Fondamenti di Informatica L-B (L-Z)
Esercitazione n°2
Abstact Data Type

A.A. 2007/08

Tutor: Barbara Pettazzoni

barbara.pettazzoni@studio.unibo.it

Login…

- Dovete loggarvi in questo modo:

  Username: lab3_numerodelvostrocomputer
  Password: lab3_numerodelvostrocomputer

  per avere "quasi" la certezza che funzioni…
- Non funziona ancora? L'unica soluzione é cambiare computer ☹
- Ora, un ripasso ultra rapido su come si crea un progetto con lcc!!!!! (Consiglio sempre di leggere l'apposita documentazione presente sul sito!)

2

Tipi di Dato Astratto - **ADT**

Un *tipo di dato astratto (ADT)* definisce una categoria concettuale con le sue proprietà:

- ***definizione di tipo* su un dominio D**
- ***insieme di operazioni ammissibili* su tale dominio.**

Si focalizza l'attenzione sulle proprietà del dominio (e quindi degli oggetti di questo tipo) *lasciando sullo sfondo la rappresentazione concreta*

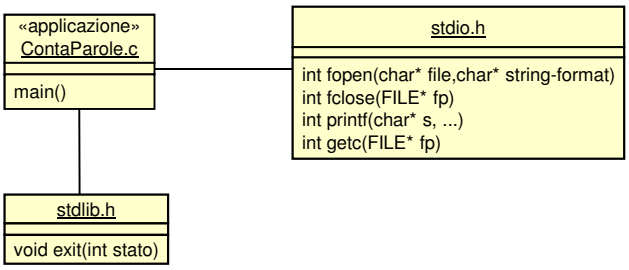
Alcune domande:

- Quali proprietà rilevanti?
- Quali operazioni dobbiamo rendere disponibili al cliente (Interfaccia)?

3

Torniamo al nostro problema:

- Scrivere un programma che conta le parole contenute in un file



- Ma questa volta contiamo le parole con un tipo di dato contatore

4

Il Contatore

Dobbiamo progettare il **tipo di dato astratto (ADT) "contatore"** e poi sfruttarlo per creare "oggetti" di tipo contatore.

Vantaggi:

- E' possibile definire tanti "oggetti" contatore quanti ne servono
- Una volta creato e implementato il componente contatore lo posso riutilizzare in tutti i progetti in cui ne ho bisogno

5

Progettare l'ADT **Contatore**

- Tipo di dato astratto **Contatore**
  1. Definire **cos'è** un contatore
  2. Definire le **operazioni** relative ad un contatore
  3. Definire la **rappresentazione interna** dell'ADT contatore
  4. Scrivere l'**interfaccia** dell'ADT contatore
  5. **Implementare** l'ADT in modo che realizzi l'interfaccia specificata

6

ADT Contatore

1. Definire **cos'è** un contatore
   - La definizione di *contatore* non è unica
     - Es: "Contatore in avanti" , "Contatore all'indietro", "Contatore Ciclico" etc....
   - La *nostra* definizione è la seguente

Un contatore è un oggetto che ha: un **valore attuale**, *un* **valore massimo** *di conteggio, uno* **stato interno** *che indica eventuali condizioni d'errore*

Il valore attuale può essere azzerato o incrementato

Il valore massimo è selezionato dall'utente. Quando si tenta di superarlo, lo stato interno commuta alla condizione d'errore e il contatore deve essere resettato. Lo stato interno può essere letto mediante una funzione specifica

ADT Contatore

2. Definire le **operazioni** relative ad un contatore

Qual è l'insieme di operazioni dell'ADT Contatore t.c. le specifiche del problema siano soddisfatte?

- In pratica stiamo definendo l'interfaccia del contatore; (dove si trova l'interfaccia di un ADT in C?)
- Nota: in questa prima fase di progetto, le operazioni possono essere espresse in un pseudo-linguaggio

ADT Contatore

2. Definire le **operazioni** relative ad un contatore

`void setMaxVal(int valore)` imposta il valore massimo

`void azzera()` resetta il contatore

`void incrementa()` incrementa il valore del contatore

`void visualizza()` stampa il valore del contatore

`int leggi()` restituisce il valore del contatore

`boolean isError()` controlla se si è verificata una condizione d'errore

9

ADT Contatore

3. Definire la **rappresentazione interna** dell'ADT Contatore
   - Esistono varie possibilità
     - Valore unico
     - Terna di valori
     - ...

Definite tutte queste proprietà possiamo andare a specificare l'interfaccia dell'ADT Contatore, cioè l'insieme di tutte le funzioni e proprietà che rendiamo disponibili al cliente

10

ADT Contatore

4. Scrivere l'**interfaccia** dell'ADT Contatore

In C l'interfaccia è contenuta negli header file .h e quindi anche per il nostro contatore dovremo creare il file header "Contatore.h" nella stessa cartella del progetto

```
/* Contatore.h */
typedef enum {NO_ERROR, OVERFLOW, BAD_MAX_VAL} StatoContatore;
typedef struct {
   int valore;
   StatoContatore stato;
  int valore_max;
} Contatore;

void setMaxVal(Contatore *c, int val);
void inizializza(Contatore *c, int val)
void azzera(Contatore *c);
void incrementa(Contatore *c);
void visualizza(Contatore c);
int leggi(Contatore c);
int isError(Contatore *c);
```

Nei metodi in cui devo cambiare lo stato del contatore è necessario passare un puntatore

Qui devo solo accedere allo stato del contatore senza modificarlo; è più opportuno il passaggio per valore

ADT Contatore

Qui abbiamo usato un intero per rappresentare il valore interno del contatore, ma potevamo usare anche una rappresentazione con una stringa di "`I`" (Notazione unaria detta anche «a stanghette») e nulla sarebbe cambiato dal punto di vista del cliente finale del contatore

5. **Implementare** l'ADT in modo che realizzi l'interfaccia specificata
   - Creare il file "Contatore.c" e inserirlo nel progetto
   - Scrivere il codice di ciascuna funzione definita nell'header

Qualche esempio di realizzazione

Rappresentazione interna come INT

```
void azzera(contatore* c)
{
  (*c).valore = 0;
  ...
}
void incrementa(contatore* c)
{
   //Controlli sul raggiungimento del valore limite
  (*c).valore++;
  ...
}

int leggi(contatore c)
{
   return c.valore;
}
```

13

Qualche esempio di realizzazione(2)

Rappresentazione interna a stanghette

Nell'interfaccia invece di int valore avremo char valore[21]

Va bene solo per contare fino a 20

```
void azzera(contatore* c)
{
  (*c).valore[0] = '\0';
   ...
}

void incrementa(contatore* c)
{
   //Controlli sul raggiungimento del valore limite
   int x = strlen((*c).valore);
   (*c).valore[x]='I'; (*c).valore[x+1]='\0';
   ...
}

int leggi(contatore c)
{
   return strlen(c.valore);
}
```

14

Come si usa un ADT:

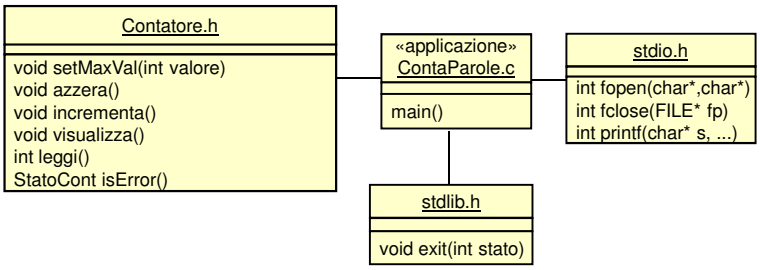
```
#include "contatore.h"

void main()
{
    int v1, v2;
    contatore c1, c2;
    azzera(&c1);
    azzera(&c2);
    setMaxVal(&c1,6);
    setMaxVal(&c2,9);
    incrementa(&c1);
    incrementa(&c1);
    incrementa(&c2);
    v1=leggi(c1);
    v2=leggi(c2);
}
```

15

Esercizio

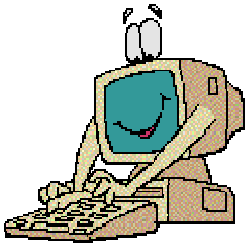
- Modificare il programma `ContaParole.c` utilizzando le funzionalità offerte dall'ADT Contatore



16

Try It Now

Provate ora a realizzare il codice



17

Qualche considerazione sugli ADT

L'APPROCCIO DEGLI ADT

- **Consente di *separare interfaccia e implementazione*: l'interfaccia rappresenta un *contratto* fra il cliente e il progettista su come deve funzionare l'ADT!!**
- **Rende il cliente *indipendente dalla struttura interna dell'ADT* (servitore)**
- **Permette al cliente di definire *tanti oggetti quanti gliene occorrono***
- **Ma non garantisce incapsulamento**
  - tutti i clienti vedono la **typedef**,
  - *conoscono la struttura interna* dell'ADT
  - e possono *violare il protocollo di accesso*

18

Una possibile soluzione

ADT Contatore: Implementazione

- **Nota**: la soluzione presentata nei lucidi seguenti è essenziale e non è commentata. Per una versione estesa e commentata si faccia riferimento ai file sorgente che sono scaricabili dalla pagina web del corso:

  http://www-db.deis.unibo.it/courses/FIL-B/

ADT Contatore: Implementazione (1)

```
/* Contatore.c */
#include "Contatore.h"

void setMaxVal(Contatore *c, int val)
{
      if(val<=0) (*c).stato = ERROR;
      else (*c).valore_max=val;
}

void azzera(Contatore *c)
{
      (*c).valore=0;
      (*c).stato=NO_ERROR;
}

StatoContatore isError(Contatore c)
{
      return c.stato;
}
```

21

ADT Contatore: Implementazione (2)

```
/* Contatore.c */
... #include <stdio.h>

void incrementa(Contatore *c)
{
   if((*c).valore >= (*c).valore_max)
   {
      (*c).stato  = ERROR;
      (*c).valore = 0;
   }
   else (*c).valore++;
}

void visualizza(Contatore c)
{
   if(!isError(c)) printf("Valore = %d.\n", c.valore);
   else printf("Errore!!!\n");
}

int leggi(Contatore c)
{
   return c.valore;
}
```

22

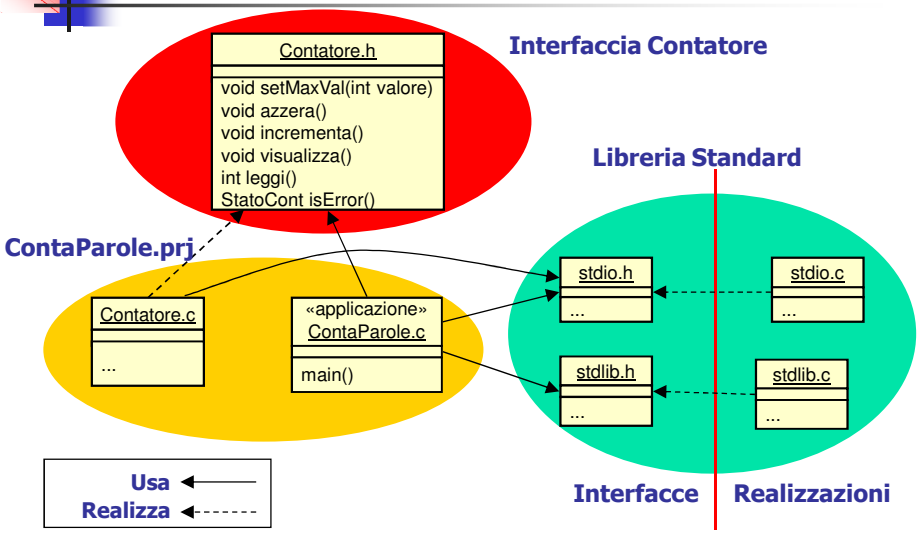
ContaParole e ADT Contatore

```
#include "Contatore.h"
...
main()
{
   FILE *fp;
   Contatore cont;
   int nletti;          /* Char letti dalla fscanf */
   char buffer[512+1];

   ... /* Apertura File */
   azzera(&cont);
   setMaxVal(&cont, 200);
   nletti = fscanf(fp, "%s",buffer);
   if(nletti>0) incrementa(&cont);
   while( !feof(fp) )
   {
        incrementa(&cont);
        fscanf(fp, "%s",buffer); printf("%s ",buffer);
   }
   printf("\nIl file contiene %d parole.\n",leggi(cont));
   printf("Contatore: ");  visualizza(cont);
   fclose(fp);
}
```

23

ContaParole e ADT Contatore



24

Es: Violare il protocollo (1)

- Violare il protocollo é banale, in realtá.
- Se leggete il file header, sapete come é strutturato internamente un contatore…
- …e potete facilmente capire come accedere alle proprietá definite dalle typedef.
- Siete costretti ad usare le funzioni specificate nel file header? ***No, ma non usarle significa non garantire la consistenza del contatore!***

25

Es: Violare il protocollo (2)

```
#include <stdio.h>
#include <stdlib.h>
#include "contatore.h"

int main()
{
   Contatore c;
   inizializza(&c,3);
   visualizza(c);
   printf("Il contatore puo' arrivare al massimo fino a 2\n");
   incrementa(&c);
   incrementa(&c);
   visualizza(c);
   …
```

26

Es: Violare il protocollo (3)

```
    printf("Ma chi mi vieta di incrementarlo cosi'?\n");
    c.valore = c.valore+1;
    c.valore = c.valore+1;
    c.valore = c.valore+1;
    visualizza(c);
    visualizzaStato(c.stato);
    printf("Non va bene...\n");
    printf("Solo usando la funzione incrementa\n");
    printf("ho la sicurezza di non andare oltre!!\n");
    incrementa(&c);
    visualizza(c);
    visualizzaStato(c.stato);
}
```

27