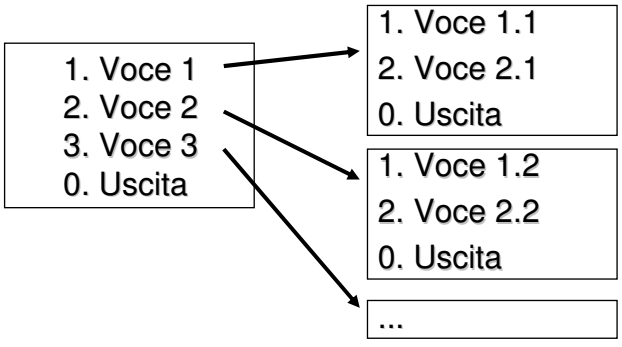
Menù V1

- Progettare un componente che consenta la visualizzazione di menù multilivello
- Ogni menù può a sua volta "contenere" un altro menù → si tratta di una struttura ricorsiva (?!)
- Per semplificare, ci si accontenta (al momento) di menù a due livelli: il menù principale più eventuali sottomenù

1

Menù V1

Esempio



2

Interfaccia

Supponendo di identificare ogni voce di menù con un indice...

- Aggiungere una voce di menù principale
  - Parametri: il testo della voce di menù
  - Valori in uscita: l'indice della voce di menù inserita
- Aggiungere una voce di sotto-menù
  - Parametri: la voce di menù principale corrispondente (il suo indice), il testo della voce di menù
  - Valori in uscita: l'indice della voce di menù inserita

3

Interfaccia

- Mostrare i menù e conoscere cosa l'utente ha scelto
  - Mostrare il menù principale
    - Se l'utente sceglie una voce "contenente" sotto-voci, mostrare le sotto-voci
      - Se l'utente sceglie una sotto-voce, terminare comunicando il successo, la voce principale e la sotto-voce
      - Se l'utente sceglie di uscire dal sotto-menù, rivisualizzare il menù principale
    - Altrimenti uscire comunicando il successo e la voce (principale) scelta
    - Se l'utente sceglie di terminare, uscire con insuccesso
  - Parametri (in ingresso): nessuno
  - Valori in uscita: la voce di menù principale scelta, la voce di sotto-menù scelta, la volontà dell'utente di terminare l'applicazione

4

Interfaccia

- Aggiungere una voce di menù principale

Testo della voce di menù

```
int addMenuItem(char itemText[]);
```

- Aggiungere una voce di sotto-menù

Indice di menù principale

```
int addSubMenuItem(int menuItemIndex, char itemText[]);
```

- Mostrare il menù ed ottenere una risposta dall'utente

Indice nel menù principale

Indice nel menù secondario

```
int showMenu(int *menuItemIndex, int *subMenuItemIndex);
```

Successo (l'utente ha scelto) – Insuccesso (l'utente vuole terminare)

5

Strutture Dati

- Memorizzare le strutture dati nell'area dati globale
  - Le strutture dati non devono essere direttamente visibili dall'esterno e devono essere manipolabili solo da funzioni/procedure del componente
- Utilizzare un array di stringhe per il menù principale e un array di array di stringhe per i menù secondari
- Il solito booleano

```
#define BOOLEAN int
#define TRUE 1
#define FALSE 0
```

- Nel file header

```
#define MENUMAXDIM 10
#define TEXTMAXDIM 50
```

- Nel file c

```
static char menuItems[MENUMAXDIM][TEXTMAXDIM];
static char
  subMenuItems[MENUMAXDIM][MENUMAXDIM][TEXTMAXDIM];
```

6

Implementazione

- Aggiunta dei menù alle strutture dati
  - Il tutto si riduce all'inserire nuove stringhe negli array contenenti le voci di menù
  - Si supponga che una stringa contenuta negli array rappresenta una voce di menù se tale stringa **non è la stringa vuota**
  - All'inizio dei tempi le strutture dati globali sono **tutte azzerate** (sono già azzerate come tutte le strutture dati globali) → non contengono voci di menù
  - L'inserimento di una stringa negli array contenenti le voci di menù si riduce all'inserimento della stringa stessa nella prima posizione che non contiene una voce di menù → **nella prima posizione che contiene la stringa vuota**

7

`addString`

- Per fattorizzare bene il codice, si introduce il metodo **`addString`** – aggiunge una stringa ad un array (come sopra...)
  - Cerca la prima posizione libera – contenente la stringa vuota
  - **Copia** la stringa
  - Restituisce la posizione di inserimento
- Parametri in ingresso
  - L'array di stringhe, la stringa da inserire
- Valori in uscita
  - La posizione di inserimento

8

Implementazione

- Ricerca della prima posizione libera nell'array: funzione **`stringCount`**
  - Poiché l'array viene riempito progressivamente, la prima posizione libera è quella il cui indice è uguale al numero di stringhe (non vuote) contenute nell'array stesso
  - Si introduce una nuova funzione **`stringCount`** (che tornerà ancora utile) che conta le stringhe contenute nel nostro array di stringhe...
- Parametri in ingresso
  - L'array di stringhe
- Valori in uscita
  - Il numero di stringhe (non vuote) contenute nell'array

9

`stringCount`

```
int stringCount(char items[][TEXTMAXDIM])
{
    int i;
    for (i = 0; i < MENUMAXDIM && items[i][0] != '\0';
            i++);
    return i;
}
```

Per verificare se la stringa è vuota, è sufficiente controllare il primo carattere...

10

addString

```
int addString(char items[][TEXTMAXDIM],
      char itemText[])
{
    int count = stringCount(items);
    if (count < MENUMAXDIM)
    {
        strcpy(items[count], itemText);
        return count;
    }
    return -1;
}
```

Copia la stringa sorgente...

...nella stringa destinazione

...se qualcosa va storto...

11

add(sub)MenuItem

```
int addMenuItem(char itemText[])
{
    return addString(menuItems, itemText);
}

int addSubMenuItem(int parentItemIndex, char itemText[])
{
    if (parentItemIndex >= 0 &&
          parentItemIndex < stringCount(menuItems))
    {
        return addString(subMenuItems[parentItemIndex],
                         itemText);
    }
    return -1;
}
```

12

Visualizzazione menù

- La "sola" visualizzazione si riduce a:
  1. Pulitura console
  2. Stampa intestazione (es: "Menù Principale")
  3. Stampa opzioni con relativo *codice*
  4. Attesa input dall'utente
  5. Validazione input – se input non valido → 4
  6. Output dei dati al chiamante
- Le operazioni di cui sopra devono essere eseguite sia nel caso di visualizzazione del menù principale, sia nel caso di sotto-menù
- È il caso di fattorizzare creando una opportuna funzione (**`internalShowMenu`**)
- Parametri in ingresso:
  - Array di stringhe che rappresenta il menù
  - Intestazione
- Valori in uscita:
  - L'utente ha scelto: una voce (successo) / di uscire (insuccesso)
  - La voce scelta

13

`internalShowMenu`

```
BOOLEAN internalShowMenu(char items[][TEXTMAXDIM], char menuName[],
                  int *itemIndex)
{
    int i, count, readMenu, paramsRead;
    system("cls");
    printf("     %s\n\n", menuName);
    count = stringCount(items);
    for (i = 0; i < count; i++)
        printf("    %2d - %s\n", i + 1, items[i]);
    printf("    0 - Esci\n\n    Opzione scelta: ");
    do
    {
        readMenu = 0;
        paramsRead = scanf("%d", &readMenu);
        while (getchar() != 10);  //Mangia fino all'LF
        if (paramsRead == 0 || readMenu < 0 || readMenu > count)
            printf("Digitare un valore fra 0 e %d\n", count);
    }
    while (paramsRead == 0 || readMenu < 0 || readMenu > count);
    if (paramsRead != 0)
    {
        *itemIndex = readMenu - 1;
        return readMenu != 0;
    }
    return FALSE;
}
```

14

internalShowMenu (1)

```
BOOLEAN internalShowMenu(char items[][TEXTMAXDIM],
      char menuName[], int *itemIndex)
{
    int i, count, readMenu, paramsRead;
    system("cls");
    printf("     %s\n\n", menuName);
    count = stringCount(items);
    for (i = 0; i < count; i++)
        printf("    %2d - %s\n", i + 1, items[i]);
    printf("    0 - Esci\n\n    Opzione scelta: ");
  ...
}
```

Pulisce lo schermo

Allinea a destra con 2 cifre (verificare!)

internalShowMenu (2)

```
int internalShowMenu(char items[][TEXTMAXDIM], char
  menuName[], int *itemIndex)
{
    ...
    do
    {
        readMenu = 0;
        paramsRead = scanf("%d", &readMenu);
        while (getchar() != 10);  //Mangia fino all'LF
        if (paramsRead == 0 || readMenu < 0 ||
                readMenu > count)
            printf("Digitare un valore fra 0 e %d\n",
                    count);
    }
    while (paramsRead == 0 || readMenu < 0 ||
      readMenu > count);
    ...
}
```

Cond. d'errore

internalShowMenu (3)

```
int internalShowMenu(char items[][TEXTMAXDIM], char
  menuName[], int *itemIndex)
{
    ...
    if (paramsRead != 0)
    {
        *itemIndex = readMenu - 1;
        return readMenu != 0;
    }
    return FALSE;
}
```

17

showMenu

```
BOOLEAN showMenu(int *menuItemIndex, int *subMenuItemIndex)
{
  BOOLEAN mainOk, stop;
  *menuItemIndex = -1;
  *subMenuItemIndex = -1;
  do
  {
    mainOk = internalShowMenu(menuItems, "Menu Principale",
             menuItemIndex);
    if (mainOk && stringCount(subMenuItems[*menuItemIndex]) > 0)
        stop = internalShowMenu(subMenuItems[*menuItemIndex],
          menuItems[*menuItemIndex], subMenuItemIndex);
    else
        stop = TRUE;
  }
  while (!stop);
  return mainOk;
}
```

Una stringa

Una array

18

Utilizzo

- Predisporre le voci di menù principale (facile)
- Predisporre le voci di sotto-menù (facile)
- Invocare **`showMenu`** (facile)
- Utilizzare la risposta di **`showMenu`** per attivare il comportamento richiesto (??!!)
  - Ogni comportamento può essere formalizzato tramite una funzione o procedura
  - La funzione o procedura da invocare può essere selezionata tramite una serie di due switch
  - In linea di principio si può prevedere uno switch per il menù principale ed uno switch per ogni sotto-menù → procedure diverse

19

Il main

```
int ok, itemIndex, subItemIndex;

populateMenus();

do
{
    ok = showMenu(&itemIndex, &subItemIndex);
    if (ok)
        handleChoice(itemIndex, subItemIndex);
}
while (ok);
```

Predispone le voci di menù

Gestisce la scelta dell'utente

20

Predisporre le voci di menù

```
void populateMenus()
{
    addMenuItem("Editing Array");    //0
    addMenuItem("Statistiche");      //1
    addMenuItem("Ricerca");          //2

    addSubMenuItem(0, "Inserimento Array");      //0.0
    addSubMenuItem(0, "Stampa Array");           //0.1

    addSubMenuItem(1, "Media");                  //1.0
    addSubMenuItem(1, "Deviazione Standard");    //1.1

    addSubMenuItem(2, "Ricerca lineare");        //2.0
    addSubMenuItem(2, "Ricerca binaria");        //2.1
}
```

21

Gestire la scelta dell'utente

Una procedura per ogni voce di menù principale a cui viene passata la sotto-voce scelta: il pattern si ripete…

```
void handleChoice(int menuIndex, int subMenuIndex)
{
    switch (menuIndex)
    {
    case 0:
        arrayEditing(subMenuIndex);
        break;
    case 1:
        stats(subMenuIndex);
        break;
    case 2:
        search(subMenuIndex);
        break;
    }
}
```

22

Gestire la scelta dell'utente

```
#define MAXARRAYDIM 100
static int workArray[MAXARRAYDIM];
static int valueCount;

void arrayEditing(int subMenuIndex)
{
    switch (subMenuIndex)
    {
    case 0:
        valueCount = readIntArray(workArray, MAXARRAYDIM);
        break;
    case 1:
        if (testValuesPresent())
            printArray(workArray, valueCount);

        break;
    }
    waitEnterKey();
}
```

Variabili globali "private" per il modulo di gestione input utente

Verifica la presenza di elementi ed eventualmente segnala l'errore

Stampa l'array (facile per noi!)

Attende la pressione del tasto invio…

23

Procedure utili…

- Strettamente correlata con il problema…

```
BOOLEAN testValuesPresent()
{
    if (valueCount <= 0)
    {
        printf("\nNessun valore inserito! Impossibile"
                    "proseguire!\n\n");
        return FALSE;
    }
    else
        return TRUE;
}
```

- Più generale…

```
void waitEnterKey()
{
    printf("\nPremere invio per continuare...");
    while (getchar() != 10);
}
```

•Attende la pressione del tasto **invio**
•Attenzione: se il buffer di input non è vuoto, svuota il buffer e **non attende**!
•Non esiste un metodo **semplice e standard** per produrre il comportamento "Press any/enter key to continue…"

24

Un progetto

- Per creare un progetto ordinato, come suddividere le funzioni e le procedure in diversi moduli / file .h e .c?

25