8. Tablouri şi pointeri.

8.1. Tablouri cu o dimensiune (vectori).

Un tablou cu o singură dimensiune este o succesiune de variabile având toate de acelaşi tip (*tipul de bază al tabloului*), care ocupă o zonă contiguă de memorie. Un tablou are:

- o dimensiune (egală cu numărul de elemente al tabloului)
- un nume (care identifică global tabloul)
- o clasă de alocare
- un tip comun tuturor elementelor tabloului

Dimensiunea tabloului precizează numărul de elemente printr-o constantă întreagă sau printr-o expresie constantă.

La declararea unui tablou se specifică: numele, tipul de bază, clasa de alocare şi dimensiunea.

```
tip nume[dimensiune];
```

sau

```
clasă tip nume[dimensiune];
```

Exemple:

```
int  x[10];          /* tablou de 10 intregi */
char litere[2*26];   /* tablou de 52 caractere */
```

Tipul elementelor tabloului poate fi un tip fundamental, enumerat, inregistrare, pointer sau un tip definit.

Numele tabloului este adresa primului element din tablou (de exemplu **x** este adresa primului element, adică **@x[0]**). Aceasta explică de ce nu este permisă o atribuire între două tablouri.

Accesul la un element din tablou se face printr-o *variabilă indexată*, formată din numele tabloului şi un *index* - o expresie cuprinsă între **0** şi **dimensiune−1**.
Primul element va fi desemnat aşadar prin **x[0]**, al doilea element prin **x[1]**,al **N**-lea prin **x[N−1]**

Un tablou declarat în interiorul unei funcţii are implicit clasa **auto**, în timp ce tablourile declarate în exteriorul tuturor funcţiilor au în mod implicit clasa **extern**.
Un tablou declarat în exteriorul tuturor funcţiilor cu specificatorul **static** este alocat la adrese fixe, fiind vizibil numai *în fişierul* în care este declarat.
Un tablou declarat în interiorul unei funcţii cu specificatorul **static** este alocat la adrese fixe, fiind vizibil numai *în interiorul funcţiei*.

Prelucrările pe tablouri se implementează cu cicluri **for**.

Exemplul 15: ***Să se afişeze elementele unui tablou citit de la intrarea standard, câte 10 pe un rând.***

```
#include <stdio.h>
int main()
/* citirea si afisarea elementelor unui vector */
{ int x[10], n, j;
  scanf("%d", &n);
  for (j=0; j < n; j++)
    scanf("%d", &x[j]);
  for (j=0; j < n; j++)
```

```
    printf("%5d%c", x[j],(j%10==9||j==n-1)?'\n':' ');
  return 0;
}
```

La declararea unui tablou, acesta poate fi şi iniţializat, dacă declaraţia este urmată de semnul = şi de o listă de valori iniţiale, separate prin virgule şi incluse între acolade.
Exemple:

```
int prime[5]={2,3,5,7,11};
char vocale[5]={'a','e','i','o','u'};
```

La declararea unui tablou iniţializat se poate omite dimensionarea, situaţie în care se ia ca dimensiune numărul de valori iniţiale:

```
char operator[]={'+','-','*','/'};
long x[]={1,10,100,1000,10000,100000};
```

Tablourile vor avea 4, respectiv 6 elemente.

Exemplul 16: ***Scrieţi un program care converteşte un şir de caractere reprezentând un număr scris cu cifre romane în corespondentul său cu cifre arabe.***

Notaţia cu cifre romane este un sistem nepoziţional, care foloseşte cifrele: **M, D, C, L, X, V, I,** având respectiv valorile: **1000, 500, 100,50, 10, 5, 1.**

Pentru a obţine valoarea numărului, se pleacă cu acesta de la 0 şi se adaugă pe rând contribuţiile cifrelor astfel: dacă valoarea cifrei romane curente este mai mare sau egală cu cifra care urmează, atunci valoarea cifrei curente se adaugă la valoarea numărului arab, altfel se scade din acesta. De exemplu numărul roman **MCMXCVIII** are ca valoare pe **1998**

Tabel 8.1. Transformarea unui număr din cifre romane în notaţie arabă

Cifra curentă	Cifra următoare	Relaţia dintre ele	Contribuţia cifrei crte	Valoare număr
M	**C**	**>**	**+1000**	**1000**
C	**M**	**<**	**−100**	**900**
M	**X**	**>**	**+1000**	**1900**
X	**C**	**<**	**−10**	**1890**
C	**V**	**>**	**+100**	**1990**
V	**I**	**>**	**+5**	**1995**
I	**I**	**=**	**+1**	**1996**
I	**I**	**=**	**+1**	**1997**
I		**>**	**+1**	**1998**

Se observă că pentru a considera şi contribuţia ultimei cifre a numărului am fost nevoiţi să "prelungim" numărul cu caracterul spaţiu liber, căruia i-am asociat valoarea 0. Pentru stabilirea corespondenţei cifră romană – valoare asociată, vom defini o funcţie **int conv(char)** care foloseşte două tablouri: **roman** şi **arab**, iniţializate respectiv cu caracterele reprezentând cifrele romane şi cu valorile acestora, definite ca externe. Numărul roman **nrom**, citit de la intrarea standard va fi terminat prin spaţiu liber.

```
#define LMAX 15
#include <stdio.h>
char roman[]="MDCLXVI ";
int arab[]={1000,500,100,50,10,5,1,0};
```

```
int conv(char);
int main(){
   char nrom[LMAX];
   int i,n=0; //n = lungimea numarului scris cu cifre romane
   int narab=0;
   int crt,urm;
   while((nrom[n++]=getchar())!=' `)
                  ;
   n--;
   for (i=0; i<n-1; i++){
      crt=conv(nrom[i]);
      urm=conv(nrom[i+1]);
      if(crt>=urm)
         narab+=crt;
      else
         narab-=crt;
  }
  for (i=0;i<n;i++)
    printf("%c",nrom[i]);
  printf("=%d\n",narab);
  return 0;
}
int conv(char c){
   int j=0;
   while(roman[j++]!=c && j<8)
           ;
   if(j<8)
      return arab[--j];
   else
      return -1;
}
```

8.2. Probleme rezolvate.

1. Definiţi o funcţie care determină poziţia elementului minim al unui tablou cu elemente reale.

```
int pmin(int n, double x[]){
   int i, pm=0;
   for(i=1; i<n; i++)
      if(x[i]<x[pm])
         pm = i;
   return pm;
}
```

2. Definiţi o funcţie care întoarce poziţia în care se află o valoare întreagă dată **y** (cheie), într-un tablou **x** nesortat cu **n** elemente întregi. Se va efectua o căutare secvenţială. Dacă valoarea căutată nu se află în tablou funcţia întoarce -1.

```
int cautsecv(int n, int x[], int y){
   int i;
   for(i=0; i<n; i++)
```

```
      if(x[i]==y) return i;
   return -1;
}
```

3. Definiţi o funcţie care întoarce poziţia în care se află o valoare întreagă dată **y** (cheie), într-un tablou **x** sortat cu **n** elemente întregi. Se va efectua o căutare binară. Dacă valoarea căutată nu se află în tablou funcţia întoarce poziţia pe care ar trebui să o ocupe cheia în tabloul sortat, precedată de semnul - .

- se compară valoarea căutată **y** cu elementul din mijlocul tabloului **x[m]**
- dacă sunt egale, elementul **y** a fost găsit în poziţia **m**
- în caz contrar se continuă căutarea într-una din jumătăţile tabloului (în prima jumătate, dacă **y<x[m]** sau în a doua jumătate dacă **y>x[m]**).

Funcţia întoarce poziţia **m** a valorii **y** în **x** sau **-i**, dacă **y** nu se află în **x, i** fiind poziţia unde ar trebui să se afle y. Complexitatea este $O(\log_2 n)$

```
int CB(int i, int j, int y, int x[]){
   int m;
   while(i <= j){
      m = (i+j)/2;
      if(y == x[m]) return m;
      if(y < x[m])
        j = m-1;
      else
        i = m+1;
   return -i;
}

int CautBin(int n, int x[], int y){
  if(n==0) return 0;
  return CB(0, n-1, y, x);
}
```

4. Definiţi funcţii care realizează operaţiile de adunare, înmulţire şi împărţire între polinoame. Un polinom va fi precizat prin gradul său şi un tablou al coeficienţilor, daţi în ordine descrescătoare a puterilor. Dacă gradul polinomului este **n**, atunci tabloul coeficienţilor va avea **n+1** elemente, chiar dacă unele din ele sunt nule.
Polinomul sumă va avea gradul **max(na, nb)**. Ultimii **min(na, nb)** coeficienţi ai polinomului sumă sunt de forma **a[i]+b[i]**, iar primii coeficienţi vor fi termenii corespunzători din **a**, dacă acesta are gradul mai mare sau din **b**, în caz contrar.

```
void adpol(int na, int a[], int nb, int b[], int c[]){
   int i, max, min;
   if(na>nb){
      max = na;
      min = na-nb;
      for(i=0; i<min; i++)
         c[i] = a[i];
   }
   else {
      max = nb;
      min = nb-na;
```

```
      for(i=0; i<min; i++)
         c[i] = b[i];
   }
   for(i=min; i<=max; i++)
      c[i] = a[i] + b[i];
}
```

Gradul polinomului produs va fi **na+nb**. Coeficienţii c_k, cu **k=0:na+nb** se calculează ca:

$$c_k = \sum_{i+j=k} a_i b_j = \sum_{\substack{k-i\geq 0,\\ k-i\leq nb}} a_i b_{k-i} = \sum_{i=k}^{nb-k} a_i b_{k-i}$$

Ceea ce se traduce prin:

```
for(k=0;k<=na+nb;k++){
   c[k] = 0;
   for(i=k; i<=nb-k; i++)
      c[k]+=a[i]*b[k-i];
```

Este posibil să folosim numai prima parte a relaţiei de mai sus, situaţie în care la fiecare iteraţie se adaugă un produs $a_i b_j$ de fiecare dată la alt termen c_{i+j}.

```
void mulpol(int na, int a[], int nb, int b[], int c[]){
   int i,j;
   for(i=0; i<=na+nb; i++)
      c[i] = 0;
   for(i=0; i<=na; i++)
      for(j=0; j<=nb; j++)
         c[i+j]+=a[i]*b[j];
}
```

La împărţirea a două polinoame, gradul polinomului cât va fi **na-nb**, iar a polinomului rest **nb-1**. Se simulează împărţirea polinoamelor. În pasul **i**, un termen din cât este: $c_i = a_i/b_0$, iar restul parţial:

$a_{i+j}=a_{i+j}-c_i b_j$

Restul se regăseşte în ultimele nb elemente din deîmpărţit, de unde este transferat în vectorul rest.

```
void divpol(int na, int a[], int nb, int b[], int c[], int r[]){
   int i, j;
   for(i=0; i<=na-nb; i++){
      c[i] = a[i]/b[0];
      for(j=0; j<=nb; j++)
         a[i+j]-=c[i]*b[j];
   }
   for(i=0; i<nb; i++)
      r[i] = a[na-nb+i+1];
}
```

5. Definiţi o funcţie care sortează un tablou **x** cu **n** elemente reale, folosind metoda selecţiei. Tabloul de sortat este partiţionat în două zone: zona din stînga – deja sortată şi zona din dreapta. În zona nesortată se determină poziţia elementului minim. Dacă aceasta nu coincide cu poziţia primului element din zona nesortată, se interschimbă elementele din cele două poziţii şi se extinde zona sortată cu o poziţie spre dreapta.

Fie **k** – începutul zonei nesortate; iniţial zona nesortată cuprinde întreg tabloul, deci **k=0**. În final, zona nesortată nu mai are elemente, deci **k=n**.

```
void sortsel(int n, double x[]){
   int pm, k, j;
   for(k=0; k<n; k++){
   /*restrange zona nesortata*/
   /*determina pozitia pm a elementului minim*/
      pm = k;
      for(j=k+1; j<n; j++)
         if(x[j] < x[pm])
            pm = j;
      if(x[pm]!=x[k]){
         t = x[pm];
         x[pm] = x[k];
         x[k] = t;
      }
}
```

6. Definiţi o funcţie care sortează un tablou **x** cu **n** elemente reale, folosind metoda inserţiei.

Tabloul este partiţionat într-o zonă sortată şi o zonă nesortată încă. Iniţial elementul **x[k], k=1** din zona nesortată este inserat în zona sortată, care se extinde cu o poziţie spre dreapta, păstrând relaţia de ordine în zona sortată.

```
void sortins(int n, double x[]){
   int k, t, p, i;
   for(k=1; k<n; k++){
      t = x[k];
      /*inserare t in zona sortata x[0]:x[k-1] in pozitia p */
      /*determinare pozitie p a primului element x[p]>=t */
      for(p=0; p<k && x[p]<t; p++)
                     ;
      if(x[p]>=t){
         for(j=p; j<k; j++)
            x[j+1] = x[j];
         x[p] = t;
      }
   }
}
```

7. Eventualele rădăcini întregi ale ecuaţiei cu coeficienţi întregi:
$x^n+a_0x^{n-1}+\ldots+a_{n-1}=0$ sunt ± 1, $\pm a_{n-1}$, $\pm d$, în care **d** este un divizor nebanal a lui a_{n-1}
Se citesc **n** şi tabloul **a**. Se cere să se găsească şi să se afişeze rădăcinile întregi simple, duble, etc ale ecuaţiei.

Indicaţie: Se vor defini şi folosi funcţiile:
int ndiv(int n, int *div); -crează un tablou al tuturor divizorilor lui **n** şi a celor cu semn schimbat şi întoarce numărul acestora
int eval(int n, int *a, int x); - stabileşte dacă polinomul are sau nu rădăcina **x** folosind schema lui Horner

`void copy(int *ns, int *as, int n, int *a);` - copiază **`n`** şi tabloul **`a`** în **`ns`** şi **`as`**
`void deriv(int n, int *a, int *nd, int *ad);` - derivează polinomul cu grad **`n`** şi coeficienţii **`a`** şi pune în **`nd`** gradul polinomului derivat şi în **`ad`** coeficienţii acestuia

8. Se consideră fracţia raţională:

$$F(x) = \frac{P_m(x)}{(x-a_0)\cdots(x-a_{n-1})} = \frac{b_0x^m + b_1x^{m-1} + \cdots b_m}{(x-a_0)\cdots(x-a_{n-1})} \text{ cu } m < n < 20.$$

Scrieţi o funcţie cu semnătura:

```
void dezv(int m, int n, int *a, int *b, int *c);
```

care calculează coeficienţii dezvoltării fracţiei raţionale: $F(x) = \frac{c_0}{x-a_0} + \cdots + \frac{c_{n-1}}{x-a_{n-1}}$

Funcţia care evaluează un polinom într-un punct dat **`x`** se consideră cunoscută.

Indicaţie: $c_j = \frac{P_m(a_j)}{\prod_{\substack{k=0,\\ k\neq j}}^{n}(a_j - a_k)}$,

```
void dezv(int m, int n, double *a, double *b, double *c){
   int j, k;
   double p=1.;
   for(j=0; j<n; j++){
      for(k=0; k<=n; k++)
         if(k!=j)
            p*=(a[j]-a[k]);
      c[j] = eval(m, b, a[j])/p;
}
```

8.9. Probleme propuse.(Tablouri cu o dimensiune)

1. Intr-un şir **`x`** cu **`n`** componente reale, să se determine media aritmetică a elementelor pozitive situate între primul element pozitiv şi ultimul element negativ al şirului, exceptând aceste elemente. Cazurile speciale vor fi clarificate prin mesaje corespunzătoare.

2. Intr.un şir **`S`** cu **`n`** elemente întregi (**`n ≤ 100`**) să se determine elementele distincte.

3. Doi vectori **`x`** şi **`y`** au **`n`**, respectiv **`m`** elemente reale distincte (**`m,n≤10`**). Să se creeze un nou vector **`z`** cu elementele comune ale celor doi vectori. (intersecţia elementelor mulţimilor reprezentate de cei doi vectori).

4. Doi vectori **`x`** şi **`y`** au **`n`**, respectiv **`m`** elemente reale distincte (**`m,n≤10`**). Să se creeze un nou vector **`z`** cu conţinând elementele celor doi vectori. Elementele comune din cei doi vectori apar în `z` o singură dată. (reuniunea elementelor mulţimilor reprezentate de cei doi vectori).

5. Se citeşte o valoare întreagă **`n ( 0 < n ≤ 100`**) şi `n` valori reale cu care se crează un vector **`x`**. Scrieţi un program C care calculează şi afişează:

- Valoarea medie

- Abaterea medie pătratică:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n-1} (x_i - xmed)^2}{n(n-1)}}$$

- Numărul de componente care depăşesc valoarea medie
- Să se creeze un vector **y** cu componentele din **x** mai mari decât valoarea medie şi să se afişeze câte 5 elemente pe o linie.

6. Se citesc **n (n ≤ 100)** coordonate reale **x, y** ale unor puncte în plan şi se crează cu acestea două tablouri **x** şi **y**.

- Să se afişeze toate tripletele de puncte coliniare.
- Să se afişeze punctele **i, j, k** pentru care aria triunghiului determinat de aceste puncte este maximă..

Indicaţie: Aria determinată de punctele **i,j,k** este:

$$s = \frac{1}{2}\begin{vmatrix} x_i & x_j & x_k \\ y_i & y_j & y_k \\ 1 & 1 & 1 \end{vmatrix}$$

Dacă punctele sunt coliniare, atunci **S=0**.

7. Să se calculeze coeficienţii binomiali $C_n^1, C_n^2, \ldots, C_n^p$ în care **n** şi **p** sunt întregi pozitivi daţi (**p < n**), cunoscând relaţia de recurenţă:

$$C_n^k = (n-k+1)/k * C_n^{k-1} \qquad \text{pornind cu} \qquad C_n^0 = 1$$

Se vor utiliza tablouri

8. Să se calculeze coeficienţii: $c_0, c_1, \ldots, c_{n-1}$, pentru n dat, ştiind că:

$$C_0/(p+1) + c_1/p + \ldots + c_p/1 = 1 \qquad \text{pentru } p=0,1,2,\ldots,n.$$

9. Să se calculeze coeficienţi polinomului Cebâşev de ordinul **n**, pornind de la relaţia de recurenţă

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \qquad k > 2$$

$$T_0(x) = 1, \quad T_1(x) = x$$

obţinând în prealabil relaţii de recurenţă pentru coeficienţi.

10. Un număr întreg este reprezentat prin cifrele sale **c[0],c[1],... c[n-1],** (**c[0]** fiind cifra cea mai semnificativă).
Să se calculeze câtul **q[0],q[1],... ,q[m-1]** obţinut prin impărţirea numărului dat prin numărul întreg **p**.

11. Dându-se o valoare întreagă **n,** să se genereze reprezentarea fracţiilor zecimale $1/2^k$ unde $k = 1,2, \ldots ,n$.

12. Să se ordoneze crescător şirul **x** cu **n** elemente utilizând observaţia că în şirul ordonat crescător orice subşir **x[i], |x[i+1],...,x[n]** cu **i=0,1,2,...,n−2** are elementul **x[i]** maxim.

13. Să se ordoneze crescător o listă având **n** componente (**n≤100**) de numere intregi, utilizând metoda contorizării inversărilor (denumită şi metoda bulelor)

14. Un număr de bare (**N≤100**) sunt date prin lungimile lor.Se dau de asemenea **P** categorii de lungimi(sau standarde) între care trebuie să se încadreze lungimile pieselor (**P≤10**).
O categorie de lungimi este precizată prin 2 limite:una minimă şi cealaltă maximă; presupunem că aceste categorii de lungimi formează intervale disjuncte.
O piesă **i** se incadrează în categoria de lungimi **j** dacă: **LMIN[j] ≤ L[i] ≤ LMAX[j].**
Să se calculeze şi să se afişeze:

- -numărul de piese din fiecare clasă
- -dimensiunea medie a pieselor din fiecare clasă de lungimi
- -numărul de rebuturi şi lungimile barelor rebutate.

15. Să se calculeze coeficienţii b_i , **i=0..n−1** din dezvoltarea produsului:

$$(x+a_0)(x+a_1)\ldots(x+a_{n-1})=x^n + b_0 x^{n-1} + \ldots + b_{n-1}$$

Se dau **n** şi coeficienţii $a_0, a_1, \ldots a_{n-1}$

16. **N** copii identificaţi prin numerele **1,2,...,N**, joacă următorul joc:se aşează în cerc în ordinea **1,2,..,N** şi începând de la copilul **k** numără de la **1** la **p** eliminând din cerc pe cel care a fost numărat cu **p**;numărătoarea începe de la următorul eliminându-se al **2**-lea ş.a.m.d.
Folosindu-se identificarea iniţială să se stabilească ordinea de ieşire a copiilor din joc.

17. Într-un şir **S** cu **n** elemente să se determine elementele distincte.
Exemplu: în şirul **2 2 5 4 5 1 2** elementele distincte sunt : **2 5 4 1**
Indicaţie

- elementele unice sunt distincte şi se tipăresc
- dintre elementele cu mai multe apariţii se tipăreşte un singur reprezentant. La întâlnirea primei egalităţi **S[i]=S[j]** se abandonează căutarea şi se tipăreşte **S[i]** numai dacă **i<j** (este prima apariţie a lui **S[i]**).

18. De pe mediul de intrare se citeşte un număr întreg **n ≠ 0** urmat de **n** valori reale. Să se introducă aceste valori într-un tablou **x** pe măsura citirii lor, astfel încât tabloul să fie ordonat crescător.

19. Dându-se o valoare **x** şi un tablou a cu **n** elemente, să se separe acest tablou în două partiţii astfel încât elementele din prima partiţie să fie mai mici sau egale cu **x**, iar cele din a doua partiţie strict mai mari decât **x.**

20. Să se determine elementul maxim dintr-un şir cu **n** elemente şi poziţia pe care el apare.În cazul unui maxim cu mai multe apariţii se va nota prima sa apariţie.

21. Se dau două şiruri **x** şi **y** ordonate strict crescător, având **M** şi respectiv **N** elemente. Să se construiască un şir **z** ordonat strict crescător conţinând elementele şirurilor **x** şi **y**.-("interclasare de şiruri").

22. Se consideră un vector **x** cu **n** componente, ordonat strict crescător şi o valoare **y**. Să se insereze această valoare în vectorul **x** astfel încât el să rămână ordonat strict crescător. Se va face o căutare rapidă a lui **y** în şir (căutare binară). Şirul iniţial şi cel rezultat se vor tipări cu câte **5** elemente pe linie.

23. Dintr-un şir dat să se determine lungimea şi poziţia subşirului strict crescător cel mai lung, format din elemente alăturate din şirul dat. Şirul dat se tipăreşte în ecou cu câte 10 elemente pe linie.Subşirul de lungime maximă se afişează cu 5 elemente pe linie.

24. Se spune că şirul **x** cu **k** elemente "intră" în şirul **s** cu **n** elemente **k≤n**, dacă există un subşir contiguu $s_i, s_{I+1}, \ldots, s_{I+k}$ cu proprietatea că elementele lui sunt identice cu elementele şirului **x**. Să se stabilească numărul de "intrări" ale lui **x** în **s** şi poziţiile în **s** ale elementelor de unde începe intrarea.

25. Doi vectori **a** şi **b** au câte **n** componente fiecare, precizate pe mediul de intrare. Să se calculeze unghiul dintre ei exprimat în radiani.

26. Două numere sunt păstrate prin tablourile cifrelor lor. Să se obţină tabloul cifrelor produsului numerelor.

27. Dându-se un număr întreg **n** să se construiască două tablouri, primul conţinând factorii primi ai lui **n**, iar cel de-al doilea multiplicităţile acestora.

28. Să se calculeze cel mai mare divizor comun cmmdc a două numere date **m** şi **n** utilizând următoarea metodă:

- se creează tablouri cu factorii primi ai celor două numere şi multiplicităţile lor
- se selectează în cmmdc factorii primi comuni la puterea cea mai mică

29. Să se calculeze cel mai mare divizor comun a două numere date **m** şi **n** prin următoarea metodă:

- pentru fiecare număr se creează un tablou cu toţi divizorii acestuia, inclusiv divizorii banali;
- se selectează apoi într-un tablou divizorii comuni şi se ia cel mai mare dintre aceştia.