**LUCRAREA DE LABORATOR 3**

**Prelucrarea tablourilor unidimensionale în TP şi C**

*Obiectivele temei*

1. Familiarizarea cu principiile prelucrării elementelor tablourilor unidimensionale.
2. Algoritmizarea şi însuşirea procedeelor stereotipe de declarare, introducere, afişare şi formare prin parcurgere şi calcule ale valorilor elementelor tablourilor unidimensionale.
3. Însuşirea procedeelor avansate de realizare a structurilor ciclice cu instrucţiunile for, while şi repeat, aplicând diverse tehnici de programare.
4. Însuşirea procedeelor de operare în interfaţa TP şi C la nivelul submeniurilor DEBUG şi BREAK/WATCH (mijloacele de depanare-urmărire a variabilelor şi punctele de întrerupere).

*Subiectele temei şi ordinea executării*

1. Studierea principiilor prelucrării (descrierii, declarării, formării, etc.) tablourilor unidimensionale -- variabilelor cu indici şi instrucţiunilor ciclice în C şi recapitularea în TP.
2. Studierea metodelor de introducere şi afişare clară a tablourilor unidimensionale.
3. Însuşirea tehnicilor fundamentale de programare a înmagazinării diferitor valori în baza elementelor: sumei şi produsului; determinării valorilor maxime şi minime; diferitor căutări, rearanjări şi transformări ale elementelor tablourilor.
4. Elaborarea algoritmului şi programului TP şi C pentru rezolvarea problemei (variantele vezi în Anexa L.lab nr.3), asigurând universalitatea.
5. Depanarea programului şi verificarea la PC a problemei trasate cu diverse combinaţii de date.
6. Controlul corectitudinei programului cu ajutorul variantei de testare.
7. Analiza eficienţei programului şi soluţiei problemei trasate.

*Conţinutul raportului*

1. Dările de seamă (rapoartele) la toate lucrările de laborator, începând cu această lucrare, se cere obligator să se includă schemele logice ale algoritmilor cu exemplificarea corectitudinii prin date concrete. Analiza erorilor admise pe parcursul efectuării lucrării şi modalităţile de excludere. Restul vezi cerinţele din lucrarea de laborator nr.1.

***Noţiuni generale***

1. ***Tipul tablou (****array, masiv****).***

Numim tablou o colectie (grup, multime ordonata) de date, de acelasi tip, situate intr-o zona de memorie continua (elementele tabloului se afla la adrese succesive). Tablourile sunt *variabile compuse (structurate)*,  deoarece grupeaza mai multe elemente. Variabilele tablou au nume, iar tipul tabloului este dat de tipul elementelor sale. Elementele tabloului pot fi referite prin numele tabloului si indicii (numere intregi) care reprezinta pozitia elementului in cadrul tabloului. Deci este o metodă de organizare a datelor este - t a b l o u l (tabele) cu şiruri (de o lungime cunoscută) de variabile de acelaşi tip. Structura:

* Ansamblu omogen de variabile numite componentele tabloului
* Toate componentele aparţin aceluiaşi tip
* Componentele sunt identificate cu ajutorul indicilor
  + - Tablouri:
      * Unidimensionale (1 – dimensionale)
      * Bidimensionale (2 – dimensionale), etc.

Un şir de elemente de acelaşi tip se mai numeşte şi vector sau tablou unidimensional. Deci tabloul este un tip de date compus dintr-un număr precizat de date de acelaşi tip. Referirea la elementele tabloului se face prin numele variabilei tablou urmat de indexul elementului pus între paranteze drepte [ ].

***1.1 Tipul tablou şi modurile de declarare în C***

In C, tablourile unidimensionale sunt alcătuite dintr-un grup de elemente de acelaşi tip (numit tip de baza) si referite printr-un nume comun.

Variabilele de tip tablou se definesc in maniera:

**tip\_de\_baza nume\_var [*dimensiune*];**

. Deci tabloul se poate caracteriza prin tip, nume şi dimensiune. Formatul comun de descriere a tablourilor este: ***tip nume[d1][d1]…[dn];***unde :

* ***tip***este tipul comun pentru toate elementele tabloului, adică tipul tabloului de\_baza. Tip al unui tablou poate fi orice tip de date deja definit: întreg, real, caracterial ş.a. *nume* este numele tabloului. In calitate de nume al tabloului este folosit orice identificator. Mai mult ca atât, deoarece numele tabloului este identificator, asupra lui se răspândeşte totul ce-i indicat în compartimentul ”Nume de variabile (identificatori)”, *d1,d2,dn*- dimensiunile tabloului (cifre întregi sau variabile şi atunci trebuie definite înainte de declararea tablourilor).
* *Dimensiunea* tabloului indica numarul de elemente prezente in tablou. Dimensiunea tabloului poate fi o expresie constanta cu rezultat intreg.

Un element al tabloului este accesat folosind ca index poziţia elementului, astfel tabloul\_meu[6] va referi al saptelea element al tabloului “tabloul\_meu”.

**Atentie**! In C, "numerotarea" elementelor tablourilor începe cu poziţia 0, astfel, daca avem definitia:

**int tabloul\_meu[100];** undeprimul element al tabloului va fi tabloul\_meu[0], iar ultimul tabloul\_meu[99].

Tablourile sunt stocate in memorie la locaţii consecutive, un tablou ocupând o zona contigua de memorie, cu primul element al tabloului aflat la adresa mai mica.

Ex.: **int x[8];**

Indexşii: [0] [1] [2] [3] [4] [5] [6] [7]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 23 |  |  |  |  | 67 |  |  |

valorile **X**

Rezultă: x[0]=23; ,,,, x[5]=67;

Atentie! O problema legata de tablouri este ca in C nu se face nici o verificare legata de "marginile" tabloului, astfel ca se pot accesa greşit elemente din afara tabloului. De exemplu, pentru definiţia:

**int tabloul\_meu[100];** daca accesam tabloul\_meu[105] nu se va semnala nici o eroare, returnându-se valoarea de la o locaţie de memorie aflata la o distanta de 5 locaţii faţă de sfârşitul tabloului, fapt ce va duce la comportări "bizare" ale programului. Aceeaşi situaţie, dar faţă de începutul tabloului, se întâmpla la accesarea tabloul\_meu[-5].

**int vect[20];**  /\* declararea tabloului vect, de maximum 20 de elemente, de tipul *int*. Elementele tabloului vect sunt : vect[0], vect[1], ., vect[19] - date de tip *int\*/*

**double p,q,tab[10];**  // declararea variabilelor simple p, q si a vectorului tab*,* de maximum 10 elemente, tip double

**#define MAX 10**

**char tab[MAX];**        **/\***declararea tabloului tab, de maximum MAX (10) elemente de tip char\*/

Consideram declaratia tabloului v cu maxim 6 elemente de tip *int*

**int v[6];**

 Elementele tabloului pot fi initializate prin atribuire:

v[0]=100;

v[1]=101;

v[2]=102;

v[3]=103;

v[4]=104;

v[5]=105;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 100 | 101 | 102 | 103 | 104 | 105 | |
| v[0] | v[1] | v[2] | v[3] | v[4] | v[5] |

**Iniţializarea tablourilor.** Deseori e necesar ca elementele tabloului să posede valori chiar la momentul descrierii tabloului. Procesul de atribuire a valorilor elementelor tabloului în timpul descrierii lui se numeşte iniţializarea tabloului. Sintaxa de iniţializare a unui tablou unidimensional este:

*tip nume[d]={v0,v1,v2,…,vn-1};*

unde *tip* este tipul tabloului, *nume* este numele tabloului, *v0,v1,v2,vn-1* valorile respective ale elementelor *nume[0],nume[1]* etc. Exemplu:

*int x[8]={1,3,15,7,19,11,13,5};*

E de menţionat faptul, că indicii tabloului se schimbă începînd dela zero.Adică la descrierea tabloului valoarea maximă a indicelui tabloului coincide cu numărul de elemente în tablou minus unu. La iniţializarea tabloului nu e numaidecât de indicat dimensiunile tabloului.Compilatorul va determina numărul elementelor după descrierea tabloului şi va forma un tablou cu mărimea respectivă. De exemplu:

*int x[]={1,3,15,7,19,11,13,5};*

 Variabilele tablou pot fi initializate in momentul declararii:

**declaratie\_tablou=lista\_valori;**

La declararea unui vector cu initializarea elementelor sale, numarul maxim de elemente ale tabloului poate fi omis, caz in care compilatorul determina automat marimea tabloului, in functie de numarul elementelor initializate. *Exemplu:*

**char tab[]=;**



          [0]                       [3]

**float data[5]=;**



    [0]                               [4]

**Accesul la elementele tabloului** Cu toate că tabloul este un tot întreg, nu se poate vorbi despre valoarea tabloului întreg. Tablourile conţin elemente cu valorile cărora se operează în program. Fiecare element în tablou îşi are indicele şi valoarea sa. În calitate de indice a unui element se foloseşte un număr întreg ce indică numărul de ordine al elementului în tablou. Enumerarea elementelor în tablou conform numărului de ordine se începe de la zero. Deci, indicele unui element poate avea valori de la 0 pîna la d-1, unde d este dimensiunea tabloului.

În calitate de valoare a unui element din tablou poate servi orice număr de tipul indicat la descrierea tabloului, adica tipul valori atribuită oricărui element din tablou trebuie să fie compatibil cu tipul tabloului. Sintaxa de acces la orice element a unui tablou este următoarea: *nume[i1][i2]..[in].* Unde *nume* este numele tabloului, *i1*– indicele elementului în dimensiunea 1, *i2*-indicele elementului în dimensiunea 2, *in* - indicele elementului în dimensiunea *n*. În cele mai dese cazuri se operează cu massive unidimensionale şi bidimensionale. Accesul la un element al unui tablou unidimensional se face în felul următor: *nume[i];* unde *nume* - numele tabloului, *i-*numarul de ordine a elementului în tablou.

## *2* Instrucţiunile ciclice şi cele adiţionale în C

Instrucţiunile precăutate mai sus redau operaţii care trebuie efectuate conform algoritmului şi fiecare din ele se îndeplinesc numai odată. În cazul, când una şi aceiaşi instrucţiune trebuie să fie executată de n ori cu diferite valori ale parametrilor se folosesc instrucţiunile ciclice. Distingem 3 instrucţiuni ciclice în C :

1) Instrucţiunea ciclică cu parametru (FOR)

2) Instrucţiunea ciclică precedată de condiţie (WHILE)

3) Instrucţiunea ciclică cu postcondiţie (DO-WHILE)

**Instrucţiunea while**

Format:

**while (*expresie*)**

***instrucţiune***

Instrucţiunea **while** este o instrucţiune de ciclu condiţionat anterior şi are formatul: while(e1) instrucţiune; antetul ciclului este while(e1) şi conţine în paranteze expresia e1 care este condiţia de repetare a ciclului. Corpul ciclului este instrucţiune şi poate fi o instrucţiune simplă sau compusă. Ea conţine acele operaţii care trebuie repetate în ciclu. Ciclul se execută astfel. Se evaluează e1 şi corpul ciclului se execută de atîtea ori de cîte ori e1 are valoarea adevăr. Deci, acest ciclu realizează construcţia ciclică următoare:

e1

false

true

corpul ciclului

Instrucţiunea se execută repetat atâta timp cât valoarea expresiei este diferită de zero. Testul are loc înaintea fiecărei execuţii a instrucţiunii. Prin urmare ciclul este următorul: se testează condiţia din paranteze dacă ea este adevărată, deci expresia din paranteze are o valoare diferită de zero, se execută corpul instrucţiunii while, se verifică din nou condiţia, dacă ea este adevărată se execută din nou corpul instrucţiunii. Când condiţia devine falsă, adică valoarea expresiei din paranteze este zero, se face un salt la instrucţiunea de după corpul instrucţiunii while, deci instrucţiunea while se termină.

Example:

a)calculează suma componentelor vectorului a de dimensiune m

suma=i=0;

while(++i< m) suma+=a[i];

b)citirea repetată de caractere până la tastarea lui 'Y'

while(getche()!='Y');

c) char \*adr; while(\*adr!=NULL) { if(\*adr=='\*')\*adr='+';

adr++; };

d) while (\*p == ' ') p++;

###### **Instrucţiunea do-while**

Ciclul do-while este un ciclu condiţionat posterior şi are formatul: do instrucţiune while(e1); Aici instrucţiune este corpul ciclului şi conţine acele instrucţiuni care trebuie repetate în ciclu.

Acest ciclu realizează următoarea construcţie ciclică:

corpul ciclului

true

e1

false

Instrucţiunea do-while se execută în felul următor: mai întîi se execută instrucţiune, adică corpul ciclului, apoi se evaluează expresia e1, care este condiţia de repetare a ciclului. Dacă e1 este adevărat, atunci se repetă execuţia corpului ciclului. În caz contrar, adică dacă e1 este 0, atunci se termină execuţia ciclului şi se trece la instrucţiunea următoare după ciclu.

Format:

**do** *instrucţiuni* **while**  (*expresie*);

Instrucţiunea se execută repetat pînă Când valoarea expresiei devine zero. Testul are loc după fiecare execuţie a instrucţiunii. Example:

i = 1; n = 1;

do {

n \*= i;

i++;

} while (i <= factorial);

**Instrucţiunea for**

Format: **for** (*expresie*-1opt; *expresie*-2opt; *expresie*-3opt)

*instrucţiune*

Această instrucţiune este echivalentă cu:

*expresie*-1;

while (*expresie*-2) {

*instrucţiune*;

*expresie*-3;

}

*Expresie*-1 constituie iniţializarea ciclului şi se execută o singură dată înaintea ciclului.

*Expresie*-2 specifică testul care controlează ciclul. El se execută înaintea fiecărei iteraţii. Dacă condiţia din test este adevărată atunci se execută corpul ciclului,

după care se execută *expresie*-3, care constă de cele mai multe ori în modificarea valorii variabilei de control al ciclului. Se revine apoi la reevaluarea condiţiei. Ciclul se termină Când condiţia devine falsă.

Oricare dintre expresiile instrucţiunii for sau chiar toate pot lipsi.

Dacă lipseşte *expresie*-2, aceasta implică faptul că clauza while este echivalentă cu while (1), ceea ce înseamnă o condiţie totdeauna adevărată. Alte omisiuni de expresii sînt pur şi simplu eliminate din expandarea de mai sus.

Instrucţiunile while şi for permit un lucru demn de observat şi anume, ele execută testul de control la începutul ciclului şi înaintea intrării în corpul instrucţiunii.

Dacă nu este nimic de făcut, nu se face nimic, cu riscul de a nu intra niciodată în corpul instrucţiunii.

Instrucţiunea for este o instrucţiune de ciclu condiţionat anterior şi are formatul:

**for(e1; e2; e3)instrucţiune;** Aici e1, e2, e3 sunt expresii. e1 este expresia de iniţializare a ciclului; e2 este expresia care determină condiţia de repetare a ciclului; e3 este expresia de reiniţializare a ciclului. Toate expresiile sunt poziţionale şi pot lipsi. instrucţiune este corpul ciclului şi poate fi o instrucţiune simplă sau compusă. Corpul ciclului sunt acele instrucţiuni de prelucrare a datelor care trebuie repetate. Corpul ciclului poate lipsi. În acest caz ciclul constă numai din antet şi instrucţiunea vidă: for(e1; e2; e3).

Instrucţiunea for se execută în felul următor: se efectuează operaţiile de iniţializare ale ciclului e1, apoi se evaluează expresia e2. Dacă e2 are o valoare diferită de 0, adică valoarea adevăr, atunci se execută instrucţiune – corpul ciclului. În caz contrar, atunci cînd e2 are valoarea fals, se termină execuţia ciclului for şi se trece la instrucţiunea următoare după ciclu. După execuţia corpului, ciclul se reiniţializează – se execută operaţiile definite de e3 şi se revine iarăşi la verificarea condiţiei de repetare a ciclului e2.

Aşadar, instrucţiunea for realizează următoarea construcţie ciclică:

e1

false

e2

corpul ciclului

e3

Exemple:

In urmatorul program, o bucla ***for*** este utilizata pentru a afisa pe ecran numerele de la 1 la 100.

#include<stdio.h>

void main ()

{ int x;

for(x=1; x<=100; x++)

printf(„%d”, x);

}

Variabila **i** este utilizata pe post de “contor” al instructiunii ***for***, numarand la a cata iteratie s-a ajuns.

Executia instructiunii ***for*** se incheie atunci cand numarul de iteratii devine egal cu **n, (100)**.

Initializarea lui **i** cu 1 se realizeaza o singura data, la inceput; **i<=100** este conditia de continuare a executiei;

**i++** se efectueaza dupa fiecare executie a ciclului (postincrementare).

for (i=0; i<100; i++)

sum += x[i];

for (i=0, t=string; i < 40 && \*t; i++, t++)

putch(\*t);

putch('\n');

Instrucţiuni de salt

C are 4 instrucţiuni care execută ramificări necondiţionate: *return, goto, break* şi *continue.* Dintre acestea, *return* şi *goto* pot să se găsească oriunde în program. Instrucţiunile *break* şi *continue* pot fi utilizate împreună cu oricare din instrucţiunile de buclare.

###### **Instrucţiunea continue**

Format:

**continue;**

Instrucţiunea continue se foloseşte numai în corpul unui ciclu şi abandonează iteraţia curentă a ciclului şi trece la iteraţia următoare a lui. Deci instrucţiunea determină trecerea controlului la porţiunea de continuare a ciclului celei mai interioare instrucţiuni while, do sau for care o conţine, adică la sfîrşitul ciclului şi reluarea următoarei iteraţii a ciclului. În while şi do se continuă cu testul, iar în for se continuă cu *expresie-*3.

Mai precis în fiecare dintre instrucţiunile:

|  |  |  |
| --- | --- | --- |
| while (...) {  ...  continue;  } | for (...) {  ...  continue;  } | do {  ...  continue;  } while (...); |

dacă apare o instrucţiune continue aceasta este echivalentă cu un salt la eticheta continue. După continue urmează o instrucţiune vidă.

Porţiunea de program din exemplul următor prelucrează numai elementele pozitive ale unui masiv.

for (i=0; i<n; i++) {

if (a[i]<0) /\* sare peste elementele negative \*/

continue;

## *... /\* prelucrează elementele pozitive \*/*

}

**Instrucţiunea return**

Instrucţiunea return se foloseşte pentru a reveni dintr-o funcţie.O instrucţiune return permite ieşirea dintr-o funcţie şi transmiterea controlului apelantului funcţiei. O funcţie poate returna valori apelantului său, prin intermediul unei instrucţiuni return.

Formate:

**return;**

return *expresie*;

În primul caz valoarea returnată nu este definită. În al doilea caz valoarea expresiei este returnată apelantului funcţiei. Dacă se cere, expresia este convertită, ca într-o atribuire, la tipul funcţiei în care ea apare.

**Instrucţiunea vidă**

Format:

**;**

Instrucţiunea vidă este utilă pentru a introduce o etichetă înaintea unei acolade drepte, într-o instrucţiune compusă, sau pentru a introduce un corp nul într-o instrucţiune de ciclare care cere corp al instrucţiunii, ca de exemplu while sau for.

*Exemplu*:

for (nc=0; s[nc]!=0; ++nc) ;

Această instrucţiune numără caracterele unui şir. Corpul lui for este vid, deoarece tot lucrul se face în partea de test şi actualizare dar sintaxa lui for cere un corp al instrucţiunii. Instrucţiunea vidă satisface acest lucru.

**Instrucţiune de salt Break** Pernmite întreruperea necondiţionată a unei secvenţe şi continuarea programului dintr-un alt punct.

## Instrucţiunea BREAK. Se utilizează în două contexte, pentru a marca încheierea secvenţei de instrucţiuni asociate unui selector case, şi instrucţiunea de ciclare, pentru a determina ieşirea forţată dintr-un ciclu while, do\_while, sau for.

**Instructiunea cu etichetă GOTO**: goto et --identificator. Are ca efect întreruperea secvenţei curente şi continuarea execuţiei de la instrucţiunea cu eticheta identificatorului, ce trebuie să se afle în aceaşi funcţie.

Totodată, avînd la dispoziţie instrucţiunile *goto* şi *if,* programatorul poate programa diferite operaţii complicate.

Exemplu :Folosirea instrucţiunilor if şi goto la organizarea ciclurilor. Acest program calculează valoarea lui y care este egală cu \_n/(n+5),unde n=1,..,50

***#define lim 50***

***main() {***

***int n=0*; *float y=0*;**

***m1: ++n*;**

***if(n<=lim) {***

***y+=n/(n+50)*;*goto m1*;*} }***

Prima linie a programului e menită preprocesorului, căruia îi se indică valoarea constantei *lim=50.* Astfel se procedează în cazul cînd vom necesita schimbarea limitei de sumare. Pentru aceasta trebuie schimbată doar directiva #define, iar substituirile corespunzătoare în textul programului preprocesorul le va face fără participarea noastră. Programul constă dintr-o singură funcţie *main(),* corpul căreia e definit cu ajutorul acoladelor exterioare. Instrucţiunea if are aici forma prescurtată, avînd ca instrucţiune un bloc, care conţine instrucţiunea goto, ce ne dă posibilitatea calculării ciclice a sumei. În încheiere menţionăm că utilizarea *if* şi *goto* la organizarea ciclurilor e un semn de cultura insuficientă de programare. Exemplu dat arată numai cum se poate organiza un ciclu cu ajutorul instrucţiunilor *if* şi *goto*, dar în nici un caz nu serveşte drept exemplu pentru copiere. La organizarea ciclurilor în limbajul de programare Turbo C se folosesc alte posibilităţi, cu mult mai fine.

....

a=0;

eticheta1: x++;

if (x<100) goto eticheta1;

***1.3 Organizarea prelucrărilor tablourilor în C***

**a)Citirea valorilor**

Ex.

int tablou[10];

printf(“\nIntroduceti dimensiunea tabloului”);

scanf(“%d”,&n);

for(i=0;I,1;i++);

{ printf(“\n tablou[%d]=”,i); //se va afişa tablou[i]= si se asteapta

scanf(“%d”,&tablou[i]); //introducerea valorii care se atribue variabilei tabloului

}

**Obs**:

1.Dimensiunea tabloului introdusă de către utilizator (n) nu trebue sa depaşeasca dimensiunea cu care a fost declarat vectorul,(in cazul anterior n introdus de utilizator nu trebue sa depaseasca 10, întrucât declaraţia a fost **tablou[10]**).

2.Identificarea zonei de memorie unde va fi reţinut tabloul se face cu ajutorul **numelui tabloului.** Acesta are ca valoare adresa primului element din tablou, ceea ce il deosebeşte de variabilele simple care nu reprezinta o adresa.

&st≡st≡&st[0];

3.Instructiunea printf() din interiorul buclei cu contorizare va afişa numai numele variabilei care se citeste.

4.Instructiunea scanf() va prelua si va memora valoarea pentru pozitia respective (tablou[i]).

5. Daca secventa de citire este alcatuita sub forma:

for(i=0;i<n;i++)

{

printf(“\n tablou[%d]=%d:”,I,tablou[i]);

scanf(“%d”,&tablou[i]);

}

Se va afişa,pentru i=0,tablou[0]=-30731 si se va executa apoi instructiunea de citire, deci se va astepta introducerea valorii care se va memoria in locatia tablou[0];

Valoarea afişata -30731 provine din faptul ca in momentul afişarii variabila tablou[i] nu este initializata si de aceea se va afişa o valoare aleatoare care va fi gasita in memorie la adresa respectivului element.

**b)Afişarea vectorilor**

Ex.

int tab[5]={1,2,3,4,5}; //sau vectorul este introdus ca la exer. de mai sus

printf(“\nElementele vectorului sunt:”);

for(i=0;i<5;i++)

printf(“\n tab[%d]=%d”,I,tab[i]);

**Obs.**  In acest caz elementele tabloului au fost initializate direct.Pentru aceasta valorile care trebuesc retinute in tablou au fost trecute intre accolade{}.

**c) Prelucrări asupra vectorilor**

**Ex.** Sa se afişeze valorile unei functii considerind ca apartine unui vector de valori.

f : D→C

D=(-∞,-3]U[3,∞)

f(x)= x{-5,-2,0,5,7,9} f(x){4,x,x,4,,}

**Obs.** Prin caracterul x sa reprezentat faptul ca respectivele valorix[i] nu apartin domeniului de definitie.

Etapele care trebuesc parcurse pentru determinarea valorilor functiei:

* Declararea vectorului de intrare xfloat x[5] si a vectorului de iesire f float f[5]
* Citirea numarului de elemente din x si a valorilor acestuia;
* Verificarea numarului de elemente din x si a valorilor fumctiei (pentru x[i]D).Daca x[i] nu apartine lui D atunci evident calculul valorii lui f nu este posibil.De aceea sa folosit variabila j care sa indice pozitia valorii care a putut fi calculate in vectorul f.Totodata valorile x[i] care apartin domeniului sunt memorate in vectorul x1;
* Afişarea vectorilo x1, sir f;

CONDITIA: Este dat tabloul unidimensional X={Xi},i=1,…,n; cu elemente reale.Alcatuiti algoritmul si programul pentru urmatoarele:obtinerea tabloului unidimensional T cu valori logice , reesind din urmatoarele:

1. variabilei t sa-I fie atribuita valoarea TRUE, daca elementele tabloului X unt ordonate strict in ordinea crescatoare, iar valoatrea FALSE in caz contrar;
2. variabilei t sa-I atribue valoarea TRUE, daca in tabloul X nu sunt prezente elementele de zero si totodata eleentele positive se alterneaza cu cele negative,iar valoarea FALSE in caz contrar;
3. variabilei k sa-I fie atribuit numarul primei intrari y in tabloul X, astfel, adica daca y nu se contine in X1 sa se calculeze K=X1+X1X2+X1X2X3+….+XX1X2…Xm, unde m este numarul primului element negative al tabloului X

Listingul programului:

# include <stdio.h>

# include <conio.h>

int i,a[10],t,n,p=0,d=0,r=0,z=1,y=0,k=0;

void main()

{

clrscr();

printf("Introducem nr. de elemente al tabloului n=");

scanf("%d",&n);

printf("Introducem y=");scanf("%d",&y);

for(i=0;i<n;i++)

scanf("%d",&a[i]);

for(i=0;i<n-1;i++)

{

if (a[i]<a[i+1]) p++;

if (p==n-1) t=1;

else t=0;}

printf("\nt=%d",t);

for(i=0;i<n;i++)

{

if(a[i]>0&&a[i+1]>0||a[i]==0) d++;

if(a[i]<0&&a[i+1]<0&&a[i+2]<0) r++;

if(d>=1||r>=1) t=0;

else t=1;

}

printf("\nt=%d",t);

for(i=0;i<n;i++)

{ if (y==a[i]) {k=i;break;}

if(a[i]>=0)

{z=z\*a[i]; k=k+z;}

}

printf("\nk=%d",k);

getche();

}

**3. Teste: Exerciţii penrtu analize:** Ce se afiseaza pe ecran la executia urmatoarelor programe? Gasiti raspunsul, dati justificarea, apoi verificati prin executie.

1.Ce valori va **afişa** programul următor?

int m=10, i=7,n,k;

do for (k=5; k >2; k--)

{for (n=4; n!=0; n--) m++; i--;

while (i>3);

printf (k=%d m=%d\n,k,m);

2. Analizaţi şi apreciaţi ce efectuiază următorul program C:

#include <stdio.h>

main(){int sum = 0, card; char answer[36];

srand( getpid()); /\* randomizare \*/

do{ printf( "Aveţi %d puncte. Inca? ", sum);

if( \*gets(answer) == 'n' ) break; /\* pentru că vor fi prea puţine \*/

printf( " %d puncte\n", card = 6 + rand() % (11 - 6 + 1));}

while((sum += card) < 21); /\* SIC ! \*/

printf(sum == 21 ? "puncte\n" :sum >21 ? "depăşit\n": "%d puncte\n", sum);

}

3. Exemplu de secventã pentru afişarea a n întregi câte m pe o linie :

for ( i=1;i<=n;i++) { printf ( "%5d%c",i, ( i%m==0 || i==n)? '\n':' ');

O variantã mai explicitã dar mai lungã pentru secventa anterioarã:

for ( i=1;i<=n;i++) { printf ("%6d ",i); if(i%m==0) printf("\n"); } printf("\n");

Una dintre conventii se referã la modul de scriere a acoladelor care încadreazã un bloc de instructiuni ce face parte dintr-o functie sau dintr-o instructiune *if, while, for* etc. Cele douã stiluri care pot fi întâlnite în diferite programe si cãrti sunt ilustrate de exemplele urmãtoare:

void main () // Afişare numere perfecte , stil Linux

{ int n,m,s,d; scanf (%d”,&n);

for (m=2; m<=n; m++) { s=0;

for (d=1; d<m; d++) { if ( m % d ==0 ) s= s+ d; }

if ( m==s) printf (%6d\n”,m); } }

// Afişare numere perfecte, stil K&R si Java

void main () { int n,m,s,d; scanf (%d”,&n);

for (m=2; m<=n; m++){ s=0;

for (d=1; d<m; d++){

if ( m % d ==0 )

s= s+ d;

}

if ( m==s) printf (%6d\n”,m);

} }

**Obs**. Specific limbajului C este utilizarea de expresii aritmetice sau de atribuire drept conditii în instructiuni *if, while, for, do* în absenta unui tip logic (boolean). Exemplu:

while (\*d++ =\*s++); // copiaza sir de la s la d

Pentru a facilita citirea programelor si trecerea de la C la Java este bine ca toate condiţiile sã aparã ca expresii de relaţie si nu ca expresii aritmetice:

while (\*s != 0)

\*d++=\*s++;

4. Exemplul formării tabloului:

int main()

{

const int array\_size = 10;

int ia[ array\_size ];

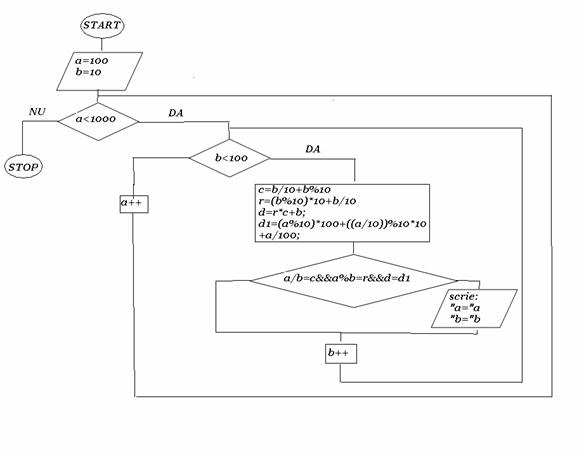
for ( int ix = 0; ix < array\_size; ++ ix )

ia[ ix ] = ix;

}

5. Am gasit doua numere:unul de trei cifre si altul de doua cifre.Daca impartim A  la B catul rezultat este egal cu suma cifrelor divizorului,iar in restul este un numar de doua cifre constand din cifrele divizorului in ordine inversa. Daca se inmulteste restul cu catul si apoi se incrementeaza cu divizorul,atunci numarul de trei cifre obtinut va consta din cifrele deimpartitului in ordine inversa.

Gasiti numerele A si B.



**TESTE în C++:** OBSERVATIE: Pe langa biblotecile utilizate de C++, se mai foloseste si biblioteca <isotream.h> care include functiile de iesire *cout* si *cin* cu urmatoarele structuri:

***cout<< expresie1<<expresie2<<expresie3<<endl;***

*cout* este o functie din biblioteca care tipareste pe monitor (daca nu este specificata o alta destinatie). Are acelasi efect ca si functia *printf(“”)* din biblioteca <stdio.h>

***cin>>nume variabila;***

***cin>>num1>>num2; //utilizata pentru valori de intrare***

**De exemplu pentru**  calcularea minimului elementelor unui sir de n numere, se utilizaza lucrul cu vectori.

**//TEST6**

#include <iostream.h>

void main()

{

int n; //numarul de elemente

int a[50]; //se defineste o variabila de tip tablou cu maxim 50 de elemente,

//deci n va trebui sa fie mai mic decat 50

int min; //variabila ce va memora minimul

int i; //contor in instructiunea for

cout<<”n=”;

cin>>n;

/\* se citesc elementele sirului \*/

for(i=0;i<n;i++)

{

cout<<”a[“<<i<<”]=”;

cin>>a[i];

}

// se calculeaza minimul

min=a[0]; //initializam minimul cu primul element din sir

for(i=1;i<n;i++)

if (a[i]<min) min=a[i];

//afisare minim

cout<<”Minimul este=”<<min;

}

Dupa cum se poate observa din programul de mai sus, fiecare element al sirului se poate utiliza ca si cum ar fi o variabila de tip *int* independenta, deci valoarea unui element al vectorului poate fi modificata independent de celelelate elemente. Sa presupunem ca utilizatorul introduce pentru n valoarea 4. Initial, elementele vectorului nu au o valoare bine definita;

Sa presupunem ca utilizatorul introduce valorile 3, 7, 2, 9 pentru cele patru elemente ale vectorului.

Valoarea oricarui element al vectorului poate fi modificata fie printr-o atribuire, fie prin introducerea unei valori de la tastatura. Daca vom introduce atribuirea a[2]=23;

valoarea elementului al treilea din vectorul a nu va mai fi 2 (vechea val.) ci 23.

Programul următor calculează media aritmetică a valorilor strict pozitive care fac parte din a(n).

**//TEST 7**

#include <stdio.h>

#include <conio.h>

//#include <iostream.h> il utilizati cand folositi cout si cin

void main()

{

float a[40],suma=0,med;

int lung,i,cont=0;

printf("Introduceti lungimea vectorului :");

scanf("%d",&lung);

//cout << "Introduceti lungimea vectorului :";

//cin >> lung;

for (i=0;i<lung;i++) //parcurgerea vectorului

{

printf("a[%d]: ",i+1);

scanf("%f",&a[i]); //citirea vectorului

/\*cout << "a["<<i+1<<"]: ";

cin >> a[i];\*/

}

for (i=0;i<lung;i++) //parcurgerea vectorului

if (a[i]>0) //cautarea doar a numerelor pozitive

{

suma+=a[i]; //suma numerelor pozitive

cont++; //contorizator al numerelor pozitive

}

med=suma/cont; //media aritmetica

printf("Media aritmetica a numerelor pozitive din vectorul a este %-8.2f",med);

//cout << "Media aritmetica a numerelor pozitive din vectorul a este "<<med;

getch();astfe//asteapta introducerea unui caracter

}

**Daca** se considera n numere intregi introduse de la tastatura, sa se afle cate numere sunt pare si cate impare.

**//TEST8**

#include <iostream.h>

#include <conio.h>

void main()

{

int n; //numarul de elemente

int v[30]; //vector cu maxim 30 elemente intregi

int pare,impare; //variabile ce memoreaza nr. Elementelor pare, respectiv impare

int j; //variabila folosita in instructiunea for

//citim numarul de elemente

cout<<”n=”;

cin>>n;

/\* se citesc elementele sirului \*/

for(i=0;i<n;i++)

{

cout<<”v[“<<i<<”]=”;

cin>>v[i];

}

//initializam variabilele

pare=0;

impare=0;

//luam fiecare element din v si testam daca acesta este sau nu par

for(j=0;j<n;j++)

if (v[j] % 2 = =0) pare++;

else impare++;

//afisam rezultatul

cout<<”Am gasit “<<pare<<” numere pare si “<<impare;

}

In programul de mai sus se verifica, pentru fiecare element, daca acesta se imparte exact la doi, caz in care s-a mai descoperit un element par. In caz contrar, numarul elementelor impare se mareste cu unu.

S-a utilizat operatorul **%** , numit si modulo aritmetic, care are ca rezultat restul impartirii lui v[j] la 2.

# **4. Întrebările de autocontrol**

3.1.Transcrieţi exemplul programului din lucrarea de laborator nr.2, utilizînd instrucţiunile ciclice.

3.2\*. Elaboraţi algoritmul şi programul pentru cazul dacă suma a trei numere reale cu valori diferite x,y,z este mai mică decât unitatea, atunci cel mai mic număr din aceste trei, de schimbat cu semisuma a celorlalte două, în caz contrar (Când suma este mai mare ) de schimbat valoarea minimală dintre x şi y cu semisuma a celorlalte valori rămase.

3.3. Modificaţi exemplul 3.2\* pentru trei tablouri unidimensionale X, Y, Z şi efectuaţi aceleaşi calcule

3.4\*. Elaboraţi algoritmul şi programul pentru cazul când sunt date 100 numere întregi pentru care trebuie de calculat diferenţa maximă şi minimimă între ele.

3.5\*. Elaboraţi algoritmul şi programul pentru determinarea. dacă un număr natural e perfect, adică care este egal cu suma tuturor divizorilor.(de exemplu 6=1+2+3).

3.6. Ce subînţelegem prin tablou şi cum se notează în program elementele unui tabel?

3.7. Cum se organizează în program introducerea şi extragerea unui tablou?

3.8. Ce operaţii se pot efectua efectiv cu elementele unui tablou?

3.9. Fie **X[1..n]** si **Y[1..n]** vectori de întregi. Care va fi valoarea lui **Y[n]** după execuţia secvenţei:

**Y[1]=x[1];**

**for (i=2;i<n;i++) y[i]=y[i-1]+x[i];**

a) x[n]+x[n-1] b) x[n]

c) x[1]+x[2]+………..+x[n]

d) nici una din valorile indicate

3.10 Fie **X[1..n]** si **Y[1..n]** vectori de numere reale. După execuţia secvenţei de program:

**y[1]=-x[1];**

**for(i=2;i<n;i++) y[i]=y[i-1]\*x[i];**

elementul **Y[n]** exprimă :

a) x[1]\*x[2]\*……\*x[n]

b) –x[1]\*x[2]\*….\*x[n]

c) (-1)n x[1]\*…..\*x[n]

d) nici una din valorile indicate

3.11 Fie **V[1..n]** vector de intregi . Secvenţa de program :

**i=1;**

**for(i=1;i<=n/2;i++) { j=n-i; aux=v[i]; v[i]=v[j]; v[j]=aux; }** are ca efect :

a) inversarea ordinii elementelor în vector

b) inversarea ordinii tuturor elementelor în vector numai când **n** este impar

c) inversarea ordinii tuturor elementelor în vector numai când **n** este par

d) nici una din variantele indicate

3.12 Ce face următoarea secvenţă ?

**scanf(“%d”,&n);**

**for(i=0;i<n;i++) scanf(“%d”,&a[i]); printf(“\n%d”,a[0]);**

**for(i=1;i<n;i++) { ex=0;**

**for(j=0;j<i;j++) if(a[i]==a[j]) ex=1; if(!ex) printf(“%d”,a[i]); }**

a) afişează numerele dintr-un şir care sunt în mai multe exemplare

b) afişează numerele cu apariţie singulară în şir

c) afişează numerele dintr-un şir

d) afişează numerele impare dintrun şir

3.13. Ce realizează următoarea secvenţă de program?

**i=0;**

**do { i++; a[i]=nr%2; nr/=2; }while(nr);**

**n=i; for(i=n;i>0;i--) printf(“%d”,a[i]);**

a) convertirea b10->b2 a unui număr fracţionar

b) convertirea b10->b2 a unui număr întreg

c) convertirea b10->b3 a unui număr întreg

d) convertirea b10->b2 a unui număr întreg pozitiv

3.14. Se dă următoarea secvenţă de cod :

**int y[5]={3,4,5,6,0};** Ce valoare conţine **y[3]** ?

a) 3 b) 5 c) 6

d) codul nu compilează pentru că nu sunt destule valori

3.15. Fie secvenţa de cod :

**int x,i,t; int y[10]={3,6,9,5,7,2,8,10,0,3};**

**while(1) { x=0;**

**for(i=0;i<9;i++) if (y[i]>y[i+1]) { t=y[i]; y[i]=y[i+1]; y[i+1]=t; x++; } if(x==0) break; }**

Cum va arăta vectorul după execuţia acestui cod ?

a) programul va rula la infinit

b) {0,2,3,3,5,6,7,8,9,10}

b) {10,9,8,7,6,5,3,3,2,0}

c) {3,6,9,5,7,2,8,10,0,3}

3.16. Există greşeli în secvenţa de calculare a mediei aritmetice?

#include<stdio.h>

#include<conio.h>

**void main(void) { int a[30],i,n=20; int s=0; float ma;**

**for(i=0;i<n;i++) { printf(“\na[%d]=”,i); scanf(“%d”,&a[i]); }**

**for(i=0;i<n;i++) s=s+a[i]; ma=s/n; printf(“\nRezultatul=%f”,ma); getch(); }**

a) nu, secvenţa este corectă

b) da, deoarece nu au fost citite toate elementele tabloului

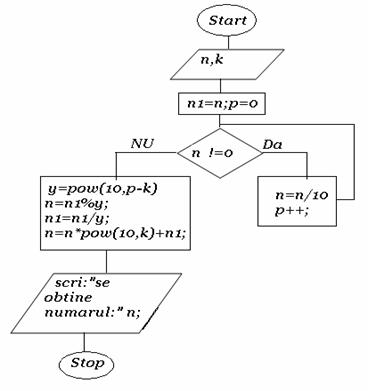
c) da, deoarece va fi afişată doar partea întreagă a rezultatului

1. da, deoarece nu au fost citite corect toate elementelevectorului

3.17. Se da un numar n cu maximum 9 cifre si k prin care se primeste un numar natural (k<=9).Sa se modifoce numarul n astfel incat primele k cifre din n de la stanga  sa ajunga ultimele cifre ale lui n.Sa se afiseze numarul astfel obtinut.

Exemplu: n=7462934

k=4      deci numarul 7462 va trece la coada lui n.Se va afisa numarul 9347462



#include<conio.h>

#include<iostream.h>

#include<math.h>

void main()

int y=pow(10,p-k);

n=n1%y;

n1=n1/y;

n=n\*pow(10,k)+n1;

cout<<"se obtine numarul"  <<n;

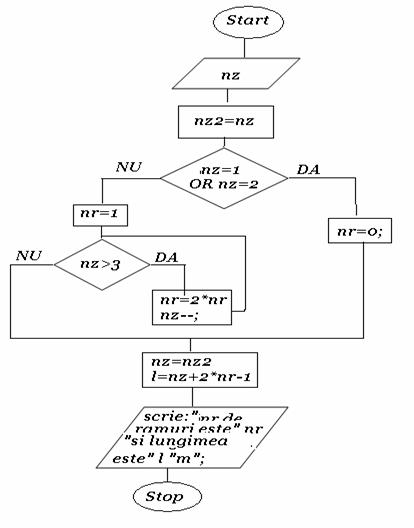
getch();

}

3.18. A fost odata un taran sarac.Intr-o zi el a gasit o samanta.A pus-o in pamant, si minune:dupa ce a rasarit a inceput sa creasca in inaltime,in fiecare zi cate un metru!Dupa doua zile apare un mugur cu o noua ramura si apoi in fiecare zi mai apare cate o ramura.Fiecare ramura noua  creste si inmugureste la fel ca prima.Dupa cateva zile taranul taie planta si are dificultati in a numara ramurile si a masura lungimea lor totala.

  Taranul va roaga sa-il ajutati:el va spune numarul de zile dupa care a taiat planta si voi sa ii spuneti cate ramuri are si ce lungime totala obtine daca ar pune cap la cap toate ramurile?

Exemplu:dupa 5 zile planta are 4 ramuri  totalizand 12 metrii.(problema din cartea de Algoritmi si structuri de date, pagina 261, problema 6)



#include<conio.h>

#include<iostream.h>

void main()

   }

nz=nz2;

l=nz+2\*nr-1;

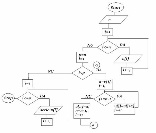
cout<<'numarul de ramuri  este '<<nr<<' si lungimea totala a plantei este de ' <<l<<"m";

getch();

}

3.19. Sa se interschimbe  elementele unui vector astfel incat primul element sa fie ultimul, al doilea element sa fie penultimul, al treilea sa fie antepenultimul etc.

Exemplu: Vectorul v=(1,2,3,4,5) cu 5 elemnte va fi v=(5,4,3,2,1);



#include<conio.h>

#include<iostream.h>

void main()

 for(i=1;i<=p;i++)

       cout<<v[i]<<' ';

getch();

}

3.20. Studiul unei metode de calcul aplicata intr-o functie matematica, utilizind pasii h1, h2, si ciclatorul while.

1. **Conditia prolemei:**  De calculate valoarea functiei, in depemdenta de conditie:

|  |  |  |
| --- | --- | --- |
| Nr. | Functia | Pasul |
| 2 | s | h1  h2 |

1. **Listingul programului:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void main(void)

{

float a,b,c,x,x1,x2,c1,c2,s,h1,h2;

clrscr();

printf("Culegeti inceputul intervalului x, x1=");

scanf("%f",&x1);

printf("Culegeti sfirsitul intervalului x, x2=");

scanf("%f",&x2);

printf("Culegeti pasul intervalului x, h1=");

scanf("%f",&h1);

printf("Culegeti inceputul intervalului c, c1=");

scanf("%f",&c2);

printf("Culegeti sfirsitul intervalului c, c2=");

scanf("%f",&c2);

printf("Culegeti pasul intervalului c, h2=");

scanf("%f",&h2);

printf("Culegeti o valuare pentru a: ");

scanf("%f",&a);

printf("Culegeti o valuare pentru b: ");

scanf("%f",&b);

x=x1;

while (x<=x2)

{

while (c<=c2)

{

if (x>c) s=(a+b\*x)/sqrt(x+1);

else

if (x<c) s=(a\*x+b\*x)/sqrt(x+1);

else s=a/x+sqrt(x+1);

printf("\nx=%2f c=%2f s=%2f",x,c,s);

c+=h2;}

x+=h1;

}

}

1. **Rezultatul executiei programului:**

Culegeti inceputul intervalului x, x1=1

Culegeti sfirsitul intervalului x, x2=325

Culegeti pasul intervalului x, h1=4

Culegeti inceputul intervalului c, c1=6

Culegeti sfirsitul intervalului c, c2=2

Culegeti pasul intervalului c, h2=1

Culegeti o valuare pentru a: 2

Culegeti o valuare pentru b: 4

x=1.000000 c=0.000000 s=4.242640

x=1.000000 c=1.000000 s=3.414214

x=1.000000 c=2.000000 s=4.242640

3.21. Exemplificarea utilizarii instructiunii for, studiata la curs.

Sa se scrie un program care ca citeasca un sir de la tastatura, sa calculeze produsul elementelor strict positive din sir si sa afiseze rezultatul final.

#include<stdio.h>

#include<conio.h>

void main()

{int sir[30],n,i,j;

float prod=1;

clrscr();

printf("Dati lungimea sirului ");

scanf("%d",&n);

for(i=0;i<n;i++)

{printf("Introduceti elementul sir[%d]=",i);

scanf("%d",&sir[i]);

}

for(j=0,i=0;i<n;i++)

if ((sir[i]>0)&&(sir[i]%2==0)) {prod=prod\*sir[i]; j++;}

if (j!=0) printf("In produs au intrat %d elemente iar valoarea lui este %.f",j,prod);

else printf("Nu au fost elemente care sa respecte conditiile impuse");

getch();

}

3.22. Sa se scrie un program care sa calculeze factorialul unui numar strict pozitiv. Sa se repete citirea numarului pana cand se introduce de la tastatura un numar pozitiv.

#include<stdio.h>

#include<conio.h>

void main()

{int n,i;

double factorial=1;

clrscr();

do

{printf("Dati n=");

scanf("%d",&n);

if (n<=0) printf("Numar mai mic sau egal cu zero. Va rugam introduceti numarul din nou \n");

} while (n<=0);

for(i=0;i<n;i++,factorial=factorial\*i);

printf("Numarul introdus este %d iar valoarea factorialului este %.f",n,factorial);

getch();

}

3.23. Exemplificarea instructiunii switch.

Sa se scrie un program de la tastatura, care sa citeasca doua numere de la tastarua, si un operator (\*,/,+,-). Daca unul din aceste caractere este introdus, sa se calculeze rezultatul operatiei. Sa se afiseze rezultatul.

#include<stdio.h>

#include<conio.h>

void main()

{int x,y,i,j=0;

float rezultat;

char operand;

clrscr();

printf("Dati x=");

scanf("%d",&x);

printf("Dati y=");

scanf("%d",&y);

printf("Introduceti operatorul de calcul");

scanf("%s",&operand);

switch (operand) {

case '\*': rezultat =x\*y; break;

case '/': if (y!=0) rezultat =x/y; else printf("Operatia nu se poate efectua"); break;

case '+': rezultat =x+y; break;

case '-': rezultat =x-y; break;

default: printf("Operator necunoscut!!"); j=1; break;

}

if ((j==0)&&(y!=0)) printf("Rezultatul operatiei este %.2f", rezultat);

getch();

}

**Exerciţii şi teste grilă**

1. Care dintre variantele de mai jos reprezintă o declaraţie corectă a unui vector **v** cu 20 de elemente numere întregi ?

**a) v[20]:integer; b) v[20] int;**

**c) int v[20]; d) int :v[20];**

**e) integer v[20];**

2. Câte erori conţine programul de mai jos?

**void main()**

**{ int n,k; int v[n]; n=4**

**for(k=0;k<n;k++) v[k]= =0; }**

a) 0 b) 1 c) 2 d) 3 e) 4

3. Care dintre secvenţele de program de mai jos calculează corect suma primelor **n** elemente ale unui vector **s** ?

**a) s=0;**

**for(i=0;i<n;i++) s+=v[i];**

**b) s=0; i=0;**

**while(i<n) { s+=v[i]; i++;}**

**c) s=0; i=0;**

**do{ s+=v[i]; i++; }while(i<n-1);**

**d) toate e) nici una**

4. Deduceţi care vor fi elementele vectorului **v** după execuţia secvenţei de program următoare:

**int n,k,x,v[7]={5,14,-3,8,-1};**

**n=5; x=v[0];**

**for(k=1;k<n;k++) v[k-1]=v[k]; v[n-1]=x;**

a) (-1,5,14,-3,8,0,0)

b) (14,-3,8,-1,0,0,5)

c) (14,-3,8,-1,5,0,0)

d) (0,0,5,-3,14,-1,8)

e) (0,0,-1,14,-3,8,5)

5. Câte elemente ale vectorului **v** vor avea valoarea 9 după execuţia programului de mai jos?

**#include<stdio.h>**

**void main()**

**{ int v[]={0,1,2,0,4,5,6};**

**int i=0,x=9;**

**do{ v[i++]=x; }while(i<6 && v[i]); }**

a) nici unul b) unul c) două d) trei e) toate

**Anexa** ***Variantele problemelor prelucrării tablourilor unidimensionale***

1. Să se elaboreze algoritmul şi programul pentru următoarele: să se calculeze tabloul F[i=1,…n] după formulele:

n

**** X***i*** / Y***i*** , dacă k<=10;

F[k] = i=1

n

**** X***i*** / ec/y***i*** , dacă k>10;

i=1

unde valorile elementelor tablourilor unidimensionale X=(x1,x2,...,xN) şi Y=(y1,y2,...,yN) sunt reale şi se întroduc de la tastatură, n<=20; k<=20.

2. Să se elaboreze algoritmul şi programul pentru următoarele: Sunt date tablourile unidimensionale A={a***i*** }; B={b***i*** }, i=1,…,n. Reglamentaţi tablourile date conform creşterii sau descreşterii valorilor lor. Folosind interclasarea grupaţi elementele acestor două tablouri într-un tablou C={c***i*** }, j=1,…,2xn; în aşa mod ca ele să fie puse în ordinea de creştere sau descreştere.

3. Să se elaboreze algoritmul şi programul pentru următoarele: Este dat tabloul unidimensional X={x***i*** }, i=1,…,n; cu elemente reale. Alcătuiţi algoritmul şi programul pentru următoarele: obţinerea tabloului unidimensional T cu valori logice, reieşind din următoarele:

1. variabilei t să fie atribuită valoarea TRUE- dacă elementele tabloului X sunt ordonate strict în ordinea crescătoare, valoarea FALSE în caz contrar;
2. variabilei t să fie atribuită valoarea TRUE dacă în tabloul X nu sunt prezente elementele de zero şi totodată elementele pozitive se alternează cu cele negative, iar valoarea FALSE în caz contrar;
3. Să se elaboreze algoritmul şi programul pentru următoarele: sunt date tablourile unidimensionale X={ x*i* }, i=1,…,n; Y={ Yj }, j=1,…,m. Să se elaboreze algoritmul şi programul pentru următoarele operaţii cu tablourile:
4. variabilei logice t să fie atribuită valoarea TRUE dacă mulţimea elementelor tabloului X este submulţimea mulţimii Y şi FALSE în caz contrar. Apoi să se aprecieze următoarele:
5. Z=X ∩ Y - intersecţia tablourilor;
6. Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat tabloul X={ x*i* }, i=1,…,n în baza căruia să se calculeze valoarea F:**

**1, dacă x*1* > x*2* >.. x*i*..> x*n* ;**

**2, dacă x*1* < x*2* <.. x*i*..< x*n* ;**

**3, dacă x*i* <2 <x <2 <...<x <2 ;**

**F= 4, dacă max x > x**

**5, dacă min x < P x ;**

**cosx sinx - în cazurile rămase.**

**6.** Să se elaboreze algoritmul şi programul pentru următoarele: **Sunt date tablourile X={ x*i* }, i=1,…,n; Y={y*i* }, i=1,…,n şi valoarea t în baza cărora să se calculeze valorile lui F:**

**n**

** (x*i* +y*i* + x*i* y*i* ), dacă k=1;**

**i=1**

**(X2*i*+X2*i*+...+X2*i*)(y2*i*+y2*i*+...+y2*i*), dacă k=2;**

**F= n**

**(y3*i* + x3*i*), dacă k=3;**

**j=1**

**7.** Să se elaboreze algoritmul şi programul pentru următoarele:  **Este dat un şir de numere x*1,* x*2*,… x*i…*. x*n*. Printre aceste numere sunt măcar două numere negative. Să se evidenţieze în această succesiune elementele tabloului unidimensional a*1*, a*2*,... a*k*, (k – necunoscut** **anticipat)- puse între o pereche de numere negative. Dacă k>1, atunci să se calculeze :**

**a) max(a2*1*, a2*2*,…a2*k*); b) min(a*1*, 2a*2*,... ka*k*);**

**c) numărul cifrelor pare printre a*1*, a*2*,... a*k*;**

**8.** Să se elaboreze algoritmul şi programul pentru următoarele:  **Este dat tabloul unidimensional V={ v*i* }, i=1,…,n, cu valori aleatorii, în baza căruia să se efectuieze următoarele:**

a) să se aprecieze numărul de ordine şi valoarea primului element pozitiv şi celui negativ din tablou, iar dacă toate elementele tabloului sunt egale cu zero, atunci de afişat mesajul respectiv;

b) să se determine numărul şi valoarea ultimului element impar al tabloului V, iar dacă nu sunt elemente impare, atunci de afişat numărul celor negative.

**9.** Să se elaboreze algoritmul şi programul pentru următoarele:  **Este dat un tablou unidimensional de numere întregi şi pozitive M=( m*1*, m*2*,... m*n*). Să se calculeze valoarile sumelor:**

### M1 N

**S=  (e(n-j) /(j+i) -i*j* 3j/ i*j* )**

**i1= j=1**

**10.** Să se elaboreze algoritmul şi programul pentru următoarele:  **Sunt date trei tablouri unidimensionale de numere întregi şi pozitive M=( m*1*, m*2*,...,m*n* ),L=(l*1*,l*2*,...l*n*), K=( k*1*, k*2*,...,k*n*), N=25. Să se calculeze valoarea sumei:**

**M1 N**

**S=  (L*j* - Ki*j* )**

**i1=L1 j=1**

**11.** Să se elaboreze algoritmul şi programul pentru următoarele:  **Sunt date valorile elementelor tabloului A={a*1*,a*2*,...,a*n*}. Să se calculeze elementele tablourilor unidimensionale Y={y*1*,y*2*,...,y*n*} după formulele:**

**a*i*/ a*i +1*/…/ a*n* dacă a*i* > 0;**

**Y*i* = 1+a*i* +a*i +1*+…+ a*n* dacă a*i* = 0;**

**a*i* a*i +1*… a*n*, dacă a*i* < 0;**

**12.** Să se elaboreze algoritmul şi programul pentru următoarele: **Să se compună un algoritm şi să se scrie un program C pentru aflarea tuturor numerelor naturale ce nu depăşesc numărul N>9999, considerat cunoscut, şi sunt egale cu suma cuburilor cifrelor lui. Toate rezultatele să se scrie într-un tablou care să fie afişat într-un mod clar.**

**13.** Să se elaboreze algoritmul şi programul pentru următoarele: **Sunt date valorile elementelor tabloului A={a*1*,a*2*,...,a*n*}. Din aceste elemente să se determine toate perechile de numere naturale din 2 cifre M N cu proprietatea că valoarea produsului (M/2)\*N nu se schimbă dacă se schimbă locul cifrelor fiecarui factor (o atare pereche va fi de exemplu 83 şi 38 => (8/2)\*3=3\*(8/2) ). Rezultatele să se înscrie în două tablouri: M în X={x*i* } şi N în Y={y*i* }, apoi de afişat în două linii.**

**14.** Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat tabloul unidimensional X={x*i*}, i=1,…,n. Transformaţi tabloul X conform regulei următoare (y*k* - valoarea elementului k a tabloului după transformare):**

**a) y*k* = max( x*i* ) pentru 1<i<k;**

**b) elementele tabloului necesită să fie arangate în ordinea inversă;**

**15.** Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat un şir de numere x*1,* x*2*,… x*i…*. x*n*. Printre aceste numere sunt măcar două numere pozitive. Să se evidenţieze în această succesiune elementele tabloului unidimensional a*1*, a*2*,... a*k*, (k – necunoscut anticipat)- puse între o pereche de numere pozitive. Dacă k>1, atunci să se calculeze :**

**a) min(a*1*, 2a*2*,... ka*k*); b) max(a*1*, a*2* a*3*,… a*k*);**

**16.** Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat tabloul unidimensional A={ a*i* }, i=1,…,n, cu valori aleatorii, în baza căruia să se obţină un nou tablou din şirul de elemente a*1*, a*2*,... a*n* conform următoarelor scheme:**

**1) a*n*, a*n-1*,... a*1*,2a*1*, 3a*2*,...(n+1)a*n*, dacă a*1* > a*n;***

**17.** Să se elaboreze algoritmul şi programul pentru următoarele: **Se consideră vectorul A={ a*i* },1<=i<=n. Să se găsească valorile minimale şi maximale, apoi segmentul de vectori de cea mai mare lungime cu proprietatea palindromică, adică segmentul de vectori în care primul element este egal cu ultimul, al doilea cu penultimul ş.a.m.d. Pentru rezultatul obţinut să fie extrasă poziţia de început a segmentului şi lungimea sa, afişînd într-un mod clar.**

**18.** Să se elaboreze algoritmul şi programul pentru următoarele: **Sunt date tablourile X={ x*i* }, i=1,…,n; Y={y*i* }, i=1,…,n şi valoarea w în baza cărora să se calculeze valorile lui V:**

**n**

**(x*i*\*x*i* +y*i* \* y*i* + x*i* \* y*i* ), dacă w=1;**

**i=1**

**V= n j**

**x*1* +  x*j* +  y*i* ), dacă w=3;**

**j=1 i=1**

19. Să se elaboreze algoritmul şi programul pentru următoarele: se citesc ciclic cate trei intregi x,i,j, pana la introducerea lui 0, pentru fiecare grup citit afisandu-se pe cate un rand nou:

* -reprezentarea lui x in zecimal, binar, hexazecimal si octal
* -reprezentarea in zecimal, binar, hexazecimal si octal a complementului lui x

20. Sa se scrie un program interactiv care prelucreaza un tablou de max N intregi ( N - cta predefinita ), prin urmatoarele comenzi: - afiseaza maximul, suma, produsul, media aritmetica si cea geometrica

21. Să se elaboreze algoritmul şi programul pentru următoarele: se citesc interactiv care prelucreaza un tablou de max N intregi ( N - cta predefinita ), prin urmatoarele comenzi: - - cauta o valoare in tablou, pe care o sterge in cazul gasirii - dimensiunea tabloului va scadea astfel cu o unitate ( nr\_elem, nu N!!!); stergerea se realizeaza prin mutarea cu o pozitie inspre indicii mici, a tuturor elementelor de tablou urmatoare celui care se sterge;       o alta varianta este de a copia ultimul element peste cel care se sterge

22. De alcătuit schema logică şi programul, ce calculează de câte ori schimbă semnul în tabloul a1, a2, ..., aN şi aranjează în ordine descrescătoare elementele negative, neschimbând poziţia numerelor pozitive, iar zerourile se plasează la sfârşitul şirului.

23. Să se elaboreze algoritmul şi programul ca să se rezolve următoarea problemă: se dă un tablou nevid de numere cu valori aleatorii x1,x2,x3,...xn. printre care sunt măcar două numere negative. Să se evidenţieze în această succesiune elementele tabloului unidimensional a1,a2,a3,...an, cuprinse într-o pereche de numere negative. Dacă n>=1, atunci să se calculeze: max(a\*a,...a\*\*n)

24. Să se elaboreze algoritmul şi programul ca să se calculeze cel mai mare divizor comun al numerelor a(i) şi b(i), unde a(i) şi b(i) sunt valorile elementelor a două tablouri unidimensionale diferite.

25. Să se elaboreze algoritmul şi programul ca să se calculeze numărul cifrelor fiecărui număr dintr-un tablou de numere pozitive care pot avea valori cu cel puţin de 4 cifre (mii) şi de salvat în alt tablou.

Să se elaboreze algoritmul şi programul pentru următoarele: sunt date tablourile unidimensionale X={ x*i* }, i=1,…,n; Y={ Yj }, j=1,…,m. Să se elaboreze algoritmul şi programul pentru următoarele operaţii cu tablourile:

1. Z=X Ư Y - unificarea tablourilor;
2. Z=X\Y - diferenţa tablourilor (în Z conţine toate elementele din X care nu se conţin în Y).

26. Să se elaboreze algoritmul şi programul pentru următoarele:  **Este dat tabloul unidimensional V={ v*i* }, i=1,…,n, cu valori aleatorii, în baza căruia să se efectuieze următoarele:**

a) să se aprecieze numărul de ordine şi valoarea primului element pozitiv şi celui negativ din tablou, iar dacă toate elementele tabloului sunt egale cu zero, atunci de afişat mesajul respectiv;

b) să se găsească numărul de ordine şi valoarea primului element par al tabloului V, iar dacă nu sunt elemente pare, atunci de afişat numărul celor valori care se găsesc într-o succesiune anumită (creştere/ descreştere);

27. Să se elaboreze algoritmul şi programul pentru următoarele: Sunt date valorile elementelor tabloului A={a*1*,a*2*,...,a*n*}. Să se calculeze elementele tablourilor unidimensionale X={ x*1*, x*2*,..., x*n* } şi Y={y*1*,y*2*,...,y*n*} după formulele:

a*i*/ a*i+1*  dacă a*i* >0;

X*i*= a*i*+a*i+1*  dacă a*i* =0;

1- a*i*/ a*i+1*, dacă a*i* <0;

28. Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat tabloul unidimensional X={x*i*}, i=1,…,n. Transformaţi tabloul X conform regulei următoare (y*k* - valoarea elementului k a tabloului după transformare):**

**a) elementele tabloului necesită să fie mutate ciclic pe m (1<=l<=n-1) poziţii la stînga: y*n* =x*m*, …, y*k* =x*k-m* pentru k=1,...,(n-m).**

**29.** Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat un şir de numere x*1,* x*2*,… x*i…*. x*n*. Printre aceste numere sunt măcar două numere pozitive. Să se evidenţieze în această succesiune elementele tabloului unidimensional a*1*, a*2*,... a*k*, (k – necunoscut anticipat)- puse între o pereche de numere pozitive. Dacă k>1, atunci să se calculeze :**

**a)numărul cifrelor negative la pătrat printre a*1*,a*2*,...a*k***

**30.** Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat un şir de numere x*1,* x*2*,… x*i…*. x*n*. Printre aceste numere sunt măcar două numere pozitive. Să se evidenţieze în această succesiune elementele tabloului unidimensional a*1*, a*2*,... a*k*, (k – necunoscut anticipat)- puse între o pereche de numere pozitive. Dacă k>1, atunci să se calculeze :**

**a) numărul cifrelor prime la pătrat printre a*1*, a*2*,...a*k*;**

31. Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat tabloul unidimensional A={ a*i* }, i=1,…,n, cu valori aleatorii, în baza căruia să se obţină un nou tablou din şirul de elemente a*1*, a*2*,... a*n* conform următoarelor scheme:**

**1) a*1*, a*2*,... a*n*, a*n*+a*n-1*,... a*2*+a*1*, dacă a*1* < a*n;***

32. Să se elaboreze algoritmul şi programul pentru următoarele: **Este dat tabloul unidimensional A={ a*i* }, i=1,…,n, cu valori aleatorii, în baza căruia să se obţină un nou tablou din şirul de elemente a*1*, a*2*,... a*n* conform următoarelor scheme:**

**1) a*1*a*2*, a*3*a*4*,...,a*n-1*a*n,* a*1*+a*2*+...+a*n*, dacă a*i*<>a*i+1;***

1. Să se elaboreze algoritmul şi programul pentru următoarele: Să se mute la sfârşitul unui tablou toate elementele nule.
2. Să se elaboreze algoritmul şi programul pentru următoarele: Să se incarce într-un vector toate numerele prime până la **n** care, inversate, sunt tot prime.
3. Să se elaboreze algoritmul şi programul pentru următoarele: Sa se ordoneze un vector astfel incat elementele de pe pozitiile impare vor fi ordonate crescator iar cele de pe pozitiile pare sa fie ordonate descrecator.
4. Să se elaboreze algoritmul şi programul pentru următoarele: Fie doi vectori x şi y, de mărime **n**. Să se calculeze:
   1. E=min(x1,y1)+min(x2,y2)+ ......min(xn,yn)
   2. E=min(x1,yn)+min(x2,yn-1)+ ......min(xn,y1)
5. Să se elaboreze algoritmul şi programul pentru următoarele: Fie doi vectori x şi y, de mărime **n**. Să se calculeze:
   1. E=(x1+y1)\* (x2+y2)\* ......\*(xn+yn)
   2. E=x1y1+ x2y2 +......+xnyn
6. Să se elaboreze algoritmul şi programul pentru următoarele: Să se afişeze şi să se numere elementele pare de pe poziţiile impare ale unui vector.
7. Să se elaboreze algoritmul şi programul pentru următoarele: Fie doi vectori a şi b cu **m**, respectiv **n** elemente numere reale. Să se afişeze câte din componentele vectorului a sunt strict mai mici decât toate componentele vectorului b.
8. Să se elaboreze algoritmul şi programul pentru următoarele: Fie v un vector de intregi. Sa se genereze doi vectori: unul care contine elementele pozitive si altul care contine elementele negative
9. Să se elaboreze algoritmul şi programul pentru următoarele: Să se calculeze coeficienţii binomiali **Cn1,Cn2,…,Cnp** în care **n** şi **p** sunt întregi pozitivi daţi (**p** **< n**), cunoscând relaţia de recurenţă:

**Cnk = (n-k+1)/k\*Cnk-1** pornind cu **Cn0 = 1**

Se vor utiliza tablouri

Bibliografie

1. T.H. Cormen, C.E. Leiserson, R.L. Rivest: Introducere in algoritmi, Libris Agora, 2000
2. D. Lucanu: Bazele proiectarii programelor si algoritmilor, Universitatea “A.I.Cuza” Iasi, 1996
3. E. Horowitz, S. Sahni, S. Anderson - Freed *Fundamentals of Data Structures in C,* Computer Science Press, 1993
4. L. Livovschi, H. Georgescu: Sinteza si analiza algoritmilor, Ed. Stiintifica si enciclopedica, 1986
5. O. Catrina, I. Cojocaru, Turbo C++, ed. Teora 1993
6. V. Petrovici, Florin Goicea, Programarea in limbajul C. Eed. Teora 1999
7. Liviu Negrescu, ,,Limbajul C” ,volumul I\_partea I-a si partea II-a19 19ditura MicroInformatica, Cluj-napoca 2001
8. [1] Donald E. Knuth. *The Art of Computer Programming*, volume 3. Massachusetts:
9. Addison-Wesley, 1973. Есть русский перевод: Д.Кнут. Искусство программирования для ЭВМ. Т.3. Изд-во “Мир”, М.1978.