Menù V3

- Rimuovere tutti i vincoli dovuti alla non dinamicità delle strutture dati...
- Un menù è composto di varie voci, ognuna delle quali può essere composta da varie sotto-voci, ognuna delle quali può essere composta da varie sotto-voci, ognuna delle quali può essere composta da varie sotto-voci, ognuna delle quali può essere composta da varie sotto-voci...
- ...serve una struttura dati ricorsiva!

1

Menù V3

- Usiamo le *linked list*!
- Un menù è una voce di menù collegata alle eventuali voci successive
- Una voce di menù può essere collegata alle eventuali sotto-voci
- Una voce di menù è rappresentata da una stringa di caratteri

→ Due puntatori: uno per la voce successiva, uno per la prima sotto-voce

2

MenùV3

- Poiché il menù può essere “infinitamente” innestato, anche la risposta può essere “infinitamente” lunga
- L’utente può aver scelto la voce 10 che sta nella voce 7 che sta nella voce 5 che sta nella voce 8 che sta nella voce... → ancora una *linked list*
- La scelta dell’utente è rappresentata da un identificatore (il numero di voce che l’utente ha scelto) e dalla voce di menù scelta; contiene anche un *link* alla scelta successiva

3

Tipi di dato – Header File

```
#ifndef MENUV3
#define MENUV3

typedef char MenuText[80];

typedef struct MenuItemStruct
{
    MenuText text;
    struct MenuItemStruct *next;
    struct MenuItemStruct *subItem;
} MenuItem;

typedef struct UserChoiceStruct
{
    int choiceId;
    MenuItem *item;
    struct UserChoiceStruct *next;
} UserChoice;

#endif
```

4

Menù V3

- Sono tutti ADT!!!
- In un'applicazione si possono avere tutti i menù che si desiderano, anche scorrelati fra loro
- Si tratta di definire la giusta interfaccia...

5

Interfaccia.1

- **Creare** una voce di menù partendo dalla stringa che lo rappresenta
- **Distruggere** una voce di menù (e tutta la catena di voci e sotto-voci)
- **Appendere** una voce di menù ad un'altra
- **Inserire** una voce di menù in una catena, prima di una voce data
- **Appendere** una sotto-voce di menù ad una voce di menù
- **Inserire** una sotto-voce di menù in una catena di sotto-voci, prima di una sotto-voce data

6

Interfaccia.2

- **Mostrare** un menù ogni cui voce può contenere sotto-menù... – gestione simile a MenùV1
  - Occorre restituire una lista che rappresenta le scelte dell'utente
  - Si esce dall'algoritmo
    - Quando si sceglie di uscire dal menù principale
    - Quando si seleziona una voce di menù che non ha sotto-voci
  - Per come sono definiti i dati e per evitare di impazzire, è il caso che l'algoritmo di gestione sia ricorsivo

7

Interfaccia.3

- Gestione delle scelte utente
  - Creare una scelta utente
  - Distruggere una scelta utente
  - Appendere una scelta utente alla lista delle scelte utente

8

Interfaccia.1.prototipi

```
MenuItem* newMenuItem(MenuText text);
void destroyMenuItem(MenuItem *item);

void appendItem(MenuItem *menu, MenuItem *toAppend);
void insertBeforeItem(MenuItem *menu, MenuItem *reference,
                                MenuItem *toInsertBefore);

void appendSubItem(MenuItem *menu, MenuItem *toAppend);
void insertBeforeSubItem(MenuItem *menu,
      MenuItem *reference, MenuItem *toInsertBefore);

int itemCount(MenuItem *item);
int subItemCount(MenuItem *item);
```

9

Interfaccia.2.prototipi

```
UserChoice* showMenu(MenuItem *menu, char title[]);
```

10

Interfaccia.3.prototipi

```
UserChoice* newUserChoice(int choiceId, MenuItem *item);

void destroyUserChoice(UserChoice *choice);

void appendChoice(UserChoice *choice, UserChoice
                                          *toAppend);
```

11

Menù – Costruzione/Distruzione

```
MenuItem *newMenuItem(MenuText text)
{
    MenuItem *item = (MenuItem*)malloc(sizeof(MenuItem));
    strcpy(item->text, text);
    item->next = NULL;
    item->subItem = NULL;
    return item;
}

void destroyMenuItem(MenuItem *menu)
{
    if (menu->subItem != NULL)
        destroyMenuItem(menu->subItem);
    if (menu->next != NULL)
        destroyMenuItem(menu->next);
    free(menu);
}
```

Distrugge ricorsivamente le sotto-voci e le voci "successive" – se invocato sulla voce radice, viene distrutto l'intero menù

12

Menù – Append

```
void appendItem(MenuItem *menu, MenuItem *toAppend)
{
    if (menu->next != NULL)
    {
        appendItem(menu->next, toAppend);
    }
    else
    {
        menu->next = toAppend;
    }
}
```

13

Menù – Insert

```
void insertBeforeItem(MenuItem *menu, MenuItem *reference,
                      MenuItem *toInsertBefore)
{
    assert(menu->next != NULL);

    if (menu->next != reference)
    {
        insertBeforeItem(menu->next, reference, toInsertBefore);
    }
    else
    {
        toInsertBefore->next = reference;
        menu->next = toInsertBefore;
    }
}
```

14

SubMenù – Append

```
void appendSubItem(MenuItem *menu, MenuItem *toAppend)
{
    if (menu->subItem != NULL)
        appendItem(menu->subItem, toAppend);
    else
        menu->subItem = toAppend;
}
```

15

SubMenù – Insert

```
void insertBeforeSubItem(MenuItem *menu,
      MenuItem *reference, MenuItem *toInsertBefore)
{
    assert(menu->subItem != NULL);

    if (menu->subItem != reference)
    {
        insertBeforeItem(menu->subItem, reference,
                                          toInsertBefore);
    }
    else
    {
        toInsertBefore->next = reference;
        menu->subItem = toInsertBefore;
    }
}
```

16

Scelte dell'utente Costruzione/Distruzione

```
UserChoice* newUserChoice(int choiceId, MenuItem *item)
{
    UserChoice *choice =
              (UserChoice*)malloc(sizeof(UserChoice));
    choice->choiceId = choiceId;
    choice->item = item;
    return choice;
}

void destroyUserChoice(UserChoice *choice)
{
    if (choice->next != NULL)
        destroyUserChoice(choice->next);
    free(choice);
}
```

Distrugge la struttura UserChoice ma NON il MenuItem... Giustamente!

17

Scelta dell'utente – Append

```
void appendChoice(UserChoice *choice,
                                  UserChoice *toAppend)
{
    choice->next = toAppend;
}
```

18

Rendering e input dell'utente

- La parte un po' più complicata...
- È opportuno separare il *rendering* dalla gestione dell'input
- ...solo perché poi è più facile cambiare il solo *rendering* mantenendo invariata la gestione dell'input
  - Word Art?
- Scelta implementativa: il codice intero di "input" assegnato ad una voce di menù, dipende dalla posizione in lista della voce stessa
  - Chi effettua il *rendering* e la parte di gestione dell'input devono gestire la cosa nello stesso modo...
  - ...basta mettersi d'accordo per tempo*!*
- 0 esce (as usual!)

19

Rendering

- Funzione *render*
  - In ingresso:
    - Una lista di menù
    - Un titolo da scrivere in testa al menù
  - In uscita:
    - Il numero di menù "renderizzati" (utile per la gestione dell'input)

```
int render(MenuItem *menu, char title[]);
```

20

Rendering

```
int render(MenuItem *menu, char title[])
{
    int choiceId;
    system("cls");
    printf("****************************************");
    printf("     %s\n", title);
    printf("****************************************\n\n");
    for(choiceId = 1; menu != NULL; choiceId++, menu=menu->next)
        printf("    %2d - %s\n", choiceId, menu->text);
    printf("    %2d - %s\n\n", 0, "Esci");
    printf("    Opzione scelta: ");
    return choiceId - 1;   //Il conteggio inizia da 1!
}
```

21

Input dell'utente.0

- Algoritmo ricorsivo

1. Si mostra un menù (una lista di voci)
   a) L'utente sceglie una voce
   b) Se l'utente ha scelto una voce diversa dalla voce di terminazione, si aggiunge la scelta corrente alla lista delle scelte dell'utente
   c) Se la voce scelta contiene delle sotto-voci, si riparte con 1.
   d) Altrimenti, si termina
   e) Se l'utente ha deciso di uscire (voce di terminazione), si termina la visualizzazione corrente e si esce con insuccesso

22

Input dell'utente.1 (Pseudocodice)

- In Ingresso:
  - Una lista di menù
  - Un titolo da mostrare
- In uscita:
  - La lista di scelte dell'utente

23

Input dell'utente.1 (Pseudocodice)

1. Se la lista ricevuta come argomento è la lista vuota (nulla), restituire la lista nulla (scelta dell'utente)
2. Visualizzare il menù (*render*) – ottenere il conteggio delle voci di menù
3. Attendere l'input dell'utente (un intero)
4. Se il parametro non è stato letto correttamente, se la scelta dell'utente non è valida (inferiore a zero o superiore al codice "massimo"), ricominciare da 3
5. Se la scelta è stata effettuata correttamente
   a. Recuperare la voce di menù dal codice inserito
   b. Se la voce scelta contiene delle sotto-voci, ripartire ricorsivamente da 1. con la lista delle sotto-voci ed il testo della voce scelta → attendere restituzione della sotto-scelta
   c. Costruire la scelta corrispondente alla voce recuperata in a.
   d. Appendere l'eventuale sotto-scelta (b.) alla scelta costruita in c.
6. Ricominciare da 2. se la voce scelta non è nulla, ha delle sotto-voci e 5.b. ha restituito una sotto-scelta nulla
7. Restituire la scelta effettuata o la scelta nulla se l'utente desidera uscire

24

Input dell'utente.2 (Codice)

- ...recuperare la voce di menù dal codice inserito...
- Si è già stabilito che il codice numerico **n** corrisponde alla n-esima voce
- Si tratta di contare le voci di una lista di menù e restituire quella giusta!

```
MenuItem* getItemFromChoiceId(MenuItem *menu, int choiceId)
{
    int currentId;
    assert(menu != NULL);  //Altre precondizioni?
    for (currentId = 1;
             menu != NULL && choiceId != currentId;
             menu = menu->next, currentId++);
    assert(menu != NULL);  //Postcondizione
    return menu;
}
```

25

Input dell'utente.2 (Codice)

```
UserChoice* showMenu(MenuItem *menu, char title[])
{
    int choiceId, paramsRead, count;
    MenuItem *chosenItem;
    UserChoice *choice = NULL, *subChoice;
    if (menu != NULL)
    {
        do
        {
            if (choice != NULL) destroyUserChoice(choice);
            choice = NULL;
            chosenItem = NULL;
            subChoice = NULL;
            count = render(menu, title);
            do
            {
                choiceId = 0;
                paramsRead = scanf("%d", &choiceId);
                while (getchar() != LINEFEED);  //Mangia fino all'LF
                if (paramsRead == 0 || choiceId < 0 || choiceId > count)
                    printf("Digitare un valore fra 0 e %d\n", count);
            }
            while (paramsRead == 0 || choiceId < 0 || choiceId > count);
```

26

Input dell'utente.2 (Codice)

```
            if (paramsRead != 0 && choiceId > 0)
            {
                chosenItem = getItemFromChoiceId(menu, choiceId);
                if (chosenItem->subItem != NULL)
                    subChoice = showMenu(chosenItem->subItem,
                                                    chosenItem->text);
                choice = newUserChoice(choiceId, chosenItem);
                appendChoice(choice, subChoice);
            }
        } while (chosenItem != NULL &&
                chosenItem->subItem != NULL && subChoice == NULL);
    }   //if (menu != NULL)
    return choice;
}
```

27

Note

- MenuItem da luogo ad una struttura ad albero (binario) di cui le "istanze" di MenuItem sono i nodi
- Ogni nodo può avere DUE nodi figli (**`next`**, **`subItem`**)
- Il nodo radice (il MenuItem senza genitore) è il nodo da cui è possibile raggiungere ogni nodo dell'albero

28