

## TIPO DI DATO ASTRATTO "Contatore"

DATI (Attributi che caratterizzano un contatore):  
valore\_cont;

OPERAZIONI (quelle permesse su un tipo di dato contatore")  
void inc();  
void dec();  
int getValue();

## Esempio: il Contatore Approccio classico:

```
main()
{
...
int cont;
cont++;
cont--;
if (cont == MAX) { ....}
cont = cont*cont;
/* può avere senso per un intero ma non ha alcun
senso
per una entità contatore */
}
```

## Vorremmo Poter Fare:

```
main()
{Contatore cont; /* definisco un TDA Contatore*/
Contatore cont2;
cont.dec(); /* invoco delle operazioni sul contatore */
cont2.dec() /* le stesse operazioni le posso invocare
chiedendole all'altro contatore: sono dello stesso tipo,
ma ovviamente avranno attributi specifici diversi!*/
if(cont.getValue() > MAX) { ....}
cont++; /* ERRORE: NON E' UN INTERO */
cont.valore_cont=cont*345;
cont.valore_cont++ /* ERRATE ENTRAMBE: il dato
valore_cont è accessibile solo tramite le operazioni di
cont */
```

## Problemi:

- dov'è l'entità contatore?
- dove sono le operazioni ammesse sul contatore?
  - stiamo usando le mosse elementari della macchina C (integer e operatori di incremento), non stiamo usando delle categorie concettuali di tipo contatore!

**DATI:**

```
char nome[20];
char cognome[20];
int matricola;
Cont num_esami_dati;
struct esami sostenuti {char * nome; int voto};
```

**OPERAZIONI:**

```
void nuovo_esame_dato(char *nome_esame, int
voto_preso);
Int calcola_media(void);
void iscrivi_anno_successivo();
```

**Approccio Classico:**

```
typedef struct Studente {
    char nome[20];
    char cognome[20];
    int matricola;
    int num_esami_dati;
    struct esami[29] {char nome[20]; int voto;}
}

void nuovo_esame_dato(Studente s, char
*nome_esame, int
voto_preso)
{...}
/* e altre funzioni varie.....*/
```

**Vorremmo Poter Fare:**

```
main()
{
    Studenti s1("Rebecca Montanari"), s2("Donald",
"Duck");
    s1.nuovo_esame(s1, "FondamentiInformatica", 28);
}
```

**GROSSO CAMBIAMENTO DI PARADIGMA:**

**NON** si richiede a funzioni di operare su dati  
nuovo\_esame(s1, "Sistemi Operativi", 28);  
**MA** si richiede alle entità di eseguire delle funzioni  
s1.nuovo\_esame(s1, "Sistemi Operativi", 28);

```
main()
{ Studenti s1, s2;
    s1.nome = "pippo"; s2.cognome = "rossi";
    /* NON BELLO: COSI' ACCEDO AI DATI
    INTERNI DI STUDENTE*/
}
```

**File stack.c**

```
static struct { int TOP;
               int S[SIZE];} SS[DIM]

void stack_push(int key, int x) { ...};
int stack_pop(int key) { ....};

/*key serve per individuare quale stack deve essere interessato all'operazione
static int KEY; /*serve per sapere se ci sono stack liberi
int stackCREATE () { ...}
```

**File main.c**

```
#include <stack.h>
main ()
{ int obj, obj1;
  obj=stackCREATE();
  ....}
```

**Stack:** tipo di dato astratto**File stack.h**

```
void stack_push(int key, int x) { ...};
void stack_pop(int key) { ....};
int stackCREATE () { ...}
```

**File stack.c**

```
....
static struct { int TOP; intS[SIZE];} stato;
void Init() { ...};
void Push(int x) { ...};
int Pop() { ...};
```

**File stack.h**

```
void Init();
void Push(int x);
int Pop();
```

**File main.c**

```
#include <stack.h>
main ()
{ ....
  Init();
  Push(10);
  printf("primo elemento estratto %d\n", Pop());
  ....}
```

**Stack:** astrazione di dato**File stack.c**

```
#include <stack.h>
void Init(STACK_TYPE *sp) { ...};
void Push(STACK_TYPE *sp int x) { . .};
int Pop(STACK_TYPE *sp) { ...};
```

**File stack.h**

```
typedef struct {
    int TOP;
    int S[SIZE];} STACK_TYPE;
void Init(STACK_TYPE *sp) { ...};
void Push(STACK_TYPE *sp int x) { . .};
int Pop(STACK_TYPE *sp) { ...};
```

**File main.c**

```
#include <stack.h>
main ()
{ STACK_TYPE s1, s2;
  Init(&s1);
  Push(&s1 10);
  ....}
```

**Stack:** tipo di dato astratto