Programmare in C su Amiga (33)

di Dario de Judicibus (MC2120)

Seconda puntata dedicata ai pulsanti a rilascio incrociato. Vedremo come avviene la gestione di tali pulsanti, e come si intercetta l'evento emesso a fronte di una selezione da parte dell'utente. Risponderemo inoltre ai quiz proposti lo scorso mese, e daremo la soluzione all'esercizio relativo alla funzione ***isInGroup()***

Introduzione

Nella scorsa puntata abbiamo visto come si definisce e si rimuove un gruppo di pulsanti a rilascio incrociato. In questa puntata vedremo come si gestiscono gli eventi emessi da questo controllo composito, più un paio di altre funzioni atte a lavorare con questo tipo di pulsanti. Come al solito, sfrutteremo anche in queste funzioni la possibilità di estendere la struttura **Gadget** per mezzo del puntatore **UserData**, a cui associeremo come di consueto l'indirizzo ad una struttura **UBUT**. Risponderemo inoltre ai *quiz* presentati lo scorso mese, e ne presenteremo altri.

Pulsanti a rilascio incrociato

Come abbiamo visto nella scorsa puntata, la **CreateXButtons()** e la **DeleteXButtons()** ci permettono di definire e di rimuovere un gruppo di pulsanti a rilascio incrociato. Tuttavia, mentre la seconda funzione si preoccupa sia di

Figura 1
DisplayButtons(). ▼

```
/*****************************************************************************
** DisplayButtons()              FUNZIONE            Versione 1.00          **
**                                                                          **
** Funzione fornita:  aggiunge dei pulsanti al contenitore e quindi li      **
**                    visualizza                                            **
**                                                                          **
** Dati in ingresso:  button      puntatore ai pulsanti (uno o gruppo)      **
**                    container   contenitore (finestra e/o quadro)         **
**                                                                          **
** Dati globali:      User        ricava da Number il numero di pulsanti    **
**                                                                          **
*****************************************************************************/
void DisplayButtons(button,container)
  IGDG *button ;
  ICNT *container ;
{
  USHORT n ;

  /*
  **  Aggiungi il pulsante al contenitore
  */
  n = ((UBUT *)(button->UserData))->Number ;
  (void)AddGList(container->w,button,EOGL,n,container->r) ;
  RefreshGList(button,container->w,container->r,n) ;
}
```

Figura 2
H_GadgetDown() ►

```
/*****************************************************************************
** H_GadgetDown: gestisce l'evento GADGETDOWN                               **
*****************************************************************************/
 int H_GadgetDown(msg)
   IMSG *msg;
 {
   IGDG *gdg;
   USHORT gid;

   /*
   **  E' stata fatta effettivamente una selezione? Se sì, 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_TGL(g,st) printf("Controllo [%s] : %s\n",(g),(st)?"ON":"OFF")
   #define PRT_GRP(g,n) printf("Controllo [%s] in %d\n",(g),(n))

   /*
   **  BLOCCO PER LA GESTIONE DEI CODICI
   */
   switch (gid)
     {
       case ATGLBID:
            PRT_TGL((gdg->GadgetText)->IText,ToggleButtonStatus(gdg)) ;
            break;
       defaults:
            break;
     }
   if (gid >= AGRVBID && gid < (AGRVBID + AGRVNUM))
   {
     PRT_GRP((gdg->GadgetText)->IText,((UBUT *)(gdg->UserData))->Number) ;
     (void)SelectXButtons(gdg,&c);
   }
   if (gid >= AGRHBID && gid < (AGRHBID + AGRHNUM))
   {
     PRT_GRP((gdg->GadgetText)->IText,((UBUT *)(gdg->UserData))->Number) ;
     (void)SelectXButtons(gdg,&c);
   }

   return(GOAHEAD);
 }
```

```
/*************************************************************************
** isInGroup()              FUNZIONE              Versione 1.00          **
**                                                                       **
** Funzione fornita:  verifica se un certo pulsante appartiene ad un     **
**                    gruppo di pulsanti o meno                          **
**                                                                       **
** Dati in ingresso:  button      puntatore al pulsante da verificare    **
**                    id          identificativo base del gruppo         **
**                                                                       **
** Dati in uscita:    result      TRUE se il pulsante è nel gruppo,      **
**                                FALSE altrimenti                       **
**                                                                       **
** Dati globali:      User        utilizza Number per sapere quanti      **
**                                pulsanti ci sono nel gruppo            **
**                                                                       **
*************************************************************************/
BOOL isInGroup(button,id)
  IGDG *button ;
  USHORT id ;
{
  USHORT gid ;
  USHORT n ;

  gid = button->GadgetID ;                  /* Identificativo selezione */
  n  = ((UBUT *)(button->UserData))->Number ; /* Numero di pulsanti       */

  return (gid >= id && gid < (id + n)) ;    /* Nel gruppo o no ?        */
}
```

Figura 3 - isInGroup().

```
/*************************************************************************
** H_GadgetDown: gestisce l'evento GADGETDOWN                           **
*************************************************************************/
int H_GadgetDown(msg)
  IMSG *msg;
{
  :

  if (isInGroup(gdg,AGRVBID))
  {
    PRT_GRP((gdg->GadgetText)->IText,((UBUT *)(gdg->UserData))->Number) ;
    (void)SelectXButtons(gdg,&c);
  }
  if (isInGroup(gdg,AGRHBID))
  {
    PRT_GRP((gdg->GadgetText)->IText,((UBUT *)(gdg->UserData))->Number) ;
    (void)SelectXButtons(gdg,&c);
  }

  return(GOAHEAD);
}
```

Figura 4 - H_GadgetDown() che utilizza la isInGroup().

```
/*************************************************************************
** SelectXButtons()          FUNZIONE              Versione 1.00         **
**                                                                       **
** Funzione fornita:  seleziona il pulsante specificato, e deseleziona   **
**                    quello corrente, aggiornando la struttura di       **
**                    servizio di tutti i pulsanti nel gruppo.           **
**                                                                       **
** Dati in ingresso:  button      puntatore al pulsante da selezionare   **
**                                                                       **
** Dati in uscita:    current     puntatore al pulsante corrente         **
**                                                                       **
** Dati globali:      User        utilizza Number, First e Selected per  **
**                                l'aggiornamento della struttura User   **
**                                                                       **
*************************************************************************/
IGDG *SelectXButtons(button,container)
  IGDG *button ;
  ICNT *container ;
{
  IGDG *first, *current, *gdg ;
  USHORT i, n ;

  n       = ((UBUT *)(button->UserData))->Number   ;
  first   = ((UBUT *)(button->UserData))->First    ;
  current = ((UBUT *)(button->UserData))->Selected ;

 /*
 **  Rimuovi il gruppo di pulsanti dal contenitore
 */
  (void)RemoveGList(container->w,first,n);


 /*
 **  Aggiorna la selezione
 */
  current->Flags &= ~SELECTED ;
  button->Flags |= SELECTED ;
  for (i = 0; i < n; i++)
  {
    gdg = first + i ;
    ((UBUT *)(gdg->UserData))->Selected = button ;
  }


 /*
 **  Riaggiungi il gruppo di pulsanti al contenitore
 */
  (void)AddGList(container->w,first,EOGL,n,container->r) ;
  RefreshGList(first,container->w,container->r,n) ;
  RefreshWindow(container->w) ;

  return(current) ;
}
```

Figura 5
Vecchia
SelectXButtons().

rimuovere i pulsanti dalla lista dei controlli associati al contenitore, sia di ripristinare l'aspetto esterno dello stesso, la prima funzione si limita solo ad allocare ed opportunamente riempire le strutture necessarie alla definizione del gruppo, come già faceva analogamente per i pulsanti a rilascio automatico e quelli a rilascio manuale la **CreateButton()**.

L'aggiornamento della lista dei controlli e la visualizzazione del gruppo nel contenitore va effettuato con un'altra funzione. Nel passato abbiamo utilizzato la **DisplayGadget()**, la quale però, pur essendo sufficientemente generale da gestire vari tipi di controlli, non era in grado di gestire gruppi di pulsanti. Si è quindi reso necessario modificare tale funzione in modo da supportare anche i pulsanti a rilascio incrociato. La nuova funzione, chiamata **DisplayButtons()** (riportata in figura 1), vale ovviamente anche per i pulsanti a rilascio automatico e manuale.

Per quello che riguarda la gestione degli eventi emessi da Intuition a fronte di operazioni utente con il gruppo di pulsanti a rilascio incrociato, questa avviene nella stessa procedura **H_GadgetDown()** usata per i pulsanti a rilascio manuale, avendo usato il valore GADGETIMMEDIATE nella definizione delle modalità di attivazione del pulsante. Il codice è riportato in figura 2. Vedremo più avanti, nella sezione *Come si usano?*, il significato di alcuni degli identificativi riportati nel codice.

Da notare che in tale codice si va a calcolare direttamente se il pulsante che ha emesso l'evento fa parte di un gruppo di pulsanti a rilascio incrociato. In particolare, nell'esempio in questione

```
/*
**  Definizioni per i gruppi di pulsanti mutualmente esclusivi
*/
 #define AGROUPV  2
 #define AGRVBID  10
 #define AGRVCOL  100
 #define AGRVROW  10
 #define AGRVNUM  3
 char *AGrpVText[] =
 {
    "Colonna 1"
   ,"Colonna 2"
   ,"Colonna 3"
 };
 #define AGROUPH  3
 #define AGRHBID  20
 #define AGRHCOL  -10
 #define AGRHROW  -30
 #define AGRHNUM  3
 char *AGrpHText[] =
 {
    "Linea 1"
   ,"Linea 2"
   ,"Linea 3"
 };
```

▲
Figura 6 - Definizioni per i gruppi di pulsanti.

si verifica tale condizione per due gruppi di pulsanti, uno a distribuzione orizzontale (pulsanti in linea), ed uno a distribuzione verticale (pulsanti in colonna). Nella scorsa puntata vi avevamo proposto come esercizio quello di provare a scrivere una funzione che verifichi se un certo pulsante fa parte di un gruppo o meno. Avevamo chiamato tale funzione **isInGroup()**. Ebbene, in figura 3 è riportata una *possibile* soluzione, mentre in figura 4 si può vedere come va modificata la **H_GadgetDown()** per utilizzare tale funzione. Ho scelto di passare a tale funzione il puntatore al pulsante che ha emesso l'evento, piuttosto che già l'identificativo, in modo da poterla utilizzare anche in altre situazioni, ma ovviamente non sarebbe cambiato nulla se aveste deciso di scrivere la **sIn-Group()** in modo da ricevere come primo parametro **gid** piuttosto che **gdg**.

Come si può notare in figura 2, nel blocco di codice che stampa sul video l'indicazione relativa all'evento ricevuto, c'è anche la chiamata ad una nuova funzione: la **SelectXButtons()**. Vediamo perché.

SelectXButtons()

Abbiamo visto come si definisce un gruppo di pulsanti a rilascio incrociato e come si fa a rimuoverlo dalla lista dei controlli associati ad un certo contenitore. Ma chi gestisce tale gruppo quando l'utente finale lavora con la vostra interfaccia? Intuition, si potrebbe rispondere.

Purtroppo le cose non sono così semplici. È vero che Intuition gestisce i vari pulsanti ed emette i messaggi relativi ad i vari eventi, come specificato

```
/*************************************************************************
** Groups: crea o cancella i gruppi di pulsanti mutualmente esclusivi    **
*************************************************************************/
 void Groups (onoff)
   BOOL onoff;
 {
   if (onoff)
   {
     if (mask & MSK_GRP)
       ShowMsgReq(w,"Pulsanti già visibili");
     else
     {
      /*
      **  Crea i gruppi di pulsanti
      */
       buttons[AGROUPV] =
          CreateXButtons(AGRVBID,AGRVNUM,AGrpVText,&c,AGRVCOL,AGRVROW,
                         COLBUTTONGROUP,1);
       if (buttons[AGROUPV] == NULL) CloseAll(EMSG_NOMEMORY) ;

      /*
      **  Attenzione: basta un solo gruppo per attivare la maschera
      */
       mask |= MSK_GRP;

       buttons[AGROUPH] =
          CreateXButtons(AGRHBID,AGRHNUM,AGrpHText,&c,AGRHCOL,AGRHROW,
                         ROWBUTTONGROUP,2);
       if (buttons[AGROUPH] == NULL) CloseAll(EMSG_NOMEMORY) ;

      /*
      **  Visualizza i pulsanti a rilascio automatico
      */
       DisplayButtons(buttons[AGROUPV],&c);
       DisplayButtons(buttons[AGROUPH],&c);
     }
   }
   else
   {
     if (!(mask & MSK_GRP))
       ShowMsgReq(w,"Pulsanti già cancellati");
     else
     {
       if (buttons[AGROUPV]) DeleteXButtons(buttons[AGROUPV],&c);
       if (buttons[AGROUPH]) DeleteXButtons(buttons[AGROUPH],&c);

       mask &= ~MSK_GRP;

     }
   }
 }
```

Figura 7 - Groups(). ►

nelle singole strutture associate ad i controlli, ma il fatto è che per lui il gruppo non esiste come tale. In realtà Intuition vede i singoli pulsanti come controlli a sé stanti, non come un unico controllo composito. Ed infatti, non dimentichiamocelo, tale controllo non esiste nell'Amiga. È una nostra invenzione. Ed allora? Semplice. Così come Intuition ci ha dato la possibilità di costruire un nuovo controllo con un suo comportamento ben preciso, così ci dà la possibilità di scrivere la procedura di gestione dello stesso. In pratica, a fronte di un evento **GADGETDOWN**, se risulta che il pulsante premuto fa parte di un gruppo, utilizzando a tal scopo la funzione **isInGadget()** che abbiamo appena visto, è necessario chiamare una funzione chiamata **SelectXButtons()** che si occupa appunto di rilasciare il pulsante precedentemente selezionato, selezionare quello premuto dall'utente, aggiornare le strutture relative al gruppo, e restaurare l'aspetto del gruppo in modo da rappresentare la nuova situazione.

La **SelectXButtons()** (vedi figura 5) ha solo due parametri in ingresso:

button
il puntatore alla struttura **Gadget** che identifica il pulsante selezionato dall'utente;

container
il puntatore alla solita struttura *contenitore*.

Per prima cosa la funzione ricava dalla struttura di servizio puntata da **UserData** tre informazioni necessarie per modificare lo stato complessivo di selezione del gruppo:

n
il numero di pulsanti facenti parte del gruppo;

first
il puntatore al primo pulsante nel gruppo, e quindi al gruppo stesso, visto

come un vettore di strutture **Gadget**;

current
il puntatore al pulsante che fino a questo momento si trovava nello stato selezionato, e che quindi va rilasciato.
QUIZ#1: esiste una verifica che può aumentare l'efficienza di questa funzione. Quale?

Una volta ottenuti tali valori, la funzione rimuove il gruppo dalla lista dei controlli, ridefinisce lo stato del nuovo e del vecchio pulsante selezionato, ed aggiorna la struttura di servizio. Infine riaggiorna la lista dei controlli ed effettua i necessari restauri al contenitore.

A questo punto la funzione **SelectXButtons()** restituisce al chiamante il puntatore al pulsante che era precedentemente selezionato, per eventuali ulteriori elaborazioni.

Come si usano?

Abbiamo ora il quadro completo delle funzioni e delle macro relative ad i pulsanti a rilascio incrociato. Vediamo adesso come si possono utilizzare queste funzioni per aggiungere ad una finestra due gruppi di pulsanti, uno a distribuzione orizzontale, ed uno a distribuzione verticale.

Innanzi tutto è necessario definire i parametri relativi ai due gruppi, come riportato in figura 6.

Inoltre definiremo una funzione **Groups()** che, analogamente alle funzioni **Buttons()** e **Toggles()** già viste nelle scorse puntate, serve a creare e rimuovere i due gruppi di pulsanti, a fronte delle richieste fatte dall'utente via menu (vedi figura 7).

Non riportiamo tutto il codice di gestione dei menu e degli eventi messi da questi in quanto segue la falsariga di quello già presentato in passato. Ovviamente avremo bisogno di una nuova costante di maschera che, per la cronaca, ho definito con

#define MSK_GRP 0×0080

Esercizio (di cui non riporterò però la soluzione nelle prossime puntate, dato l'enorme numero di possibili soluzioni proponibili).

Ora che avete ben tre classi diverse di pulsanti da gestire nella **gdgmain.c**, ma anche molte ridondanze dovute al riutilizzo sempre degli stessi scheletri (basti pensare alle tre funzioni **Buttons()**, **Toggles()** e **Groups()**), provate a scrivere delle funzioni di carattere generale che migliorino la leggibilità e le prestazioni del codice del programma principale, e che eliminino del tutto tali ridondanze.

Le risposte ad i quiz

Ecco, come promesso, le risposte ad i cinque *quiz* presentati nella 32ª puntata.

QUIZ #1: nel codice della **CreateXButtons()** *mancano alcuni controlli che vanno assolutamente effettuati per prevenire un utilizzo improprio della funzione, e garantire così che esso venga*

Vecchio Codice

```
/*****************************************************************************
** CreateXButtons                FUNZIONE              Versione 1.00      **
**                                                                        **
** Dati globali:      User        memorizza in Size la memoria utilizzata **
```

```
  :

  n1 = n - 1;
```

Nuovo Codice

```
/*****************************************************************************
** CreateXButtons                FUNZIONE              Versione 1.10      **
**                                                                        **
** Dati globali:      User        memorizza in Size la memoria utilizzata **
**                                in Number il numero di pulsanti nel     **
**                                gruppo, in First il puntatore al gruppo **
**                                ed in Selected il pulsante inizialmente **
**                                selezionato                             **
```

```
  :

 /*
 **  Safe filter
 */
  n1 = n - 1;
  if (n < 2 || buton < 0 || buton > n1) return(NULL) ;
```

Una soluzione più completa è la seguente:

```
 /*
 **  Verifica la correttezza dei parametri d'ingresso
 */
  if (   (id == 0)
      || (txt == NULL)
      || (container == NULL)
      || !(class & (ROWBUTTONGROUP|COLBUTTONGROUP))
      || (n < 2)                                    /* Fondamentale */
      || !(buton >= 0 && buton < n)                 /* Fondamentale */
      )
      return (NULL) ;
```

◀ *Figura 8 - Risposta al quiz #1.*

Vecchio Codice

```
UsrSize = sizeof(UBUT)*n ; /* Una struttura UsrButton per pulsante  */

:

for (i = 0, l = 0, h = 0; i < n; i++)
{

:

  Gdg[i].UserData     = (APTR)&User[i] ;       /* Struttura di servizio */
  User[i].Size        = TotSize ;  /* Qui metto quanta memoria ho preso */
  User[i].Number      = n ;        /* Numero di pulsanti nel gruppo     */
  User[i].First       = &Gdg[0] ;    /* Primo pulsante del gruppo       */
  User[i].Selected    = &Gdg[buton] ;/* Pulsante selezionato nel gruppo */

:

}
```

Nuovo Codice

```
UsrSize = sizeof(UBUT) ;   /* Una sola struttura UsrButton per tutti */

:

for (i = 0, l = 0, h = 0; i < n; i++)
{

:

  Gdg[i].UserData     = (APTR)User ; /* Struttura di servizio (una sola)*/

:

}
User->Size        = TotSize ;       /* Qui metto quanta memoria ho preso */
User->Number      = n ;             /* Numero di pulsanti nel gruppo     */
User->First       = (APTR)&Gdg[0] ;     /* Primo pulsante del gruppo     */
User->Selected    = (APTR)&Gdg[buton] ; /* Pulsante selezionato          */
```

▲ *Figura 9 - Risposta al quiz #2.*

```
/*************************************************************************
** SelectXButtons()          FUNZIONE              Versione 1.00         **
**                                                                       **
** Funzione fornita:  seleziona il pulsante specificato, e deseleziona   **
**                    quello corrente, aggiornando la struttura di       **
**                    servizio di tutti i pulsanti nel gruppo.           **
**                                                                       **
** Dati in ingresso:  button      puntatore al pulsante da selezionare   **
**                    container   contenitore (finestra e/o quadro)      **
**                                                                       **
** Dati in uscita:    current     puntatore al pulsante corrente         **
**                                                                       **
** Dati globali:      User        utilizza Number, First e Selected per  **
**                                l'aggiornamento della struttura User   **
**                                                                       **
*************************************************************************/
IGDG *SelectXButtons(button,container)
  IGDG *button ;
  ICNT *container ;
{
  IGDG *first, *current ;
  USHORT n ;

  n       = ((UBUT *)(button->UserData))->Number   ;
  first   = (IGDG *)((UBUT *)(button->UserData))->First    ;
  current = (IGDG *)((UBUT *)(button->UserData))->Selected ;

 /*
 **  Rimuovi il gruppo di pulsanti dal contenitore
 */
  (void)RemoveGList(container->w,first,n);


 /*
 **  Aggiorna la selezione
 */
  current->Flags &= ~SELECTED ;
  button->Flags |= SELECTED ;
  ((UBUT *)(first->UserData))->Selected = (APTR)button ;


 /*
 **  Riaggiungi il gruppo di pulsanti al contenitore
 */
  (void)AddGList(container->w,first,EOGL,n,container->r) ;
  RefreshGList(first,container->w,container->r,n) ;
  RefreshWindow(container->w) ;

  return(current) ;
}
```

◀ *Figura 10 Nuova SelectXButtons().*

Vecchio Codice

```
for (i = 0, l = 0, h = 0; i < n; i++)
{
:

  h = max(h,container->w->RPort->TxHeight) ; /* Altezza massima finora  */
}
```

Nuovo Codice

```
for (i = 0, l = 0, h = 0; i < n; i++)
{
:

}
h = container->w->RPort->TxHeight ;  /* Altezza del testo               */
```

▲ *Figura 11 - Risposta al quiz #3.*

eseguito correttamente. Quali sono queste verifiche?

A parte le verifiche fondamentali che richiedono che parametri come l'identificativo del gruppo, il puntatore al vettore dei testi per i pulsanti, quello al contenitore, e la configurazione del gruppo non assumano valori nulli, è opportuno verificare che il numero di pulsanti nel gruppo sia *strettamente maggiore* di uno, e che il pulsante da selezionare inizialmente abbia un indice

Vecchio Codice

```
for (i = 0, l = 0, h = 0; i < n; i++)
{
:

 /*
 **  Associa al pulsante il suo testo
 */
  Txt[i] = textModel ;              /* Copia il prototipo del controllo */
  Txt[i].IText = (UBYTE *)txt[i] ;  /* Copia il testo vero e proprio    */
  l = max(l,ITXTL(&Txt[i])) ;               /* Lunghezza massima finora */
  h = max(h,container->w->RPort->TxHeight) ; /* Altezza massima finora  */
}
Gdg[n1].NextGadget = NULL ;          /* Ultimo pulsante del gruppo       */
:

/*
**  Alloca i campi variabili della struttura Gadget
*/
for (i = 0; i < n; i++)
{
  Txt[i].LeftEdge = l/8 ;           /* Ascissa del testo nel controllo  */
  Txt[i].TopEdge  = h/3 ;           /* Ordinata del testo nel controllo */
  Gdg[i].Width    = l + 2*Txt[i].LeftEdge ;
  Gdg[i].Height   = h + 2*Txt[i].TopEdge ;
}
```

Nuovo Codice

```
for (i = 0, l = 0, h = 0; i < n; i++)
{
:

 /*
 **  Associa al pulsante il suo testo
 */
  Txt[i] = textModel ;              /* Copia il prototipo del controllo */
  Txt[i].IText = (UBYTE *)txt[i] ;  /* Copia il testo vero e proprio    */
  Txt[i].LeftEdge = ITXTL(&Txt[i]) ;/* Memorizza per ora la lunghezza   */
  l = max(l,Txt[i].LeftEdge) ;      /* Lunghezza massima finora         */
}
h = container->w->RPort->TxHeight ;/* Altezza del testo                */
Gdg[n1].NextGadget = NULL ;         /* Ultimo pulsante del gruppo       */
:

/*
**  Alloca i campi variabili della struttura Gadget
*/
for (i = 0; i < n; i++)
{
  Txt[i].LeftEdge = l/8 +           /* Ascissa del testo nel controllo  */
                    (l - Txt[i].LeftEdge)/2 ; /* Centra il testo        */
  Txt[i].TopEdge  = h/3 ;           /* Ordinata del testo nel controllo */
 ,Gdg[i].Width    = 5*l/4 ;
  Gdg[i].Height   = 5*h/3 ;
}
```

Figura 12 - Risposta al quiz #4.

compreso fra zero ed **n,** quest'ultimo non compreso, come riportato in figura 8.

QUIZ #2: esiste un modo per risparmiare memoria ed ottimizzare così l'insieme delle strutture che definiscono il gruppo?

Sì. Usare una sola struttura **UsrButton** invece di creare una struttura **UsrButton** per ogni pulsante del gruppo, e far puntare tutti i campi **UserData** del vettore di strutture **Gadget** che definisce il gruppo di pulsanti, a quest'unica struttura di servizio. Questa, infatti, contiene informazioni relative all'intero gruppo, e non ha quindi senso duplicarla tante volte quanti sono i pulsanti nel gruppo.

In figura 9 sono riportate le modifiche da effettuare alla funzione **CreateXButtons()** se si intende applicare questa tecnica. Anche la **SelectXButtons()** appena vista, è ovviamente impattata dall'aver utilizzato una sola struttura **UsrButton**, piuttosto che un vettore di tali strutture. In figura 10 è riportata la nuova funzione. Analizzando con attenzione le differenze tra questa, e la funzione riportata in figura 5, vi accorgerete di altre modifiche, non legate all'ottimizzazione effettuata con la tecnica in esame. Vedremo nella prossima puntata il perché di tali cambiamenti, che interessano la stessa struttura **UsrButton**.

QUIZ #3: nel codice contenuto in questo ciclo, c'è un'istruzione che potrebbe essere tranquillamente portata fuori ciclo. Quale?

Quella che calcola l'altezza del testo sulla base dell'altezza del *font* utilizzato nel *raster* del contenitore, come riportato in figura 11.

QUIZ #4: nel codice contenuto in questo ciclo, c'è un errore che altera il risultato che si vorrebbe ottenere. Qual è?

Nel blocco di codice che associa al pulsante il suo testo, viene calcolata la lunghezza massima fra quelle di tutti i testi relativi ai pulsanti del gruppo. Tale lunghezza viene poi utilizzata per definire, sia la larghezza di tutti i pulsanti, sia *la posizione del testo in ogni pulsante.* E qui sta l'errore.

Infatti, in questo modo, il testo di ogni pulsante viene ad essere allineato a sinistra alla stessa distanza dal bordo sinistro del pulsante, per *tutti* i pulsanti del gruppo. Dato però che ogni pulsante ha la stessa larghezza, ma i vari testi hanno lunghezze diverse, questi non risultano centrati nel rettangolo che rappresenta il pulsante, con un effetto estetico discutibile.

Per ovviare a ciò (vedi figura 12), si è utilizzata la seguente tecnica.

All'interno del ciclo principale, viene calcolata la lunghezza di ogni singolo testo, e memorizzata temporaneamente in **Txt[i].LeftEdge**, per non sprecare inutilmente memoria definendo un altro vettore di interi. Ovviamente si continua a calcolare anche la lunghezza massima dei vari testi.

Quindi, nel secondo ciclo, si utilizza tali valori (lunghezza massima e lunghezza del singolo testo) per centrare il testo nel pulsante. Ora l'effetto risultante è decisamente migliore.

QUIZ #5: notate come l'istruzione che seleziona il pulsante indicato dal valore di **button**, *si trovi dopo i due blocchi di codice che posizionano i vari pulsanti, e non prima. Perché?*

Perché in tali blocchi il campo **Flags** del primo o dell'ultimo pulsante, viene usato come modello per quello degli

CASELLA POSTALE

Stringhe e puntatori

Egregio Dott. de Judicibus,
seguo con vivo interesse la Sua rubrica che reputo interessante e completa. Sono interessato da qualche tempo, alla realizzazione di un input controllato in linguaggio C utilizzando solo ed esclusivamente i messaggi inviati da Intuition.

Dopo aver realizzato un primo prototipo, ho riscontrato uno strano bug (a mio parere) che si verifica quando è necessario utilizzare la routine ripetutamente per riempire ad esempio un Array. Le propongo il seguente programma che reagisce esattamente come la routine da me creata.

```
char *MyInput()
{
    char *stringa;
    gets (stringa);
    return(stringa);
}

main()
{
    char *dati[11];
    char *input;
    int i;
    for (i=0;i<10;i++)
    {
        input = MyInput ();
        strcpy (dati[i],input);
    }
    for (i=0;i<10;i++)
        printf("%s\n",dati[i]);
}
```

Dove si annida l'errore? Ringraziandola per la Sua disposizione le posticipo che utilizzo il compilatore Manx v3.6a e i sistemi operativi 1.2 ed 1.3. Cordiali Saluti.

Bta Amiga Club 2000

Di errori, od imprecisioni, ce n'è più di una. La maggior parte si può comunque ricondurre ad un errore «classico» per chi programma in C, e cioè la differenza che esiste tra il definire un vettore di caratteri, ed il dichiarare un puntatore ad una stringa di caratteri.

Consideriamo le seguenti definizioni:

```
char *buffer[20] ;          /* 1 */
char *ptrstr ;              /* 2 */
char *ptrbuf = buffer ;     /* 3 */
```

Nella prima dichiarativa, abbiamo definito sia un vettore di caratteri, che un puntatore a tale vettore. In pratica, da un punto di vista fisico, abbiamo riservato un'area ampia 20 byte, riservata agli elementi di **buffer** indicizzati da **0** a **19** (20 elementi), più un'area ampia due o quattro byte, a seconda delle specifiche di compilazione e/o del sistema operativo, che viene automaticamente inizializzata con l'indirizzo del primo elemento del vettore. Le due aree sono strettamente legate, non è cioè possibile modificare il valore del puntatore, perché questo porterebbe ad invalidare l'equivalenza:

pointer = &(*pointer)

Nella seconda dichiarativa abbiamo definito un puntatore ad una stringa non inizializzato (od inizializzato a zero, a seconda dei compilatori). A tale puntatore, tuttavia, *non è associata nessuna area di memoria in grado di ricevere una stringa di caratteri* al seguito, ad esempio, di una operazione di lettura.

La terza dichiarativa definisce anch'essa un puntatore ad un carattere (o strin-

altri pulsanti. Questo vuol dire che, se il pulsante che deve essere inizialmente selezionato è proprio il primo o l'ultimo, tutti gli altri pulsanti risultano selezionati. Viceversa, se i pulsanti «modello» non sono inizialmente selezionati, il codice contenuto nei due cicli di posizionamento finirebbe per deselezionare anche il pulsante indicato dal valore di **button**.

Conclusione

Bene. Anche questa volta siamo arrivati al capolinea. Lo scopo principale di questa puntata e della precedente era di dimostrare come Intuition ci permetta di fare molto più di quello che è riportato nei vari manuali dell'Amiga. Questi devono infatti essere pensati come una base di partenza, da cui il programmatore deve prendere spunto per sviluppare via via cose sempre nuove. Provate a sviluppare nuovi tipi di controlli, magari di tipo composito. Per ogni controllo definite tre funzioni: una di creazione, una di rimozione, ed una di gestione degli eventi da esso generati, con al più qualche funzione secondaria e qualche macro. Tanti auguri e, ovviamente, buon divertimento!

MC

---

ga), ma in più inizializza tale puntatore con il valore dell'indirizzo del primo elemento del vettore definito sopra, essendo

buffer = &buffer[0]

È importante quindi capire cosa stiamo facendo quando dichiariamo qualcosa nel nostro programma, in termini di prenotazione di spazi di memoria e loro interrelazioni. Per questo il C non è un semplice linguaggio di alto livello, come ad esempio il Pascal, ma può essere considerato piuttosto una specie di super-Assembler.

Nel caso in questione, ci sono svariati errori di questo tipo. Innanzi tutto la funzione **MyInput()**, dove è stato definito un puntatore ad una stringa, **stringa** appunto, che doveva, nelle intenzioni dell'autore, essere utilizzato per caricarvi la stringa di caratteri letta dalla **gets()**. Il problema è che in questo modo non si è definito fisicamente lo spazio per caricare tale stringa, ma solo un puntatore, per giunta non inizializzato. Affinché la funzione funzioni correttamente, è necessario passare alla **gets()** un puntatore ad un'area di memoria sufficientemente larga per caricarvi la stringa in questione. Per far ciò ci sono tre alternative, riportate nella figura pubblicata in questa pagina. Nella prima si definisce il vettore direttamente nella funzione di lettura. Nella seconda tale spazio è allocato dinamicamente, cosa che diminuisce le prestazioni della funzione dato il tempo richiesto per l'allocazione, ma che può rendersi necessaria in certi tipi di programmi per i quali è richiesta la rientranza (vedi la penultima puntata). La terza possibilità consiste nel definire l'area di ingresso dei dati esternamente alla **MyInput()**, e passare quindi a quest'ultima il puntatore a tale area come parametro. Ovviamente nel nostro caso le tre soluzioni sono più o meno equivalenti, mentre una delle tre si può rivelare più efficiente in situazioni e programmi più complessi ed articolati.

```
/*************************************************************************
**  Prima possibilità: vettore definito in MyInput()                    **
**  ------------------------------------------------------------------- **
**  NOTA: al posto di 256, sarebbe consigliabile utilizzare la          **
**        costante BUFSIZ definita in stdio.h od il valore esterno      **
**        _bufsize nel caso si usi il Lattice 5.xx                      **
*************************************************************************/
char *MyInput()
{
    char stringa[256] ;
    return (gets(stringa)) ;
}

/*************************************************************************
**  Seconda possibilità: allocazione dinamica                           **
**  ------------------------------------------------------------------- **
**  NOTA: al posto di 256, sarebbe consigliabile utilizzare la          **
**        costante BUFSIZ definita in stdio.h od il valore esterno      **
**        _bufsize nel caso si usi il Lattice 5.xx                      **
*************************************************************************/
char *MyInput()
{
    char *stringa ;
    stringa = (char *)calloc(1, 256) ;
    return (gets(stringa)) ;
}

/*************************************************************************
**  Terza possibilità: vettore esterno                                  **
*************************************************************************/
char *MyInput(stringa)
char *stringa ;
{
    return (gets(stringa)) ;
}
```

MyInput().

Lo stesso errore fatto nella **MyInput()**, è presente anche nella funzione principale, in quanto sia **input** che i vari **dati[i]** altro non sono che semplici puntatori.

Ora, mentre per **input** non c'è alcun problema, dato che serve solo come variabile temporanea (a condizione che **stringa** punti ad un'area dichiarata in memoria), i primi dieci puntatori del vettore **dati** devono ritenere il contenuto immessovi ai fini dell'elaborazione successiva, e cioè il ciclo sulla **printf()**, per cui è necessario definire anche qui dieci aree dati in grado di ricevere le stringhe lette.

Un'altra piccola imprecisione riguarda la definizione di **dati**. Quando si definisce un vettore, si specifica il *numero degli elementi* in quel vettore, elementi che vengono numerati in C a partire da zero.

Scrivere

char *dati[11];

vuol dire definire un vettore di *undici* puntatori a stringhe di caratteri, da **dati[0]** a **dati[10]**. Il ciclo, tuttavia, riempie solo i primi dieci, dato che esso continua fintanto che l'indice *è minore* di dieci. L'undicesimo elemento, cioè **dati[10]**, non è mai interessato dai due cicli del programma **main()**.

Dato che quello riportato nella lettera non è il programma che presenta il problema ma, a quanto ho capito, un programmino equivalente, non posso essere sicuro che tale problema sia dovuto proprio a questi errori. Certo è, comunque, che se errori del genere sono presenti anche nel programma vero, non mi stupisce che il tutto non funzioni correttamente. Spero di esserLe stato di aiuto.

MC

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

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 Comunicare tra 2 PC	mc104	
COM/02	PROCOMM Noto programma di comunicazione		Hard disk
COM/03	OMEGA LINK Comunicazione a finestre	mc106	
COM/04	BACKCOMM Programma di comunicazione TSR	mc103	

DATABASE

CODICE	TITOLO&DESCRIZIONE	REC.	HARDWARE
DBS/01	EASY LABELS Per creare etichette		
DBS/02	VIDEO DATABASE Gestione video cassette	mc105	Hard disk
DBS/03	HOME MANAGER DataBase, calcolatrice e calendario		Hard disk
DBS/04	MAIL-MONSTER Gestire Mailing list	mc103	
DBS/05	MAKE MY DAY Per organizzare il lavoro		
DBS/06	PC-FILE+ Data Base	mc106	
DBS/07	TASK MASTER Projet Plaining		
DBS/08	RELIANCE MAILING LIST Mailing per associazioni culturali		
DBS/09	DMS Utility per memorizzare file di diverso genere	mc107	

EDUCATIVO

CODICE	TITOLO&DESCRIZIONE	REC.	HARDWARE
EDU/01	ABC FUN KEYS Per imparare le lettere dell'alfabeto	mc103	
EDU/02	COMPUTER TUTOR Auto-apprendimento del computer		
EDU/03	PC-FASTYPE Imparare professionalmente ad usare la tastiera		CGA

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'SADVENTURE Clone di Donkey Kong	mc103	EGA/VGA
GIO/05	CAESAR Strategia		BASIC+EGA/VGA
GIO/07	CLONE INVADERS Clone di Space Invaders		
GIO/08	EGAINT Tetris	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 Siete in miniera	mc104	VGA
GIO/21	MOSAIX Puzzle		VGA
GIO/22	OTHELLO EGA Othello	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		
GIO/35	MR.SPOCK Filetto 3D	mc105	EGA/VGA
GIO/36	MONUMENTS OF MARS Siete su Marte	mc106	
GIO/37	PHARAOH'S TOMB Esplorate la grande piramide	mc106	
GIO/38	POKER Gioco del poker da bar	mc107	EGA/VGA

GRAFICA

CODICE	TITOLO&DESCRIZIONE	REC.	HARDWARE
GRF/01	FINGER PAINT Programma di disegno		
GRF/02	PC-KEY-DRAW Per fare slide show	mc107	CGA
GRF/03	H&P CALENDAR Calendario grafico	mc103	
GRF/04	PC-DEMO SYSTEM Progenitore di Presentation Manager	mc105	
GRF/05	GRAPHICWORKSHOP Convertitore di formati grafici	mc106	

SPREADSHEET

CODICE	TITOLO&DESCRIZIONE	REC.	HARDWARE
SPD/01	AS-EASY-AS Spreadsheet compatibile 1-2-3	mc103	
SPD/02	EXPRESS-CALC Richiede solo 256 Kbyte di RAM	mc104	
SPD/03	EZ-SPREADSHEET Calcoli di budget		
SPD/04	INSTACALC Spreadsheet TSR	mc107	
SPD/05	QUEBECALC Spreadsheet 3D		

UTILITY

CODICE	TITOLO&DESCRIZIONE	REC.	HARDWARE
UTI/01	PC-DESK-TEAM Utility TSR	mc107	
UTI/02	HARD DISK UTILITIES Per gestire l'Hard Disk		Hard Disk
UTI/03	DOS HELP Help dei comandi DOS	mc104	
UTI/04	DISK SPOOL II Spooler di stampa	mc103	
UTI/05	LOCKTITE Protegge i file con password		
UTI/06	VIRUS SCAN Cerca i virus		
UTI/07	LHARC Compressore di dati	mc105	
UTI/08	ARJ Compressore di dati	mc105	
UTI/09	LZEXE Compressore di eseguibili	mc105	
UTI/10	DIET Compressore di eseguibili	mc105	
UTI/11	PKLITE Compressore di eseguibili	mc105	
UTI/12	NEWSPACE Compressore di file dell'Hard Disk	mc105	
UTI/13	CATDISK Catalogatore di file	mc105	
UTI/14	POINT&SHOOT Programma di Back Up	mc105	
UTI/15	SHEZ Shell per compressori di dati	mc106	
UTI/16	ZZAP Shell per compressori di dati	mc106	
UTI/17	GUARDIAN ANGEL Per programmare con i TSR	mc107	
UTI/18	STORE Per rinominare file in automatico	mc107	
UTI/19	TXT Per gestire i file TXT	mc107	

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 Per suonare al computer	mc104	

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
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 Buon numero di ricette in inglese	mc104
VAR/08	PERSONAL C COMPILER Semplice compilatore C	mc105
VAR/09	MOUSE.TPU & NEWEXEC Unit in TP per Mouse & Company	mc106
VAR/10	TRS, PRINT & GESTECC Unit in TP per stampa e gestione degli errori	mc106
VAR/11	ARIANNA Programma di CAE	mc106

WORDPROCESSOR

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

AMIGA

GIOCO

AMGI/02	WELLTRIX Clone di Tetris	mc105
AMGI/03	SYS Sulla falsa riga di Pac Man	mc105

GRAFICA

AMGR/01	PRINTSTUDIO Gestisce la stampa di testi	mc104
AMGR/02	TEXTPAINT Editor ANSI	mc105
AMGR/03	SCREENX Per chi lavora su più finestre	mc105
AMGR/04	SETPAL Cambia lo schermo da NTSC a PAL	mc105

SPREADSHEET

AMSP/01	SPREAD	mc104

UTILITY

AMUT/01	MACH III Tool per mouse	mc104
AMUT/02	RULER Visualizza la finestra di workbench	mc104
AMUT/03	HEX File editor in esadecimale	mc104
AMUT/04	MOM Aggiunge menu al workbench	mc104
AMUT/05	CB Log delle funzioni di I/O	mc104
AMUT/06	ZETAVIRUS Anti Virus	mc104
AMUT/07	DIRMASTER Utility per gestire i file	mc105
AMUT/08	KDC Per muoversi nelle directory	mc105
AMUT/09	XCOPYIII Copiatore veloce	mc105
AMUT/10	CD2TAPE Se usate oltre al PC il Cd Audio	mc105
AMUT/11	BBS & Log Gestione Packet Radio	mc106
AMUT/12	UTILITIES Comandi CLI Dos-Like	mc106
AMUT/13	VIEW80 II Visualizzatore di testi	mc106
AMUT/14	MATCALC Matematica matriciale	mc106
AMUT/15	ICONMASTER Editor di icone	mc106
AMUT/16	HERMIT Cattura lo schermo	mc106
AMUT/17	TURBO IMPLODER Compressore di file eseguibili	mc106
AMUT/18	FONTS2PRINTER Font per stampanti	mc107
AMUT/19	SVD Anti Virus	mc107
AMUT/20	MC-PROGRAMS Collezione di utility	mc107
AMUT/21	CHP&SAVE-PREFS Preference del computer	mc107

MACINTOSH

EDUCATIVO

MIED/01	KID PIX Per disegnare	mc107
MIED/02	NUMBER TALK Per imparare a contare	mc107
MIED/03	ALPHA TALK Per impare lo spelling in inglese	mc107

GIOCO

MIGI/01	STELLA OBSCURA Gioco spaziale in 3D	mc106
MIGI/02	PARARENA RollerBoard sullo Skeatboard	mc106
MIGI/03	VIDEO POKER FOR FUN Siete a Las Vegas	mc106
MIGI/04	SPACE STATION PHETA Su e giù per le scale	mc106
MIGI/05	STRATEGO Strategia con le carte	mc106
MIGI/06	THE LAWNZAPPER Falciate il prato	mc107
MIGI/07	MACTRIS Tetris sul Mac	mc107
MIGI/06	CANFIELD Solitario con le carte	mc107

GRAFICA

MIGR/01	CALENDAR MAKER Calendari personalizzati	mc106

UTILITY

MIUT/01	OLIVER'S BUTTONS INIT che sostituisce i bottoni di sistema	mc107
MIUT/02	POPCHAR Font	mc107

Compilare e spedire a

MCmicrocomputer - Via Carlo Perrier 9, 00157 Roma

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"

1)Codice: 5)Codice: 9)Codice:

2)Codice: 6)Codice: 10)Codice:

3)Codice: 7)Codice: 11)Codice:

4)Codice: 8)Codice: 12)Codice:

Nome e Cognome

Indirizzo

CAP/Città

Telefono

EXpress
Office Automation

PROMOZIONE VALIDA FINO AL 31.05.1991
PREZZI RISERVATI ESCLUSIVAMENTE AD
OPERATORI DEL SETTORE
E AD AZIENDE

I CASH & CARRY DELL'INFORMATICA

ROMA
Via Casale Agostinelli, 140
(Loc. Morena)

MILANO
Via Mecenate 76/4
Tel. 02 / 58010800

BOLOGNA
Via Dello Stallo 2/5/A
(Zona Massarenti)

TORINO
Via Ventimilia 16/3
Tel. 011 / 697353-6960502
(Prossima apertura via Reinier, 29

PARMA
Via Buffolara 68
Tel. 0521 / 96412
(Da Settembre: Via Colorno
di fronte al Centro Torri)

REGGIO EMILIA
SEDE AMMINISTRATIVA
MAGAZZINO
PUNTO DI DISTRIBUZIONE
Via F. Cavallotti, 22
tel. 0522/512751 fax 0522/513129

PROSSIME APERTURE:
MODENA - FERRARA -
FIRENZE - PIACENZA -
MANTOVA - PERUGIA

VISITATE IL CASH & CARRY
DELLA VOSTRA CITTA'
PIU' DI 500 ARTICOLI A
VOSTRA DISPOSIZIONE

SCHEDE PER COMPUTER

P 120	SCHEDA VGA 256K 2 LAYER	L. 74.976
P 121	SCHEDA VGA 256K 4 LAYER	L. 86.592
P 130	SCHEDA VGA 512K 4 LAYER	L. 139.920
M 203	SCHEDA MADRE 286 12 MHZ	L. 115.104
M 221	SCHEDA MADRE 286 16 MHZ	L. 167.904
M 301	SCHEDA MADRE 386 25MHZ	L. 288.288
P 022	SCHEDA MULTII/O IDE+CAVI	L. 44.352
P 020	SCHEDA CONTROLLER ATBUS	L. 23.232

CASSE E TASTIERE

L 004	C. FLIP TOP AT + ALIM.	L. 100.848
L 007	C. C91-01 CON DISPLAY	L. 127.776
L 003	C. SLIDE AT + ALIM.	L. 160.512
L 002	C. ELEGANT AT+ ALIM. + DISP.	L. 227.568
L 021	C. SLIM SLIDE AT + LED + ALIM.	L. 199.056
L 030	MINITOWER CON ALIMENTATORE	L. 149.952
L 040	TOWER MEDIO + ALIM. E DISP.	L. 199.056
L 050	TOWER BIG + ALIM. E DISP.	L. 253.440
L 302	TASTIERA 102 TASTI	L. 45.091
L 303	TASTIERA MINI 84 TASTI	L. 74.976
L 300	TASTIERA MICROSW. 102 T.	L. 60.192

CONFIGURAZIONI "EXPRESS"

ASSEMBLATE E TESTATE - COMPLETE DI TASTIERA ITALIANA

R 550-B	PC EXPRESS 286-12 DESK + DUAL +FDD1.2	L. 529.584
R 550B2	PC EXPRESS 286-12 DESK + DUAL + FDD1.4	L. 529.584
R 552	PC EXPRESS 286-12 TOW. + VGA + FDD1.4	L. 529.584
R 555	PC EXPRESS 286-16 TOW. + VGA + FDD1.4	L. 593.472
R 558	PC EXPRESS 286-20 TOW. + VGA + FDD1.2	L. 620.400
R 561	PC EXPRESS 386-25 TOWER + VGA	L.1.192.752
R 57O	M/F-LAN 01 SPOT LAN STATION 286-16 + 1MB RAM + CASSA SLIM + FDD1.44	L. 616.176

ACCESSORI

T 004	MOUSE PER PC + SOFTWARE	L. 18.057
T 090	CONNETTORE 9/25 PIN M/F	L. 3.326
A 340	TAPPETINO PER MOUSE	L. 3.743
T 092	MULTIPRESA ITALIANA	L. 28.300
T 060	CAVO CENTRONICS PARALL.	L. 3.432
T 061	CAVO CENTR.-CENTR.	L. 4.224
T 062	CAVO MASCHI-FEMMINA	L. 4.224
T 100	DATA SWITCH PARAL. 2 VIE	L. 17.793
T 101	DATA SWITCH SER. 2 VIE	L. 17.793
A 261	DISCHETTO PULIZIA 5"1/4	L. 2.059
A 260	DISCHETTO PULIZIA 3"1/2	L. 2.423
A 303	VASC. P.DISK 3"1/2 40 POS	L. 6.705
A 304	VASC. P.DISK 3"1/2 80 POS	L. 8.025
A 305	VASC. P.DISK 5"1/4 50 POS	L. 7.022
A 306	VASC. P.DISK5"1/4 100 POS	L. 8.395
A 385	COPERTINA PER PC	L. 5.808
A 320	SUPPORTO PER PC IN VERT.	L. 10.032
A 310	SUPPORTO IN PLEX. 80 COL.	L. 31.152
A 311	SUPPORTO IN PLEX. 136 COL.	L. 45.302
A 321	SUPPORTO STAMP. ECONOMICO	L. 6.388
T 050	TRACKBALL	L. 60.139
A 230	DISCO FLOPPY BULK 3"1/2 1MB	L. 710
A 231	DISCO FLOPPY BULK 3" 1/2 2MB	L. 1.350
A 251	DISCO FLOPPY BULK 5" 1/4 HD	L. 740
A 252	DISCO FLOPPY BULK 5" 1/4 2D	L. 420

LAP TOP

R 500	PORTATILE 286-16 1MB RAM HARD DISK 40MB	L 2.692.000
R 510	PORTATILE 286-12 1MB RAM HARD DISK 20MB	L. 2.639.472

EXpress
Office Automation

RICHIEDETE IL LISTINO COMPLETO.
DISPONIAMO A MAGAZZINO DI UNA AMPIA
GAMMA DI PRODOTTI.
SONO GRADITI GLI ORDINI E LE RICHIESTE DI INFORMAZIONI VIA FAX