60 mega - 6,5 ore di autonomia in programmazione continua. **L. 4.550.000**

NOTEBOOK 386 SL

Processore 80386/25 - display vga 640x480 retroilluminato con 32 toni di grigio - monitor esterno - 5 mega ram espandibile - hard disk da 60 mega espandibile a 120 mega - piu' di 8 ore di autonomia in programmazione continua! - velocita' oltre 30 mhz. L.6.800.000

Disponiamo inoltre della gamma:

TOSHIBA COMPAQ

STAMPANTI

9 AGHI

EPSON LX400	L.340.000
PANASONICKX/P1081	L..320.000
PANASONIC KX/1180	L.360.000
MANNESMANN MT81	L.270.000
STAR LC20	L.258.000
EPSON LX1050	L.690.000
EPSON FX850	L.690.000
EPSON FX1050	L.845.000
PANASONIC KX/P 1695	L.760.000
STAR LC200	L.410.000
STAR LC15	L.520.000

24 AGHI

STAR LC24200	L.495.000
STAR LC24200 COLORE	L. 590.000
PANASONIC KXP1124	L.550.000
PANASONIC KXP1624	L.860.00
NEC P20	L.540.000
NEC P30	L.750.000
EPSON LQ400	L. 498.000
EPSON LQ850	L.890.000

LASER

EPSON EPL 7100	L.1.900.000
NEC S60	L.2.100.000
NECS60 POSTSCRIPT	
PANASONIC KX/P4420	
PANASONIC KXP4450I	

MEMORIE DI MASSA

HD 40 SEAGATE	L.330.000
HD 80 SEAGATE	L.540.000
HD 120 SEAGATE	L.720.000
HD 211 SEAGATE	L.980.000
DRIVE 1,44 MEGA	L. 95.000
DRIVE 1,2 MEGA	L. 95.000

SCHEDE GRAFICHE

VGA 256K	L.85.000
VGA 512 K	L.148.000
VGA 1024K L.	L.218.000

COPROCESSORI

IIT 80C287	L.135.000
IIT 80C387/25	L.325.000
IIT 80C387/33	L.345.000
IIT 80387/20SX	L.210.000

COMPATIBILI IBM

AT 286

A partire da **L.450.000**

AT 386sx

A partire da **L.820.000**

At 386/25

A partire da **L.1.050.000**

AT 486/33

A partire da **L.2.500.000**

MONITOR

TRL 14" VGA MONO **L.169.000**

GOLDSTAR 14" VGA L.440.000

TRL 14" VGA 0.28PITH **L.540.000**

TRL 14" MULTISINK **L.650.000**

NEC

NEC 2A	L. 640.000
NEC 3D	L.840.000
NEC 4D L.	L.1.580.000
NEC 5D L.	L.2.800.000

La SER.COM. s.r.l. opera su tutto il territorio nazionale, spedizioni in 24 ore dall-ordine.

Le nostre condizioni di garanzia sono totali, 12 mesi dal momento dell-acquisto e rimborso se entro 10 giorni viene effettuata una valida contestazione sulla merce.

SERCOM s.r.l.
V.le Parioli 55/A ROMA
TELEFONI:
06 8587787 8587792