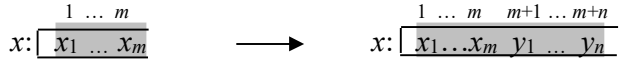
ALTE EXEMPLE DE PROGRAME CU LISTE IMPLEMENTATE STATIC (VECTORI)

R1.

Enunţul problemei: **Să se determine un algoritm prin care să se adauge elementele unui vector la sfârşitul altui vector.** De exemplu, pentru x = (1 2 3 4) şi y=(5 7 9), după adăugarea elementelor vectorului y la sfârşitul vectorului x, se obţine x = (1 2 3 4 5 7 9).

Metoda de rezolvare: Considerăm primul vector (x_1,…, x_m) şi al doilea vector (y_1,…, y_n) – acestea sunt datele de intrare. Adăugând elementele vectorului y la sfârşitul vectorului x se obţine:



Pentru aceasta: $x_{m+1} \leftarrow y_1$ … $x_{m+n} \leftarrow y_n$

$\Leftrightarrow$ $x_{m+i} \leftarrow y_i$, $i = 1,2,\ldots,n$.

Descrierea algoritmului în pseudocod:

```
citeşte m,x1,…,xm,n,y1,…,yn
pentru i = 1,n,1 repeta
        xm+i←yi
scrie x1,…,xm+n
```

Descrierea algoritmului în C++:

```
#include<iostream>
#define NMAX 50
using namespace std;
int n,m;
float x[NMAX],y[NMAX];

void CitireVector(float x[NMAX], int &n){
//se citeste dimensiunea unui vector si elementele sale
 cout<<"Dati dim vect: "; cin>>n;
 cout<<"Dati elementele: ";
 for (int i=0;i<n;i++) cin>>x[i];
}
void AfisareVector(float x[NMAX], int n) {
//afisarea elementelor unui vector
 for (int i=0;i<n;i++) cout<<x[i]<<" ";
}
void AdaugareSfarsit() {
 for (int i=0;i<n;i++) x[m+i] = y[i];
}
int main() {
 cout<<"Primul vector."<<endl;
 CitireVector(x,m);
 cout<<"Al doilea vector."<<endl;
 CitireVector(y,n);
 AdaugareSfarsit();
 cout<<"Dupa adaugarea elem celui de-al doilea vector la
sfarsitul primului vector:"<<endl;
 AfisareVector(x,n+m);
 return 0;
}
```

Rulare:

```
      Primul vector.
      Dati dim vect: 2
      Dati elementele: 5 5
      Al doilea vector.
      Dati dim vect: 3
      Dati elementele: 7 8 9
      Dupa adaugarea elem celui de-al doilea vector la sfarsitul
primului vector:
      5 5 7 8 9
```

R2.

Enunţul problemei: **Să se determine un algoritm pentru adaugarea elementelor unui vector la începutul altui vector.** De exemplu, pentru x = (1 2 3 4) şi y = (5 7 9), după adăugarea elementelor vectorului y la începutul vectorului x, se obţine x = (5 7 9 1 2 3 4).

Metoda de rezolvare: Considerăm primul vector $x_1,\ldots, x_m$ şi al doilea vector $y_1,\ldots, y_n$ – acestea sunt datele de intrare. Adăugând elementele vectorului y la începutul vectorului x se obţine:

$$x: \overbrace{x_1 \ \ldots \ x_m}^{1 \ \ldots \ m} \longrightarrow x: \overbrace{y_1 \ \ldots \ y_n \ x_1 \ldots x_m}^{1 \ \ldots \ n \quad n+1 \ \ldots \ n+m}$$

Dar pentru inserarea celor n elemente din y la început, trebuie să "facem loc" pentru n componente, adică să deplasăm elementele din x spre dreapta (de la ultimul din grup la primul), nu cu câte o poziţie, ci cu câte n poziţii spre dreapta, adică

$$\begin{vmatrix} x_{n+m} \leftarrow x_m \\ \ldots \\ x_{n+1} \leftarrow x_1 \end{vmatrix}$$

$\Leftrightarrow \ x_{n+i} \leftarrow x_i$, $i = m, m\text{-}1\ldots,1$

Apoi inserăm la început elementele vectorului y:

$$\begin{vmatrix} x_1 \leftarrow y_1 \\ \ldots \\ x_n \leftarrow y_n \end{vmatrix}$$

$\Leftrightarrow \ x_i \leftarrow y_i$, $i = 1,2,\ldots, n$.

Descrierea algoritmului în pseudocod:

```
citeşte m,x1,…,xm,n,y1,…,yn
pentru i = m,1,-1 repeta    *deplasarea elem din x
      xn+i←xi
pentru i = 1,n,1 repeta     *inserarea elem din y
      xi←yi
scrie x1,…,xm+n
```

Descrierea algoritmului în C++:

```
#include<iostream>
#define NMAX 50
using namespace std;
int n,m;
float x[NMAX],y[NMAX];

void CitireVector(float x[NMAX], int &n) {
//se citeste dimensiunea unui vector si elementele sale
 cout<<"Dati dim vect: "; cin>>n;
 cout<<"Dati elementele: ";
 for (int i=0;i<n;i++) cin>>x[i];
}
```

```
void AfisareVector(float x[NMAX], int n) {
//afisarea elementelor unui vector
 for (int i=0;i<n;i++) cout<<x[i]<<" ";
}
void AdaugareInceput() {
//deplasarea elem din x cu n pozitii spre dreapta, primul
//elem.ce se "deplaseaza" fiind cel mai din dreapta vectorului
 int i;
 for (i=m-1;i>=0;i--) x[n+i]=x[i];
                  //elem de pe poz.i se copiaza pe pozitia n+i
 //inseram elem din y
 for (i=0;i<n;i++) x[i] = y[i];
}
int main() {
 cout<<"Primul vector."<<endl;  CitireVector(x,m);
 cout<<"Al doilea vector."<<endl;  CitireVector(y,n);
 AdaugareInceput();
 cout<<"Dupa adaugarea elem celui de-al doilea vector la
inceputul primului vector:"<<endl;
 AfisareVector(x,n+m);
 return 0;
}
```

Rulare:

```
Primul vector.
Dati dim vect: 2
Dati elementele: 5 5
Al doilea vector.
Dati dim vect: 3
Dati elementele: 7 8 9
Dupa adaugarea elem celui de-al doilea vector la sfarsitul
primului vector:
7 8 9 5 5
```

R3.

Enunţul problemei: **Să se determine un algoritm pentru deplasarea elementelor unui vector cu o poziţie spre dreapta, ultimul element devenind primul (~permutare circulară).** De exemplu, pentru x = (1 2 3 4), după deplasarea elementelor vectorului x cu o poziţie spre dreapta se obţine x = (4 1 2 3).

Metoda de rezolvare: Ideea constă în a reţine ultimul element într-o variabilă suplimentară, apoi penultimul element se "deplasează" spre dreapta cu o poziţie, ş.a.m.d., primul element se "deplasează" spre dreapta cu o poziţie şi la final, pe poziţia "rămasă liberă" se pune valoarea reţinută în variabila suplimentară.

Descrierea algoritmului în pseudocod:

```
citeşte n,x1,…,xn
a ← xn      *se retine valoarea ultimului elem intr-o var suplim
pentru i = n-1,1,-1 repeta *deplasam spre dreapta elem 1..n-1
   xi+1 ← xi       *deplasarea se incepe de la penultimul element
x1 ← a      *pe prima poz pun ce retinusem in var suplim
```

Descrierea algoritmului în C++:

```
#include<iostream>
#define NMAX 50
using namespace std;
int n;
float x[NMAX];

void CitireVector(float x[NMAX], int &n) {
```

```
//se citeste dimensiunea unui vector si elementele sale
 cout<<"Dati dim vect: "; cin>>n;
 cout<<"Dati elementele: ";
 for (int i=0;i<n;i++) cin>>x[i];
}
void AfisareVector(float x[NMAX], int n) {
//afisarea elementelor unui vector
 for (int i=0;i<n;i++) cout<<x[i]<<" ";
}
void DeplasareSpreDreapta() {
 float a = x[n-1];
 for (int i=n-2;i>=0;i--) x[i+1]=x[i];
                    //elem. de pe poz.i se copiaza pe pozitia i+1
 x[0] = a;
}
int main()
{
 CitireVector(x,n);
 DeplasareSpreDreapta();
 cout<<"Dupa deplasarea elementelor spre dreapta: "<<endl;
 AfisareVector(x,n);
}
```

R4.

Enunţul problemei: **Să se determine un algoritm pentru eliminarea elementelor impare ale unui vector.** De exemplu, pentru x = (1 2 3 4), după eliminarea elementelor impare se obţine x = (2 4), iar pentru vectorul x = (1 3 3 5), după eliminarea elementelor impare se obţine vectorul vid.

Metoda de rezolvare: Se parcurge vectorul $x_1,\ldots, x_n$ şi dacă elementul curent (de pe poziţia i) are valoare impară atunci acesta se elimină din vector (prin deplasarea elementelor de pe pozitia i+1 până la n cu o poziţie spre stânga, descrescând apoi numărul efectiv de elemente din vector, $n \leftarrow n$-1).

Descrierea algoritmului în pseudocod:

```
citeşte n,x1,…,xn
pentru i = 1,n,1 repeta *parcurgem vectorul
   daca x[i]%2 = 1 atunci *val curenta este impara si-o elim
      pentru j = i+1,n,1 repeta
          xj-1 ← xj
       n ← n-1
       i← i-1 *ramanem pe loc pt ca pe poz i a venit un nou elem
daca n=0 atunci
      scrie "vector vid"
altfel
      scrie x1,…,xn
```

Descrierea algoritmului în C++:

```
#include<iostream>
#define NMAX 50
using namespace std;
int n;
int x[NMAX];

void CitireVector(int x[NMAX], int &n){
//se citeste dimensiunea unui vector si elementele sale
 cout<<"Dati dim vect: "; cin>>n;
 cout<<"Dati elementele: ";
 for (int i=0;i<n;i++) cin>>x[i];
```

```
}
void AfisareVector(int x[NMAX], int n) {
 if (n) //sau if(n!=0)
 {
  cout<<"Elementele vectorului: ";
  for (int i=0;i<n;i++) cout<<x[i]<<" ";
 }
 else cout<<"Vectorul este vid";
}
void EliminarePoz(int i0, int x[NMAX], int &n) {
 //elimin elem de pe poz i din vect x de dim n (dim se modif)
 //prin depl. elementelor i+1...n-1 cu o pozitie spre stanga
 for (int i=i0+1;i<n;i++) x[i-1]=x[i];
 n--; //numarul efectiv de elemente scade cu o unitate
}
void EliminareValImpare(int x[NMAX], int &n) {
 for (int i=0;i<n;i++)
  if (x[i]%2==1) //sau if (x[i]%2)
    {
     EliminarePoz(i,x,n);
     i--; //ramanem pe loc caci a venit un nou elem pe poz i
     }
}
int main() {
 CitireVector(x,n);
 EliminareValImpare(x,n);
 cout<<"Dupa eliminarea valorilor impare: "<<endl;
 AfisareVector(x,n);
 return 0;
}
```

Rulare:

```
Dati dim vect: 4
Dati elementele: 1 2 3 4
Dupa eliminarea valorilor impare:
Elementele vectorului: 2 4
sau
Dati dim vect: 4
Dati elementele: 1 3 3 5
Dupa eliminarea valorilor impare:
Vectorul este vid
```

O altă variantă a funcției EliminareValImpare este

```
void EliminareValImpare(int x[NMAX], int &n) {
 int i=0;
 while (i<n)
  if (x[i]%2) //sau if (x[i]%2==1)
     EliminarePoz(i,x,n); //si ramanem pe loc
  else
     i++; //mergem la urmatorul element
}
```

R5.

Enunţul problemei: **Să se determine un algoritm pentru eliminarea elementelor de pe poziţii pare ale unui vector.** De exemplu, pentru x = (5 7 2 4), după eliminarea elementelor impare se obţine x = (5 2).

Metoda de rezolvare: Algoritmul pare similar, însă după eliminarea unui element poziţiile elementelor de după el se schimbă faţă de poziţiile iniţiale:

- iniţial:

	1	2	3	4	5	6
x:	5	~~8~~	8	6	7	4

- după eliminarea elementului de pe poziţia a 2-a:

	1	2	3	4	5
x:	5	8	~~6~~	7	4

- la pasul următor trebuie eliminată valoarea 6 care iniţial era pe poziţia 4, acum însă este pe poz 3 şi se obţine:

	1	2	3	4	5
x:	5	8	7	~~4~~	

- la pasul următor trebuie eliminată valoarea 4 care iniţial era pe poziţia 6, acum însă este pe poz 4.

Aşadar, am eliminat în ordine poziţia 2, apoi 3, 4, 5…

Descrierea algoritmului în pseudocod:

```
citeşte n,x1,…,xn
pentru i = 2,n,1 repeta
    pentru j = i+1,n,1 repeta
        xj-1 ← xj
    n ← n-1
daca n=0 atunci
        scrie "vector vid"
altfel
        scrie x1,…,xn
```

Descrierea algoritmului în C++:

```
#include<iostream>
#define NMAX 50
using namespace std;
int n;
int x[NMAX];
void CitireVector(int x[NMAX], int &n){
//se citeste dimensiunea unui vector si elementele sale
 cout<<"Dati dim vect: "; cin>>n;
 cout<<"Dati elementele: ";
 for (int i=0;i<n;i++) cin>>x[i];
}
void AfisareVector(int x[NMAX], int n) {
 if (n!=0) //sau if(n)
 {cout<<"Elementele vectorului: ";
  for (int i=0;i<n;i++) cout<<x[i]<<" ";
 }
 else cout<<"Vectorul este vid";
}
void EliminarePoz(int i0, int x[NMAX], int &n) {
 //elimin elem de pe poz i din vect x de dim n (dim se modif)
 //prin depl elementelor i+1...n-1 cu o pozitie spre stanga
 for (int i=i0+1;i<n;i++) x[i-1]=x[i];
 n--; //numarul efectiv de elemente scade cu o unitate
}
void EliminarePozPare(int x[NMAX], int &n) {
 for (int i=1;i<n;i++) EliminarePoz(i,x,n);
}
int main() {
 CitireVector(x,n);
```

```
 EliminarePozPare(x,n);
 cout<<"Dupa eliminarea pozitiilor pare: "<<endl;
 AfisareVector(x,n);
 return 0;
}
```

Rulare:

```
Dati dim vect: 4
Dati elementele: 1 2 3 4
Dupa eliminarea pozitiilor pare:
Elementele vectorului: 1 3
sau
Dati dim vect: 5
Dati elementele: 1 2 3 4 5
Dupa eliminarea pozitiilor pare:
Vectorul este 1 3 5
```

R6.

Enunţul problemei: **Să se determine un algoritm pentru eliminarea duplicatelor dintr-un vector.** De exemplu, pt x = (5 7 5 9 5 5 7), după eliminarea duplicatelor se obţine x = (5 7 9). x = (**5** 7 5 9 5 5 7) -> x = (5 **7** 9 7) -> x = (5 7 **9**)

Metoda de rezolvare: Se parcurge vectorul şi se verifică dacă valoarea curentă, să zicem x_i, se mai găseşte şi mai departe (pe poziţiile i+1,…,n) – în caz afirmativ se elimină respectiva poziţie.

Descrierea algoritmului în pseudocod:

```
citeşte n,x1,…,xn
pentru i = 1,n-1 repeta *parcurgem elem vectorului fara ultimul
  j←i+1 *incepand cu poz i+1 ma uit dupa dubluri
 cat_timp j≤n repeta
  daca xj = xi atunci *am dublura => o elimin
   pentru k=j+1,n repeta *deplasez blocul j+1..n spre st
     xk-1 ← xk
   n←n-1 *nr de elem scade
     *si raman pe poz j (a venit aici alt elem)
  altfel j←j+1 *n-a fost dublura deci trec la urmatorul j
scrie x1,…,xn
```

Descrierea algoritmului în C++:

```
#include<iostream>
#define NMAX 50
using namespace std;
int n,x[NMAX];
void CitireVector(int x[NMAX], int &n){
//se citeste dimensiunea unui vector si elementele sale
 cout<<"Dati dim vect: "; cin>>n;
 cout<<"Dati elementele: ";
 for (int i=0;i<n;i++) cin>>x[i];
}
void AfisareVector(int x[NMAX], int n) {
 cout<<"Elementele vectorului: ";
  for (int i=0;i<n;i++) cout<<x[i]<<" ";
}
void EliminarePoz(int i0, int x[NMAX], int &n) {
//elimin.elem de pe poz i din vect x de dim n (dim se modifica)
 //prin deplasarea elementelor i+1...n-1 cu o pozitie spre st.
 for (int i=i0+1;i<n;i++) x[i-1]=x[i];
 n--; //numarul efectiv de elemente scade cu o unitate
```

```
}
void EliminareDuplicate(int x[NMAX], int &n) {
 for (int i=0;i<n-1;i++) //pt fiecare element fara ultimul
  for (int j=i+1;j<n;j++) //ma uit dupa el daca gasim val egala
   if (x[j]==x[i])
    {
     EliminarePoz(j,x,n);
     j--;
    }
}
int main() {
 CitireVector(x,n);
 EliminareDuplicate(x,n);
 cout<<"Dupa eliminarea duplicatelor: "<<endl;
 AfisareVector(x,n);
 return 0;
}
```