

4. TABLOU

TABLOUL


Structură de date statică.

Definiție: Fie C o colecție de elemente de tipul TE și $[n]=\{1,2,3,\dots,n\}$ cu $n \in \mathbb{N}^*$ o mulțime de indici.

Aplicația $f: [n_1] \times [n_2] \times \dots \times [n_k] \rightarrow C$ care atașează fiecărui tuplu de k indici (i_1, i_2, \dots, i_k) unde $i_j \in [n_j]$, un element c_{i_1, i_2, \dots, i_k} definește un **tablou** k dimensional $T(n_1, n_2, \dots, n_k)$

Secvențele sunt deci aplicații ale domeniului indecșilor pe domeniul de valori.

Deci pentru $k = 1$ se obține un tablou unidimensional numit *vector*, $f: [n] \rightarrow C$, având elementele $c_1 = f(1); c_2 = f(2); \dots; c_n = f(n)$.



Un tablou este **static**: nu pot fi inserate sau șterse elemente de tablou (celule). Identificarea elementelor se face cu ajutorul indicilor.

În memoria internă elementele unui tablou static vor ocupa o zonă compactă (contiguă) de memorie, adică locații succesive de memorie, ceea ce asigură accesul direct (random) la un element - reprezentare secvențială.



Pentru tablourile statice limbajele de programare au un **tip de dată** predefinit:

array în Pascal; [] C/C++ ,

Accesul la elementul **x[i]**:

adresa de început a tabloului (a) + poziția în vector * dimensiunea ocupată în memorie de un element

Tablou Unidimensional:

x: *array*[n] of TE

înseamnă : x_0, \dots, x_{n-1}

$$x[i] = a + i * \text{sizeof}(TE)$$

x: *array*[$i_{in}..i_{fin}$] of TE

$$x[i] = a + (i - i_{in}) * \text{sizeof}(TE)$$



Tablou Multidimensional (de k dimensiuni)

reprezentarea în memorie este liniară

La tablourile bidimensionale:

„linie după linie“ (indicii variază de la dreapta la stânga)

„coloană după coloană “ (indicii variază de la stânga la dreapta)

Tablourile k dimensionale - interpretate ca vectori de tablouri k-1 dimensionale.

Accesul la un element $x[i_1, i_2, \dots, i_{k-1}, i_k]$:

x : *array*[n_1, n_2, \dots, n_k] of TE

$$x[i_1, i_2, \dots, i_{k-1}, i_k] = a + \left(\sum_{j=1}^{k-1} (i_j - 1) * \prod_{j < i < k+1} n_i \right) * \text{sizeof}(\text{TE})$$

Domeniul indecșilor depinde de limbaj.

Există 3 implementări importante:

$\text{index} \in \{0, 1, \dots, n-1\}$

$\text{index} \in \{1, \dots, n\}$ (Matlab, Fortran)

$\text{index} \{l_{\text{in}}, \dots, l_{\text{fin}}\}$

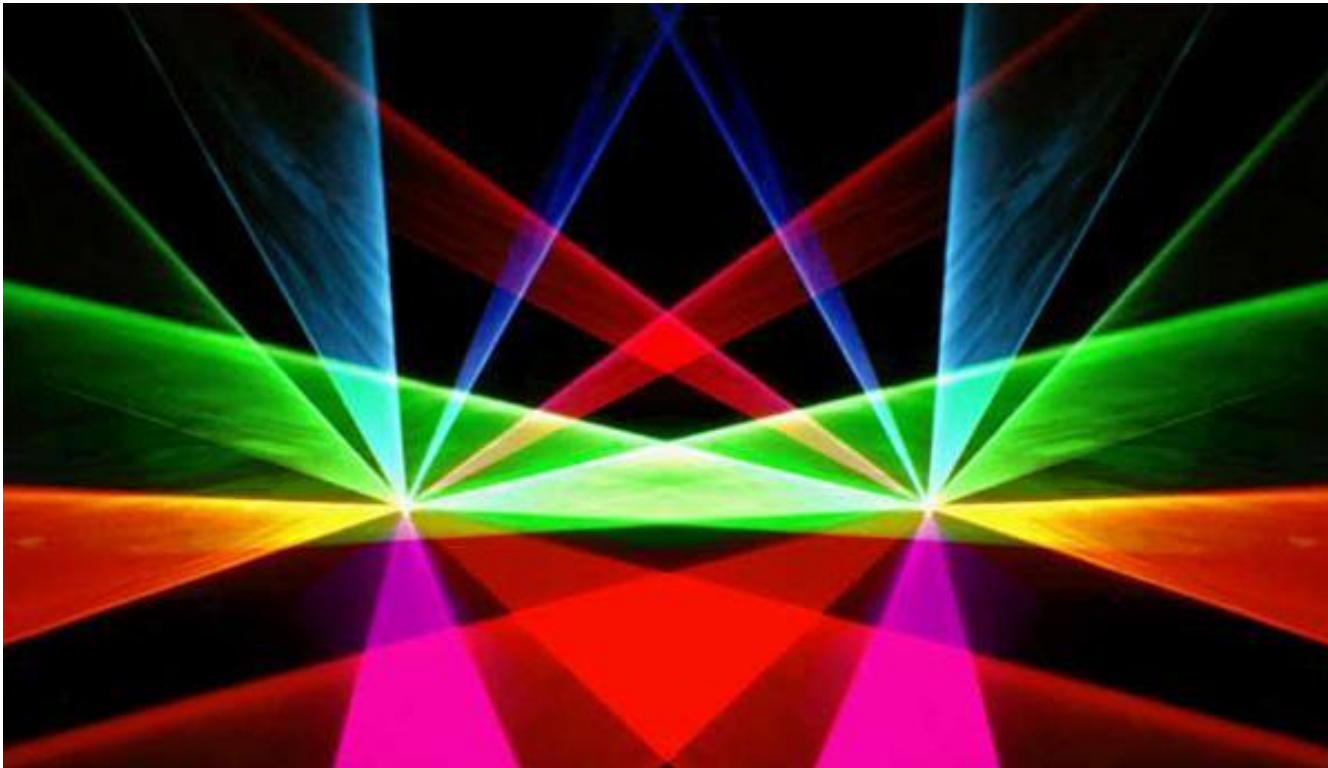
	Default Base index	Specifiable Index Type	Specifiable Base Index	Bound Check	Multi- dimensional
C	0	No	No	unchecked	array of array
C++	0	No	No	unchecked	array of array
Java	0	No	No	checked	array of array
Pascal	index type	Yes	Yes	varies	yes

Tablourile k dimensionale interpretate ca vectori de tablouri $k-1$ dimensionale.

Exemplu:

Imagine RGB 3 dimensiuni

Vector de 3 matrice (una pentru fiecare culoare)



990033 R: 153 G: 000 B: 051	FF3366 R: 255 G: 051 B: 102	CC0033 R: 204 G: 000 B: 051	FF0033 R: 255 G: 000 B: 051	FF9999 R: 255 G: 153 B: 153	CC3366 R: 204 G: 051 B: 102	FFCCFF R: 255 G: 204 B: 255	CC6699 R: 204 G: 051 B: 153	993366 R: 153 G: 051 B: 102	660033 R: 102 G: 000 B: 051	CC3399 R: 204 G: 051 B: 153	FF99CC R: 255 G: 153 B: 204	FF66CC R: 255 G: 102 B: 204	FF99FF R: 255 G: 153 B: 255	FF6699 R: 255 G: 102 B: 153	CC0066 R: 204 G: 000 B: 102
FF0066 R: 255 G: 000 B: 102	FF3399 R: 255 G: 051 B: 153	FF0099 R: 255 G: 000 B: 153	FF33CC R: 255 G: 051 B: 204	FF00CC R: 255 G: 000 B: 204	FF66FF R: 255 G: 102 B: 255	FF33FF R: 255 G: 051 B: 255	FF00FF R: 255 G: 000 B: 255	CC0099 R: 204 G: 000 B: 153	990066 R: 153 G: 000 B: 102	CC66CC R: 204 G: 102 B: 204	CC33CC R: 204 G: 051 B: 204	CC99FF R: 204 G: 153 B: 255	CC66FF R: 204 G: 102 B: 255	CC33FF R: 204 G: 051 B: 255	993399 R: 153 G: 051 B: 153
CC00CC R: 204 G: 000 B: 204	CC00FF R: 204 G: 000 B: 255	9900CC R: 153 G: 000 B: 204	990099 R: 153 G: 000 B: 153	CC99CC R: 204 G: 153 B: 204	996699 R: 153 G: 102 B: 153	663366 R: 102 G: 051 B: 102	660099 R: 102 G: 000 B: 153	9933CC R: 153 G: 051 B: 204	660066 R: 102 G: 000 B: 102	9900FF R: 153 G: 000 B: 255	9933FF R: 153 G: 051 B: 255	9966CC R: 153 G: 102 B: 204	330033 R: 051 G: 000 B: 051	663399 R: 102 G: 051 B: 153	6633CC R: 102 G: 051 B: 204
6600CC R: 102 G: 000 B: 204	9966FF R: 153 G: 102 B: 255	330066 R: 051 G: 000 B: 102	6600FF R: 102 G: 000 B: 255	6633FF R: 102 G: 051 B: 255	CCCCFF R: 204 G: 204 B: 255	9999FF R: 153 G: 153 B: 255	9999CC R: 153 G: 153 B: 204	6666CC R: 102 G: 102 B: 204	6666FF R: 102 G: 102 B: 255	666699 R: 102 G: 102 B: 153	333366 R: 051 G: 051 B: 102	333399 R: 051 G: 051 B: 153	330099 R: 051 G: 000 B: 153	3300CC R: 051 G: 000 B: 204	3300FF R: 051 G: 000 B: 255
3333FF R: 051 G: 051 B: 255	3333CC R: 051 G: 051 B: 204	0066FF R: 000 G: 102 B: 255	0033FF R: 000 G: 051 B: 255	3366FF R: 051 G: 102 B: 255	3366CC R: 051 G: 102 B: 204	000066 R: 000 G: 000 B: 102	000033 R: 000 G: 000 B: 051	000099 R: 000 G: 000 B: 153	000099 R: 000 G: 000 B: 153	0033CC R: 000 G: 051 B: 204	000066 R: 000 G: 000 B: 102	336699 R: 051 G: 102 B: 153	0066CC R: 000 G: 102 B: 204	99CCFF R: 153 G: 204 B: 255	6699FF R: 102 G: 153 B: 255
003366 R: 102 G: 051 B: 102	6699CC R: 102 G: 153 B: 204	006699 R: 000 G: 102 B: 153	3399CC R: 051 G: 153 B: 204	0099CC R: 000 G: 153 B: 204	66CCFF R: 102 G: 204 B: 255	3399FF R: 051 G: 153 B: 255	003399 R: 000 G: 051 B: 153	0099FF R: 000 G: 153 B: 255	33CCFF R: 051 G: 204 B: 255	00CCFF R: 000 G: 204 B: 255	99FFFF R: 153 G: 255 B: 255	66FFFF R: 102 G: 255 B: 255	33FFFF R: 051 G: 255 B: 255	00FFFF R: 000 G: 255 B: 255	00CCCC R: 000 G: 204 B: 204
009999 R: 000 G: 153 B: 153	6699CC R: 102 G: 153 B: 204	99CCFF R: 153 G: 204 B: 255	33CCFF R: 051 G: 204 B: 255	33CCCC R: 051 G: 204 B: 204	66CCCC R: 102 G: 204 B: 204	339999 R: 051 G: 153 B: 153	336666 R: 051 G: 102 B: 102	006666 R: 000 G: 102 B: 102	003333 R: 000 G: 051 B: 051	00FFCC R: 000 G: 255 B: 204	33FFCC R: 051 G: 255 B: 204	33CC99 R: 051 G: 204 B: 153	00CC99 R: 000 G: 204 B: 153	66FFCC R: 102 G: 204 B: 204	99FFCC R: 153 G: 255 B: 204
00FF99 R: 000 G: 255 B: 153	339966 R: 051 G: 153 B: 102	006633 R: 000 G: 102 B: 051	336633 R: 051 G: 102 B: 051	669966 R: 102 G: 153 B: 102	66CC66 R: 102 G: 204 B: 102	99FF99 R: 153 G: 255 B: 153	66FF66 R: 102 G: 255 B: 102	339933 R: 051 G: 153 B: 051	99CC99 R: 153 G: 204 B: 153	66FF99 R: 102 G: 255 B: 153	33FF99 R: 051 G: 255 B: 153	33CC66 R: 051 G: 204 B: 102	00CC66 R: 000 G: 204 B: 102	66CC99 R: 102 G: 204 B: 153	009966 R: 000 G: 153 B: 102
009933 R: 000 G: 153 B: 051	33FF66 R: 051 G: 255 B: 102	00FF66 R: 000 G: 255 B: 102	CCFFCC R: 204 G: 255 B: 204	CCFF99 R: 204 G: 255 B: 153	99FF66 R: 153 G: 255 B: 102	99FF33 R: 153 G: 255 B: 051	00FF33 R: 000 G: 255 B: 051	33FF33 R: 051 G: 255 B: 051	00CC33 R: 000 G: 204 B: 051	33CC33 R: 051 G: 204 B: 051	66FF33 R: 102 G: 255 B: 051	00FF00 R: 000 G: 255 B: 000	66CC33 R: 102 G: 204 B: 051	006600 R: 000 G: 102 B: 000	003300 R: 000 G: 102 B: 000
009900 R: 000 G: 153 B: 000	33FF00 R: 051 G: 255 B: 000	66FF00 R: 102 G: 255 B: 000	99FF00 R: 153 G: 255 B: 000	66CC00 R: 102 G: 204 B: 000	00CC00 R: 000 G: 204 B: 000	33CC00 R: 051 G: 204 B: 000	339900 R: 051 G: 153 B: 000	99CC66 R: 153 G: 204 B: 102	669933 R: 102 G: 153 B: 051	99CC33 R: 153 G: 204 B: 051	336600 R: 051 G: 102 B: 000	669900 R: 102 G: 153 B: 000	99CC00 R: 153 G: 204 B: 000	CCFF66 R: 204 G: 255 B: 102	CCFF33 R: 204 G: 255 B: 051
CCFF00 R: 204 G: 255 B: 000	999900 R: 153 G: 153 B: 000	CCCC00 R: 204 G: 204 B: 000	CCCC33 R: 204 G: 204 B: 051	333300 R: 051 G: 051 B: 000	666600 R: 102 G: 102 B: 000	999933 R: 153 G: 153 B: 051	CCCC66 R: 204 G: 204 B: 102	666633 R: 102 G: 102 B: 051	999966 R: 153 G: 153 B: 102	CCCC99 R: 204 G: 204 B: 153	FFFFCC R: 255 G: 255 B: 204	FFFF99 R: 255 G: 255 B: 153	FFFF66 R: 255 G: 255 B: 102	FFFF33 R: 255 G: 255 B: 051	FFFF00 R: 255 G: 255 B: 000
FFCC00 R: 255 G: 204 B: 000	FFCC66 R: 255 G: 204 B: 102	FFCC33 R: 255 G: 204 B: 051	CC9933 R: 204 G: 153 B: 051	996600 R: 153 G: 102 B: 000	CC9900 R: 204 G: 153 B: 000	FF9900 R: 255 G: 153 B: 000	CC6600 R: 204 G: 102 B: 000	993300 R: 153 G: 051 B: 000	CC6633 R: 204 G: 102 B: 051	663300 R: 102 G: 051 B: 000	FF9966 R: 255 G: 153 B: 102	FF6633 R: 255 G: 102 B: 051	FF9933 R: 255 G: 153 B: 051	FF6600 R: 255 G: 102 B: 000	CC3300 R: 204 G: 051 B: 000
996633 R: 153 G: 102 B: 051	330000 R: 051 G: 000 B: 000	663333 R: 102 G: 051 B: 051	996666 R: 153 G: 102 B: 102	CC9999 R: 204 G: 153 B: 153	993333 R: 153 G: 051 B: 051	CC6666 R: 204 G: 102 B: 102	FFCCCC R: 255 G: 204 B: 204	FF3333 R: 255 G: 051 B: 051	CC3333 R: 204 G: 051 B: 051	FF6666 R: 255 G: 102 B: 102	660000 R: 102 G: 000 B: 000	990000 R: 153 G: 000 B: 000	CC0000 R: 204 G: 000 B: 000	FF0000 R: 255 G: 000 B: 000	FF3300 R: 255 G: 051 B: 000
CC9966 R: 204 G: 153 B: 102	FFCC99 R: 255 G: 204 B: 153	FFFFFF R: 255 G: 255 B: 255	CCCCCC R: 204 G: 204 B: 204	999999 R: 153 G: 153 B: 153	666666 R: 102 G: 102 B: 102	333333 R: 051 G: 051 B: 051	000000 R: 000 G: 000 B: 000								

Imagine RGB reprezentare:

- matrice bidimensională de triplete sau
- vector de 3 matrice bidimensionale



Memorarea tablourilor : înlănțuit (cazul matricilor rare)

Tabloul este folosit ca SD pentru mai multe TAD-uri:

Containere;

Vectori; Matrice;

Liste;

Dicționare

Exemplu:

Matrice de numere complexe

```
typedef struct {      /*definim tipul elementului */
    double real;
    double imag;
} complex;
```

```
/*definim unele operatii*/
```

```
complex cadd ( complex z1 , complex z2 )
{
    complex z;
    z.real = z1.real + z2.real;
    z.imag = z1.imag + z2.imag;
    return z;
}
```

```
complex cmul ( complex z1 , complex z2 )
{
    complex z;
    z.real = z1.real * z2.real - z1.imag * z2.imag;
    z.imag = z1.real * z2.imag + z1.imag * z2.real;
    return z;
}
```

```
complex conj ( complex z1 )
```

```
{
```

```
    complex z;
```

```
    z.real = z1.real;
```

```
    z.imag = -z1.imag;
```

```
    return z;
```

```
}
```

```
void cprn ( complex z )
```

```
{
```

```
    printf("(%lf) + i(%lf)", z.real, z.imag);
```

```
}
```

```
#define MAXROW 10
#define MAXCOL 15

typedef struct    /*definim matricea*/
{
int rowdim;      /* nr. linii*/
int coldim;      /* nr. coloane*/
complex entry[MAXROW][MAXCOL];
} matrix;
```

```

matrix msetid ( int n )    /* matricea unitate*/
{
matrix C;
int i, j;
if ((n > MAXROW) || (n > MAXCOL)) {
    fprintf(stderr, "msetid: Matrice prea mare\n");
    C.rowdim = C.coldim = 0;
    return C;
}
C.rowdim = C.coldim = n;
for (i = 0; i < C.rowdim; ++i) {
    for (j = 0; j < C.coldim; ++j) {
        A.entry[i][j].real = (i == j) ? 1 : 0;
        A.entry[i][j].imag = 0;
    }
}
return C;
}

```

```

matrix madd ( matrix A , matrix B )    /* adunarea a 2 matrice*/
{
  matrix C;
  int i, j;

  if ((A.rowdim != B.rowdim) || (A.coldim != B.coldim)) {
    fprintf(stderr, "madd: Matrice de dimensiuni incompatibile\n");
    C.rowdim = C.coldim = 0;
    return C;
  }
  C.rowdim = A.rowdim;
  C.coldim = A.coldim;
  for (i = 0; i < C.rowdim; ++i)
    for (j = 0; j < C.coldim; ++j)
      C.entry[i][j] = cadd(A.entry[i][j],B.entry[i][j]);
  return C;
}

```

matrix **mmul** (matrix A , matrix B) */*inmultirea a 2 matrice*/*

```
{  
  matrix C;  
  int i, j, k;  
  complex z;  
  
  if (A.coldim != B.rowdim) {  
    fprintf(stderr, "mmul: Matrice de dimensiuni incompatibile\n");  
    C.rowdim = C.coldim = 0;  
    return C;  
  }  
  C.rowdim = A.rowdim;  
  C.coldim = B.coldim;  
  for (i = 0; i < A.rowdim; ++i) {  
    for (j = 0; j < B.coldim; ++j) {  
      C.entry[i][j].real = 0;  
      C.entry[i][j].imag = 0;  
      for (k = 0; k < A.coldim; ++k) {  
        z = cmul(A.entry[i][k], B.entry[k][j]);  
        C.entry[i][j] = cadd(C.entry[i][j],z);  
      }  
    }  
  }  
  return C;  
}
```

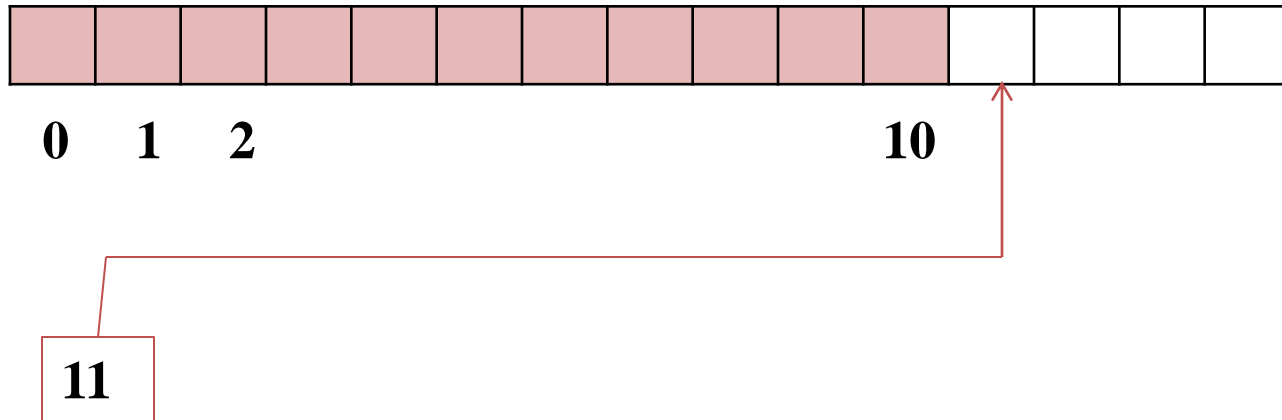


O impementare completa a unuei matrice vezi :

Liviu Negrescu – Limbajele C si C++ pentru incepatori ; Editura Albastra

<http://cse.iitkgp.ac.in/~pds/notes/swf/list1.html>

Vector dinamic



Constă:

- dintr-un vector (normal) care nu este complet ocupat cu elemente
- dintr-un indice care indică prima poziție neocupată (primul element liber al vectorului)

Atât timp cât noul element mai încapă în vector el va fi inserat =>
inserarea unui element: $O(1)$

Dacă noul element nu mai încapă în câmp (vector) :

- Se generează un nou vector mai mare;
- Se copiază elementele existente în noul vector;
- Se adaugă la sfârșit noul element

Practic:

de câte ori dimensiunea devine insuficientă se generează un vector de dimensiune dublă și se copiază elementele existente.

În realitate factorul folosit de diferitele limbaje este:

Sun-Java: 1,5

Gnu-Java: 2

C# (Mono): 2

Python: 1,125

Precondiție:

Afirmație care trebuie să fie adevărată înainte de apelul operației (sarcina utilizatorului)

Postcondiție

Afirmație care trebuie să fie adevărată în urma efectuării operației
=> formează baza argumentării corectitudinii TAD-ului

TAD Vector Dinamic

```
 $\beta = 2$  //factorul de creștere  
 $\alpha = 4$  //consum max suplimentar de memorie  
 $w = 1$  //dimensiunea curentă a câmpului  
 $n = 0$  //numărul curent de elemente  
 $v = \text{new TE}[w]$  //vector static
```



Dăm unele operații pentru TAD vector dinamic

Procedure Creare (V,n,TE) $O(n)$

Crează un vector V de n elements de tip TE

Pre : n, TE

Post: $V \in D$

Procedure Set(V,i,e) $O(1)$

Setează e in poziția i în vectorul V; //de obicei cu []

Pre : $V \in D, i < n, e \in TE$

Post: $V[i]=e$

Function Get(V,i) $O(1)$

Returnează valoarea elementului de pe poziția i din vectorul v.

Pre : $V \in D, i < n,$

Post: $Get = V[i]$

Size (dimensiune curentă)

```
set(int i, TE& e) {  
    assert(0 ≤ i < n);  
    v[i] = e;  
}
```

```
TE get(int i) {  
    assert(0 ≤ i < n);  
    return v[i];  
}
```

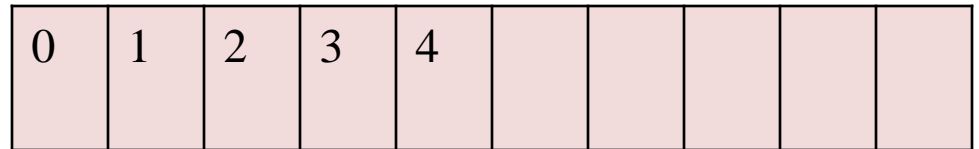
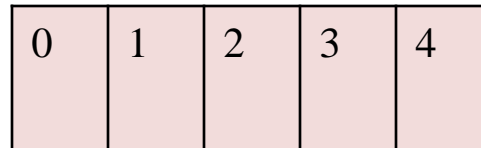
```
int size() {  
    return n;  
}
```

```

void insert (TE e) {
if (n==w)
    reallocate( $\beta$ *n);
v[n]=e;
n++;
}

```

n=5, w=5



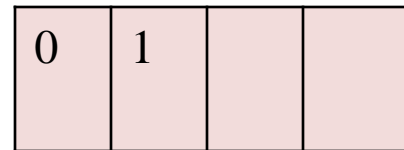
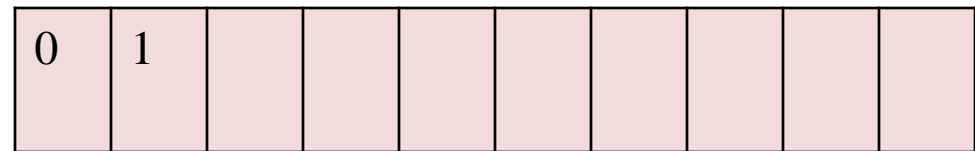
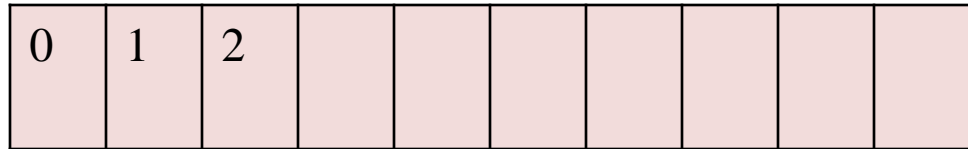
n=6, w=10

```


void remove () {
  assert(n>0);
  n--;
  if ( $\alpha * n \leq w$  and  $n > 0$ )
    reallocate( $\beta * n$ );
}

```

$n=3, w=10$



$n=2, w=4$



```
void realloc (int new_w) {  
w = new_w;  
TE[ ] new_v = new TE[new_w];  
for (i=0; i<n; i++)  
    new_v[i] = v[i];  
v = new_b;  
}
```




Foarte curând oamenii se vor împărți în două categorii: oameni bătrâni și oameni care știu să lucreze la calculator.

(Grigore Moisil)

Grigore Constantin Moisil

(n. [10 ianuarie 1906](#), [Tulcea](#) -

d. [21 mai 1973](#), [Ottawa](#),

[Canada](#)) a fost un

[matematician român](#),

considerat părintele

[informaticii](#) românești cu

[invenția](#) de circuite [electronice](#)

tristabile