PRONTUARIO DESIGN PATTERN

PATTERN SENZA MEMORIZZAZIONE

PATTERN CON MEMORIZZAZIONE

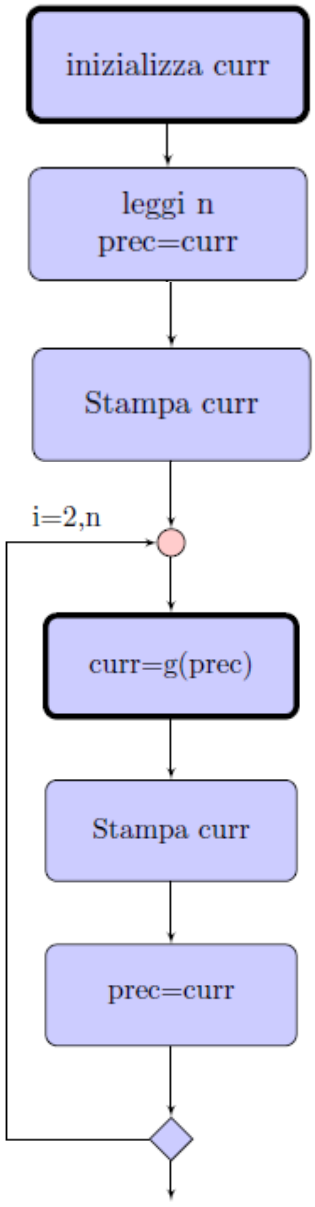
PATTERN SENZA MEMORIZZAZIONE

GENERAZIONE DI SEQUENZA

Il processo di generazione può essere formulato in termini generali introducendo una opportuna funzione generatrice ***g*** che ha l'obiettivo di creare per passi successivi tutti gli elementi della sequenza stessa.

La generazione di una sequenza, nell'ipotesi di generatrice nella forma $x_i = g(x_{i-1})$, che genera ogni elemento della sequenza solo sulla base del precedente valore.

a. Caso in cui viene richiesto di generare una sequenza di *lunghezza nota* ***n***.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Senza tappo
Dipendenze	Valore Precedente
Tipo	Generazione

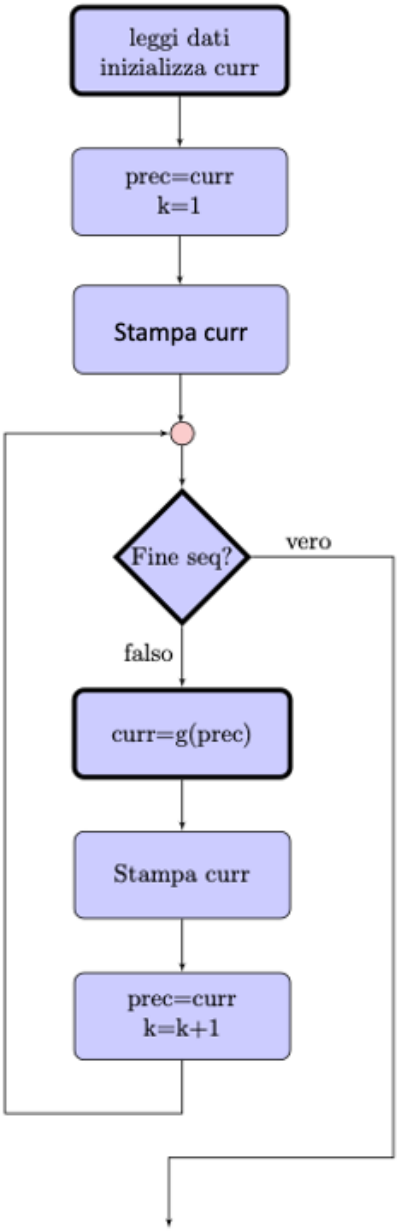
Come specializzare il pattern?

1. Inizializzazione del primo elemento (ovvero, *curr*).
2. Definizione della funzione generatrice *g*.

Codice C

```
int curr ; /* Contiene il risultato all' iterazione corrente */
int prec ; /* Contiene il risultato dell' iterazione precedente */
int i; /* Variabile contatore */
int n; /* Numero di valori della sequenza */

curr = ... ;       /* Inserire inizializzazione del primo elemento*/
printf ("Inserisci  il  valore  di n: ");
scanf ("%d" ,&n);
prec =curr;
printf ("%d\n",curr );

/* generazione della sequenza desiderata */
for (i =2; i <=n; i ++){
     curr = ... ;     /*Definizione della funzione generatrice g  */
     printf ("%d\t",curr );
     prec = curr ;
}
```

b. Caso in cui viene richiesto di generare una sequenza *con tappo*.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Con tappo
Dipendenze	Valore Precedente
Tipo	Generazione

Come specializzare il pattern?

1. Inizializzazione del primo elemento (ovvero, *curr*);
2. Definizione blocco decisionale per verificare la condizione di terminazione;
3. Definizione della funzione generatrice *g*.

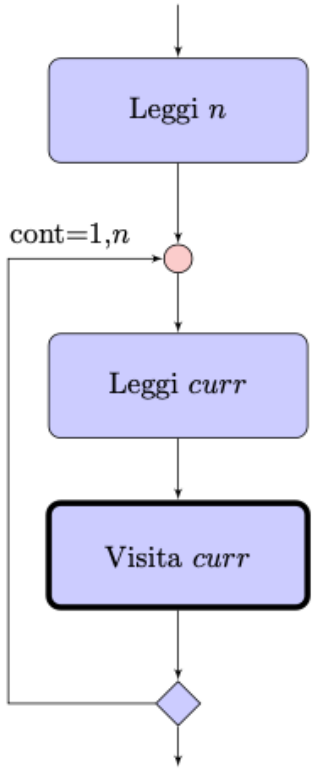
Codice C

```
int curr ; /* Contiene il risultato all' iterazione corrente */
int prec ; /* Contiene il risultato dell' iterazione precedente */
int k; /* Variabile contatore */

/* lettura dei dati */
... ;

/* inizializzazione delle variabili*/
curr = ... ;    /*Inserire inizializzazione del primo elemento */
prec =curr;
k = 1;
printf ("%d\n",curr );

/* generazione della sequenza desiderata */
while(...) {        /*Inserire la condizione di terminazione */
    curr = ... ;    /* Definizione della funzione generatrice g. */
    printf ("%d\t", curr );
    prec = curr ;
    k = k+1;
}
```

VISITA DI SEQUENZA

La visita consiste nel raggiungere tutti gli elementi in ordine prefissato, che in genere è quello crescente a partire dal primo elemento.

a. Visita di una sequenza di *lunghezza nota* ***n***.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Senza tappo
Tipo	Visita

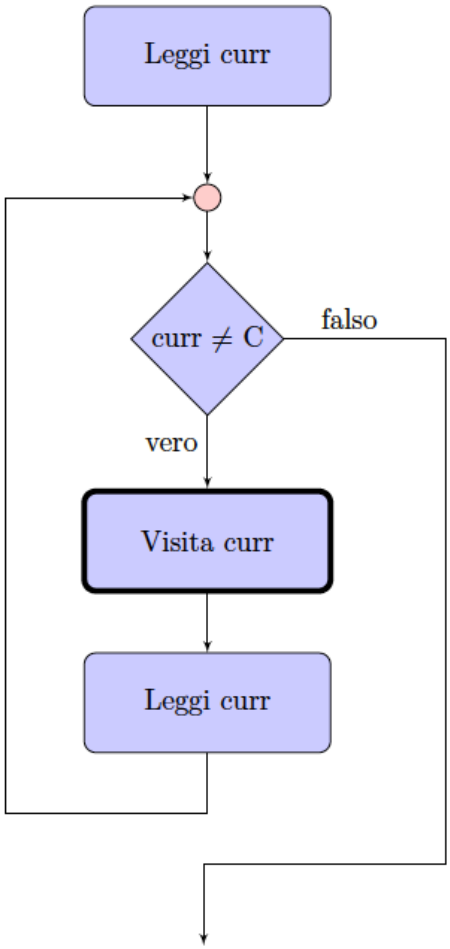
Come specializzare il pattern?

1. Definizione della visita dell'elemento *curr*.

Codice C

```
int curr ; /* Valore corrente della sequenza */
int i ;     /* Variabile contatore */
int n ;   /* Numero di elementi della sequenza*/


/* lettura dei  coefficienti */
printf ("Inserisci  il  valore  di n: ");
scanf ("%d" ,&n);

 for(i=1; i<=n; i++){
       /* lettura dei coefficienti */
        printf("Inserisci il valore di curr:");
        scanf("%d",&curr);
        printf("%d\n",curr); /* visita dell'elemento */

 }
```

b. Visita di una sequenza *con tappo*.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Con tappo
Tipo	Visita

Come specializzare il pattern?

1. Inizializzazione del primo elemento (ovvero, curr);
2. si visita l'elemento corrente, e si ritorna in ciclo alla lettura di un nuovo elemento della sequenza

Codice C

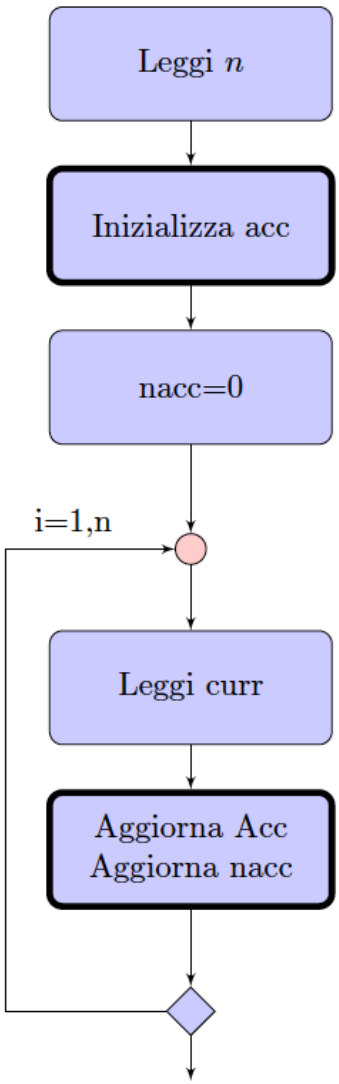
```
/* si ipotizza che sia stato definito un tappo C */
int curr ; /* Valore corrente della sequenza */

/* lettura dei coefficienti */
printf ("Inserisci  il  valore  di curr: ");
scanf ("%d" ,&curr);

/* visita della sequenza */
while( curr!=C){   /* C = tappo */

      printf("%d\n",curr); /* visita dell'elemento */
      /* lettura dei coefficienti */
      printf("Inserisci il valore di curr:");
      scanf("%d",&curr);
}
```

ACCUMULAZIONE SU UNA SEQUENZA

L'accumulazione può essere finalizzata a:

- calcolare la cardinalità (numerosità), ***nacc,*** di un opportuno sottoinsieme di elementi della sequenza che soddisfano una data proprietà;
- totalizzare i valori degli elementi che compongono la sequenza, ***acc***, o al limite un sottoinsieme di essa;
- tenere in conto la posizione dell'elemento nella sequenza.

a. Accumulazione su una sequenza di *lunghezza nota* ***n***.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Senza tappo
Tipo	Accumulazione

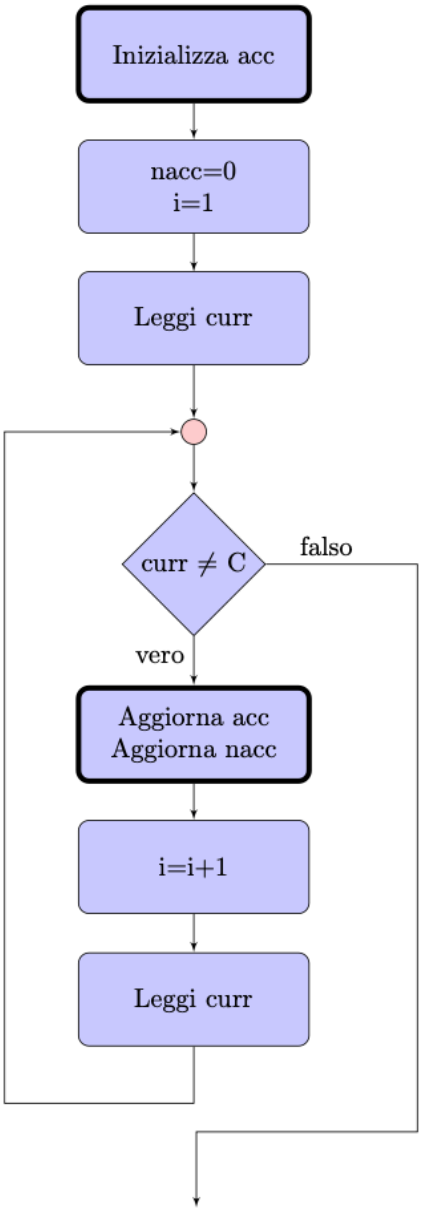
Come specializzare il pattern?

1. Inizializzazione dell'accumulatore, *acc*;
2. Definizione del blocco relativo all'aggiornamento dell'accumulatore e del contatore, *acc* e *nacc*,

Codice C

```
int curr ; /* Valore corrente della sequenza */
int i ;    /* Variabile contatore */
int n ;  /* Numero di elementi della sequenza*/
float acc ;  /* Variabile accumulatore */
int nacc ; /* Numero di elementi aggiunti all'accumulatore */

 /* lettura dei coefficienti */
printf ("Inserisci  il  valore  di n: ");
scanf ("%d" ,&n);

acc = ...; /* inizializzazione dell'accumulatore*/
nacc = 0; /* inizializzazione del contatore accumulatore*/

for(i=1; i<=n; i++){
  /* lettura dei coefficienti */
  printf("Inserisci il valore di curr:");
  scanf("%d",&curr);

  acc = ...; /* aggiornamento dell'accumulatore*/
  nacc = ...; /*aggiornamento num elementi accumulatore*/
}
```

b. Accumulazione su una sequenza *con tappo*.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Con tappo
Tipo	Accumulazione

Come specializzare il pattern?

1. Inizializzazione dell'accumulatore, *acc*;
2. Definizione del blocco relativo all'aggiornamento dell'accumulatore e del contatore, *acc* e *nacc*

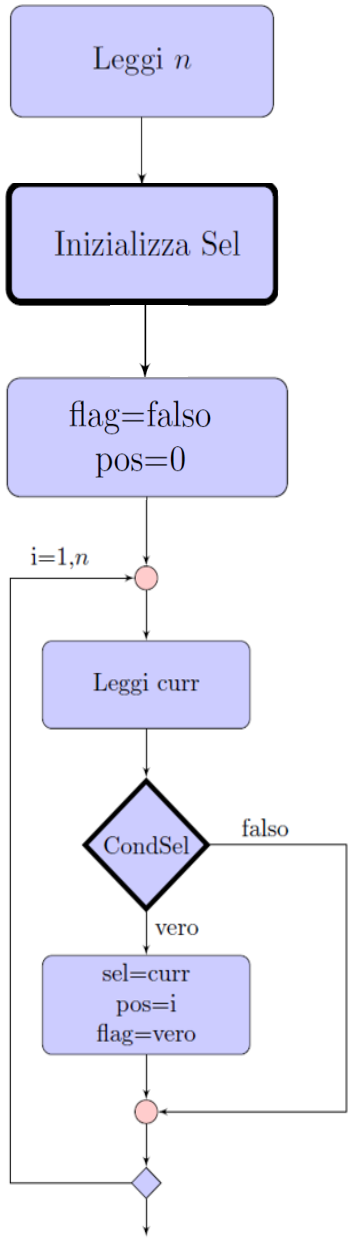
Codice C

```
/* si ipotizza che sia stato definito un tappo C */
 int curr ; /* Valore corrente della sequenza */
 int i ;     /* Variabile contatore */
 float acc ;  /* Variabile accumulatore */
 int nacc ; /* Numero di elementi aggiunti all'accumulatore */

 acc = … ; /* inizializzazione dell'accumulatore*/
 nacc = 0; /* inizializzazione del contatore accumulatore*/
 i = 1;      /* inizializzazione del contatore */

 /* lettura dei coefficienti */
 printf("Inserisci il valore di curr:");
 scanf("%d",&curr);

 while (curr != C) {    /* C = tappo */

   acc = …; /* aggiornamento dell'accumulatore*/
   nacc = …;  /* aggiornamento num elementi accumulatore*/

    i = i+1;  /* aggiornamento contatore*/
   /* lettura dei coefficienti */
   printf("Inserisci il valore di curr:");
   scanf("%d",&curr);
 }
```

SELEZIONE DI UN ELEMENTO IN UNA SEQUENZA

Si basa sulla visita della sequenza; ad ogni iterazione bisogna verificare, mediante un'opportuna condizione ***ConSel***, specializzata per la risoluzione del particolare problema, se bisogna selezionare o meno l'elemento corrente. Se ciò non accade, si passa ovviamente al prossimo elemento, altrimenti assegniamo il valore dell'elemento ***curr*** alla variabile ***sel***, destinata a contenere il valore selezionato. Se viene richiesto di memorizzare il posto occupato dall'elemento della sequenza usiamo una variabile ***pos***. Inoltre, si introduce una variabile binaria denominata ***flag*** per memorizzare la condizione che la selezione è già stata effettuata almeno una volta con successo.

a. Selezione in una sequenza di *lunghezza nota* ***n***.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Senza tappo
Tipo	Selezione

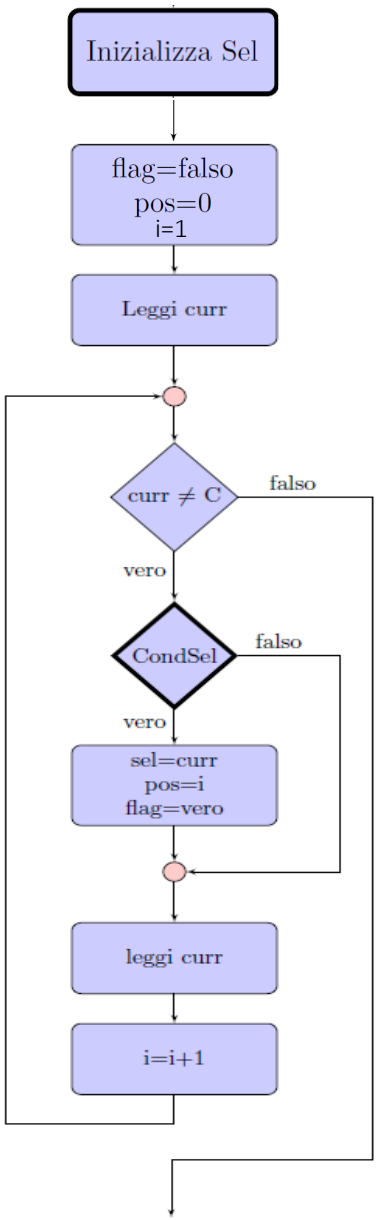
Come specializzare il pattern?

1. Inizializzazione della variabile *sel*.
2. Definizione blocco decisionale per selezionare l'elemento della sequenza.

Codice C

```
int curr ; /* Valore corrente della sequenza */
int i ;      /* Variabile contatore */
int flag;  /* Variabile che verifica la avvenuta selezione  */
int pos ; /* posizione del valore selezionato */
int sel ;  /* valore selezionato */
int n;     /* dimensione della sequenza */

/* lettura dei coefficienti */
printf("Inserisci il valore di n:");
scanf("%d",&n);
sel = … ; /* inizializzazione della variabile sel*/
flag = 0; /* inizializzazione della variabile di verifica*/
pos = 0; /* inizializzazione della variabile pos*/

for (i=1; i<=n; i++) {
  /* lettura dei coefficienti */
  printf("Inserisci il valore di curr");
  scanf("%d",&curr);
  if ( … ) { /* definizione condizione di selezione */
     sel  = curr; /* aggiornamento della selezione*/
     pos = i;  /* aggiornamento della posizione del selezionato*/
     flag = 1;  /* aggiornamento variabile di verifica*/
   }
}
```

b. Selezione in una sequenza *con tappo*.



Proprietà	
Memorizzazione	No
Tipo Terminazione	Con tappo
Tipo	Selezione

Come specializzare il pattern?

1. Inizializzazione della variabile *sel*.
2. Definizione blocco decisionale per selezionare l'elemento della sequenza

Codice C

```
/* si ipotizza che sia stato definito un tappo C */
int curr ; /* Valore corrente della sequenza */
int i ;     /* Variabile contatore */
int flag;  /* Variabile che verifica la avvenuta selezione  */
int pos ; /* posizione valore selezionato */
int sel ;  /* valore selezionato */

sel = … ; /* inizializzazione sel*/
flag = 0; /* inizializzazione della variabile di verifica*/
pos = 0; /* inizializzazione della posizione*/
i = 1;  /* inizializzazione del contatore*/

/* lettura dei coefficienti */
printf("Inserisci il valore di curr:");
scanf("%d",&curr);

while (curr != C) {  /* C = tappo */
   if ( … ) { /* definizione condizione di selezione */
      sel  = curr; /* aggiornamento della selezione*/
      pos = i;  /* aggiornamento della posizione del selezionato*/
      flag = 1;  /* aggiornamento variabile di verifica*/
   }
   /* lettura dei coefficienti */
  printf("Inserisci il valore di curr:");
  scanf("%d",&curr);
  i = i+1;
}
```

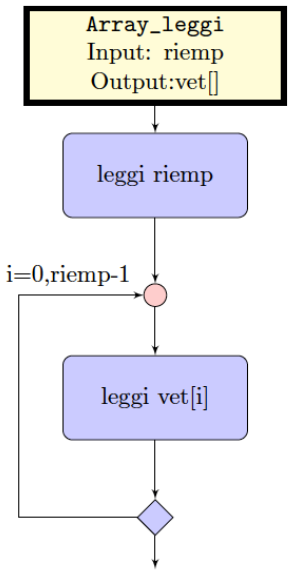
PATTERN CON MEMORIZZAZIONE

LETTURA E MEMORIZZAZIONE

Un vettore ha una cardinalità stabilita all'atto della propria dichiarazione (vettori statici) e può ospitare un numero massimo di elementi che il programmatore deve specificare nel momento in cui realizza il programma.

a. Caso in cui viene richiesto di leggere e memorizzare una sequenza di *lunghezza nota* ***riemp***.

Proprietà	
Memorizzazione	Si
Tipo Terminazione	Senza tappo
Tipo	Lettura e Memorizzazione

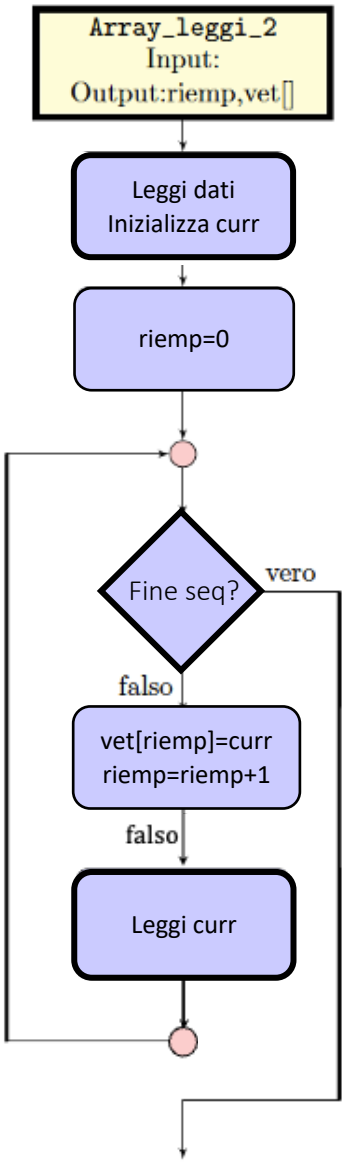


Come specializzare il pattern?

Codice C

```
int i; /* Variabile contatore */
int riemp; /* Riempimento del vettore */

/* lettura dei coefficienti*/
printf ("Numero degli elementi del vettore: ");
scanf ("%d" ,&riemp);

/* generazione della sequenza desiderata */
for (i =0; i <=riemp-1; i ++){
    printf ("lettura dell'elemento di indice %d : ", i);
    /* Aggiornamento del vettore */
    scanf("%d", &vet[i]) ;
}
```

b. Caso in cui viene richiesto di leggere e memorizzare una sequenza *con tappo*.



Proprietà	
Memorizzazione	Si
Tipo Terminazione	Con tappo
Tipo	Lettura e Memorizzazione

Come specializzare il pattern?

Codice C

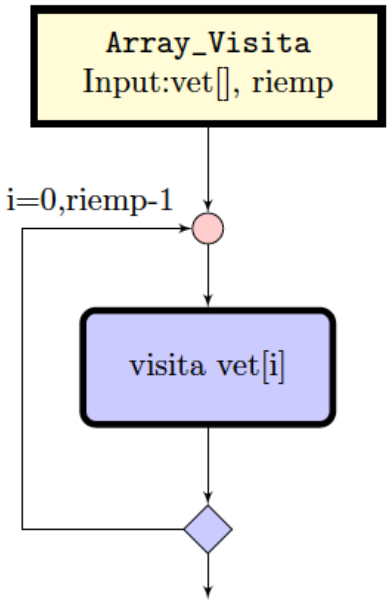
```
    int curr; /* identificazione */

    /* lettura dei dati */
     ... ;
/* inizializzazione dei coefficienti*/
    curr = ... ;   /*Inserire inizializzazione del primo elemento */
    riemp=0;

    /* generazione della sequenza desiderata */
    while(...) {        /*Inserire la condizione di terminazione */
          vet[riemp] = curr;
          riemp=riemp+1 ;
          curr = ... ;   /* Definizione della funzione generatrice g. */
      }
    }
```

VISITA DI SEQUENZA

La visita consiste nel raggiungere tutti gli elementi in ordine prefissato. Il contatore di ciclo parte da zero, si arresta all'ultimo elemento, collocato nella posizione di posto riemp-1.

Visita di una sequenza di *lunghezza nota* ***riemp***.

Proprietà	
Memorizzazione	Si
Tipo Terminazione	Senza tappo
Tipo	Visita



Come specializzare il pattern?

Codice C

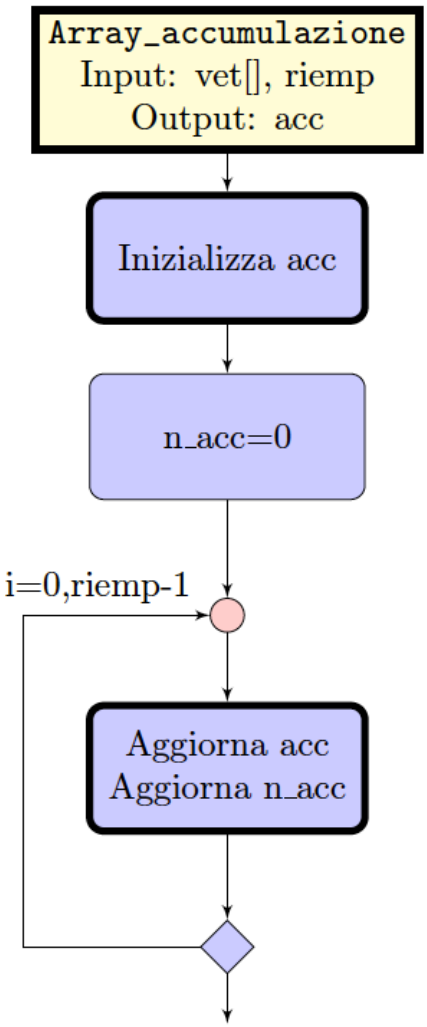
```
int i ;     /* Variabile contatore */

for(i=0; i<riemp-1; i++){
/* lettura dei coefficienti */
  printf("L'elemento con indice %d ha valore %d ", i, vet[i]);
}
```

ACCUMULAZIONE SU UNA SEQUENZA

I problemi di accumulazione sono caratterizzati da dover disporre, per ogni generica iterazione del ciclo, delle seguenti informazioni:

- il valore dell'accumulatore **acc**
- il numero di incrementi dell'accumulatore **n_acc**
- l'indice dell'iterazione corrente (che denota anche il posto occupato dall'elemento) **i**
- il valore dell'elemento corrente **curr**

Accumulazione su una sequenza di *lunghezza nota* ***riemp***.



Proprietà	
Memorizzazione	Si
Tipo Terminazione	Senza tappo
Tipo	Accumulazione

Come specializzare il pattern?

1. Inizializzazione dell'accumulatore, *acc*;
2. Definizione del blocco relativo all'aggiornamento dell'accumulatore e del contatore, *acc* e *n_acc*,

Codice C

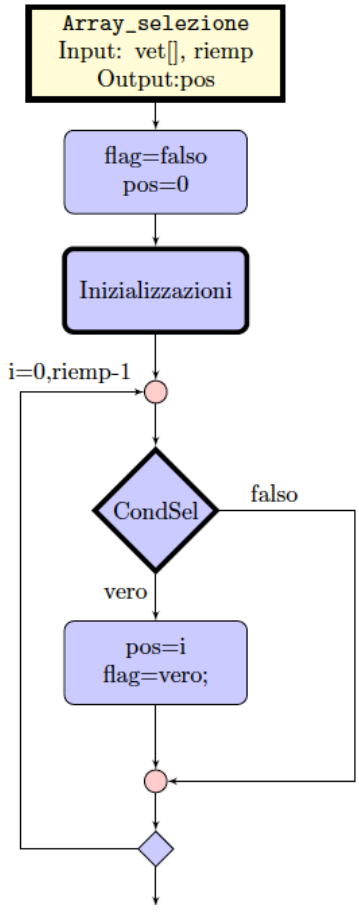
```
int i ;     /* Variabile contatore */
float acc ;  /* Variabile accumulatore */
int n_acc ; /* Numero di elementi aggiunti all'accumulatore */


acc = ...; /* inizializzazione dell'accumulatore*/
nacc = 0; /* inizializzazione del contatore accumulatore*/

for(i=0; i<=riemp-1; i++){
  /* aggiornamento dei coefficienti */
  acc = ...; /* aggiornamento dell'accumulatore*/
  nacc = ...; /*aggiornamento num elementi accumulatore*/
}
```

SELEZIONE DI UN ELEMENTO IN UNA SEQUENZA

Il pattern di selezione realizza la visita (iterativamente) della sequenza, analizzando al generico passo se si verifica la condizione ***ConSel*** per selezionare l'elemento corrente ***curr***; in tal caso si memorizza la posizione dell'elemento selezionato in ***pos***; si introduce una variabile binaria denominata ***flag*** per memorizzare la condizione che la selezione è già stata effettuata almeno una volta con successo.

Selezione in una sequenza di *lunghezza nota* ***riemp***.

Proprietà	
Memorizzazione	No
Tipo Terminazione	Senza tappo
Tipo	Selezione



Come specializzare il pattern?

1. eventuali inizializzazioni per il problema dato.
2. Definizione blocco decisionale per selezionare

Codice C

```
int i ;       /* Variabile contatore */
int flag;    /* Variabile che verifica la avvenuta selezione  */
int pos ;   /* posizione dell'elemento selezionato */
int riemp; /* dimensione della sequenza */

flag = 0; /* inizializzazione della variabile di verifica*/
pos = 0; /* inizializzazione della variabile pos*/
/*inserire un ulteriore blocco di eventuali inizializzazioni*/

for (i=0; i<=riemp-1; i++) {
  /* aggiornamento dei coefficienti */
     if ( … ) { /* definizione condizione di selezione */
         pos = i;  /* aggiornamento della posizione del selezionato*/
         flag = 1;  /* aggiornamento variabile di verifica*/
     }
 }
```