

Bogdan Pătruț

Carmen Violeta Muraru

APLICAȚII ÎN C și C++

**Editura EduSoft
Bacău - 2006**

Copyright © 2006 Editura EduSoft

Toate drepturile asupra prezentei ediții sunt rezervate Editurii EduSoft. Reproducerea parțială sau integrală a conținutului, prin orice mijloc, fără acordul scris al Editurii EduSoft este interzisă și se va pedepsi conform legislației în vigoare.

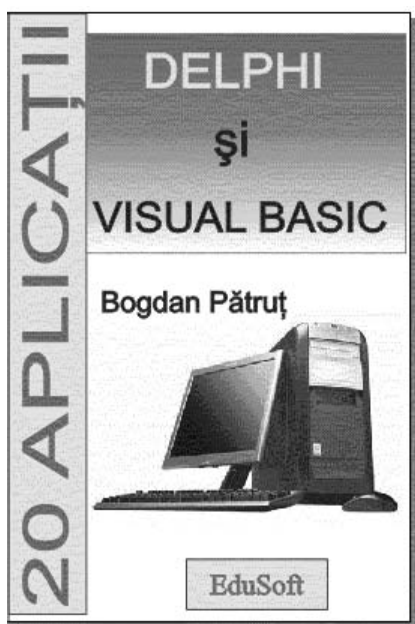
Editura EduSoft

600065 Bacău, str. 9 Mai, nr. 82, sc. C, ap. 13

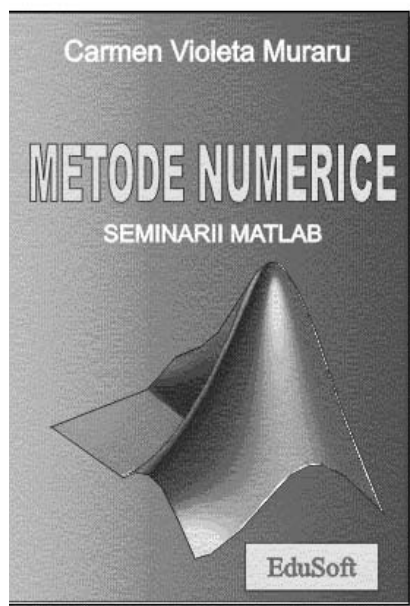
E-mail: contact@edusoft.ro, Web: www.edusoft.ro

Tel. 0234/206090, 0723 187198, 0741 638182

De aceeași autori la Editura EduSoft:



**20 aplicatii in Delphi
si Visual Basic,**
Bogdan Patrut,
ISBN 973-87655-3-6



Metode numerice.
Seminarii MATLAB,
Carmen Violeta Muraru,
ISBN 973-87655-4-4.

ISBN 973-8934-05-2

www.edusoft.ro

Introducere

Deviza acestei cărți este: *“mai clar, mai simplu, mai multe exemple.”*

Cartea se dorește a fi un supliment la manualele de C/C++ existente pe piața românească a lucrărilor de informatică. Ea este o culegere de exerciții, dar nu se limitează la nivelul acesta. După un capitol introductiv, care se constituie ca un memento al limbajului C, fiecare capitol din următoarele cuprinde două părți: una cu probleme rezolvate și una cu probleme propuse spre rezolvare.

În prima parte a fiecărui capitol, cititorul este invitat să rezolve împreună cu autorul mai multe probleme, grupate în capitole ce urmăresc, în mare măsură programa de liceu (clase de informatică). Problemele sunt prezentate gradat și au rezolvările cât mai scurte și, credem noi, mai clare, fiind însoțite și de comentarii ce urmăresc să pună în evidență unele aspecte teoretice legate fie de elementele de limbaj folosite în respectivul program, fie de algoritmul care stă la baza rezolvării problemei.

Unele dintre aceste comentarii și observații sunt marcate cu unul din simbolurile:



- reprezentând observații mai puțin importante sau ușor de reținut, referitoare, de obicei, doar la subiectul tratat în respectivul exemplu;



- reprezintă observații mai importante, cu caracter de generalitate mai mare, care se referă la elemente teoretice mai complexe și care ar fi indicat ca cititorul să și le noteze.

Programele alese sunt foarte diferite între ele (algoritmi matematici, operații la nivel de biți, algoritmi recursivi, structuri de tip înregistrare, ordonare, prelucrarea fișierelor, tehnici de programare, liste și arbori, bioritm, joc cu strategie, clase de intervale), însă odată cu trecerea la un nou program se introduc și se explică noi elemente ce țin de limbajul C.

Toate aceste programe care rezolvă problemele au fost scrise și rulate pe un calculator PC, cu compilatorul *Borland C++ 3.1*. Astfel, majoritatea exemplurilor din lucrare, chiar dacă sunt scrise în limbajul C standard (deci neobiectual), folosesc un minim de facilități aduse de limbajul C++, cum ar fi comentariile cu simbolurile *“//”* și declararea variabilelor simultan cu inițializarea lor, în cadrul unor instrucțiuni repetitive.

Cea de a doua parte a fiecărui capitol cuprinde o serie de probleme care se rezolvă foarte asemănător celor rezolvate și au diferite grade de dificultate, unele dintre ele au putând fi folosite în concursuri de programare pentru elevi și studenți. Cele opt capitole totalizează un număr de peste 400 de probleme propuse.

Mulți programatori începători consideră dificil de învățat limbajul C. El este un limbaj "greu" mai ales pentru cei care s-au obișnuit să lucreze în *Pascal*. Ei nu înțeleg adesea cum se evaluează unele expresii sau aritmetica pointerilor, precum și folosirea structurilor sau a fișierelor.

Prin problemele rezolvate prezentate în această lucrare, am dorit să venim tocmai în întâmpinarea acelor programatori, care fac trecerea la limbajul C, plecând de la un alt limbaj de programare (*Pascal*, *Basic*, *Fortran*), fie că aceștia sunt elevi de liceu (cărora cartea li se adresează cu precădere), studenți, profesori de informatică sau chiar programatori experimentați, tuturor celor care vor să pătrundă cu ușurință în minunata lume a puternicului limbaj *Borland C/C++*.

Problemele propuse spre rezolvare vin în sprijinul elevilor și studenților informaticieni, dar și al profesorilor lor, care vor găsi în această lucrare o bogată sursă de exerciții și probleme ce pot fi rezolvate la tablă, în clasă, sau se pot constitui în teme pentru acasă. De aceea am inclus un capitol special cu probleme recapitulative.

Cu speranța că parcurgând atent această carte, cititorii vor privi cu ochi mai prietenoși limbajul C, le dorim lectură plăcută și... compilare rapidă și fără erori!

Autorii

Capitolul 0. Scurtă introducere în limbajul C

Limbajul C a fost dezvoltat la începutul anilor 1970 de Ken Thompson și Dennis Ritchie, care aveau nevoie de un limbaj simplu și portabil pentru scrierea nucleului sistemului de operare Unix. Este implementat pe marea majoritate a platformelor de calcul existente azi, și este cel mai popular limbaj de programare pentru scrierea de software de sistem.

Identificatori

În C numele de variabile, constante, tipuri de date definite de utilizator, funcții, etc sunt denumite identificatori. Aceștia sunt succesiuni de litere sau cifre, în care primul caracter să nu fie cifră.

Constantele

Sunt valori care nu se modifică pe parcursul programului. Dacă li se asociază un identificator, ele pot fi referite apoi pe parcursul programului cu acest identificator

Exemplu

Const int a=10 ;

Utilizarea calificativului **const** din declarație certifică faptul că valoarea respectivă nu poate fi modificată de către program.

Tipuri de date: reprezintă o mulțime omogenă de date. Tipul de date se identifică printr-un identificator, plajă de valori atașată tipului. Toate datele dintr-un anumit tip sunt reprezentate intern, în memoria calculatorului, în același mod. În C tipurile de date pot fi predefinite (de bază) și definite de utilizator (derivate). Un tip de date este caracterizat de dimensiunea zonei de memorie alocate unui element. Dimensiunea și domeniul de valori ale acestor tipuri de date pot varia însă de la un procesor la altul. Totuși întotdeauna un caracter echivalează cu un octet.

Tipul	Intervalul	Numar de octeti
char,	[-128, 128]	1
unsigned char	[0, 255]	
short,	[-32768, 32767]	2
unsigned short	[0, 65535]	
int,	[-32768, 32767]	2
unsigned int	[0, 65535]	

long,	[-2147483648,2147483647]	4
unsigned long	[0, 429467259]	
float	Valoarea absoluta in [3,4*10 ⁻³⁸ , 3.4*10 ³⁸]	4
double	Valoarea absoluta in [1.7*10 ⁻³⁰⁸ , 1.7*10 ³⁰⁸]	8
long double	Valoarea absoluta in [3.4*10 ⁻⁴⁹³² , 1.1*10 ⁴⁹³²]	10

Tabel 1

Variabilele sunt caracterizate de elementele: *nume, adresa de memorie, un tip de date și valoarea*. Locația de memorie (adresa), stochează valoarea variabilei la un moment dat. Prin urmare variabilele sunt date care își pot modifica valoarea pe parcursul execuției programului. Toate variabilele dintr-un program trebuie declarate înainte. Inițializarea variabilei se poate face simultan cu declararea sa. Forma generală a unei declarații este :

Tip lista `_variabile` ;

Dacă aveți mai multe variabile de același tip ele pot fi declarate, înșirând numelor lor separate de virgulă în dreptul tipului de bază, pe aceeași linie.

Exemplu :

```
int a, b ;
int m=1 ;
char c ;
float x, y=5 ;
```

Variabilele locale

Variabilele declarate în interiorul unor funcții sunt numite variabile locale. Aceste variabile în C, se mai numesc și variabile automate. Aceste variabile nu vor fi recunoscute în afara blocului funcției.

Exemplu : `void functie_1(void)`

```
{
    int x,y ;
    x=10 ;
    y=x+3 ;
}

void functie_2(void)
{
    int x ;
    x=-123 ;
}
```

De remarcat că variabila x, declarată în **funcție_1** nu are nici o legătură cu variabila x din **funcție_2**. În C avem posibilitatea să declarăm variabilele nu numai imediat după acolada, ci și în interiorul blocului funcției.

```
void f (void)
{
    int s ;}
```

Un avantaj al declarării unei variabile locale în interiorul unui bloc este că memoria pentru variabila va fi alocată numai dacă e necesar. Variabilele locale sunt create la fiecare întoarcere în blocul funcției și distruse la ieșirea din el.

Observație *Limbajul C face diferența dintre literele mici și cele mari.*

Inserarea de comentarii

Uneori pentru o mai bună înțelegere a programului, putem adăuga acestuia comentarii.

Comentariile se inserează în program în două moduri. Prima posibilitate este să inserați caracterele slash //, înaintea textului vostru.

```
// Acesta este un comentariu
```

A altă posibilitate este să scrieți textul între caractere slash și asteriscuri, astfel

```
/* Acesta este un comentariu*/
```

La întâlnirea simbolurilor pentru inserarea comentariilor, compilatorul ignoră aceste mesaje, ele având importanță doar pentru programator.

Parametrii formali

Într-o funcție care folosește argumente trebuie declarate variabilele care acceptă valorile argumentelor, variabile numite parametri formali ai funcției

Observație

Parametrii formali declarați trebuie să fie de același tip cu argumentele folosite la apelarea funcției. Un parametru este o valoare transmisă unei funcții. Cele mai multe dintre programe transmit parametri către funcția **printf**:

```
printf ("valoarea este %d\n", rezultat);
```

Exemplu

```
void info_angajat (int varsta, float salariu, int nr_marca).
```

```
{
```

```
// instrucțiunile funcției;
```

}

Variabilele `vârstă`, `salariu`, `nr_marca`, reprezintă parametrii formali ai funcției **info_angajat**

Parametrii formali se alocă pe stivă ca și variabilele automate. De aceea, ei se consideră a fi variabile locale și sunt utilizabili numai în corpul funcției în antetul căreia sunt declarați.

La apelul unei funcții, se alocă pe stiva parametri formali, dacă există, li se atribuie valorile parametrilor efectivi care le corespund. Apoi se alocă pe stiva variabilele automate declarate la începutul corpului funcției respective.

La revenirea din funcție, se realizează curățirea stivei, adică sunt eliminate de pe stivă (dezalocate) atât variabilele automate, cât și parametrii. În felul acesta, la revenirea din funcție, stiva ajunge la starea dinaintea apelului.

Variabile globale

Spre deosebire de cele locale, acestea sunt recunoscute de către toate componentele programului.

Observație Dacă o variabilă locală și una globală au același nume, trimerile la numele acelei variabile din interiorul blocului de cod în care a fost declarată variabila locală se vor referi numai la variabila locală și nu vor afecta variabila globală. Zona de alocare a memoriei pentru variabilele locale este o regiune fixă de memorie, specială destinată în acest scop de către compilatorul de C. Variabilele globale sunt utile când un număr mare de funcții din programul d-voastră folosesc aceleași date. În schimb variabilele globale inutile vor ocupa spațiu de memorie pe toată durata execuției programului.

Expresii

Expresiile sunt formate din unul sau mai mulți operanzi și operatori. Un **operand** poate fi o constantă, numele unei variabile, numele unei structuri, numele unei funcții sau chiar o expresie între paranteze rotunde.

Prioritatea operatorilor este cea care determină ordinea de efectuare a operațiilor. În C se definesc următorii operatori: aritmetici, relaționali, logici și pe biți.

În funcție de aceasta avem următoarea grupare a operatorilor:

Operatorul de atribuire se folosește în interiorul oricărei expresii valide în C. Forma generală a acestui operator este:

`nume_variabila=expresie`

Observație

Când variabilele de un tip sunt combinate cu variabile de alt tip se produce o conversie de tip. Regula acestei conversii este ca valoarea membrului drept (expresia), este convertită la tipul membrului stâng.

Operatorii de incrementare și decrementare

Operatorii de incrementare ++ și decrementare -- sunt doi operatori utili ce în general nu mai apar în alte limbaje de programare. Operatorul de incrementare adaugă o unitate la operand, iar cel de decrementare scade o unitate.

Exemplu :

$x=x+1$ este echivalent cu $++x$;

și

$x=x-1$ este echivalent cu $--x$;

Cei doi operatori pot precede sau urma operandului, existând diferențe între cele două forme.

Exemplu

$a=2$;

$b=++a$;

stabilește valoarea lui b la 3 ;

$a=2$;

$b=a++$;

stabilește valoarea lui b la 2.

Operatorii de pointer & si *

Pointer-ul este adresa de memorie a unei variabile. Operatorul de pointer & este un operator unar care returnează operandului său adresa de memorie.

Exemplu

$a=\&num$;

unde în a se va plasa adresa variabilei **num**, adresa care reprezintă doar localizarea internă a variabilei și nu are legătură cu valoarea ei.

Operatorul * este complementul operatorului adresa și calculează valoarea variabilei localizate la adresa ce urmează.

$m=*a$, unde a este adresa variabilei **num**.

*Variabilele care memorează adrese (pointeri) trebuie declarate cu simbolul * înaintea numelui variabilei.*

Astfel în exemplul **char** *ch, ch nu este o variabilă caracter ci un pointer la caracter.

Operatorul modulo

În cazul în care vă interesează restul unei împărțiri, limbajul C vă oferă operatorul *modulo* (rest)- %

Exemplu

```
# include <stdio.h>
void main (void)
{
    int rest;
    rest =110%3;
    printf ("110 împărțit la 3 dă restul %d\n", rest);
}
```

Orice program în C are următoarea structură:

- directive de procesare
- declarații globale de variabile și funcții;
- definiții de funcții ;

Orice program în C poate fi împărțit pe mai multe secțiuni. Pentru a asocia anumite instrucțiuni unei secțiuni, se includ acele instrucțiuni între acolade ({}). Aceste acolade fac parte din sintaxa limbajului. Orice acoladă deschisă trebuie să avari și una închisă.

Preprocesarea

Înainte de a fi compilat, un program poate fi prelucrat (preprocesat). Utilizatorii limbajului C și C++ pot folosi o serie de funcții aflate în bibliotecile standard ale acestor limbaje. Prototipurile acestor funcții se găsesc în fișiere **header (.h)**.

Una din directivele de preprocesare este **#define**, folosită pentru includerea unui fișier header poate avea una din formele :

include "specificator_de_fișier "

include <specificator_de_fișier>

Modul de încadrare a numelui fișierului controlează modul de cautare a fișierului indicat : " ", căutarea se face mai întâi în directorul curent și apoi în cele standard pentru include și <>, căutarea se face doar în directoarele standard.

Dacă nu ați inclus într-un program fișierul sursă de care programul are nevoie, compilatorul va afișa un mesaj de eroare de forma :

Warning...Function ' printf ' should have a prototype in function main().

Limbajul C pune la dispoziția utilizatorului funcții pentru prelucrarea datelor, grupate în biblioteci.

- stdio.h și io.h pentru citire/scriere
- conio.h pentru interfata cu consola
- graphics.h pentru interfata grafica
- ctype.h pentru prelucrarea caracterelor

- stdlib.h si math.h pentru prelucrari numerice
- dos.h pentru interfața cu sistemul de operare DOS

Exemplu :

include <stdio.h>

Directiva **#define** definește o macrocomandă ce înseamnă definirea unui simbol care înlocuiește o porțiune de cod C. În momentul preprocesarii, fiecare apariție a numelui macrocomenzii va fi înlocuită cu definiția sa.

Exemplu :

define *nume* succesiune_de_caractere

define PI 3.14159

Ca urmare PI va fi înlocuit cu valoarea specificată mai sus, până la sfârșitul codului de program.

define XY acesta este un test

Astfel XY se va înlocui acolo unde se întâlnește cu sirul de caractere " acesta este un test ". Foarte utilă este reprezentarea macrocomenzilor de tip funcție.

Exemplu

define Abs (a) (a)<0 ? -(a) :(a)

E interesant de observat că numele macrocomenzii poate avea argumente.

Să definim macrocomenzile pentru pătrat și cub :

define pătrat(x) ((x)*(x))

define cub(x) ((x)*(x)*(x))

O altă categorie de directive o reprezintă cele condiționale. În această categorie avem : **#if**, **#ifndef**, **#else**, **#elif** și **#endif** ce controlează care părți ale fișierului sursă vor fi compilate și în ce condiții.

Funcții de intrare-ieșire formatată

În C, funcțiile care pot scrie și citi date în diferite formate stabilite de programator sunt funcțiile **printf ()** și respectiv **scanf ()**.

Funcția de scriere cu format printf ()

Funcția realizează afișarea datelor în consolă , folosind specificatorii de format ce indică modul în care vor fi afișate argumentele care urmează. Un specificator de format începe cu simbolul % și este urmat de codul formatului.

Prototipul funcției este :

printf (const char *format [, lista argumente]) ;

Numărul de argumente trebuie să fie egal cu cel al specificatorilor de format.

Această funcție se află în STDIO.H, biblioteca funcțiilor de intrare-ieșire.

Funcția **printf ()** acceptă o gamă largă de specificatori de format:

cod	Format
%d	intreg in baza 10 cu semn
%i	intreg in baza 8,10 sau 16 cu semn
%o	octal fara semn
%x	hexazecimal fara semn (litere mici)
%u	intreg fara semn
%e %E	Numar real de forma iii.zzzzzz (numarul implicit de zecimale 6)
%f	numar real de forma i.zzzzzz (nr de zecimale implicit 6)
%g %G	nr. real ce suprima caracterele terminale ce nu influenteaza valoarea, adica cifre de zero
%c	caracter
%s	sir de caractere

Tabel 2

Observații:

Pentru a genera la ieșire o valoare fără semn se folosește **%u**. Specificatorul de format **%f** afișează număr în virgulă mobilă, iar specificatorii **%e** si **%E** pentru argumente în format **double**.

Exemplu

1. **printf** (" Acesta este un %s", "test.");

afiseaza pe ecran

Acesta este un test.

2. **# include** <stdio.h>

void main (void)

{

int x;

for (x=1;x<10;x++)

printf ("%d", x)

}

Programul afisează pe ecran secvența de numere:

1, 2, 3, 4, 5, 6, 7, 8, 9

3. **# include** <stdio.h>

void main (void)

{

int a,b;

```

.....
printf ("suma numerelor este %d", a+b );
}

```

Programul va afișa suma numerelor a și b introduse de la tastatură.

Funcția **printf ()** se folosește de altfel și pentru afișarea de mesaje.

```
printf ( " Acesta este un test \n " ) ;
```

Modificatori de format

Modificatorul de format se înserează între simbolul procentului și codul formatului, specificând lățimea minimă a câmpului, numărul de cifre după virgulă sau alinierea la stânga.

Specificatorul lățimii minime de câmp este un întreg plasat între simbolul procentului și codul formatului, indicând numărul minim de spații pe care se afișează numărul. Dacă șirul sau numărul este mai lung decât minimum, va fi afișat în întregime.

Exemplu

```

. # include <stdio.h>
    void main (void)
    { double a ;
      a=10.3456 ;
      printf ( " %f \n " , a ) ;
      printf ( " %10f \n " ,a ) ;
      printf ( " %012fm " ,a ) ;
    }

```

Programul va afișa :

10.345600 - adaugă două zerouri până la 6, valoarea implicită a numărului de zecimale

10.345600 - câmpul minim pe care se tipărește numărul este de 10 spații

00010.345600 - se adaugă zerouri în fața până se completează cu dimensiunea câmpului minim

Directivile de formatare pot avea opțional și alte componente :

% fanion dimensiune. specificator de precizie

Fanioane pentru funcția printf(), cu excepția lui * (pt scanf)

- : aliniaza valoarea la stânga într-un câmp de dimensiune data
- + : pune + înaintea unui număr pozitiv de tip cu semn
- spatiu* : pune spațiu înaintea unui număr pozitiv de tip cu semn

Specificatorul de precizie

Acest specificator urmează specificatorului lăţimii minime de câmp (când acesta există). Se reprezintă printr-un întreg precedat de un punct. Dacă se aplică numerelor reale, acesta specifică numărul de zecimale cu care va fi afişat numărul. De exemplu **%9.3f** afişează un număr de cel puțin 9 caractere având 3 cifre după virgulă.

Exemple de scriere formatată

```
printf (" %f\n ",1.0) ; /* 1.000000 : 6 cifre zecimale
```

```
printf (" %.2f\n ",1.009) : /*1.01 : 2 cifre zecimale
```

```
printf (" %e\n ",1.0) ; /* 1.000000+00e+00 : 6 cifre zecimale
```

Scriere de numere întregi în formă de tabel :

```
printf (" |%6d| ", -12) ; /* | -12| */
```

```
printf (" |%-6d| ", -12) ; /* |-12 | */
```

```
printf (" |%+6d| ", -12) ; /* | +12| */
```

```
printf (" |%06d| ", -12) ; /* |-00012| */
```

Când se doreşte ca **printf()** să înceapă afişarea pe o linie nouă, trebuie să includeti un caracter special, linie nouă (**\n**), în interiorul textului pe care **printf** urmează să-l afişeze. Astfel la întâlnirea tabulatorului, **printf** va avansa cursorul la linia următoare.

Exemplu

```
# include <stdio.h>
```

```
void main (void)
```

```
{  
    printf (" Aceasta este linia unu \n");  
    printf (" Aceasta este linia a doua");  
}
```

Funcția de citire cu format **scanf ()**

Cu ajutorul acestei funcții se face introducerea datelor de la tastatură.

Prototipul acestei funcții este :

```
scanf ( const char *format [, lista_adrese_variabile])
```

și o găsim tot în biblioteca de funcții **STDIO.H**. Constantele format le găsim în Tabelul 2 pentru diferite tipuri de date. Funcția citește o secvență de argumente de intrare, caracter cu caracter până la terminarea șirului de argumente sau până la apăsarea tastei Enter.

De exemplu, pentru citirea unei linii de forma:

3, 4, 5

se va scrie următoarea secvență:

```
int a, b, c;
```

```
scanf ("%d %d %d", &a, &b,, &c);
```

Observații

Toate variabilele folosite pentru a primi valori cu funcția **scanf ()**, se transmit utilizând adresele acestora, așa cum s-a văzut și în exemplul de mai sus.

Funcții pentru citirea și afișarea caracterelor

Cele mai simple funcții pentru citirea și afișarea caracterelor la consolă sunt **getchar()**, care citește un caracter de la tastatură și **putchar()**, care scrie un caracter pe ecran. Funcția **getchar ()** așteaptă apăsarea unei taste și apoi returnează valoarea Ascii a acesteia. Tasta apăsată se va scrie pe ecran. Forma celor două funcții :

```
int getchar (void);
```

```
int putchar (int c);
```

Aceste funcții se găsesc în fișierul antet **STDIO.H**

Următorul cod de program va citi și va afișa caractere până la caracterul linie nouă:

```
do
```

```
{ litera=getchar ();
```

```
    putchar (litera);
```

```
}
```

```
while (litera !='\n');
```

La funcția **getchar()** există și funcții alternative, două dintre cele mai folosite sunt : **getch()** și **getche()**. Prototipurile acestor funcții sunt :

```
int getch (void);
```

```
int getche (void);
```

și pot fi găsite în fișierul **CONIO.H**.

Cele două funcții așteaptă apăsarea unei taste, după care returnează imediat. Deosebirea e că la **getch()** caracterul nu apare pe ecran pe când la **getche()** el se tipărește pe ecran.

Exemplu :

```
# include <stdio.h>
```

```
# include <ctype.h>
```

```
# include <conio.h>
```

```
void main (void)
```

```
{
```

```
    int litera ;
```

```
    do
```

```
    { litera=getche ();
```

```
        litera =toupper (litera) ;
```

```
    }
```

```
    while (litera !='A') ;
```

Programul de mai sus citește caractere de la tastatură utilizând citirea directă. Se introduc litere până la tastarea lui 'A', litere ce sunt transformate în majusculă.

Funcții

O funcție are o definiție și atâtea apeluri într-un program, câte sunt necesare. O definiție de funcție are formatul:

1. antet

2. corp

tip nume (lista declarațiilor parametrilor formali)// antet

```
{declaratii de variabile locale  
    instrucțiuni terminate cu ;  
}
```

O funcție poate folosi sau nu parametrii. “Un parametru este o informație pe care programul o transmite funcției” [K. Jamsa]. Dacă nu se folosesc parametri atunci după numele funcției trebuie să plasați între paranteze cuvântul **void**. Utilizarea unei funtii în program se numește defapt **apel de funcție**. Orice program în C conține o funcție principală, **main**, în care sunt apelate celelalte funcții ale programului, dacă ele există.

Exemplu

```
#include <stdio.h>
```

```
void main (void)  
{  
    lista_instrucțiuni urmate de ;  
}
```

Apelul unei funcții trebuie să fie precedat de definiția sau de prototipul ei.

El poate avea același format ca și antetul funcției, în plus este urmat de punct și virgulă.

Prototipul unei funcții conține informații asemănătoare cu cele din antetul ei:

1. **tipul valorii returnate;**

2. **numele functiei;**

3. **tipurile parametrilor.**

Tip- tipul valorii returnate de funcție (implicit este întreg). Pentru cele care nu returnează nimic tipul este void. Funcțiile care nu returnează nimic sunt

echivalentul procedurilor din Pascal. Lista declarațiilor parametrilor formali, separați prin virgule trebuie specificați cu numele și tipul lor. Această listă în anumite condiții poate fi vidă.

Exemplu

```
void f (void)
int g ()
main ()
```

Parametrii utilizați la apel se unesc parametri actuali sau efectivi. Transmisia parametrilor se face prin valoare. Folosind apelul prin valoare , modificările parametrilor funcției vor exista doar în cadrul funcției apelate. Astfel la sfârșitul execuției funcției, valorile variabilelor transmise de program funcției rămân nemodificate.

Orice program în C conține o funcție principală ce conține instrucțiunile care se execută cât și apelurile de alte funcții care apar într-un program.

Observație

1. Toate funcțiile care returnează valori trebuie să conțină instrucțiunea **return..** Dacă nu există nici o instrucțiune return, atunci valoarea returnată a funcției este nedefinită din punct de vedere tehnic.
2. Parametrii actuali să corespundă cu cei formali prin ordine și tip
3. La apel se atribuie parametrilor formali valorile parametrilor actuali, apoi execuția se continuă cu prima instrucțiune a funcției apelate.

Exemplu de funcție cu return

```
int a_patrat ( int valoare)
{
    return (valoare*valoare);
}
```

Să scriem acum codul funcției principale în care se face apelul funcției de mai sus pentru o anumită *valoare*.

```
#include <stdio.h>
```

```
int a_patrat ( int valoare)
{
    return (valoare*valoare);
```

```
void main (void )
```

```
{
    printf ( " Pătratul lui 4 este %d\n ", a_patrat(4)) ;
}
```

Comentariile încep cu secvența de caractere `/*` și se termină cu `*/`. Ele sunt ignorate de compilator.

Instrucțiuni în C

În C instrucțiunile sunt grupate în câteva categorii:

- selecție
- iterație
- salt
- expresie
- eticheta
- bloc

Instrucțiunile de selecție (decizie)

Valorile de adevăr și fals sunt date de expresiile condiționale. În C, o valoare de "adevăr" înseamnă orice valoare diferită de zero, inclusiv numere negative. O valoare "fals", înseamnă o valoare de zero.

Avem două instrucțiuni de selecție în C : **if** și **switch**. Mai există un operator în C care uneori poate constitui o alternativă la **if**.

Exemplu :

`min=(a<b) ?a :b`

Instrucțiunea if (dacă)

Forma acestei instrucțiuni este :

if (expresie) instrucțiune ;
else instrucțiune ;

Instrucțiunea ramifică programul în două în urma evaluării unei expresii logice . Componenta **else** este opțională. Dacă valoarea expresiei este adevărată se execută instrucțiunea de după **if** ; dacă nu, se va executa instrucțiunea de după **else** dar niciodată amândouă.

Testarea unei egalități se realizează tot cu ajutorul instrucțiunii **if**, utilizând semnul egal dublu (`==`). De remarcat că un singur egal se utilizează doar la instrucțiunea de atribuire.

if (`a== 0`)
 instrucțiune

Observație

Limbajul C nu operează cu tipul boolean, valorile de adevăr fiind codificate numeric, după următoarea regulă: o valoare nulă este echivalentă cu fals iar o valoare nenulă cu adevărat.

Exemplu

Determină minimul a doua numere:

```
int a,b ;
if (a<b) min=a;
    else min=b;
```

O variantă pentru instrucțiunea **if** este operatorul ternar **?**, care are următoarea formă generală:

exp1? exp2: exp3

Valoarea unei expresii **?** se determină astfel: se evaluează exp1. Dacă această este adevărată, atunci se evaluează și exp2, care devine valoarea întregii expresii. Pentru exp1 falsă, atunci se evaluează exp3 și valoarea ei devine valoarea întregii expresii.

Astfel codul :

```
x=3 ;
y=x>2 ? 30 :200
```

este echivalent cu instrucțiunea **if** :

```
x=3;
if (x>2) y=30;
    else y=200
```

În cazul în care avem mai multe căi de decizie, vom folosi structuri **if** imbricate:

Exemplu

```
if (x<0) e=x*x;
    else if (x==0) e=5.0;
        else e=x+4;
```

Se observă că în C nu e greșit să punem **;** înaintea lui **else**, așa cum eram obișnuiți în Pascal.

Instrucțiunea switch

Este o instrucțiune de decizie multiplă și are următoarea sintaxă:

```
switch (expresie)
{ case const_1 : instrucțiune_1 ;[break ;]
  case const_2 : instrucțiune_2 ;[break ;]
  .....
  case const_n : instrucțiune_n ;[break ;]
  [default: secvența]
}
```

Se compară valoarea expresiei cu fiecare din valorile constantelor exprimate în instrucțiunile **case**. Când se descoperă o egalitate, secvența de instrucțiuni

asociată acelei instrucțiuni **case** se execută până la instrucțiunea **break** sau până la sfârșitul instrucțiunii **switch**. Instrucțiunea **default** se execută când nu se găsește nici o egalitate. Instrucțiunile **break** și **default** sunt facultative.

Observatii

1. Instrucțiunea **switch** spre deosebire de **if**, efectuează doar teste de egalitate, în timp ce **if** poate evalua orice expresie de tip relațional sau logic.

2. Instrucțiunea **switch** se folosește cu succes în cazul prelucrării comenzilor provenite de la tastatură, cum ar fi selecția meniurilor.

Exemplu

```
switch (op){
    case '*': înmulțește(a, b);
        break;
    case '+': adună(a, b);
        break;
    case '-': scade(a, b);
        break;
    default : printf("eroare");}
```

Instrucțiuni de ciclare

Instrucțiunea for

Este o instrucțiune repetitivă cu număr cunoscuți de pași.

Are următoarea formă generală :

for (inițializare ; condiție ; increment)

Pe lângă aceste forme mai există și alte variații în funcție de contextul problemei.

Inițializarea este în general o instrucțiune de atribuire care stabilește valoarea de început a contorului ciclului. *Condiția* este o expresie relațională care determină când se încheie ciclul. *Incrementul* definește regula de modificare a variabilei contor pe parcursul buclei.

Instrucțiunea **for** se execută atâta timp cât condiția este adevărată. Când aceasta devine falsă programul continuă cu instrucțiunea imediat următoare lui **for**.

Variațiile instrucțiunii **for** rezultă din ideea că nu este necesară existența tuturor celor trei elemente ale buclei **for**, prezentate mai sus.

Exemplu

```
for (x=1 ; x !=100 ; )
    scanf(" %d ", &x)
```

Se observă inexistența zonei rezervate "increment". Aceasta înseamnă că la fiecare rulare a ciclului **x** este comparat cu 100 și atâta timp cât nu găsește

egalitate se reia bucla. Dacă se introduce însă 100 de la tastatura se încheie execuția buclei și programul continuă cu instrucțiunea de citire a variabilei. Există situații când variabila contor se inițializează în afara instrucțiunii **for** și atunci “ inițializare ” poate lipsi. Astfel :

Exemplu

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

Se observă în ambele cazuri ca semnul ; nu lipsește din sintaxa chiar dacă unele elemente lipsesc.

Deasemenea cu ajutorul unei variații a ciclului **for** se poate exprima un ciclu infinit astfel :

```
for ( ; ; )
```

Exemplu :

```
for ( ; ; )  
{ ch=getchar() ;/* scrie un caracter  
  if (ch == 'A') break ; /*se încheie ciclul  
}
```

Ciclul de mai sus se va executa până când se va tasta un A de la tastatura. Instrucțiunea **break** va determina încheierea imediată a instrucțiunii **for**, în acest caz.

Limbajul C va permite să inițializați și să incrementați mai multe variabile într-o bucla **for**, separându-le prin virgula.

```
for(i=0,j=0 ;i<=100 ;i++ , j++)
```

Dacă se folosesc variabile multiple în **for**, atunci vom utiliza *bucle imbricate* (vezi matrici).

Instrucțiunea while

Este o instrucțiune repetitivă cu test inițial și număr necunoscut de pași. Expresia generală a instrucțiunii este :

```
while (conditie) instructiune ;
```

Condiție poate fi orice expresie, care dacă este adevărată (valoare diferită de zero) va determina continuarea buclei **while**. Când condiție va fi falsă, execuția programului continuă cu instrucțiunea imediat următoare lui **while**.

Instrucțiunea **while** poate fi considerată o variație a instrucțiunii **for**, caz în care lipsesc elementele: “inițializare” și “ increment”.

```
for ( ; conditie ; ) instructiune
```

Exemplu 1

```
int n,i,s;  
.....  
i=0; s=0;  
while (i<n) {s+=i; i++}
```

Exemplu 2 (K. Jamsa)

```
printf (" doriți să continuați ? (D/N) : " ) ;
litera= getch ( ) ;
    litera =toupper (litera) ;
while ((litera !='D') && (litera !='N'))
{
    putchar (7)    // emite un sunet
    litera =getch() ;
    litera =toupper (litera) ;
}
```

Instrucțiunea do-while

Spre deosebire de ciclurile **for** și **while** care testează condiția executării la început , ciclul **do-while** verifică condiția la sfârșit. Prin urmare un ciclu **do-while** se execută cel puțin odata. Forma generala este:

```
do {
    instructiune;
} while (conditie);
```

De amintit că acoladele nu se folosesc în cazul unei singure instrucțiuni ci doar când avem bloc de instrucțiuni.

Exemplu

```
int i,n, s ;
.....
i=1; s=0;
do {s+=i;
    i++; } while (i<=n);
```

Observatie

Ciclul **do-while** este foarte des utilizat la functii de selectie a meniurilor.

Exemplu 2 scris cu ajutorul instrucțiunii **while**, poate fi modificat cu **do-while**, astfel :

Exemplu (K. Jamsa)

```
printf (" doriți să continuați ? (D/N) : " ) ;
do
{ litera= getch ( ) ;
  litera =toupper (litera) ;
  if ((litera !='D') && (litera !='N'))
      putchar (7);    //emite un sunet
}
while ((litera !='D') && (litera !='N'))
}
```

Următorul cod de program realizează suma numerelor pozitive introduse de la tastatură. Programul se încheie la introducerea primului număr negativ.

Exemplu

```
sum = 0.0;
scanf ("%d", &x);
do {
    sum += x;
    scanf ("%d", &x);
}
while (x > 0.0);
```

Instrucțiuni de salt

1. Instrucțiunea return

Este o instrucțiune ce nu trebuie să lipsescă din corpul unei funcții care returnează o valoare, deoarece obține rezultatul funcției. Este considerată de salt pentru că determină programul să revină la punctul în care s-a făcut apelarea funcției. Dacă instrucțiunii **return** i se asociază o valoare, acea valoare devine valoarea funcției.

Forma generală a instrucțiunii :

return expresie ;

unde expresie este opțională și când există va deveni valoarea returnată de funcție.

Astfel o funcție declarată **void** nu poate conține o instrucțiune **return** care specifică o valoare.

2. Instrucțiunea break

Această instrucțiune așa cum s-a văzut și mai sus , poate fi folosită în primul rând pentru a încheia execuția unei instrucțiuni **case** aflate într-o instrucțiune **switch**. Dar poate fi folosită și pentru încheierea unui ciclu, prin omiterea testului conditional al ciclului.

```
# include <stdio.h>
```

```
void main (void)
```

```
{
```

```
    int x;
```

```
    for (x=1; x<20;x++) {
```

```
        printf ("%d",x);
```

```
        if (x==10) break;
```

```
    }
```

```
}
```

În exemplul de mai sus se scriu pe ecran numerele de la 1 la 10, și când s-a tipărit 10, se iese imediat din ciclu, ignorând testul condițional $x < 20$, datorită instrucțiunii **break**.

3. Instrucțiunea continue

Spre deosebire de instrucțiunea **break**, instrucțiunea **continue** forțează următoarea iterație a ciclului, prin omiterea oricăror linii de program dintre cele două iterații. Astfel în ciclul **for** această instrucțiune realizează saltul la pasul de reinițializare. Pentru ciclurile **while** și **do-while**, realizează direct evaluarea expresiei ce decide continuarea ciclului.

Exemplu

4. Instrucțiunea goto

Are sintaxa următoare:

goto eticheta;

Se realizează saltul necondiționat la instrucțiunea a cărei etichetă este cea specificată în dreptul instrucțiunii. De reținut este faptul că nu se face salt în afara funcției curente. Este recomandabil restrângerea utilizării acestei instrucțiuni doar atunci când este neapărat necesar, de exemplu pentru optimizarea programului.

Exemplu

```
#include <stdio.h>
void main (void)
{
    int numar=5 ;
    eticheta :
        printf (" %d ", numar ++);    //afișează numerele de la 5 la 100
        if (numar <=100)
            goto eticheta;
}
```

Tablouri și șiruri

Tabloul (array) este o structură de variabile de același tip. Tipul variabilelor se numește tipul de bază al tabloului, iar numărul componentelor unui tablou (variabile) este determinat de numărul de valori ale tipului indicelui. Pentru declararea unui tablou trebuie specificate tipul de bază cât și dimensiunea acestuia, adică numărul maxim de elemente. La declarare compilatorul de C alocă suficientă memorie pentru a putea reține toate elementele. Prima intrare este locația 0, iar ultimul element va apărea la o locație cu unul mai mică decât dimensiunea tabloului.

Observație

C nu are facilități de verificarea depășirii limitelor pentru tablouri. Așa că rămâne în sarcina programatorului să se asigure de depășirea limitelor acolo unde este nevoie. Liniile respective vor fi compilate fără erori dar sunt incorecte.

Fiecare element dintr-un tablou este dat de un indice (vector) sau doi indici (matrici). *Acești indici sunt obligatoriu numere întregi și indică poziția elementului în cadrul tabloului.* Elementele tabloului sunt ordonate de acești indici. Tablourile pot avea una sau mai multe dimensiuni. În cadrul variabilelor de tip tablou identificăm vectorii și matricile pentru cazul tablourilor unidimensionale, respectiv bidimensionale. Tablourile trebuiesc declarate la fel ca orice altă variabilă.

tip nume_variabilă[dimensiune] */ pentru variabile tablou unidimensionale

tip nume_variabilă[dimensiune][dimensiune] */ pentru variabile tablou biidimensionale

Exemplu :

```
int a[100] ;
```

```
float lista[30] ;
```

```
float matrice [20][30]
```

Pentru a putea accesa un element al tabloului se specifică numele tabloului urmat de indicele elementului între paranteze pătrate.

Exemplu

```
lista [3]=30 ;
```

```
a[8] ;
```

```
a[i] ;
```

Pentru parcurgerea elementelor unui tablou putem utiliza o instrucțiune repetitivă folosind ca variabilă de ciclare indicele tabloului *i*. În cazul tablourilor bidimensionale (matrici) parcurgerea elementelor se face după doi indici corespunzători liniilor și coloanelor, *i* respectiv *j*.

```
# include <stdio.h>
```

```
void main (void)
```

```
{
```

```
    int lista [6]={20, 30, 40, 50, 60, 70};
```

```
    int i;
```

```
    printf(" valorile tabloului sunt \n") ;
```

```
    for (i=0; i<6;i++)
```

```
        printf ("lista[%d] = %d\n", i, lista[i] );
```

```
}
```

Exemplu

```
int vector[20] ;
```

```
int matrice[20][20] ;
```

Următorul cod de program realizează citirea unui vector de la tastatură:

```
printf(" n= "); scanf ("%d", &n) ;
```

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

```
{ printf (" a[%d]= ", i) ;
```

```
scanf (" %f ", &a[i]) ; }
```

Citirea unei matrice pătratică de dimensiune n :

```
printf(" n= "); scanf ("%d", &n) ;
```

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

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

```
{ printf (" a[%d][%d]= ", i,j) ;
```

```
scanf (" %f ", &a[i][j]) ; }
```

Afișarea unui vector la consolă :

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

```
printf (" %.3f ", a[i]) ;
```

Declararea șirurilor de caractere

Un șir de caractere reprezintă defapt o secvență de caractere ASCII. Pentru toate șirurile citite de la tastatură, programele le atribuie automat caracterul NULL la sfârșitul lor, pentru a-i marca sfârșitul. În C un șir de caractere reprezintă defapt un tablou de caractere care se încheie cu caracterul nul (zero). De reținut că șirurile de caractere trebuiesc declarate cu caracter mai lung decât cel mai lung șir pe care îl pot memora. Astfel pentru a declara un șir ce memorează 20 de caractere veți scrie :

```
char șir [21] ;
```

Compilatorul de C poate crea șiruri capabile să stocheze 256 de caractere, indexate de la 0 la 255.

Observație Când se lucrează cu șiruri de caractere e de datoria programatorului să se asigure de includerea caracterului NULL, pentru reprezentarea sfârșitului de șir.

Constantele șir sunt liste de caractere încadrate între ghilimele duble.

Să observăm astfel diferența dintre caracterele 'A' și " A ". În primul caz este reprezentat caracterul A prin valoarea ei Ascii, în al doilea caz este reprezentat defapt un șir ce conține litera A dar și caracterul Null. Aceste două simboluri sunt stocate diferit și trebuie avut grijă la diferența dintre ele.

C dispune în bibliotecile sale de o varietate de funcții pentru manipularea șirurilor de caractere

- strcpy** (s1,s2) - copiază s2 în s1
- strcat** (s1,s2) - concatenează s2 la sfârșitul lui s1
- strlen** (s1) - calculează lungimea lui s1
- strcmp** (s1, s2) - returnează valoare 0 dacă s1 și s2 sunt identice, negativ dacă s1<s2 și pozitiv dacă s1>s2

Citirea și scrierea șirurilor de caractere

S-a văzut că un șir de caractere este un tablou cu valori de tip char.

Funcțiile care permit citirea și scrierea șirurilor de caractere sunt **gets()** și **puts()**.

Cu ajutorul funcției **gets()** se pot introduce caractere de la tastatură până la apăsarea tastei ENTER. Prototipul funcției este :

char *gets (char *str) ;

și se găsește în fișierul **STDIO.H**

Str este un tablou de caractere care primește caracterele introduse de utilizator.

Exemplu : se citește un șir de caractere

include <stdio.h>

include <string.h>

void main (void)

{
char *str*[80];

gets (*str*);
}

Funcția **puts()** are prototipul :

int puts (const char *str) ;

și ea își scrie argumentul pe ecran. Acționează ca și funcția **printf ()**, dar este incapabilă să genereze numere sau să efectueze conversii de format, fiind destinată exclusiv afișării șirurilor de caractere. Prin urmare rulează mai repede decât **printf()**, necesitând mai puține resurse.

Exemplu :

puts (" asta este un test ") ;

Pentru a crea o constantă de tip șir de caractere, în program trebuie plasate caracterele între ghilimele.

" Acesta este un șir de caractere "

Funcții pentru șiruri de caractere

În C pentru a indica sfârșitul unui șir de caractere se utilizează în mod obișnuit caracterul NULL.

Când se lucrează cu șiruri de caractere, pentru a determina numărul de caractere dintr-un șir C pune la dispoziție funcția **strlen**, care returnează numărul de caractere din șir.

Prototipul funcției:

strlen (**const** char *sir);

și este inclus în fișierul STRING.H.

Când doriți să copiați conținutul unui șir în altul C vă pune la dispoziție funcția **strcpy** care copiază caracterele dintr-un șir sursă în altul, destinație.

Prototipul funcției :

***strcpy** (char *destinatie, **const** char *sursa) ;

și si este inclus în fișierul STRING.H.

Această funcție returnează un pointer care indică începutul șirului destinație.

Pentru a adauga conținutul unui șir la altul C vă pune la dispoziție funcția **strcat()**. Procesul de adaugare a unui șir la altul se numește concatenare de șiruri.

Prototipul funcției :

***strcat** (char *destinatie, **const** char *sursa) ;

și este inclus în fișierul STRING.H.

Transmiterea tablourilor către funcții în C se realizează cu ajutorul pointerilor. Astfel, funcției i se poate transmite un pointer către tablou, indicând numele tabloului fără a fi urmat de indicele tabloului.

Exemplu

void main (**void**)

```
{  
    int vector [10] ;  
    funct_1 (vector) ;  
    .....  
}
```

În această situație parametrul formal al funcției poate fi declarat :

- ca pointer
- ca tablou dimensionat
- ca tablou nedimensionat

Exemple

1. **void** funct_1 (**int** *x) /*pointer
 {.....
 }
2. **void** funct_1 (**int** x[10]) /*tablou dimensionat
 {.....
 }
3. **void** funct_1 (**int** x[]) /* tablou nedimensionat
 {.....
 }

Inițializarea tablourilor

La atribuirea de valori inițiale variabilelor tablou, acestea trebuie transmise între acolade ({}). În cazul șirurilor de caractere valoarea inițială a variabilei se transmite între ghilimele duble.

Exemple

```
char titlu[60]= " Să învățăm C " ;  
int puncte [7]= {20, 30, 50, 88, 70, 28, 56} ;  
int matrice [2][3]= {{1, 3, 5},  
                      {6, 11, 8}} ;
```

Afișarea valorilor conținute în variabila matrice.

```
# include <stdio.h>  
void main (void)  
{  
    int linie, coloana;  
    float matrice [3][4]= {{1.0, 2.3, 3.4, 5.6},  
                            {6.0, 8.7, 6.9, 4.0},  
                            {11.0, 2.6, 7.9, 33.0}} ;  
    for (linie=0 ; linie <3 ; linie++)  
        for (coloana=0 ; coloana<4 ; coloana++)  
            printf (" matrice [%d][%d]= %f\n", linie, coloana, matrice  
[linie][coloana]) ;  
}
```

Recursivitate

Un program în C poate fi împărțit în părți mai mici, numite funcții, astfel programul va fi mai ușor de manipulat și de testat. În cadrul funcției principale a oricărui program în C, se pot apela mai multe funcții, care la randul lor pot apela alte funcții. Limbajul C permite ca o funcție să se apeleze pe sine însăși.

O astfel de funcție se numește **funcție recursivă**, iar procesul acesta de apelare se numește **recursivitate**.

Un exemplu des întâlnit este cel al calculului factorialului unui număr.

$\text{factorial}(n) = n * \text{factorial}(n-1)$.

Următorul program creează o funcție recursivă *factorial*, care se va apela în funcția principală, *main*.

La apelare această funcție primește o valoare a parametrului, care este testată cu 1. Dacă această valoare este 1, funcția returnează valoarea 1, altfel returnează rezultatul înmulțirii dintre valoare și factorialul valorii minus 1.

```
#include <stdio.h>
int factorial (int valoare)
{
    if (valoare==1)
        return 1;
    else
        return (valoare*factorial (valoare-1))
}
void main (void)
{
    int i;
    for (i=1; i<=5;i++)
        printf ("factorial de %d este %d\n", i, factorial (i));}
```

Câteva funcții matematice

Funcția *abs*, returnează valoarea absolută a unei expresii de tip întreg. Se utilizează astfel:

```
#include <stdlib.h>
int abs (int expresie);
```

```
#include <stdlib.h>
#include <stdio.h>
void main (void)
{
    printf (" valoarea absolută a lui %d este %d\n", 5, abs (5)) ;
}
```

Pentru ridicarea la putere limbajul C dispune de funcția *pow*, care returnează rezultatul ridicării unei valori la o putere dată, astfel :

```
#include <math.h>
double pow (double val, double putere) ;
```

```

# include <stdio.h>
# include <math.h>
void main (void)
{
    int putere;
    for (putere= 2; putere<6; putere++)
        printf ("5 ridicat la %d e %f\n", putere, pow(5.0, putere));
}

```

Limbajul C oferă două funcții pentru generarea de numere aleatoare: *rand*, *random*, care returnează numere întregi aleatoare:

```

# include <stdlib.h>
int rand (void);
int random (int limită)

```

Pentru operația de extragere a rădăcinii pătrate, limbajul C vă oferă funcția *sqrt*.

```

# include <math.h>
double sqrt (double val)

```

Funcția lucrează numai valori pozitive, altfel returnează eroare.

```

# include <stdio.h>
# include <math.h>
void main (void)
{
    double val;
    for (val=0.0 ; val< 100 ;val+=10)
        printf (" valoarea %f    rădăcină pătrată %f\n ", val, sqrt (val)) ;
}

```

Pentru calculul logaritmului natural se utilizează funcția *log*

```

# include < math.h>
double log (double val) ;

```

Capitolul 1. Elemente de bază ale limbajului C/C++

- variabile • expresii • instrucțiuni • apelul funcțiilor •
- transmiterea parametrilor prin valoare și folosind adrese •
- intrări și ieșiri • ecranul în mod text •

Probleme rezolvate

1. Să se scrie un program care să calculeze numărul de picioare dintr-o curte, în care se află *g* găini, *p* pisici și un om.

Includem header-ul cu operațiile de intrare și ieșire standard:

```
#include <stdio.h>
```

Programul este constituit dintr-o singură funcție, funcția `main`:

```
void main()  
{
```

```
    int g, p;
```

S-au declarat variabilele de intrare, care apoi se citesc cu `scanf`:

```
    printf("Dati numarul de gaini: ");
```

```
    scanf("%d", &g);
```

```
    %d inseamnă "număr întreg"
```

```
    printf("Dati numarul de pisici: ");
```

```
    scanf("%d", &p);
```

&p înseamnă că valoarea lui `p` este preluată în funcția `scanf`. În continuare, se declară și se afișează data de ieșire, numărul total de picioare.

Firește, o găina are două picioare, o pisică patru, iar omul are și el două:

```
    int np=2*g+4*p+2;
```

Mai sus avem atât o declarație, cât și un calcul al variabilei `np`

```
    printf("\nNumarul total de picioare: %d", np);
```

"\n" de la începutul apelului lui `printf` reprezintă trecerea la un nou rând.

2. Să se verifice dacă un număr real *x* se află în intervalul [*a*,*b*).

```
#include <stdio.h>
```

```
void main()  
{
```

```
    Se declară cele trei variabile reale:
```

```
    float a,b,x;
```

```
    Se citesc capetele intervalului:
```

```
    printf("Dati capetele intervalului:\n");
```

```
    scanf("%f %f", &a, &b);
```

```
    Apoi se citește x:
```

```
    printf("Dati numarul cu pricina:");
```

```
    scanf("%f", &x);
```

```
    Se verifică dacă x este în intervalul căutat: && înseamnă "și"
```

```
    if ((a<=x) && (x<b))
```

Dacă da, atunci se afișează textul: "Numărul *x* se află..." Valoarea lui *x* se afișează pe 6 poziții, din care 3 zecimale (o poziție o ocupă și punctul zecimal)

```
        printf("Numărul %6.3f se află...", x);
```

```
    else
```



```

... numărul nu se află în intervalul [a,b) ...
    printf("Numărul %6.3f nu se află...",x);
}

```

3. Se dau trei numere reale a,b,c. Pot reprezenta ele lungimile laturilor unui triunghi.



Firește a, b, c pot fi lungimile laturilor unui triunghi doar dacă ele sunt toate numere pozitive, iar suma oricăror două este mai mare decât al treilea număr.



Veți observa cu ușurință că programul nici nu pare scris în limbajul C, ci mai degrabă într-o combinație de limba română și C. Și totuși, programul merge și face ce trebuie să facă. Tot secretul reușitei lui constă în definițiile de mai jos, în care relația "&&" este înlocuită cu cuvântul "si", în loc de "if" apare "daca", iar pentru "else" folosim "altfel".

```

#define si &&
#define daca if
#define altfel else
#include <stdio.h>
void main()
{
    float a,b,c;
    printf("Dati lungimile laturilor:\n");
    scanf("%f %f %f",&a,&b,&c);
    Mai jos se verifică condiția de triunghi:
    daca ((a>=0) si (b>=0) si (c>=0) si
        (a<b+c) si (b<a+c) si (c<a+b))
        printf("%4.2f, %4.2f
            si %4.2f formeaza triunghi.",
            a,b,c);
    altfel
        printf("Nu formeaza triunghi.");
}

```

4. Să se scrie un program care să verifice dacă trei numere reale a, b și c pot reprezenta lungimile laturilor unui triunghi. Dacă da, atunci să se precizeze de ce tip este triunghiul: echilateral, isoscel, dreptunghic sau oarecare.

De data aceasta, dacă tot am scris programul anterior, în care verificăm dacă a, b, c pot fi lungimile laturilor unui triunghi, vom rescrie funcția main() din acel program, dându-i numele FormeazaTriunghi în acest nou program. Astfel, când în funcția main() se va ajunge la instrucțiunea if FormeazaTriunghi(a,b,c) ..., programul își va continua execuția cu primul rând din funcția sus numită. În locul lui x (primul parametru al funcției) va veni a, adică valoarea sa, în locul lui y va veni valoarea lui b, iar în locul lui z - valoarea lui c. Firește, parametrii (argumentele) funcției FormeazaTriunghi ar putea fi numite chiar și a, b, c, (eventual în altă ordine), aceste variabile

neavând nimic comun cu variabilele `a, b, c` din funcția `main()` (în afară de nume).



Se spune că apelul funcției se face **prin valoare**. Comunicarea `main()`-`FormeazaTriunghi()` este într-un singur sens, de la prima funcție către a doua. Astfel, chiar dacă, să zicem, `x` s-ar modifica în interiorul celei de a doua funcție, aceasta nu ar afecta în nici un fel valoarea variabilei `a`, adică a parametrului efectiv cu care `main()` apelează pe `FormeazaTriunghi()`. Puteți verifica aceasta singuri sau urmăriți exemplul următor. Funcția `FormeazaTriunghi` returnează o valoare de adevăr, adică *adevărat* sau *fals*, lucru care se traduce în limbajul C printr-o valoare nenulă, pentru *adevărat*, respectiv prin valoarea 0, pentru *fals*. Variabilele `echil`, `isos` și `drept` au valorile 1 dacă triunghiul este respectiv echilateral, isoscel, dreptunghic. Firește, un triunghi echilateral va fi și isoscel. E greu să se obțină triunghiuri dreptunghic isoscele, deoarece e greu să se introducă (pentru `scanf()`) valori potrivite.

```
#include <stdio.h>
int FormeazaTriunghi(int x, int y, int z)
{
    return ((x>=0) && (y>=0) && (z>=0) && (x<y+z)
           && (y<x+z) && (z<x+y));
}
void main()
{
    float a,b,c;
    printf("Dati lungimile laturilor:\n");
    scanf("%f %f %f",&a,&b,&c);
    if (FormeazaTriunghi(a,b,c))
    {
        int echil=((a==b) && (b==c));
        int isos=((a==b) || (b==c) || (c==a));
        int drept=((a*a==(b*b+c*c)) ||
                  (b*b==(a*a+c*c)) || (c*c==(a*a+b*b)));
        if (echil) printf("Triunghi echilateral.\n");
        if (isos) printf("Triunghi isoscel.\n");
        if (drept) printf("Triunghi dreptunghic.\n");
        if ((!echil) && (!isos) && (!drept))
            printf("Triunghi oarecare.\n");
    }
    else printf("Nu formeaza triunghi.\n");
}
```

5. Să se scrie un program care să returneze succesul unui număr întreg dat.

Este vorba despre un program banal, însă vom încerca să punem în evidență un anumit aspect, anume o simulare a transmiterii prin valoare a parametrilor, în cazul funcțiilor.

```
#include <stdio.h>
int succesul(int x)
```

```

{
    int y=x+1;
... evident, y este succesorul lui x.
    x++;
... îl creștem pe x cu o unitate. Oare se va transmite în exterior valoarea nouă?
    return y;
}
void main()
{
    int a;
    printf("Dati numarul a: "); scanf("%d",&a);
    printf("Deocamdata a = %d\n",a);
    int b=succesor(a); printf("Succesorul b = %d\n",b);
    printf("Iar acum a = %d\n (ala vechi!)\n",a);
    int c=succesor(3);
    printf("Succesorul lui 3 este = %d\n",c);
    printf("Iar 3 tot 3 ramane, nu-i asa ?\n");
    int d=succesor(a+3);
    printf("In fine, succesorul lui 3+%d este = %d\n",a,d);
}

```

Se va observa cu ușurință, că valoarea modificată a lui `x` nu se transmite și lui `a`. Motiv pentru care puteți apela funcția `succesor` și cu un argument constant sau cu o expresie ca în exemplul anterior. Iată de ce e nevoie de un nou mecanism de transmitere a parametrilor, pentru a realiza comunicarea de la funcția apelată către cea apelantă (de exemplu, de la `succesor` la `main`).



Un astfel de mecanism se numește **transmitere prin referință a parametrilor** și aceasta se realizează în felul următor: în loc să punem ca argument formal în apel numele variabilei respective, vom pune simbolul "&" în fața variabilei. Aceasta schimbă semnificația! Dacă `x` este o variabilă oarecare, prin `&x` vom înțelege adresa din memorie unde se află variabila `x` (indiferent pe unde șade ea acolo!). Astfel, trebuie ca și funcția care primește pe `&x` ca argument să știe să lucreze cu o adresă a unei variabile de tipul lui `x` (practic este vorba tot despre un apel prin valoare, însă valorile sunt adrese de memorie sau **pointeri**).

Un exemplu îl constituie exercițiul următor.

6. Să se scrie un program care să returneze succesorul unui număr întreg dat, dar să încreezeze cu 1 valoarea numărului inițial.

Vom încerca să punem în evidență un anumit aspect, anume transmiterea prin referință a parametrilor, în cazul funcțiilor

```

#include <stdio.h>
int succesor(int *x)
{
    int y=(*x)+1;
... evident, y este succesorul valorii lui x
    (*x)++;
... creștem valoarea lui x cu o unitate.

```

```

        return y;
    }
void main()
{   int a;
    printf("Dati numarul a: "); scanf("%d",&a);
    printf("Deocamdata a = %d\n",a);
    int b=succesor(&a);
    printf("Succesorul b = %d\n",b);
    printf("Iar acum a = %d\n (ala vechi!)\n",a);
    // printf("Succesorul lui 3 = %d",succesor(&3);
}

```



Funcția `succesor` ar fi putut arăta și astfel:

```

int succesor(int *x)
{
    return ++(*x);
}

```

dar nu și așa:

```

int succesor(int *x)
{
    return (*x)++;
}

```



Firește, atât `++m`, cât și `m++` reprezintă incrementări ale lui `m` cu o unitate, însă diferența e de finețe: pe când în primul caz are loc întâi incrementarea lui `m`, apoi folosirea sa, în cel de al doilea caz are loc întâi utilizarea sa (a valorii neincrementate încă) și apoi incrementarea. Atenție deci, căci puteți avea probleme! Se va observa că valoarea lui `x` se transmite în exterior variabilei `a`.



Înainte de a trece mai departe, să analizăm încă puțin exemplul dat. Dacă până acum noi puteam apela funcțiile descrise nu doar cu variabile, ci și cu constante sau chiar expresii (în fond cu valorile lor), de această dată, un apel de genul celui pus în comentariu în finalul programului anterior semnalează eroare. Nici apelul `succesor(3)` nu e mai bun! Dar, dacă `&m` înseamnă adresa variabile `m`, acest lucru ne obligă ca în antetul funcției care o folosește, să definim ceva de genul `nume_functie(tip * mm)`, în care `tip` este tocmai tipul lui `m`, `mm` este argumentul funcției, iar steluta “*” semnalează faptul că `mm` nu este un număr întreg, ci un pointer la un număr întreg, adică un indicator către o zonă de memorie unde se află un număr întreg (practic, în momentul execuției programului, apelând funcția cu valoarea adresei lui `m` (`&m`), `mm` va conține tocmai această adresă (ceea ce se numește că `mm` este un “indicator” către `m`).

Oare funcția `scanf()` nu lucrează la fel? Ba da, e și normal, deoarece ea trebuie să trimită către exterior (adică funcției apelante) rezultatele citirilor, deci întotdeauna trebuie să avem grijă cum folosim funcția `scanf()`.

Fiindcă `printf()` nu modifică în nici un fel valorile afișate, e normal ca la `printf()` să nu avem nevoie de `&`. Decât, dacă dorim să obținem adresa unei variabile, ca în exemplul următor:

```
#include <stdio.h>
void main()
{ int x=3; printf("Adresa lui x: %ld",&x); }
```

Dar un asemenea exemplu nu ne ajută cu nimic, de obicei, fiind pur didactic...

7. Să se scrie un program care să calculeze valoarea mediei aritmetice a două valori reale a și b. Se vor folosi două funcții, una care va trimite rezultatul prin numele funcției, iar alta printr-unul din parametri.

```
#include <stdio.h>
float media(float x, float y)
{
    return (x+y)/2;
}
void proc_media(float x, float y, float * m)
{
    *m = (x+y)/2;
}
void main()
{
    float a, b, m1, m2;
    printf("Dati cele doua numere:\n");
    scanf("%f %f",&a,&b);
    m1 = media(a,b);
    proc_media(a,b,&m2);
    printf("Media returnata prin numele functiei: %7.3f\n",
        m1);
    printf("Media returnata ca si parametru      : %7.3f\n",
        m2);
    printf("Nici o diferenta !\n");
}
```



De fapt, dacă prima funcție folosește `return` pentru a obține valoarea medie (adică se comportă ca o funcție “adeverată”), cea de a doua se comportă ca o procedură din limbajul Pascal, motiv pentru care nici nu returnează nimic (apare `void` ca tip al funcției), iar rezultatul se transmite prin intermediul parametrului de tip pointer `m`.

Observați diferența dintre `x` și `y`, pe de o parte, și `m` pe de altă parte, în antetul funcției `proc_media`, cât și modul în care este apelată ea!

8. Să se determine maximul și minimul a două numere întregi, fără a folosi instrucțiunea “if”.

Programul ar fi banal, firește, însă dacă nu trebuie să folosim cuvântul “if”, înseamnă că vom utiliza interesantul operator condițional “`? :`”.

Astfel, vom putea defini două macroui, cu numele `minim` și `maxim`, după cum urmează:

```
#define max(x,y) x > y ? x : y
#define min(x,y) x < y ? x : y
#include <stdio.h>
```

```

void main()
{
    int x, y;
    printf("Ia sa vedem care-i mai tare !\n");
    printf("Da x si y:"); scanf("%d %d",&x,&y);
    printf("\nCel mai tare este: %d",max(x,y));
    printf("\nIar cel mai slab : %d",min(x,y));
}

```

Simplu, elegant și frumos, nu-i așa?



Nu vă faceți griji dacă lucrați cu numere reale în loc de numere întregi, macrourele funcționează perfect în orice situație!

```

#define max(x,y) x > y ? x : y
#include <stdio.h>
void main()
{
    float x, y;
    printf("Ia sa vedem care-i mai tare !\n");
    printf("Da x si y:"); scanf("%f %f",&x,&y);
    printf("\nCel mai tare este: %7.3f",max(x,y));
}

```



E simplu de înțeles acum cum funcționează operatorul ternar “?:”: să considerăm că avem 3 expresii e_1 , e_2 și e_3 , în care e_2 și e_3 trebuie să fie, în mare fie spus, cam de același tip. Se evaluează e_1 . Dacă e_1 e o valoare nenulă (*adevărat*), atunci rezultatul este e_2 , altfel e_3 .

9. Să se calculeze puterea a n-a a lui 2, unde n este un întreg dat de la tastatură.

Pentru că tot am vorbit în exemplul anterior despre unul din operatorii speciali, pe care îi are limbajul C, vom prezenta în exemplul de față operatorul de “shiftare” (“șiftare”, “rotire”, “deplasare”) spre stânga “<<”.



Fie x un număr întreg. Prin $x<<1$ înțelegem deplasarea spre stânga a biților din reprezentarea pe biți (în baza 2) a lui x , completându-se cu 0 în dreapta. Astfel, de exemplu, dacă $x=6$, adică 101, atunci, deplasarea spre stânga va duce la obținerea numărului $x=12$, adică 1010. De fapt, o shiftare spre stânga cu o unitate reprezintă o înmulțire a lui x cu 2. O shiftare spre stânga cu p poziții a lui x reprezintă o înmulțire a lui x cu 2 de p ori.

Tot la fel, o shiftare spre dreapta ar însemna o împărțire succesivă la 2 (în care nu se ține cont de rest).

Folosind acest artificiu (al deplasărilor spre stânga), am calculat și noi puterea lui 2, conform programului următor:

```

#include <stdio.h>
void main()
{

```

```

    int n, m=1;

```

... aici se declara atât variabila n , cât și m , care se și inițializează la valoarea 1

```

printf("Dati n: ");
scanf("%d",&n);
printf("2^%d = %d\n",n,m=(m<<n));
}

```

Al doilea apel al funcției `printf` are un element interesant: se afișează valoarea expresiei `"m=(m<<n)"`.

Valoarea unei expresii de forma `m=e`, care evident este și o instrucțiune de atribuire, este tocmai valoarea expresiei `e`. Astfel, când scriem `m=(m<<n)`, înseamnă că `m` este shiftat spre stânga cu `n` poziții, deci `m`, inițial 1, devine egal cu puterea `n` a lui 2. Aceasta este și valoarea expresiei `m=(m<<n)`, care va fi afișată.

10. Să se determine aria unui triunghi în funcție de lungimile laturilor sale.

Soluția pe care o prezentăm în continuare nu face apel la funcția `FormeazaTriunghi`, pe care am prezentat-o într-un exemplu anterior. De această dată, presupunem utilizatorul programului bine intenționat, astfel încât ne vom axa pe formula de calcul a ariei, când se cunosc lungimile `a`, `b`, `c` ale celor trei laturi, formulă cunoscută sub denumirea de *formula lui Heron*. Elementul interesant din acest program este folosirea funcției `sqrt(x)`, care ne dă rădăcina pătrată a unui număr real `x`, precum și a funcției `getch()`, care ne returnează valoarea tastei apăsate (un caracter). Fiecare din cele două funcții necesită includerea în textul sursă al programului a doua fișiere `.H`, anume `<math.h>` și respectiv `<conio.h>`.

```

#include <stdio.h>
#include <math.h>
#include <conio.h>
void main()
{
    float a,b,c,p,aria;
    printf("Dati lungimile laturilor:\n");
    scanf("%f %f %f",&a,&b,&c); p=(a+b+c)/2;
    aria=sqrt(p*(p-a)*(p-b)*(p-c));
    printf("Aria este: %7.3f.",aria); getch();
}

```

Spre deosebire de toate funcțiile folosite până acum, `getch()` este utilizată într-un alt mod. Practic, se comportă întocmai ca o procedură din Pascal. Chiar dacă ea returnează caracterul apăsător la tastatură, acesta contează mai puțin, important este "*efectul lateral*" pe care îl are această funcție, aici concretizându-se în așteptarea apăsării unei taste. Acest lucru este foarte util, în sensul că după fiecare rulare a programului, nu vom mai fi nevoiți să acționăm combinația de taste `Alt-F5` pentru a comuta între ecranul cuprinzând rezultatul execuției programului și cel al mediului de programare *Borland*.

11. Să se determine rădăcina pătrată a unui număr real `a`, fără a utiliza funcția `sqrt`.



Această problemă se poate soluționa pornind de la *șirul recurent al lui Newton*:

$$x_0 = 1, x_n = (x_{n-1} + a/x_{n-1})/2, \text{ pentru } n \geq 1$$

De fiecare dată vom avea nevoie doar de doi termeni succesivi ai acestui șir, fie ei x_n și x_{n1} . Vom calcula pe x_n plecând de la x_{n1} , apoi vom actualiza pe x_{n1} cu valoarea lui x_n , iar procedeul continuă repetitiv până când diferența dintre x_n și x_{n1} este nesemnificativă.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
float radical(float a)
{ const eps=0.00001; // o constanta,
// avand valoarea egala cu precizia dorita in calculul
radicalului
    float xn1=1, xn=(1+xn1)/2;
    do { xn1=xn;
        xn = (xn1+a/xn1)/2;
    } while (fabs(xn-xn1)>eps);
    // fabs(x) = |x|, unde x este numar real
    return xn;
// rezultatul este ultimul termen calculat al sirului
}
void main()
{ clrscr(); // se curata ecranul
  float a;
  printf("Dati a:"); scanf("%f",&a);
  printf("\nsqrt      (%7.3f)=%10.6f",a,sqrt(a));
  // se compara cele doua functii
  printf("\nradical(%7.3f)=%10.6f",a,radical(a));
  printf("\nCinci zecimale exacte!");
  getch();
}
```

Observați folosirea instrucțiunii repetitive `do... while...`, având semnificația: *execută ... atât timp cât ...*. Este vorba despre o instrucțiune repetitivă în care testul se face la sfârșitul ciclului.

12. Ce va afișa următorul program (care folosește operatorul ",") ?

```
#include <conio.h>
#include <stdio.h>
void main()
{
    clrscr(); int x=1,y; int z = (x = 2, y=x+3);
    printf("%d, %d, %d",x,y,z); getch();
}
```



În atribuirea `z = (x=2, y=x+3)` au loc următoarele: mai întâi x primește valoarea 2, apoi y valoarea 5, iar z tot valoarea 5. Deci se va afișa 3, 5, 5.

13. Se citesc litere pâna la întâlnirea caracterului ".". Să se contorizeze numărul de vocale citite.

De data aceasta, vom folosi macroul predefinit `getchar()`, care returnează tasta apăsată, de fapt el citește din intrarea standard, `stdin`.

```
#include <stdio.h>
void main()
{
    char c; int nr_voc; nr_voc=0;
    printf("Dati textul:\n");
    while ((c = getchar()) != '.')
    {
        printf("%c", c);
        if ((c=='a') || (c=='e') || (c=='i') ||
            (c=='o') || (c=='u')) nr_voc++;
    }
    printf("\nTotal vocale: %d",nr_voc);
}
```

Înlocuind '.' cu '\n', `getchar()` va citi până la apăsarea lui Enter.

14. Să se deseneze un dreptunghi de dimensiune $m \times n$, folosind caracterele speciale pentru colțuri.

Vom folosi macroul `putchar()` pentru a afișa pe ecran. Colțurile au următoarele valori (in hexazecimal):

```
#include <stdio.h>
#include <conio.h>
#define LEFT_TOP 0xDA
#define RIGHT_TOP 0xBF
#define HORIZ 0xC4
#define VERT 0xB3
#define LEFT_BOT 0xC0
#define RIGHT_BOT 0xD9
void main()
{
    int m, n;
    char i, j;
    clrscr();
    printf("Dati m si n: ");
    scanf("%d %d",&m,&n);
    /* marginea de sus */
    putchar(LEFT_TOP);
    for (i=0; i<n; i++)
        putchar(HORIZ);
    putchar(RIGHT_TOP);
    putchar('\n');
    /* mijlocul */
    for (i=0; i<m; i++)
    {
        putchar(VERT);
        for (j=0; j<n; j++)
            putchar(' ');
        putchar(VERT);
    }
}
```

```

        putchar('\n');
    }
    /* marginea de jos */
    putchar(LEFT_BOT);
    for (i=0; i<n; i++)
        putchar(HORIZ);
    putchar(RIGHT_BOT);
    putchar('\n'); getch();
}

```



Puteți remarca utilizarea variabilelor `i`, `j` ca numere întregi, deși ele au fost declarate ca fiind de tip `char`. Trebuie precizat că C-ul poate face astfel de conversii, iar un `char` ocupă doar un octet (iar un `int` doi), și cum valorile pentru `i` și `j` sunt mici, se poate folosi `char`.

15. Să se miște o literă pe ecranul text, folosind cele patru taste de cursor.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    char tasta; char x=40,y=12;
    gotoxy(x,y); putchar('X');
    do {
        tasta=getch(); gotoxy(x,y); putchar(' ');
        switch(tasta) {
            case 72: { if (y>1) y--; break; }
            case 80: { if (y<24) y++; break; }
            case 75: { if (x>1) x--; break; }
            case 77: { if (x<80) x++; }
        }
        gotoxy(x,y); putchar('X');
    } while (tasta!=27);
}

```



În exemplul anterior, am folosit afișarea cu `putch()` (care scrie doar pe ecran, direct în memoria video). Poziționarea cursorului se face cu funcția `gotoxy` (declarată în `conio.h`). Ecranul are 25 de linii și 80 de coloane, numerotate de la 1 la 25, respectiv de la 1 la 80. Noi am evitat afișările pe linia 25, deoarece o afișare în punctul de pe această linie și ultima coloană are ca efect "ridicarea" textului cu o linie în sus, pe ecran.

Observați și utilizarea instrucțiunii de decizie multiplă `switch`. În cadrul ei am folosit instrucțiunea `if` pentru a evita ieșirile din ecran, precum și instrucțiunea `break`, care determină continuarea execuției programului cu instrucțiunea de după `switch`.

Tasta acționată se citește cu `getch()` (dar nu se și afișează). Ea poate fi cursor sus (72), stânga (75), dreapta (77) sau jos (80), pentru deplasare, respectiv `Escape` (27), pentru terminarea programului.

16. Să se simuleze mișcarea unei bile de sus în jos, pe coloana 40 a ecranului. Afișarea se va repeta, de fiecare dată bila având o culoare aleasă la întâmplare.



Noutățile aduse de acest exemplu sunt:

- utilizarea fișierului `<dos.h>`, care conține declarația funcției `delay`. Apelul `delay(p)` face o pauză (în execuția programului) de `p` milisecunde. Astfel, `delay(500)` înseamnă o pauză de o jumătate de secundă.
- utilizarea unor funcții declarate în `<stdlib.h>`, pentru obținerea de numere aleatoare (de fapt generate după o anumită formulă, într-un șir de numere, pornind de la o anumită valoare inițială):
 - `void randomize()` schimbă valoarea inițială a acestui șir de valori aleatoare;
 - `int random(int n)` returnează o valoare aleatoare întreagă, din intervalul `[0,n-1]`. Astfel, pentru a obține una din cele 15 culori (exceptând negrul=0), am folosit `textcolor(1+random(15))`.
- o altă noutate este folosirea lui `puts` pentru a afișa un șir de caractere;
- răspunsul la întrebarea din finalul ciclului `do-while` este afișat pe ecran.

El este o literă, preluată cu `getche()`. Dacă aceasta este 'n', programul se oprește. Diferența între `getche()` și `getch()` este că prima afișează caracterul apăsător (deci este cu "ecou"), pe când cea de a doua nu.

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
void main()
{
    randomize();
    do {
        delay(500); clrscr(); textcolor(1+random(15));
        char y=1;
        while (y<=25)
        {
            gotoxy(40,y); putch(' ');
            gotoxy(40,++y); putch('o'); delay(50);
        }
        puts("\nMai doriti odata ? ->");
    } while (getche()!='n');
}
```

17. Să se determine cel mai mare divizor comun și cel mai mic multiplu comun pentru două numere întregi a și b.



Pentru a determina cel mai mic multiplu comun al lui `a` și `b`, trebuie ca mai întâi să determinăm `c` = cel mai mare divizor comun al lui `a` și `b`. Dacă cel puțin unul din cele două numere (`a` sau `b`) este 0, vom considera celălalt

număr ca fiind c . Dacă ambele numere sunt diferite de 0, atunci vom scădea numărul mai mic din numărul mai mare, până vom obține egalitatea numerelor a și b . În acest moment, $c=a$. În fine, metoda se numește **algoritmul lui Euclid cu scaderi**, iar pentru a obține cel mai mic multiplu comun înmulțim pe a și b inițiali și împărțim la c .

```
#include <stdio.h>
#include <conio.h>
long cmmdc(long a, long b)
{
    if (!a) return b;
    else if (!b) return a;
        else while (a-b) // adica daca a-b != 0, adica a !=b
            if (a>b) a-=b; else b-=a;
    return a;
}
long cmmmc(long a, long b)
{ long c=cmmdc(a,b);
  if (c) return a*b/c; else return 0;
}
```

Când ambele numere sunt 0, `cmmmc` nu există, dar returnăm 0.

```
void main()
{
    long a,b,c,d;
    printf("\nDati a, b: "); scanf("%ld %ld",&a,&b);
    c=cmmdc(a,b);
    d=cmmmc(a,b); // conversia rezultatului impartirii
                  // la numarul intreg d se face automat
    printf("Cmmdc(%ld,%ld)=%ld\n",a,b,c);
    printf("Cmmmc(%ld,%ld)=%ld\n",a,b,d); getch();
}
```

De subliniat faptul că o declarație de forma `long x` este echivalentă cu o declarație de forma `long int` (întreg lung). "`%ld`" permite afișarea de întregi lungi.

18. Să se simuleze mișcarea unei bile pe o masă de biliard, fără frecare.

Bila va fi caracterizată de patru variabile întregi: x și y vor indica coordonatele bilei; ($1 \leq x \leq 80$, $1 \leq y \leq 24$);

- v_x și v_y vor reprezenta direcțiile de deplasare ale bilei:

Astfel, la fiecare pas, bila se va deplasa din poziția (x,y) , în poziția $(x+v_x, y+v_y)$, unde v_x și v_y pot lua valorile 1 sau -1. De pildă, o deplasare în direcția stânga-jos este dată de $v_x=-1$ și $v_y=1$.

```
#include <conio.h>
#include <dos.h>
void HideCursor()
{
    /* sa scrieti exact asa: "asm {" si nu cu acolada pe randul
    urmator !! */
    asm {
        mov ax, 0x0100
        mov cx, 0x2607
```

```

        int 0x10
    }
}
void ShowCursor()
{
    asm {
        mov ax, 0x0100
        mov cx, 0x0506
        int 0x10
    }
}
void main()
{
    clrscr(); HideCursor();
    char x=40, y=12, vx=1, vy=1;
    while (!kbhit())
    {
        gotoxy(x,y); putchar(' ');
        if ((x==1) || (x==80)) vx=-vx;
        if ((y==1) || (y==24)) vy=-vy;
        x+=vx; y+=vy;
        gotoxy(x,y); putchar('o'); delay(50);
    }
    ShowCursor();
}

```



Acest exemplu are următoarele elemente de noutate:

- funcția `kbhit()` - returnează 1 (adevărat) dacă fu apăsată o tastă (tocmai s-a apăsat o tastă), respectiv 0, în caz contrar;
- se pot scrie în textul C secvențe de program în limbaj de asamblare, folosind declarația `"asm {"` și `"}"`, ca în funcțiile `HideCursor()` și `ShowCursor()`.
- cursorul pâlpâitor din modul text se poate ascunde cu prima funcție și reafișa cu cea de a doua.

19. Să se afișeze primele n numere prime.

Mai întâi, vom scrie o funcție care va verifica dacă un număr întreg este prim sau nu.



Pentru a vedea dacă x este prim, se compară, mai întâi, x cu 0, 1 și 2. Dacă $x \geq 2$, atunci se verifică dacă x este par sau nu. În caz că este impar, se împarte succesiv la toate numerele impare începând cu 3, până se obține un divizor d al lui x (caz în care numărul nu este prim) sau se ajunge la partea întreagă a rădăcinii pătrate a lui x (caz în care numărul e prim)

```

#include <stdio.h>
#include <conio.h>
#include <math.h>
int EstePrim(int x)
{
    if ((x==0) || (x==1)) return 0;

```

```

else
    if (!(x%2)) //adica daca x modulo 2 este zero...
        if (x==2) return 1;
        else return 0;
    else
        { int d=1, radical=sqrt(x);
          while ((d<=radical) && (x%(d+=2)));
          /* adica cat timp d ≤ radical, iar x se nu imparte exact
             la d (care creste cu doua unitati, incepand cu 1 (deci prima
             valoare testata va fi 3)) */
          return (d > radical);
        }
}

void main()
{
    clrscr();
    int i=0, j=1, n;
    printf("Dati nr. de numere: "); scanf("%d",&n);
    for ( ; j<=n, getch(); i++)
        if (EstePrim(i)) { printf("%d,",i); j++; }
}

```

Acest program poate că este cel mai ciudat dintre toate cele prezentate până acum. În funcția `EstePrim` am pus în evidență un mod interesant de a folosi instrucțiunea `while`, care nu are decât condiție (nu și instrucțiune)! Însă, în cadrul condiției se ascunde, totuși, o simplă instrucțiune de incrementare cu 2 a valorii lui `d`. Astfel, instrucțiunea `while` de acolo este echivalentă cu:

```
while ((d<=radical) && (x%d!=0)) d=d+2;
```

Valoarea returnată este 1, dacă `d>radical` (adică s-a depășit valoarea maximă de test pentru eventualii divizori ai lui `x`, respectiv 0, în caz contrar.

În privința programului principal (funcția `main`), observați modul special de folosire a instrucțiunii `for`. Practic, inițializarea lui `i` la 0 este deja făcută (când se declară `i`), apoi apare și operatorul virgulă „, ”, care conține testul de terminare a ciclului `for`, precum și un apel al lui `getch()` care va determina așteptarea apăsării unei taste pentru fiecare număr (prim sau nu), până la sfârșit.

20. Să se transforme un număr din baza 10 într-o bază p , $2 \leq p \leq 10$.



Vom împărți succesiv pe x (numărul dat) la p , reținând cifrele din baza p într-un număr x_{pi} , care va fi apoi răsturnat (oglintit), pentru a obține numărul x_p , văzut ca reprezentarea în baza p a lui x . Pentru a nu avea probleme în cazul în care prima cifră a lui x_{pi} ar fi 0, folosim o incrementare cu 1 a cifrelor la transformarea $x \rightarrow x_{pi}$, respectiv o decrementare la transformarea $x_{pi} \rightarrow x_p$.

```

#include <conio.h>
#include <stdio.h>
void main()
{

```

```

clrscr(); int x, xpi, xp, p;
printf("Dati x in baza 10: "); scanf("%d",&x);
printf("Dati noua baza p : "); scanf("%d",&p);
for (xpi=0; x; xpi=10*xpi+x%p+1, x=x/p);
for (xp=0; xpi; xp=10*xp+(xpi-1)%10, xpi=xpi/10);
printf("Numarul in baza %d este: %d",p,xp); getch();
}

```

De remarcat, în acest exemplu, eleganța folosirii operatorului “,” în cadrul instrucțiunilor `for`. De pildă, primul `for` se interpretează astfel:

- se inițializează `xpi` cu 0;
- cât timp `x` este diferit de 0 execută:
 - `xpi` devine $10 \cdot xpi + x \% p + 1$;
 - `x` se împarte la `p` (și se face conversia la `int`)

21. Să se calculeze suma $s = 1 - 2 + 3 - 4 + \dots \pm n$, pentru un n dat de la tastatură.

Vom folosi o variabilă `semn` care va lua pe rând valorile 1 și -1.

```

#include <conio.h>
#include <stdio.h>
void main()
{
    clrscr(); int n;
    printf("Dati numarul n: "); scanf("%d",&n);
    for (int i=1, suma=0, semn=1; i<=(n+1);
        suma+=(semn*i), i++, semn=-semn);
    printf("Suma dorita este: %d",suma); getch();
}

```



Firește, suma e foarte ușor de determinat matematic. Am vrut, însă, să punem în evidență următoarele aspecte:

- declararea și inițializarea variabilelor `i`, `suma` și `semn`, în cadrul primei expresii din instrucțiunea `for`;
- folosirea condiției `i<=(n+1)` pentru a verifica dacă $i \leq n$ sau nu;
- folosirea operatorului virgulă în cadrul expresiilor ce apar în “`for`”;
- inexistența unei alte instrucțiuni în cadrul lui “`for`”.

De remarcat puterea de care dă dovadă instrucțiunea `for` din limbajul C, în comparație, de pildă, cu cea din cazul limbajului Pascal.

22. Să se deseneze în ecranul text un dreptunghi, specificat prin coordonatele a două colțuri diagonale opuse.

Vom citi `x1`, `y1` și respectiv `x2`, `y2`, coordonatele acestor colțuri. Folosind operatorul ternar “?:” vom determina în `x1`, `y1` minimele, iar în `x2`, `y2` maximele (ceea ce va însemna că, în final, aceste coordonate vor fi ale colțurilor stânga-sus și dreapta-jos). Apoi vom folosi două instrucțiuni `for` pentru a desena. Caracterul va avea codul hexazecimal B0 (adică 178) și va fi afișat cu o funcție `PrintAt`.

```

#include <conio.h>
#include <stdio.h>

```

```

void PrintAt(char x, char y, char c)
{
    gotoxy(x,y); putchar(c);
}
void main()
{
    int x1,y1,x2,y2,aux,x,y;
    char cc=0xB0; // numar hexazecimal = 178, in baza 10
    clrscr();
    printf("\nDati x1: "); scanf("%d",&x1);
    printf("\nDati y1: "); scanf("%d",&y1);
    printf("\nDati x2: "); scanf("%d",&x2);
    printf("\nDati y2: "); scanf("%d",&y2);

```

Daca $x1 > x2$, folosind variabila aux, se inter schimbă valorile lui $x1$ și $x2$

```

     $x1 > x2$  ? aux=x1, x1=x2, x2=aux: 1;

```

... aici, în loc de 1 ar fi putut fi orice;

Facem la fel pentru $y1$ și $y2$:

```

     $y1 > y2$  ? aux=y1, y1=y2, y2=aux: 1;

```

```

    clrscr();

```

```

    for (x=x1; x<=x2; x++)

```

```

        for (y=y1; y<=y2; y++)

```

```

            PrintAt(x,y,cc);

```

```

            //... se face conversie de la int la char

```

```

        getch();

```

```

    }

```

Probleme propuse

- Care este valoarea variabilei p după efectuarea următoarelor operații?
 a) $p=i=1$; while ($i<7$) $i++$; $p*=i$;
 b) $p=1$; $i=7$; while ($i>1$) { $p*=i$; $i-=1$;}
- Să se calculeze suma: $S=1-1 \times 3+1 \times 3 \times 5-\dots \pm 1 \times 3 \times 5 \times \dots \times (2n-1)$.
- Ce este greșit în secvența de instrucțiuni de mai jos:
 if ($n<5$) $x=while$; else $i:=i+1$; while ($p<4$) $k++$;
- Fie a, b, c trei numere reale. Scrieți o funcție care să verifice dacă a, b pot reprezenta lungimile laturilor unui dreptunghi, respectiv c să fie diagonală în acel dreptunghi.
- a) Elaborați un program care să tipărească tabela de temperaturi Fahrenheit și echivalențele lor în centigrade sau grade Celsius, folosind formula: $C=(5/9) \times (F-32)$, între valorile extreme 0 și 300 grade Fahrenheit.
 b) Elaborați un program care să tipărească tabela corespunzătoare Celsius-Fahrenheit.
- Ce execută următorul program C ? Rescrieți-l folosind instrucțiunea for.

```

#include <stdio.h>
void main()
{
    long nc=0;
    while (getchar() != EOF) ++nc;
    printf("%ld\n",nc);
}

```


- }
7. Scrieți un program care să numere spațiile, tab-urile și new-line-urile (trecherile la un nou rând) date de la tastatură.
 8. Scrieți un program care să copieze intrarea în ieșire, înlocuind fiecare șir de unul sau mai multe spații cu un singur spațiu.
 9. Scrieți un program care să înlocuiască fiecare tab printr-o secvență de trei spații.
 10. Scrieți o funcție `int power(int x, int n)` care ridică pe x la puterea n .
 11. Scrieți un program care să determine cel mai mare divizor comun a două numere întregi, prin algoritmul clasic al lui Euclid (cu împărțiri repetate).
 12. Să se determine dacă două numere sunt prime între ele sau nu.
 13. Să se scrie programe pentru a deplasa o bilă pe ecran, de la stânga la dreapta și de la dreapta la stânga, până la acționarea unei taste.
 14. Scrieți un program care să citească mai multe numere întregi, până la întâlnirea lui zero, și să determine cel mai mic număr și cel mai mare număr citit.
 15. Scrieți un program care să calculeze valoarea unui polinom într-un punct dat. Polinomul este dat prin coeficienții săi. Nu se vor utiliza tablourile!
 16. Scrieți un program care să calculeze factorialul unui număr întreg n .
 17. Scrieți funcții care să calculeze aria unui cerc în funcție de: a) raza cercului; b) diametrul cercului; c) lungimea cercului; d) latura triunghiului echilateral înscris în cerc; e) latura pătratului înscris în cerc.
 18. Aceeași problemă ca și cea anterioară, dar cu citiri/afișări colorate, în diferite poziții ale ecranului text.
 19. Să se determine perimetrul și aria unui triunghi echilateral înscris într-un cerc de diametru d .
 20. Să se determine cel de al treilea unghi al unui triunghi când se cunosc celelalte două.
 21. Să se rezolve un triunghi (adică să se determine elementele sale necunoscute (lungimi de laturi sau măsuri de unghiuri)), când se cunosc:
 - a) lungimile celor trei laturi;
 - b) lungimea a două laturi și măsura unghiului dintre ele;
 - c) lungimea unei laturi și măsurile unghiurilor adiacente ei
 22. Să se determine aria unui trapez când se cunosc lungimile bazelor și a înălțimii.
 23. Să se determine lungimea unui cerc în funcție de aria sa.
 24. Să se determine aria pătratului circumscris unui cerc, cunoscând aria pătratului înscris în cerc.
 25. Catetele unui triunghi dreptunghic au lungimile a și b . Să se determine ipotenuza, perimetrul și aria sa.
 26. O carte are n foi și r rânduri pe fiecare pagină. Câte rânduri are cartea?
 27. Un punct material se află la distanța x_0 de origine, la momentul inițial t_0 , când începe să se miște rectiliniu uniform. Țiind că la momentul t se află la distanța x față de origine, să se determine viteza v de mișcare a punctului material, la momentul t .

28. Scrieți o funcție care să transforme un unghi exprimat în radiani în valoarea sa exprimată în grade și una care să facă transformarea inversă.
29. Scrieți o funcție care să determine diferența dintre două momente de timp, date prin ore, minute și secunde.
30. Viteza unui automobil este de v km/h. Exprimați această viteză în m/s.
31. Un muncitor lucrează n zile pentru a termina o lucrare. Să se determine câte zile sunt necesare pentru a termina aceeași lucrare o echipă de m muncitori.
32. Două echipe de muncitori au în componența lor m_1 , respectiv m_2 muncitori. Prima echipă termină o lucrare în z_1 zile, iar cea de a doua în z_2 . Să se determine:
 - a) z_2 în funcție de m_1, z_1, m_2 ;
 - b) m_2 în funcție de m_1, z_1, z_2 .
33. Într-o magazie sunt următoarele produse: fasole, cartofi și roșii în cantitățile f, c, r . Un kilogram de fasole costă cf lei, unul de cartofi cc lei. Să se determine costul unui kilogram de roșii, știind că toate legumele din magazie costă c lei.
34. Scrieți un program care să rezolve ecuația de gradul I $ax+b=0$.
35. Scrieți un program care să rezolve ecuația de gradul II $ax^2+bx+c=0$, inclusiv pentru $a=0$ sau când ecuația admite soluții complexe, nereale.
36. Să se determine dacă un număr a este divizibil cu toate numerele b_1, b_2 și b_3 .
37. Să se scrie un program care să citească două numere reale a și b . Apoi să pună utilizatorului o întrebare: *Ce doriți să calculăm ? Media aritmetică (1) sau geometrică (2)?*. Dacă se va răspunde prin 1, se va calcula și afișa media aritmetică, iar pentru 2 media geometrică (numai dacă numerele sunt pozitive !, iar de nu, se va afișa 'eroare !'). Dacă nu se răspunde prin 1 sau 2 se va produce un sunet în difuzor.
38. Să se scrie un program care să citească trei numere reale a, b și c , apoi să pună o întrebare de genul: *Ce doriți să calculăm? Aria sau perimetrul?*. Dacă se va răspunde prin 'aria' atunci se va calcula și afișa aria, altfel perimetrul. Ce se va întâmpla dacă se va răspunde prin 'arie' ?
39. Să se scrie un program care să citească un număr întreg, iar dacă acest număr este 1 să afișeze *luni*, dacă este 2 să afișeze *marți*, ... 7 - *duminică*, iar dacă nu este cuprins între 1 și 7 să afișeze cuvântul *eroare*. Să se scrie programul folosind: a) instrucțiunea `if`; b) instrucțiunea `switch`.
40. Să se scrie un program care să calculeze aria unui cerc în funcție de diametru, dacă răspunsul la întrebarea: *'Aria în funcție de diametru (1) sau lungime (2)?'* este fie caracterul '1', fie caracterul 'D', sau să calculeze aria cercului în funcție de lungime, dacă răspunsul este '2' sau 'L', iar dacă răspunsul este diferit de aceste patru posibilități să se producă un sunet.
41. Venitul unei societăți comerciale este de $venit$ milioane lei, iar cheltuielile sunt de $chelt$ lei. Să se precizeze dacă profitul societății

este mai mare sau nu decât o valoare constantă $\text{profit_minim} = 500$ (milioane lei).

42. Să se precizeze dacă un triunghi cu lungimile celor trei laturi a , b și c este sau nu isoscel. În caz afirmativ, să se producă un sunet și să se afișeze *Tra-la-la* în mijlocul ecranului, altfel să se rezolve ecuația $ax^2+bx+c=0$, unde a și b sunt lungimile primelor două laturi ale triunghiului, iar c este semiperimetrul său.
43. Să se realizeze următorul joc: o bilă se mișcă pe ecran ca pe o masă de biliard (vezi problema 2). O paletă de lungime 9 (caractere), situată pe ultima linie a ecranului, se deplasează stânga-dreapta cu ajutorul a două taste (vezi problema 3). Când bila lovește paleta, în difuzor se aude un sunet mai lung, iar când ea lovește pereții laterali, un alt sunet, mai scurt. Jocul se oprește când se apasă o anumită tastă de stop.
44. Se citesc de la tastatură caractere, până la întâlnirea a două caractere care sunt identice. Să se determine câte caractere sunt cifre dintre cele citite.
45. De la tastatură se introduce o listă de numere întregi. Se cere să se afișeze valoarea maximă depistată în listă. Dimensiunea listei este precizată înainte de a fi introduse elementele ce o compun.
46. Să se scrie un program care să deseneze un dreptunghi umplut cu o anumită culoare (o cutie). Dreptunghiul se va specifica prin coordonatele colțurilor stânga-jos și lungimile laturilor, care, alături de culoare, vor fi introduse de la tastatură.
47. Să se scrie un program care să deseneze pe ecran dreptunghiuri de dimensiuni aleatoare, în poziții aleatoare și de culori aleatoare pe ecran, până se acționează o tastă numerică.
48. Să se “inverseze” (oglindească) un număr (care nu se termină cu cifra 0). De exemplu, pentru numărul 10758 să obținem numărul 85701.
49. Să se realizeze următoarea “piramidă” a numerelor:

```

1
1 2
1 2 3
.....
1 2 3 ... n

```

în care numărul n este citit de la tastatură.

50. Folosind, pe rând, fiecare din cele trei instrucțiuni repetitive, să se calculeze valoarea următorului produs:

$$P = \left(1 - \frac{1}{2^2}\right) \left(1 - \frac{1}{3^2}\right) \dots \left(1 - \frac{1}{n^2}\right).$$

51. Să se afișeze “piramida” de numere de mai jos:

```

n n-1 n-2 ... 3 2 1
.....
2 1
1

```

52. Să se producă în difuzor un sunet crescător, apoi descrescător și tot așa, până se acționează o tastă cu o literă mare.

53. Să se afișeze piramida de numere:

```
1
1 2 3
1 2 3 4 5
```

.....
1 2 3 4 ... (2n-1)

54. Să se calculeze, folosind instrucțiunea `do`, suma $S=2+4+6+\dots+(2n)$, unde $n \geq 1$.
55. Să se producă în difuzor un sunet aleator, până se acționează o tastă.
56. Realizați un program care să afișeze numele anotimpului corespunzător unui număr întreg citit de la tastatură (1..4). Scrieți două variante de program: folosind `if` și folosind `switch`.
57. Scrieți un program care să pună întrebarea: "În ce an s-a născut Eminescu ?" și, dacă răspunsul este corect (1850), atunci să afișeze "Foarte bine", altfel un text corespunzător. Folosiți instrucțiunea `switch`.
58. Se citește un șir de numere întregi până la întâlnirea numărului 0. Să se calculeze media aritmetică a numerelor din șir.
59. Se citesc 3 numere naturale n , p și k , apoi un șir de n numere naturale. Câte dintre acestea, împărțite la p dau restul k ?
60. Să se calculeze produsul a două numere naturale prin adunări repetate.
61. Efectuați împărțirea întreagă a două numere, fără a utiliza operatorii `/` și `%`, ci doar scăderi repetate.
62. Se citește un număr natural. Câte cifre conține ?
63. Un număr se numește "palindrom" dacă citit invers este același număr. Să se verifice dacă un număr este sau nu palindrom.
64. Să se afișeze toate numerele prime mai mici sau egale cu un număr m dat.
65. Să se afișeze primele n numere prime care au suma cifrelor $\leq m$.
66. Să se afișeze toate numerele de 3 cifre care, citite invers, sunt tot numere prime.
67. Calculați și afișați suma: $1/(1 \times 2) + 1/(2 \times 3) + 1/(3 \times 4) + \dots + 1/(n \times (n+1))$.
68. Să se verifice dacă un număr natural este sau nu pătrat perfect.
69. Să se scrie un program care să rezolve câte o ecuație de gradul II, până utilizatorul programului nu mai vrea acest lucru.
70. Să se verifice dacă un număr natural este sau nu cub perfect.
71. Să se afișeze toate numerele de forma a^2+b^3 , cu $1 \leq a \leq 5$ și $1 \leq b \leq 5$.
72. Să se determine toate triunghiurile diferite cu laturi numere întregi pozitive și perimetru p .
73. Să se listeze toate numerele $\leq n$, a căror sumă a cifrelor este divizibilă prin 5.
74. Să se transforme un număr din baza 10 în baza $p < 10$. Să se transforme un număr din baza $p < 10$ în baza 10.
75. Să se transforme un număr din baza p în baza q , unde $p, q \leq 10$.

76. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze soluțiile tuturor ecuațiilor de gradul I $ax+b=0$, unde a și b sunt toate perechile de numere citite consecutiv, în care b este divizibil prin a .
77. Se citesc numere naturale până la introducerea unui număr real. Să se calculeze suma S a tuturor numerelor citite, precum și câtul și restul împărțirii lui S la suma cifrelor lui S .
78. Se citesc numere naturale până la întâlnirea numărului 12. Să se afișeze toate tripletele de numere citite consecutiv, în care al treilea număr este restul împărțirii primului la al doilea.
79. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate tripletele de numere citite consecutiv, în care al treilea număr este media aritmetică (geometrică) dintre primul și al doilea.
80. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate perechile de numere citite consecutiv, cu proprietatea că al doilea număr este egal cu suma cifrelor primului număr.
81. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate perechile de numere citite consecutiv, cu proprietatea că al doilea număr reprezintă restul împărțirii primului număr la suma cifrelor sale.
82. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate perechile de numere citite consecutiv, cu proprietatea că al doilea număr reprezintă numărul de apariții ale cifrei 3 în pătratul primului.
83. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate perechile de numere citite consecutiv, cu proprietatea că al doilea număr reprezintă pătratul numărului de apariții ale cifrei 1 în primul.
84. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate tripletele de numere citite consecutiv, cu proprietatea că al treilea număr este suma dintre primul și al doilea.
85. Se citesc numere reale până la întâlnirea numărului 0. Să se afișeze toate tripletele de numere citite consecutiv, cu proprietatea că ele pot reprezenta laturile unui triunghi.
86. Se citesc numere reale până la întâlnirea numărului 0. Să se afișeze toate tripletele de numere citite consecutiv, cu proprietatea că ele pot reprezenta laturile unui triunghi isoscel.
87. Se citesc numere reale până la întâlnirea numărului 0. Să se afișeze toate tripletele de numere citite consecutiv, cu proprietatea că ele pot reprezenta laturile unui triunghi dreptunghic.
88. Se citesc numere reale până la întâlnirea numărului 0. Să se afișeze soluțiile tuturor ecuațiilor de gradul 2 $ax^2+bx+c=0$, unde a , b și c sunt toate tripletele de numere citite consecutiv, în care $b^2-4ac=0$.
89. Se citesc numere reale până la întâlnirea numărului 0. Să se afișeze soluțiile tuturor ecuațiilor de gradul II $ax^2+bx+c=0$, unde a , b și c sunt toate tripletele de numere citite consecutiv, în care $b^2-4ac>0$.
90. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate perechile de numere (n_1, n_2) citite consecutiv, cu proprietatea că $n_1 > n_2$ și suma cifrelor lui n_1 este mai mică decât suma cifrelor lui n_2 .

91. Să se găsească toate reprezentările posibile ale numărului natural n ca sumă de numere naturale consecutive.
92. Să se calculeze determinantul unei matrice pătratice de ordinul doi și a unei matrice pătratice de ordinul trei.
93. La un concurs sportiv s-au înscris patru echipe A, B, C și D. La sfârșitul tuturor probelor cele patru echipe au obținut respectiv X, Y, Z și T puncte. Să se afișeze valoarea de adevăr a propoziției: "Echipa A a obținut cel mai mare punctaj."
94. Scrieți un program care să se rezolve ecuația de gradul III: $ax^3+bx^2+cx+d=0$, a, b, c, d numere reale, $a \neq 0$, folosind *metoda lui Cardano*.
95. Generați trei numere reale din intervalul $[0, 1)$ și apoi calculați și afișați media lor aritmetică.
96. Prima defectare a unui calculator survine după o durată de funcționare specificată prin trei numere întregi: h ore, m minute și s secunde. Scrieți un program care să determine durata de funcționare în secunde.
97. Scrieți un program care să rezolve un sistem de două ecuații liniare cu două necunoscute: $a_1x+b_1=c_1$, $a_2x+b_2=c_2$.
98. Scrieți un program care să determine elementele mulțimii $\{x | x \in \mathbb{R}, \cos(n \times \arccos(x)) = 0\}$.
99. Calculați media armonică a trei numere reale x, y și z .
100. Scrieți funcții și apoi dezvoltați programe C pentru rezolvarea următoarelor probleme:
 - a) Fie $A(x_1, x_2)$ și $B(x_2, y_2)$ două puncte din planul euclidian. Afișați:
 - distanța euclidiană de la A la B;
 - coordonatele mijlocului segmentului $|AB|$.
 - b) Fie $A(x_1, y_1)$, $B(x_2, y_2)$ și $C(x_3, y_3)$ cele trei puncte din planul euclidian ce reprezintă colțurile unui triunghi ABC. Să se determine:
 - coordonatele mijloacelor laturilor;
 - lungimile laturilor;
 - lungimile medianelor.
 - c) Pentru datele de la punctul anterior, să se determine coordonatele cercului înscris în triunghi și a celui circumscris triunghiului, precum și razele acestor cercuri.
 - d) Să se determine coeficienții a, b și c ai dreptei $ax+by+c=0$, care trece prin punctele P și Q de coordonate date.
101. Scrieți un program care să verifice dacă trei puncte din planul euclidian, date prin coordonatele lor (x_1, y_1) , (x_2, y_2) și (x_3, y_3) sunt sau nu coliniare.
102. Să se verifice dacă trei drepte, date prin ecuațiile lor de forma $ax+by+c=0$, sunt sau nu concurente.
103. Să se calculeze distanța de la un punct $P(x_p, y_p)$ la dreapta $ax+by+c=0$.
104. Dându-se trei numere reale a, b și c , să se rezolve ecuația $ax^4+bx^2+c=0$, $a \neq 0$.

105. Scrieți un program C care să studieze natura și semnul soluțiilor unei ecuații de gradul doi.
106. Se citește un număr natural n și se cere să se afișeze descompunerea lui în factori primi.

Capitolul 2. Tablouri și pointeri

- șiruri de caractere • tablouri • legătura pointerilor cu tablourile •
- operații pe biți •

Probleme rezolvate

1. Să se scrie un program care să contorizeze aparițiile fiecărei cifre, a fiecărui caracter de spațiere (blanc, tab, linie nouă) și a tuturor celorlalte caractere.

```
#include <stdio.h>
void main()
{
    int c, i, n_sp=0, n_alte=0;
    int n_cifre[10];
    for (i=0; i<10; n_cifre[i++]=0);
    while ((c = getchar()) != EOF)
        if (c >= '0' && c <= '9')
            ++n_cifre[c-'0'];
        else if (c == ' ' || c == '\n' || c == '\t')
            ++n_sp;
        else
            ++n_alte;
    printf("Cifrele: ");
    for (i=0; i<10; i++)
        printf(" %d", n_cifre[i]);
    printf("\nCaractere de spatiere: %d, alte
           caractere: %d\n", n_sp, n_alte);
}
```



Exemplul ne permite să punem în evidență mai multe aspecte ale limbajului:

- declarația `int n_cifre[10]` reprezintă crearea unui tablou unidimensional (vector, șir) de 10 numere întregi, indexate de la 0 la 9 (unde `n_cifre[c]` reprezintă numărul de apariții ale cifrei `c` în fișierul standard de intrare (de exemplu tastatura)); indicele poate fi orice expresie întreagă, inclusiv variabilele întregi ca `i` sau constantele întregi;
- incrementarea `++n_cifre[c-'0']` pune în evidență diferența între două caractere, adică diferența între codurile lor *ASCII*, care se transformă în număr întreg, adică într-un indice cu valori între 0 și 9;
- instrucțiunea `for` pune în evidență un mecanism de utilizare a operatorului `++` pentru indicii `i`; astfel, ciclul `for` de acolo face o inițializare a elementelor vectorului cu 0; dacă am fi scris `n_cifre[++i]`, diferența era că, în acest al doilea caz, mai întâi se incrementa `i`, apoi se folosea pe post de indice, ceea ce ar fi însemnat că `i` ar fi fost folosit ca indice de la 1 la 10 și nu de la 0 la 9, ca în cazul descris;

- reprezentarea caracterelor de trecere la o noua linie (*Enter*): '`\n`' și de *TAB*: '`\t`';
- folosirea constantei `EOF` care reprezintă sfârșitul de fișier; practic, la rularea programului, pentru a termina ciclul "`while`", va trebui tastat *Ctrl-Z*.

2. Să se scrie un program care citește mai multe linii și o tipărește pe cea mai lungă.



Vom pune în evidență, cu ajutorul acestui exemplu, lucrul cu tablourile de caractere. Schița programului este simplă:

```
while (mai există o altă linie)
    if (este mai lungă decât linia anterioară)
        salveaz-o pe ea și lungimea ei;
tipărește linia cea mai lungă
```

Vom scrie o funcție `getline` care să citească următoarea linie de la intrare (un fel de generalizare a lui `getchar`). `getline` va trebui, în cel mai defavorabil caz, să returneze un semnal despre un eventual sfârșit de fișier; ea va returna lungimea liniei, respectiv 0, când se întâlnește sfârșitul de fișier; zero nu este niciodată o lungime validă de linie, deoarece orice linie are cel puțin un caracter, chiar și o linie ce conține doar caracterul "linie nouă" are lungimea 1. Salvarea unei linii mai lungi decât linia anterioară se va face cu funcția `copy`.

```
#include <stdio.h>
#define lung_max 1000 // lungimea maxima a unei linii
int getline(char s[], int limita)
```

Primul argument al funcției este un tablou de caractere (nu se precizează lungimea), iar cel de al doilea un număr întreg; comunicarea cu `main()` se face astfel: `main`→`getline` : prin argumente; `getline`→`main` : prin rezultatul funcției `getline`.

```
{
    int c, i;
    for (i=0; i < limita-1 && (c = getchar()) != EOF
        && c != '\n'; ++i)
```

Atât timp cât nu s-a atins limita maximă admisă și nu s-a ajuns la sfârșitul de fișier (deci nu s-a tastat *Ctrl-Z*, de pildă) și nici nu s-a ajuns la o linie nouă (tastarea lui *Enter*), se preia în `c` un caracter de la intrarea standard.

```
s[i] = c; // acesta se pune in tabloul s, pe pozitia i
if (c == '\n')
```

Dacă ultimul caracter introdus era '`\n`', acesta se adaugă liniei `s`.

```
s[i++] = c;
s[i] = '\0';
```

La sfârșitul liniei se pune caracterul cu codul 0 care simbolizează, în cazul tablourilor de caractere sfârșitul de sir: caracterul `NULL`.

```
return i;
```

```
}
void copy(char s1[], char s2[])
```

Copiază în tabloul `s2` conținutul tabloului `s1`, firește până se întâlnește caracterul de sfârșit de șir '`\0`'.

```

{
    int i=0;
    while ((s2[i++] = s1[i]) != '\0');
}

    Mai întâi s2[i] primește valoarea lui s1[i], iar apoi i crește cu 1.
void main()
{
    int lung, max;
// lungimea liniei curente si a celei maxime pana acum
    char linie[lung_max];
// linia curenta introdusa = un vector de caractere
    char linie_maxima[lung_max]; // cea mai lunga linie
    max=0;
    while ((lung = getline(linie,lung_max)) > 0)
        if (lung > max)
            { max = lung; copy(linie,linie_maxima); }
    if (max > 0)
        printf("Linia maxima este: %s Ea are lungimea: %d.",
                linie_maxima,max);
    else printf("Nici o linie citita...");
}

```



Dacă rulați programul, veți vedea că funcția `copy` lucrează corect și ne referim aici la faptul că valoarea primită de `s2` este trimisă în exterior, așa cum se întâmpla în exemplele unde vorbeam despre transmiterea parametrilor prin referință.

Acest lucru este posibil, deoarece în C o expresie de forma:

`<exp1>[exp2]`

este, de fapt, definită ca un pointer:

`*((exp1) + (exp2)).`

Rulând programul veți observa că mesajul *“Ea are lungimea ...”* va fi afișat pe o linie nouă, iar asta deoarece `linie_maxima` conține drept ultim caracter pe `'\n'`. `%s` folosește pentru afișarea șirurilor de caractere.

3. Să se rescrie programul de la exemplul anterior folosind declarații de variabile globale.



O variabilă globală trebuie să fie definită în afară oricărei funcții (deci nici măcar în funcția `"main"`); acest lucru face să se aloce memorie reală pentru ea. Astfel de variabile pot fi accesate prin numele lor de orice funcție din program. Ele își păstrează valorile și după ce funcția care le-a folosit și, eventual, setat și-a terminat execuția.

Vom rescrie programul precedent, în care `linie`, `linie_maxima` și `max` vor fi declarate variabile globale, externe oricărei funcții. Astfel, antetele, corpurile și apelurile funcțiilor se vor schimba.

```

#include <stdio.h>
#define lung_max 1000 // lungimea maxima a unei linii
char linie[lung_max], linie_maxima[lung_max];
int getline()

```

```

{
    int c, i;
    for (i=0; i < lung_max-1 && (c = getchar())
        != EOF && c != '\n'; ++i) linie[i] = c;
    // acesta se pune in tabloul s, pe pozitia i
    if (c == '\n')
    // daca ultimul caracter introdus era '\n', acesta se adauga
        linie[i++] = c; // liniei s
    linie[i] = '\0';
    // la sfarsitul liniei se pune caracterul cu codul 0,
    // care simbolizeaza, in cazul tablourilor de caractere,
    // sfarsitul de sir
    return i;
}
void copy()
{
    int i=0; while ((linie_maxima[i++] = linie[i]) != '\0');
}

void main()
{
    int lung, max;
    // lungimea liniei curente si a celei maxime pana acum
    max=0;
    while ((lung = getline()) > 0)
        if (lung > max) // !!
            { max = lung; copy(); }
    if (max > 0)
        printf("Linia maxima este: %s Ea are lungimea: %d.",
            linie_maxima,max);
    else printf("Nici o linie citita...");
}

```

Observație: dacă se vor introduce mai multe linii de aceeași lungime maximă, se va observa că programul o afișează pe prima linie din ele. Dacă relația notată '!!' s-ar scrie: "lung >= max", atunci s-ar afișa ultima dintre aceste linii maxime.

4. Scrieți o funcție care să returneze lungimea unui șir de caractere.

Funcția se numește strlen; ea nu ia în considerare caracterul NULL.

```

#include <stdio.h>
int strlen(char s[])
/* functia este o rescriere a lui 'strlen' din fisierul
"string.h", care contine mai multe declaratii referitoare la
sirurile de caractere si prelucrarea lor */
{
    int i=0; while (s[i++] != '\0');
    return i-1;
}
void main()
{
    char s[20];
    puts("\nDati sirul: "); gets(s); int l=strlen(s);
    printf("Lungimea sirului este: %d",l);
}

```

5. Scrieți o funcție care convertește un șir de caractere într-un întreg.

De fapt, avem de rescris funcția `atoi`, definită în `<stdlib.h>`:

```
#include <stdio.h>
int atoi(char s[])
{
    int i, n=0;
    for (i=0; s[i] >= '0' && s[i] <= '9'; i++)
        n = 10*n + s[i] - '0';
    /* conversia se opreste odata cu intalnirea unui caracter ce nu
    este cifra */
    return n;
}
void main()
{
    char s[10];
    puts("\nDati sirul: "); gets(s);
    printf("Numarul este: %d.", atoi(s));
}
```



Dupa cum am arătat și în exemplul **Q1**, expresia `s[i] - '0'` reprezintă valoarea numerică a caracterului aflat în `s[i]`, deoarece valorile caracterelor `'0'`, `'1'` etc. formează un șir crescător pozitiv și contiguu.

6. Scrieți un program care citește un șir de caractere și afișează șirul de caractere doar cu litere mici.

Vom scrie o funcție `lower` care va transforma în literă mică o majusculă din setul ASCII. #include <stdio.h>

```
#include <conio.h>
char lower(int c)
// se foloseste conversia intre char si int !
{
    if (c >= 'A' && c <= 'Z') return c+'a'-'A';
    else return c;
}
void main()
{
    clrscr();
    char s[30], t[30];
    puts("Dati sirul initial: "); gets(s);
    for (int i=0; s[i]; t[i] = lower(s[i]), i++);
    t[i] = NULL; // se marcheaza sfarsitul sirului t
    printf("Sirul final este: \n");
    printf("%s", t);
    getch();
}
```

De observat că se putea folosi o instrucțiune `for` ca următoarea:

`for (int i=0; i<=strlen(s); ...)`, adică folosind funcția descrisă într-un exemplu anterior.

7. Să se elimine toate aparițiile caracterului spațiu dintr-un text.

Vom scrie o funcție mai generală `elimina(char s[], char c)` care va elimina toate aparițiile caracterului `c` din șirul `s`.

```
#include <stdio.h>
void elimina(char s[], char c)
{
    int d=c;
    // se foloseste conversia unui caracter la un intreg
    int i,j;
    for (i = j = 0; s[i]; i++) // o atribuire multipla: i=j=0
        if (s[i] != d)
            s[j++] = s[i];
        // intai se foloseste j ca indice in s, apoi j=j+1
    s[j] = NULL; // adica '\0'
}
void main()
{
    char s[40];
    puts("Dati textul: "); gets(s); elimina(s, ' ');
    puts("A ramas textul: "); puts(s);
}
```

8. Scrieți o funcție `getbits(x, p, n)` care să returneze (ca drapt la dreapta) câmpul de lungime `n` biți al lui `x`, care începe la poziția `p`.

Presupunem că bitul 0 este cel mai din dreapta și că `n` și `p` sunt valori pozitive sensibile. De exemplu, `getbits(x, 4, 3)` returnează 3 biți în pozițiile 4, 3 și 2, ca drapti la dreapta.

```
#include <stdio.h>
unsigned getbits(unsigned x, unsigned p, unsigned n)
{
    return ((x >> (p+1-n)) & ~(~0 << n));
}
void main()
{
    int x=13; // 00001101
    printf("\nx=%d", x);
    int y=getbits(x, 4, 3);
    printf("\ny=%d", y); // 3, adica 011
}
```

Funcția `getbits` are argumente de tip `unsigned`, iar tipul returnat este tot `unsigned`. `unsigned`, alături de `signed`, `short` și `long` se numesc **modificatori de tip**. Un astfel de modificador schimbă semnificația tipului de bază pentru a obține un nou tip.

Fiecare dintre acești modificatori poate fi aplicat tipului de bază `int`.



Modificatorii `signed` și `unsigned` pot fi aplicați și tipului `char`, iar `long` poate fi aplicat lui `double`.

Când un tip de bază este omis din declarație, se subînțelege că este

```
int. Exemple:      long      x;      → int este subînțeles
unsigned char      ch;
signed int          i;      → signed este implicit
unsigned long int   l;      → nu era nevoie de int
```

Acest exemplu pune în evidență și un alt aspect al limbajului C, cum ar fi **operatorii pentru manipularea biților**, din care noi am folosit doar câțiva.



Operatorii pentru manipularea biților sunt:

- & {!, bit cu bit
- | SAU inclusiv, bit cu bit
- ^ SAU exclusiv, bit cu bit
- << deplasare la stanga a bitilor (despre care am mai vorbit)
- >> deplasare la dreapta a biților
- ~ complement față de 1 (este un operator unar)

Acestia nu se pot aplica lui `float` sau lui `double`.

Operatorul “{!, bit cu bit” “&” este folosit adesea pentru a masca anumite mulțimi de biți.

De exemplu, `c = n & 0177` pune pe zero toți biții lui `n`, mai puțin bitul 7, (cel mai tare).

Operatorul “SAU, bit cu bit” “|” este folosit pentru a pune pe 1 biți:

`x = x | MASK` pune pe 1, în `x`, biții care sunt setați pe 1 în `MASK`.

Trebuie să distingeți cu grijă operatorii pe biți “&” și “|” de conectorii logici “&&” și “||”. De exemplu, dacă `x` este 1 și `y` este 2, atunci `x & y` este 0, dar `x && y` este 1!

Operatorul unar “~” dă complementul față de 1 al unui întreg, adică el convertește fiecare bit de 1 în 0 și invers.

Un exemplu de utilizare ar fi: `x&~077`. Aici se maschează ultimii 6 biți ai lui `x` pe 0. De notat că `x&~077` este independent de lungimea cuvântului și deci, preferabil, de exemplu, lui `x&0177700`, care vede pe `x` ca o cantitate de lungime 16 biți.

În funcția `getbits`, `x` este declarat `unsigned` pentru a ne asigura că la shiftarea spre dreapta, biții vacanți vor fi umpluți cu 0 și nu cu biții de semn (independent de implementarea de C!). `~0` este cuvântul cu toți biții pe 1. Prin operația `~0 << n` creăm o mască cu zerouri pe cei mai din dreapta `n` biți și 1 în rest; complementându-l apoi cu `~`, facem o mască cu 1 pe cei mai din dreapta `n` biți.

Propunem cititorului să rescrie `getbits` pentru a număra biții de la stânga la dreapta. Un alt exercițiu ar putea fi scrierea unei funcții `right_rot(n,b)` care să rotească întregul `n` la dreapta cu `b` poziții.

9. Scrieți o funcție `bitcount(n)` care să contorizeze numărul de biți pe 1 dintr-un argument întreg.

```
#include <stdio.h>
int bitcount(unsigned n)
{
    for (int b=0; n; n ==>> 1) // este echivalent cu n=n>>1
        if (n & 01)
            b++;
    return b;
}
```

```

void main()
{
    long x; printf("\nDati x: "); scanf("%ld",&x);
    printf("Avem %d biti pe 1 in %ld",bitcount(x),x);
}

```

10. Dându-se o expresie (cuprinzând paranteze rotunde, operanzi litere mici ale alfabetului latin și operatorii +, -, * și /), să se scrie în forma poloneză posfixată.

De exemplu, pentru expresia "a*(b+c)" se obține abc+*, iar pentru expresia a*(b+c)-e/(a+d)+h se obține abc+*ead+/-h+.

Fie *s* expresia inițială și *fp* forma sa poloneză. Algoritmul de rezolvare a problemei folosește o stivă a operatorilor, notată cu *st*. Se citește câte un caracter *s[i]* din expresie și:

a) dacă el este '(', atunci se trece în stivă;

b) dacă este ')' se trec în *fp* toți operatorii, până la întâlnirea lui '(', care se șterge din stivă;

c) dacă *s[i]* este un operator aritmetic, se compară prioritatea sa cu a celui operator aflat în vârful stivei și:

dacă este mai mică, se copiază din stivă în *fp* toți operatorii cu prioritate mai mare sau egală decât acesta, apoi acest operator e trecut în stivă;

dacă nu, se trece direct în stivă acest operator;

în orice alt caz, rămâne că s-a citit un operand, iar acesta se trece în *fp*.

Programul C care rezolvă problema este dat mai jos:

```

/* FormaPoloneza */
#include <stdio.h>
#include <string.h>
int pr(char c)
{
    switch(c) {
        case '(': return 0;
        case '+':
        case '-': return 1;
        case '*':
        case '/': return 2;
    }
    return -1;
}

void main()
{
    char t[30]="", s[30]="", fp[30]="", st[30]="";
    int i,sp=-1,k=-1;

    printf("Expresia: "); scanf("%s",&t);
    strcpy(s,"("); strcat(s,t); strcat(s,")");
    for (i=0; i<=strlen(s); i++)
        switch (s[i]) {
            case '(': st[++sp]='('; break;
            case ')': while (st[sp]!='(') fp[++k]=st[sp--];
                       sp--; break;
            case '+': case '-': case '*':

```

```

        case '/':
            if (pr(st[sp])>=pr(s[i]))
            {
                while (pr(st[sp])>=pr(s[i]))
                { fp[++k]=st[sp]; sp--; }
                st[++sp]=s[i];
            }
            else st[++sp]=s[i];
            break;
        default: fp[++k]=s[i];
    }
    printf("Forma poloneza este: %s\n", fp);
    fflush(stdin); getchar();
}

```

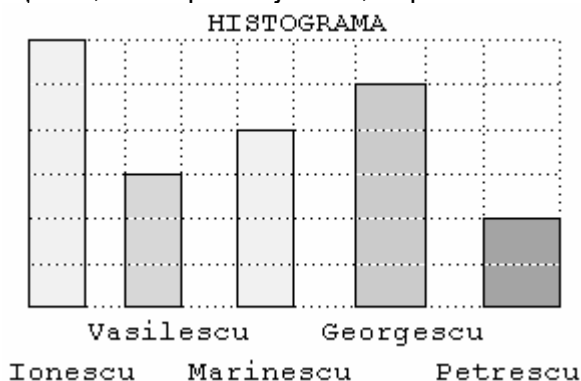
Probleme propuse

1. Se citesc de la intrare mai multe linii. Scrieți un program care să tipărească toate liniile mai scurte de 80 de caractere.
2. Să se elimine spațiile nesemnificative (cele de după un caracter diferit de spațiu sau tab) din fiecare linie de intrare și, de asemenea, ștergeți liniile care conțin doar spații.
3. Elaborați un program care să "împăturească" liniile de intrare lungi după ultimul caracter diferit de spațiu care apare înainte de a n -a coloană a intrării, unde n este un parametru. Asigurați-vă că programul dumneavoastră lucrează inteligent cu liniile foarte lungi, chiar dacă nu e nici un tab sau spațiu înainte de coloana respectivă.
4. Scrieți o funcție `void rightrot(unsigned n, unsigned b)` care rotește întregul n la dreapta cu b poziții.
5. Scrieți o funcție `void invert(unsigned x, unsigned p, unsigned n)` care inversează cei n biți ai lui x , care încep de la poziția p , lăsându-i pe ceilalți neschimbați.
6. Scrieți o funcție care șterge fiecare caracter din șirul $s1$ care se potrivește cu vreun caracter din șirul $s2$.
7. Scrieți o funcție `lower` care să convertească literele mari în litere mici pentru un șir de caractere, folosind o expresie condițională și nu `if`.
8. Scrieți programe care să realizeze inversarea unui vector: a) în același vector și fără a utiliza un vector suplimentar; b) într-un alt vector.
9. Scrieți un program C care să determine simultan minimul și maximul unui vector de numere reale.
10. Scrieți o funcție care să expandeze notațiile scurte de tipul $a-z$ într-un șir $s1$ în lista echivalentă și completă $abc..xyz$ în șirul $s2$. Sunt permise atât litere mari, mici, cât și cifre. Se vor trata și cazuri de tipul $a-b-c$ și $a-z0-9$ sau $-a-z$.
11. Pentru un vector de numere reale dat, să se determine a) $S1$ = suma componentelor sale; b) $S2$ = suma pătratelor componentelor vectorului.
12. Să se afișeze doar elementele pare dintr-un vector de numere întregi.

13. Să se afișeze doar elementele de pe poziții impare dintr-un vector oarecare.
14. Să se determine numărul elementelor negative și a celor pozitive dintr-un vector de numere reale.
15. Se consideră un șir de n persoane, notat cu x . Să se verifice dacă, pentru o persoană p dată, există o persoană în x cu același nume ca și p . Dacă nu, să se insereze pe poziția k în x noua persoană p . Discuție după valoarea lui k .
16. Să se afișeze toate numere prime dintr-un tablou de numere întregi pozitive.
17. Fie declarațiile: `float a[20]; int n; float e`. Să se determine valoarea e în fiecare din cazurile:
 - $e = x_0 + x_1 + \dots + x_{n-1}$ (adică $x[0] + x[1] + \dots + x[n-1]$);
 - $e = x_1 + x_3 + \dots + x_{..}$;
 - $e = x_0 \cdot x_1 \cdot x_2 \cdot \dots \cdot x_n$;
 - $e = x_1 \cdot x_3 \cdot x_5 \cdot \dots \cdot x_{..}$;
 - $e =$ media aritmetică a componentelor din vector;
 - $e =$ suma cuburilor componentelor negative din vector;
 - $e = x_0 - x_1 + x_2 - x_3 + \dots \pm x_{n-1}$;
18. Fie doi vectori de numere reale x și y , din care doar componentele $0..n-1$ se folosesc. Să se determine vectorul de numere reale z , astfel încât:
 - $z[i] = x[i] + y[i]$;
 - $z[i] = x[i] - y[i]$;
 - $z[i] =$ minimul dintre $x[i]$ și $y[i]$.
19. Fie doi vectori x și y , din care doar primele n componente se folosesc. Să se determine expresia e calculată astfel:
 - $e = (x_0 + y_0) \cdot (x_1 + y_1) \cdot \dots \cdot (x_{n-1} + y_{n-1})$;
 - $e = x_0 \cdot y_0 + x_1 \cdot y_1 + \dots + x_{n-1} \cdot y_{n-1}$ // produsul scalar a doi vectori
 - $e = \min(x_0, y_{n-1}) + \min(x_1, y_{n-2}) + \min(x_2, y_{n-3}) + \dots + \min(x_{n-1}, y_0)$;
 - $e = x_0^2 \cdot y_1 + x_1^2 \cdot y_2 + \dots + x_{n-1}^2 \cdot y_0$.
20. Memorați în primele n componente ale unui vector x de numere întregi primele n numere prime.
21. Memorați în primele n componente ale unui vector x de numere întregi primele n numere prime mai mari decât 999, care citite invers sunt tot numere prime.
22. Fie un vector x de numere întregi. Să se formeze un vector y de numere întregi, în care $y[i]$ să fie reprezentarea în baza 2 a numărului $x[i]$.
23. Fie un vector x de numere întregi. Să se formeze un vector y de numere întregi, în care $y[i]$ să fie restul împărțirii lui $x[i]$ la suma cifrelor lui $x[i]$. Complicați exercițiul făcând împărțirea prin scăderi repetate!
24. Fie un vector x de numere întregi. Să se determine un număr p , care să fie cel mai mare număr prim din cadrul vectorului. Dacă nu există, atunci

p să fie egal cu 0. Dacă p nu este 0, atunci să se împartă (ca numere întregi) toate componentele lui x la suma cifrelor lui p . Complicați exercițiul, încercând să determinați radicalul prin metoda lui Newton, iar împărțirea să o faceți prin scăderi repetate!

25. Realizați, folosind tablouri, o aplicație practică referitoare la un concurs de admitere într-un liceu cu un singur profil. Există n elevi candidați, iar numărul de locuri este m . Fiecare elev ia două note. Primii m elevi sunt considerați reușiți dacă fiecare notă a lor este mai mare sau cel puțin egală cu 5. Aplicația va permite, așadar, ordonarea elevilor după media obținută, dar se cere și o ordonare alfabetică, precum și implementarea căutării unui elev după nume.
26. Să presupunem, în continuare, că au avut loc ordonările descrescătoare ale mediilor pentru două licee unde s-au susținut concursuri de admitere. Inspectoratul Școlar Județean ar putea fi interesat de situația pe ansamblu a notelor obținute de candidații de la ambele licee, deci ordinea descrescătoare, după medie, a reuniunii celor două mulțimi de candidați. O primă soluție ar fi să realizăm reuniunea mulțimilor de candidați, apoi să o sortăm. Dar ar fi mai bine să ne folosim de faptul că avem deja ordonați elevii pe două liste. De aceea, vă propunem să scrieți un program care face o interclasare a celor două liste.
27. Să presupunem că avem informații despre un număr de maxim 10 elevi (numele și punctajele lor obținute la un concurs de informatică). Vă propunem să realizați o histogramă a rezultatelor lor. Fiecare elev va fi trecut cu numele său și cu o bară verticală, de o anumită culoare, ce reprezintă situația sa, adică punctajul său, după modelul din figură:



28. Să presupunem că avem doi vectori A (de m elemente distincte) și B (de n elemente distincte). Să realizăm reuniunea și respectiv intersecția lor.
 - pentru reuniune, vom copia în vectorul rezultat ($Reun$) toate elementele din A , după care vom adăuga acele elemente din B care nu se găsesc și în A ;
 - pentru intersecție, vom pune în mulțimea rezultat ($Inters$) toate elementele din A care se regăsesc în B .

Cele două mulțimi ar putea cuprinde candidații la un concurs de admitere. Se poate verifica, făcând intersecția mulțimilor, dacă nu cumva

- există un elev care să se fi înscris la concurs la ambele licee, încălcând, astfel, legea!
29. Determinați valoarea unui polinom într-un punct. Coeficienții polinomului sunt memorați într-un tablou de numere reale.
 30. Memorând în tablouri coeficienții a două polinoame, determinați suma și diferența a două polinoame, precum și produsul lor. Să se împartă două polinoame, obținând în doi vectori câtul și restul împărțirii.
 31. Un efect grafic interesant (pentru începutul unui program mai complex) am putea obține prin vizualizarea mișcării unor bile de diferite culori pe ecran. Acestea să se deplaseze ca pe o masă de biliard, dar să se supună legii reflexiei, nu și a frecării. Vom avea astfel nevoie de 5 vectori: doi pentru coordonatele bilelor (x și y), unul pentru culorile bilelor (Culoare) și încă doi pentru deplasările relative (pe orizontală și pe verticală) ale celor n bile (dx și dy , având componente ce iau doar valorile -1 și $+1$). Scrieți un astfel de program.
 32. Scrieți o funcție care transformă un șir scris cu litere mici în același șir, cu litere mari. Apoi scrieți o funcție care face conversia inversă.
 33. Limba păsărească. Problema cere înlocuirea fiecărei vocale V dintr-un șir prin grupul pVp .
 34. Scrieți un program care să despartă o propoziție în cuvinte și să le numere.
 35. Afișarea unui șir prin litere care cad. Jocurile pe calculator, pe care le puteți realiza și dumneavoastră, cu cunoștințele de până acum, pot avea prezentări dintre cele mai frumoase. Un efect interesant și atractiv îl reprezintă căderea unor litere; ele se depun pe un rând al ecranului, formând un text. Realizați un astfel de program.
 36. Scrisul defilant. Pe linia 20 va apare, defilând de la dreapta la stânga, un mesaj. El apare din extrema dreaptă a ecranului și intră în cea stângă. Procesul se repetă până se apasă o tastă. Scrieți un program care să realizeze acest lucru.
 37. Într-o școală sunt m clase, fiecare având câte n elevi. Se cere să se determine care elev are media la învățătură cea mai mare.
 38. Într-o clasă există NrB băieți și NrF fete care îi iubesc, însă fiecare fată iubește numai anumiți băieți. Se cere să se determine băiatul pe care îl iubesc cele mai multe fete, precum și topul tuturor băieților. Dacă există cel puțin doi băieți cu punctajele egale, atunci nu există un unic "fericit" și se va afișa un mesaj corespunzător.
 39. Un elev din clasa I are la dispoziție n litere mici din alfabetul latin, din care m distincte. Doamna învățătoare îi cere următoarele lucruri: a) Să verifice dacă există litere care apar de mai multe ori și să păstreze toate literele distincte o singură dată. b) Să așeze aceste litere în ordine alfabetică. *Exemplu:* $n=6$, a,b,a,d,c,c sunt cele 6 litere. a) litere distincte: a,b,d,c ; b) ordinea alfabetică: a,b,c,d .
 40. Pentru un grup de n persoane, se definesc perechi de forma (i, j) cu semnificația *persoana i este influențată de persoana j* . Se cere să se determine persoanele din grup care au cea mai mare influență.

41. O matrice cu n linii și n coloane (n impar) care conține toate numerele naturale de la 1 la n^2 și în care suma elementelor de pe fiecare linie, coloană și diagonală este aceeași, se numește *pătrat magic* de ordinul n . De exemplu, un pătrat magic de ordinul 3 este:
42. Să se scrie un program care verifică - printr-o singură parcurgere - dacă o matrice cu n linii și n coloane este pătrat magic, adică o matrice pătratică în care suma de pe fiecare linie și cea de pe fiecare coloană, precum și sumele elementelor de pe diagonale coincid.
43. Se dă un șir a cu n elemente numere întregi. Să se scrie un program care determină ce element se află pe poziția k (dată) în șirul obținut din șirul a prin ordonare, fără a-l ordona pe a și fără a utiliza alte șiruri.
44. Numerele $C_1, C_2, C_3, \dots, C_n$ reprezintă valorile colesterolului a n persoane. O persoană este considerată sănătoasă dacă are colesterolul între două limite date a și b . Se cere să se afle numărul persoanelor sănătoase și media aritmetică a colesterolului persoanelor bolnave.
45. O persoană are de cumpărat din m magazine n produse care au prețuri diferite. Să se întocmească un program care să indice pentru fiecare produs magazinul în care acesta are prețul minim. Cunoscând cantitățile ce trebuie cumpărate din fiecare produs, să se determine suma ce urmează a fi cheltuită.
46. Se consideră n aruncări cu un zar. Se cere: a) Să se determine de câte ori apare fiecare față și la a câta aruncare. b) Să se determine toate perechile de forma (F_i, K_i) , cu proprietatea că $F_i + K_i$ este un număr par, unde F_i reprezintă numărul feței, iar K_i = numărul de apariții ale feței F_i .
47. Se dă un număr natural N . Să se verifice dacă numărul obținut prin eliminarea primei și ultimei cifre din N este pătrat perfect.
48. Să se determine cel mai mare divizor comun a n numere.
49. Se dă o propoziție în care cuvintele sunt separate fie prin spațiu, fie prin caracterele punct, virgulă, punct și virgulă, semnul exclamării și semnul întrebării. Despărțiți propoziția în cuvinte.
50. Dintr-o matrice dată M , să se listeze valorile tuturor punctelor sa și poziția lor. $M[i, j]$ este considerat punct sa dacă este minim pe linia i și maxim pe coloana j .
51. Să se elimine dintr-o matrice A (cu m linii și n coloane) linia l și coloana k și să se listeze matricea rămasă. (Nu se va folosi o altă matrice.).
52. Să se bordeze o matrice A (având m linii și n coloane) cu linia $m+1$ și coloana $n+1$, unde $A[m+1, j]$ să fie suma elementelor de pe coloana j , cu j de la 1 la n , iar $A[i, n+1]$ să fie suma elementelor de pe linia i , cu i de la 1 la m .
53. Fiind dat un vector V cu n componente elemente întregi, să se formeze un nou vector W , în care $W[i]$ să conțină suma (produsul) cifrelor elementelor lui $V[i]$.
54. Să se genereze aleator cuvinte de lungime 4.
55. Să se insereze între oricare două elemente dintr-un vector de numere reale media lor aritmetică.

56. Să se determine de câte ori se întâmplă ca într-un vector de numere reale să existe un element ce reprezintă produsul elementelor sale vecine.
57. Să se scrie în spirală numerele de la 1 la n^2 într-o matrice pătratică cu n linii și n coloane începând: a) din centrul matricei (n impar); b) din colțul stânga-sus. Se vor considera, pe rând, ambele sensuri de deplasare.
58. Se dă o matrice pătratică de ordin n . Se consideră că cele două diagonale împart matricea în patru zone: *nord*, *sud*, *est* și *vest* (elementele de pe diagonală nu fac parte din nici o zonă).
- Să se calculeze suma elementelor din *nord*, produsul elementelor din *sud*, media aritmetică a elementelor din *est* și numărul elementelor nule din *vest*.
 - Să se obțină simetrica matricei inițiale față de diagonala principală.
 - Să se obțină simetrica matricei inițiale față de diagonala secundară.
 - Să se obțină simetrica matricei inițiale față de o axă orizontală ce trece prin centrul matricei.
 - Să se obțină simetrica matricei inițiale față de o axă verticală ce trece prin centrul matricei.
59. Pentru o matrice cu m linii și n coloane, cu elemente numere reale, să se listeze toate liniile, precum și numerele sale de ordine, care conțin cel puțin k elemente nule.
60. Scrieți un program care verifică dacă o matrice este simetrică față de diagonala principală sau dacă are toate elementele nule.
61. Se dau două șiruri de numere întregi x cu n_x elemente și y cu n_y elemente, unde $n_x > n_y$. Să se decidă dacă y este un subșir al lui x , adică dacă un număr k , astfel încât: $x_k = y_1, x_{k+1} = y_2, \dots, x_{k+n_y-1} = y_{n_y}$. În caz afirmativ, se va afișa valoarea lui k .
62. Pentru două matrice cu elemente reale, A , cu m linii și n coloane, și B , cu n linii și p coloane, se numește *produsul matricelor* A și B matricea C , cu m linii și p coloane, în care elementul $C[i, j]$ este suma $A[i, 0] \times B[0, j] + A[i, 1] \times B[1, j] + \dots + A[i, n-1] \times B[n-1, j]$, pentru orice i între 1 și m și orice j între 0 și $p-1$. Să se scrie un program care citește două matrice A și B și calculează și afișează produsul lor, C .
63. Fie dat un vector $x = (x_0, x_1, \dots, x_{n-1})$. Să se modifice vectorul astfel încât, în final, să avem:
- $x = (x_1, x_2, \dots, x_{n-1}, x_0)$;
 - $x = (x_{n-1}, x_0, \dots, x_{n-2})$;
 - $x = (x_1, x_0, x_3, x_2, \dots, x_{n-1}, x_{n-2})$ (n - par).
64. Să se transforme un număr din baza 10 în baza 2, memorând numărul rezultat: a) într-un vector; b) într-un șir de caractere.

65. Să se realizeze deplasarea unei ferestre pe ecran, folosind tastele de cursor și, de asemenea, redimensionarea sa, la apăsarea altor două taste (de pildă `Home` și `End`).
66. Să se realizeze un joc de tip *"puzzle"* (*"perspico"*), în care jucătorul dispune de o matrice cu 4×4 căsuțe. În acestea sunt puse, pe rând, niște pătrățele cu numerele 1, 2, ..., 15, ultima căsuță fiind liberă. Se amestecă aceste numere, prin deplasarea căsuței libere sus, jos, stânga sau dreapta. Simulați amestecarea pătrățelelor și dați posibilitatea utilizatorului să refacă matricea, cu ajutorul tastelor de cursor. Generalizare pentru $n \times n$ căsuțe..
67. Numerele de la 1 la n sunt așezate în ordine crescătoare pe circumferința unui cerc, astfel încât n ajunge lângă 1. Începând cu numărul s , se elimină din cerc, numerele din k în k , după fiecare eliminare, cercul strângându-se. Care va fi numărul ce va rămâne?
68. Aceeași problemă ca cea anterioară, dar numerele nu se elimină, ci se marchează, până unul din ele va fi marcat de două ori. Câte numere au rămas nemarcate?
69. Fie A o matrice cu m linii și n coloane, cu elemente reale. Să se determine linia l și coloana k pe care suma elementelor este maximă.
70. Fie A o matrice cu m linii și n coloane, cu numere reale și un vector V cu n componente reale. Să se determine dacă acest vector apare ca linie în matricea A și, în caz afirmativ, să se afișeze numărul acestei linii.
71. Să se determine toate soluțiile de tip `short int` ale ecuației $7x - 4y = 3$.
72. Fie V un vector cu elemente de tip `Char`. Să se construiască un vector W , astfel încât $W[i] = \text{numărul de apariții ale lui } V[i] \text{ în vectorul } V$.
73. Se dă un vector de numere întregi pozitive. Să se determine cea mai lungă subsecvență de elemente consecutive prime, ale căror oglindite sunt tot numere prime.
74. Se dă un vector de numere întregi. Se cere să se afișeze subsecvența palindromică de lungime maximă. (Elementele subsecvenței sunt elemente consecutive în vector.)
75. Pentru un număr întreg n să se determine toate permutările șirului 1, 2, ..., n . Aceeași problemă pentru un șir de numere x_1, x_2, \dots, x_n .
76. O instituție cuprinzând n persoane trebuie să trimită într-o inspecție o echipă formată din m persoane. Câte echipe se pot forma și care sunt acestea?
77. Într-un raft încap m cărți din cele n de aceeași grosime, de care se dispune. Care sunt toate posibilitățile de a aranja cărțile pe raft?
78. Să se scrie un program care să determine produsul cartezian al n mulțimi, fiecare cu un număr finit p_i de elemente.
79. Să considerăm că avem o mulțime de m băieți și o mulțime de n fete. Ne punem problema de a cupla băieții cu fetele în toate modurile posibile, pentru a dansa fiecare din cele p dansuri, la o petrecere. Scrieți un program pentru soluționarea acestei probleme.
80. Scrieți un program care să genereze toate submulțimile unei mulțimi cu n elemente.

81. Cu ajutorul unui rucsac de greutate maximă admisibilă GG trebuie să se transporte o serie de obiecte din n disponibile, având greutatea G_1, G_2, \dots, G_n , aceste obiecte fiind de utilitățile C_1, C_2, \dots, C_n . Dacă pentru orice obiect i putem să luăm doar o parte $x_i \in [0, 1]$ din el, atunci spunem că avem *problema continuă a rucsacului*, iar dacă obiectul poate fi luat doar în întregime sau nu, spunem că avem *problema discretă a rucsacului*. Scrieți programe C care să rezolve ambele probleme.
82. Pe o tablă de șah cu $n \times n$ pătrate, să se așeze n dame care să nu se atace între ele.
83. Pe o tablă de șah cu $n \times n$ pătrate, să se așeze n piese care să nu se atace între ele și astfel încât piesele să fie câte una pe fiecare coloană a tablei de șah. Piesele mută sub forma: a) unui cal; b) unui nebun; c) unei ture; d) ca un cal și un nebun împreună; e) ca un cal și o tură împreună.
84. Se dă un șir (un vector) de n numere întregi. Se cere să se determine cel mai lung subșir crescător al acestuia. De exemplu, pentru vectorul $V = (4, 1, 8, 5, 8)$ de lungime $n=5$, răspunsul este: 4, 8, 8. Se va folosi metoda programării dinamice.
85. Unor elevi li se cere să pună $n < 200$ cuvinte formate doar din litere mici într-o ordine firească. De exemplu: *platon kant marx stalin havel* (ordine cronologică). Fiecare elev îi dă profesorului o succesiune de cuvinte pe care o consideră corectă. Problema pe care trebuie să o rezolve profesorul este de a puncta fiecare elev cu o notă între 1 și n , reprezentând numărul de cuvinte plasate într-o succesiune corectă. Pentru exemplul anterior, avem următoarele secvențe date de elevi și notele obținute:
- marx stalin kant platon havel 3
 - havel marx stalin kant platon 2
 - havel stalin marx kant platon 1
- Ajutați-l pe profesor !
86. Problema platourilor. Fie un tablou a : `array[0..n-1]` cu elemente de tip întreg cu proprietatea $a[0] \leq a[1] \leq \dots \leq a[n-1]$. Un *platou* este o subsecvență maximală de elemente consecutive în tablou care conține aceeași valoare. Lungimea unui platou este numărul de elemente ce compun tabloul. Se cere să se determine lungimea p a celui mai lung platou.
87. Să se scrie un program care să determine pentru un vector a numărul platourilor (vezi problema anterioară). Definim noțiunea de *pantă* ca fiind o secvență de elemente consecutive din a astfel încât $a[i] = a[i-1] + 1$. Să se determine panta de lungime maximă.
88. Numere pitagoreice. Să se determine, pentru un număr întreg n dat, toate tripletele de numere pitagoreice (a, b, c) (deci $c^2 = a^2 + b^2$) astfel încât $1 \leq a, b \leq n$.
89. Problema steagului național olandez. Se consideră n pionii de trei culori (roșu, alb și albastru), aranjați în linie. Să se rearanjeze, prin

interschimbări, acești pionii, astfel încât întâi să avem pionii roșii, apoi cei albi, apoi cei albaștri. Pionii se pot interschimba doi câte doi.

90. Să se scrie un program care să caute binar un element într-un vector. Se va folosi metoda iterativă "*divide et impera*".
91. La un concurs sunt prezentate 5 melodii A, B, C, D, E. Să se afle toate posibilitățile de a le prezenta, știind că melodia B va fi interpretată înaintea melodiei A.
92. Să se așeze $2n-2$ nebuni pe o tablă de șah cu n^2 pătrate astfel încât nici o pereche de nebuni să nu se amenințe.
93. Să se așeze pe o tablă de șah cu n^2 pătrate cât mai multe dame care să nu se atace între ele.
94. Pe produsul cartezian $N \times N$ se definește operația: $(a, b)(b, c) = (a, c)$, pentru orice a, b și $c \in N$, despre care știm că:
 - este asociativă: $((a, b)(b, c))(c, d) = (a, b)((b, c)(c, d))$;
 - efectuarea ei necesită exact $a \times b \times c$ secunde.

Fiind date $x_1, x_2, \dots, x_n \in N, n \geq 3$, care este timpul minim și cel maxim în care se poate efectua produsul $(x_1, x_2)(x_2, x_3) \dots (x_{n-1}, x_n)$? De exemplu, pentru produsul $(7, 1)(1, 9)(9, 3)$ avem rezultatul $(7, 3)$, care se poate obține cel puțin în 48 secunde și cel mult în 4 minute și 12 secunde.

95. Fie s o permutare a mulțimii $\{1, 2, \dots, n\}$. Să se scrie un program care să afișeze cel mai mic număr nenul k pentru care $s^k = e$, unde e este permutarea identică.
96. Fie s o permutare a mulțimii $\{1, 2, \dots, n\}$. Să se determine numărul de inversiuni și signatura permutării s .
97. Să se determine coeficienții polinomului $P(X) = X^n + a_1 X^{n-1} + \dots + a_n$, dacă se cunosc toate rădăcinile sale.
98. Scrieți un program C care să afișeze toate numerele întregi de n cifre, egale cu de k ori produsul cifrelor lor ($k \in N$).
99. Scrieți un program C care să afișeze toate numerele întregi de n cifre, egale cu suma pătratelor cifrelor lor.
100. Să se determine toate numerele de n cifre care adunate cu oglinditul lor dau un pătrat perfect.
101. Scrieți un program care, folosind un număr minim de interschimbări, să rearanjeze toate componentele unui vector de numere reale, astfel încât mai întâi să avem elementele negative, iar apoi cele pozitive.
102. Să se genereze primele 100 numere din mulțimea $M = \{1, 3, 4, \dots\}$, definită astfel:

a) $1 \in M$;

b) dacă $x \in M$, atunci și $2x+1$ și $3x+1 \in M$ (regula se poate aplica de un număr finit de ori);

c) nici un alt element nu mai poate fi din M .

Cum putem decide dacă un număr natural x este sau nu în M , fără a genera elementele lui M mai mici sau egale cu x ?

103. Scrieți un program C care, pentru două numere întregi, cu n cifre, memorate sub forma unor vectori, să determine suma, produsul lor,

precum și câtul și restul împărțirii primului laq al doilea, memorând rezultatele tot vectorial.

104. Pentru o matrice reală pătratică de dimensiune $n \times n$ să afișeze liniile crescător după: a) primul element al liniei; b) suma elementelor liniei; c) elementul minim al liniei.
105. Pentru o matrice reală pătratică de dimensiune $n \times n$ să se utilizeze proprietățile determinanților pentru a-i calcula determinantul.
106. Generați toate matricele cu 6 linii și 6 coloane, cu elemente din mulțimea $\{-1, 1\}$, astfel încât pe fiecare linie și coloană să fie un număr impar de -1 .
107. Se consideră n puncte în plan, date prin coordonate carteziene. Să se determine în care din aceste puncte putem alege centrul cercului de rază minimă ce conține cele n puncte date.
108. Se citesc n numere reale a_0, a_1, \dots, a_{n-1} . Să se calculeze coeficienții polinomului $f_n(x) = (x-a_1)(x-a_2) \dots (x-a_n)$.
109. Un profesor dispune de n culegeri de probleme, care conțin, respectiv p_1, p_2, \dots, p_n probleme. El vrea să găsească acel set de culegeri care totalizează un număr de probleme ce poate fi împărțit exact la numărul n de elevi ai săi.
110. Se dă un rucsac cu greutatea maximă admisibilă G . Se cere să se transport cu el acele obiecte, din n disponibile, astfel încât suma utilităților obiectelor luate în rucsac să fie cât mai apropiată (în valoare absolută) de o valoare dată. Obiectele au utilitățile u_1, \dots, u_n și greutatea g_1, \dots, g_n . Se vor considera două cazuri: a) obiectele nu pot fi secționare (se pun întregi); b) obiectele pot fi secționare.
111. Să se genereze și să se afișeze atât în baza 2, cât și în baza 10, toate numerele ale căror reprezentări au a cifre de 1 și b cifre (semnificative) de 0.
112. Pentru o mulțime finită M cu n elemente se dă o lege de compoziție sub forma unei matrice cu $n \times n$ căsuțe. Se cere să se verifice dacă această lege de compoziție determină pe M o structură de grup abelian.
113. Scrieți un program care generează aleator matrice de dimensiuni corespunzătoare, cu elemente reale și calculează produsul lor.
114. Scrieți un program care, dându-i-se un număr în cifre tipărește acel număr în cuvinte. De exemplu, pentru numărul 152365 se obține "o sută cinci zeci și două de mii trei sute șaiszeci și cinci".
115. Scrieți un program care să transcrie cu cifre romane un număr scris cu cifre arabe.
116. Să se scrie un program care să opereze asupra unei matrice pătratice M , de numere întregi cuprinse între 0 și 255, în felul următor. Se alege un număr p la întâmplare între 1 și lungimea curentă a matricei (n). Se adaugă la vectorul v , inițial vid, elementul de la intersecția liniei p cu coloana p , după care se elimină din matrice atât linia p , cât și coloana p . Procedura continuă, până când matricea se "golește" definitiv. Câte elemente va conține vectorul v în final?

Capitolul 3. Recursivitate

- funcții recursive • operații în vectori • divide et impera • turnurile din Hanoi •
 - sortare prin interclasare • căutare binară • problema damelor •
 - backtracking recursiv • problema labirintului •

Probleme rezolvate

1. Se consideră următoarele declarații și convenții:

```
typedef int vector[20];
```

x este, de obicei, un vector (șir de elemente), n = lungimea sa ($n \geq 1$), iar k și e numere întregi.

Se cere să se scrie funcții recursive pentru a determina, pentru un vector x de lungime n, următoarele mărimi:


- suma componentelor;**
- produsul componentelor;**
- numărul componentelor negative;**
- produsul componentelor pozitive;**
- suma elementelor pare din vector;**
- produsul elementelor de pe poziții impare;**
- suma elementelor pare de pe poziții divizibile prin k;**
- suma pătratelor componentelor cu cubul mai mic decât k;**
- suma numerelor prime din vector;**
- cel mai mare divizor comun al componentelor;**
- existența elementului e în vector;**
- prezența elementului e pe o poziție impară din vector;**
- numărul componentelor pozitive pare;**
- numărul componentelor impare din intervalul $[-2, 3]$;**
- de câte ori o componentă este media aritmetică întreagă a componentelor vecine; presupunem că există cel puțin 3 componente în vector;**
- de câte ori o componentă este egală cu produsul componentelor vecine; presupunem că există cel puțin 3 componente în vector;**
- media aritmetică a componentelor.**


Vom rezolva doar câteva din punctele problemei, scriind funcții corespunzătoare și apelându-le din main.


```
#include <stdio.h>
#include <conio.h>
typedef int vector[20];
// a. suma componentelor;
int Suma(vector x, int n)
{
    if (n== -1) return 0;
    else return (x[n]+Suma(x,n-1));
}
```

```

}
// b. produsul componentelor;
int Produs(vector x, int n)
{
    if (n==1) return 1;
    else return (x[n]*Produs(x,n-1));
}

 // c. numarul componentelor negative;
int NumarNegative(vector x, int n)
{
    if (!n) return (x[n]<0);
    else return (x[n]<0) + NumarNegative(x,n-1);
}

 // d. produsul componentelor pozitive;
int ProdusPozitive(vector x, int n)
{
    if (!n) return (x[n]>0 ? x[n] : 1);
    else return (x[n]>0 ? x[n] : 1)*ProdusPozitive(x,n-1);
}

 // r. media aritmetica a elementelor;
float MediaElementelor(vector x, int m, int n)
/* n este dimensiunea reala a vectorului, iar m este elementul
ultim considerat, adica cel dupa care are loc inductia */
{
    return (float)x[m]/n + (m ? MediaElementelor(x,m-1,n) : 0);
/* conversie la float ! */
}

void main()
{
    vector x; int n;
    printf("\n\nDati n: "); scanf("%d",&n);
    for (int i=0; i<n; i++)
    {
        printf("Dati x[%d]: ",i); scanf("%d",&x[i]);
    }
    // se apeleaza functia pentru ultimul element al vectorului
    printf("Suma: %d\n", Suma(x,n-1));
    printf("Produsul: %d\n", Produs(x,n-1));
    printf("Numar negative: %d\n", NumarNegative(x,n-1));
    printf("Produs pozitive: %d\n", ProdusPozitive(x,n-1));
    printf("Media aritmetica: %6.2f\n",
        MediaElementelor(x,n-1,n));
    getch();
}

```

2. Problema "turnurilor din Hanoi": se dau trei turnuri numerotate cu 1, 2 și 3. Pe turnul 1 se află n discuri, de diametre 1, 2, 3, ..., n (de sus în jos). Acestea trebuie mutate pe turnul 2, folosind eventual turnul 3, în cât mai multe mutări. O mutare constă în luarea unui disc din vârful unui turn

și așezarea sa în vârful altui turn, cu condiția ca diametrul discului pe care se așază să fie mai mare decât al discului în cauză.



Problema se rezolvă prin metoda “*divide et impera*”, folosind recursivitatea. Astfel, pentru a muta n discuri de la un turn p la un turn q , în condițiile impuse, va trebui să mutăm (recursiv) primele $n-1$ turnuri de la p la r (turnul auxiliar), apoi discul al n -lea se va muta direct de la p la q , urmând ca, în final, celelalte $n-1$ discuri să se aducă (recursiv) de la r la q . Algoritmul de bază este descris în funcția `hanoi` (vezi programul).

În continuare prezentăm o variantă cu grafică și animație (în mod text) pentru această problemă, în care numărul de discuri este constant, egal cu 8.

```
#include <stdio.h>
#include <dos.h> // pentru pauzele date de 'delay'
#include <conio.h>
#include <stdlib.h>
// declaratii globale
// virf[i] reprezinta linia de afisare al discului din varful
// turnului (tijeii) a i+1 -a
int virf[3];
// pauza intre doua miscari din timpul animatiei
int pauza=35;
```

Această funcție afișează un caracter `c`, însă testează și dacă s-a apasat o tastă sau nu. În caz afirmativ se iese forțat din program, printr-un apel al funcției `exit` (declarată în `<stdlib.h>`).

```
void punecar(char c)
{
    if (kbhit()) exit(1); else putchar(c);
}
/* determina coloana unde se afla turnul 'tija' */
int col_tija(int tija)
{
    return (7*tija+9+17*(tija-1));
}
void muta_dreapta(int disc, int tija1, int tija2)
{
    int i,k;
    for (i=col_tija(tija1)-disc; i<col_tija(tija2)-disc;i++)
    {
        delay(pauza); gotoxy(i,3);
        for (k=0;k<2*disc+1;k++) punecar(' ');
        gotoxy(i+1,3);
        for (k=0;k<2*disc+1;k++) punecar('²');
    }
}
void muta_stinga(int disc, int tija1, int tija2)
{
    int i,k;
    for (i=col_tija(tija1)-disc; i>col_tija(tija2)-disc;i--)
    {
        delay(pauza); gotoxy(i,3);
        for (k=0;k<2*disc+1;k++) punecar(' ');
    }
}
```

```

        gotoxy(i-1,3);
        for (k=0;k<2*disc+1;k++) punecar('²');
    }
}
void coboara(int disc, int tija)
{
    int i,k;
    for (i=3;i<virf[tija-1]-1;i++)
    {
        delay(pauza); gotoxy(col_tija(tija)-disc,i);
        for (k=0;k<2*disc+1;k++) punecar(' ');
        gotoxy(col_tija(tija)-disc,i+1);
        for (k=0;k<2*disc+1;k++) punecar('²');
    }
    virf[tija-1]--;
}
void ridica(int disc, int tija)
{
    int i,k;
    for (i=virf[tija-1];i>3;i--)
    {
        delay(pauza); gotoxy(col_tija(tija)-disc,i);
        for (k=0;k<2*disc+1;k++) punecar(' ');
        gotoxy(col_tija(tija)-disc,i-1);
        for (k=0;k<2*disc+1;k++) punecar('²');
    }
    virf[tija-1]++;
}

```

Această funcție mută discul din vârful turnului tija1 în vârful lui tija2, apelând funcțiile de mai înainte:

- îl ridică pe tija1;
- îl mută de la tija1 la tija2;
- îl coboară pe tija2:

```

void muta(int disc,int tija1,int tija2)
{
    ridica(disc,tija1);
    if (tija1 < tija2) muta_dreapta(disc,tija1,tija2);
    else muta_stinga(disc,tija1,tija2);
    coboara(disc,tija2);
}

```

Procedura recursivă ce implementează algoritmul descris este:

```

void hanoi(int n,int tija1,int tija2,int tija3)
{
    if (n == 1)
        muta(1,tija1,tija2);
    else
    {
        hanoi(n-1,tija1,tija3,tija2);
        muta(n,tija1,tija2);
        hanoi(n-1,tija3,tija2,tija1);
    }
}
void initializari(void)
{

```

```

    int k,disc;
    virf[0]=13;
    virf[1]=virf[2]=22;
    clrscr();
    for (disc=1;disc<=9;disc++)
    {
        gotoxy(col_tija(1)-disc,virf[0]+disc-1);
        for (k=0;k<2*disc+1;k++)
            punecar('2');
    }
}

void main(void)
{
    clrscr(); initializari(); gotoxy(32,1);
    puts("~Turnurile din Hanoi~");
    hanoi(8,1,2,3); getch();
}

```

3. Să se scrie un program care ordonează un șir de numere prin metoda sortării prin interclasare.

```

#include <stdio.h>
#include <conio.h>
typedef int vector[20];
void Citeste(vector x, int *n)
{
    int p;
    printf("Dati nr. de elemente: "); scanf("%d",&p);
    *n=p;
    for (int i=0; i<p; i++)
    {
        printf("Dati x[%d]=" ,i);
        scanf("%d",&x[i]);
    }
}

void Afiseaza(vector x, int n)
{
    printf("Elementele lui x sunt: ");
    for (int i=0; i<n; i++)
        printf("%d",x[i]);
    printf("%c.\n\n",8);
}

```



Următoarea funcție interclasează cele două părți ale unui vector *a*, situate între *inceput* și *mijloc*, pe de o parte, și *mijloc+1* și *sfarsit*, pe de alta, unde *mijloc* este $(inceput+sfarsit)/2$. (Vezi și exemplul Q13.)

```

void Interclasare(vector a, int inceput,
                  int mijloc, int sfarsit)
{
    int i, j, k; vector c;
    i = inceput; j = mijloc+1; k = 0;
    while (i<=mijloc && j<=sfarsit)
        if (a[i]<a[j]) c[k++]=a[i++];
        else c[k++]=a[j++];
}

```

```

while (i<=mijloc) c[k++]=a[i++];
while (j<=sfarsit) c[k++]=a[j++];
for (i=0; i<k; i++) a[i+inceput]=c[i];
}

```



Ordonarea se realizează recursiv astfel: dacă șirul conține doar un element, înseamnă că el este deja ordonat; în caz contrar, el se împarte în două subșiruri, care se ordonează la fel (recursiv) și apoi se interclasează.

```

void SortarePrinInterclasare(vector a, int start, int stop)
// start si stop sint capetele intre care se face sortarea
{
    int mij=(start+stop)/2;
    if (start<stop)
    {
        SortarePrinInterclasare(a, start, mij);
        SortarePrinInterclasare(a, mij+1, stop);
        Interclasare(a, start, mij, stop);
    }
}

void main()
{
    int i, n; vector a;
    clrscr(); Citeste(a,&n); Afiseaza(a,n);
    SortarePrinInterclasare(a,0,n-1); Afiseaza(a,n);
    getch();
}

```

4. Să se scrie un program care caută existența unui element într-un șir $x = (x_1, x_2, \dots, x_n)$ ordonat crescător. Se va folosi metoda căutării binare.



Vom aplica, din nou, **metoda “divide et impera”** astfel:

- se compară elementul căutat cu elementul din mijlocul șirului;
- dacă ele coincid, înseamnă că elementul există în șir și procesul se întrerupe;
- dacă primul este mai mic decât al doilea, nu are sens o căutare dincolo de elementul din mijloc, ci doar în față, deci în stânga;
- dacă, însă, elementul căutat este mai mare decât cel din mijloc, căutarea se va face doar în partea din dreapta elementului din mijloc.

Acest algoritm este implementat în funcția recursivă `CautareBinara`.

```

#include <stdio.h>
#include <conio.h>
typedef int vector[20];
void Citeste(vector x, int *n)
{
    int p;
    printf("Dati nr. de elemente: "); scanf("%d",&p);
    *n=p;
    for (int i=0; i<p; i++)
    {
        printf("Dati x[%d]=",i+1);

```

```

        // consideram indexarea de la 1 la n !
        scanf("%d",&x[i]);
    }
}
void Afiseaza(vector x, int n)
{
    printf("Elementele lui x sunt: ");
    for (int i=0; i<n; i++)
        printf("%d",x[i]);
    printf("%c.\n\n",8);
}

```



În continuare avem aplicarea metodei “*divide et impera*” în varianta recursivă; sugerăm cititorului să scrie și o variantă iterativă a acestei funcții.

```

int CautareBinara(vector a, int elem, int inceput, int sfarsit)
{
    if (inceput<=sfarsit)
    {
        int mijloc=(inceput+sfarsit)/2;
        if (elem==a[mijloc])
            return mijloc+1; // atentie la indici !
        else
            if (elem<a[mijloc])
                return
                    CautareBinara(a, elem, inceput, mijloc-1);
            else
                return
                    CautareBinara(a, elem, mijloc+1, sfarsit);
    }
    else return 0;
}

void main()
{
    int n,p,e; vector a;
    clrscr(); Citeste(a,&n); Afiseaza(a,n);
    printf("Dati numarul cautat: "); scanf("%d",&e);
    printf("\n");
    if (p=CautareBinara(a,e,0,n-1))
        printf("%d exista pe pozitia %d in sir !",e,p);
    else
        printf("%d nu exista in sir !",e);
    getch();
}

```

Funcția `CautareBinara` returnează poziția `p` pe care se află elementul căutat în cadrul vectorului dat, respectiv 0, în caz că nu există. De remarcat că `p` este un indice care este cu o unitate mai mare decât indicele real, ținând cont că în C lucrăm cu vectori indexați începând cu 0 și nu cu 1.

5. Să se scrie un program care caută existența unui element într-un șir $x = (x_1, x_2, \dots, x_n)$ ordonat crescător. Se va folosi metoda căutării prin interpolare.

Căutarea prin interpolare este o ameliorare a metodei căutării binare, care se bazează pe strategia adoptată de o persoană când caută un cuvânt într-un dicționar. Astfel, dacă cuvântul căutat începe cu litera C, deschidem dicționarul undeva mai la început, iar când cuvântul începe cu litera V, deschidem dicționarul mai pe la sfârșit. Dacă e este valoarea căutată în vectorul $a[ic..sf]$, atunci partiționăm spațiul de căutare pe poziția $m=ic+(e-a[ic])*(sf-ic)/(a[sf]-a[ic])$. Această partiționare permite o estimare mai bună în cazul în care elementele lui a sunt numere distribuite uniform.

```

/* Cautare prin interpolare */
#include <stdio.h>
#include <conio.h>
typedef int vector[20];
void Citeste(vector x, int *n)
{
    int p;
    printf("Dati nr. de elemente: "); scanf("%d",&p);
    *n=p;
    for (int i=0; i<p; i++)
    {
        printf("Dati x[%d]=",i+1);
        // consideram indexarea de la 1 la n !
        scanf("%d",&x[i]);
    }
}
void Afiseaza(vector x, int n)
{
    printf("Elementele lui x sunt: ");
    for (int i=0; i<n; i++)
        printf("%d,",x[i]);
    printf("%c.\n\n",8);
}
int CautInterp(vector a, int e, int ic, int sf)
{
    if (ic<=sf)
    {
        int m; m=ic+(e-a[ic])*(sf-ic)/(a[sf]-a[ic]);
        if (e==a[m])
            return m+1; // atentie la indici !
        else
            if (e<a[m])
                return CautInterp(a, e, ic, m-1);
            else
                return CautInterp(a, e, m+1, sf);
    }
    else return 0;
}
void main()
{
    int n,p,e; vector a;
    clrscr(); Citeste(a,&n); Afiseaza(a,n);
    printf("Dati numarul cautat: "); scanf("%d",&e);
    printf("\n");
}

```

```

if (p=CautInterp(a,e,0,n-1))
    printf("%d exista pe pozitia %d in sir !",e,p);
else printf("%d nu exista in sir !",e);
getch();
}

```

6. "Problema celor 8 regine". Să se aseze n dame pe o tablă de șah cu $n \times n$ căsuțe, astfel încât damele să nu se atace între ele. O damă atacă toate câmpurile de pe aceeași linie, coloană și de pe diagonalele pe care se află.

Vom rezolva problema damelor prin metoda **"back-tracking recursivă"**. Astfel, să facem o analiză preliminară a problemei. În primul rând, observăm că nu putem așeza doua dame pe o aceeași coloană, deoarece ele s-ar ataca reciproc pe verticală.



Prin urmare, vom pune fiecare damă pe o altă coloană. Așadar, vom așeza damele în ordinea coloanelor, dama numărul k fiind pusă pe coloana a k -a. Firește, ea trebuie așezată astfel încât să nu atace damele deja așezate. Procedând astfel, ne asigurăm că nu le va ataca nici pe cele ce vor fi așezate după ea. Așezarea damei a k -a se face la intersecția dintre coloana a k -a și o linie, să zicem $x[k]$, cuprinsă între 1 și n .

În continuare, să observăm că $x[k]$ trebuie să fie diferită de $x[i]$, pentru orice $1 \leq i < k$, ceea ce înseamnă evitarea atacului pe orizontală. Pentru a evita atacul pe diagonală între dama k și o damă i dinaintea sa, să observăm că aceasta înseamnă că diferența coloanelor să fie diferită de cea a liniilor celor două dame, deci $\text{abs}(x[k]-x[i]) \neq k-i$.

Toate aceste elemente se regăsesc în funcția `PotContinua` care returnează 1 în momentul în care așezarea damei k corespunde restricțiilor, deci se poate trece la așezarea damei $k+1$. Apelul recursiv din cadrul funcției `Dama` realizează tocmai acest lucru.

În cazul în care nu se poate trece la dama $k+1$, iar toate valorile pentru $x[k]$ au fost încercate ($1 \leq x[k] \leq n$), funcția `Dama` eșuează și se revine la dama anterioară. (Într-o variantă recursivă s-ar scrie $k--$.)

În funcția `main` se pleacă de la dama 1.

```

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
const max=11;
typedef int vector[max]; // vom folosi indicii incepand cu 1
int NrSol; // variabila globala reprezentand numarul de solutii
void Scrie(int n, vector x)
{
    int i; NrSol++;
    printf("Solutia nr: %d:\n",NrSol);
    for (i=1; i<= n; i++)
        printf("Dama de pe coloana %d e pe linia %d.\n",i,x[i]);
    printf("\n"); getch();
}
int PotContinua(vector x, int k)

```

```

{
    for (int atac=0, i=1; i<k && !atac;
        atac=(x[i]==x[k] || abs(x[i]-x[k])==k-i), i++);
    return !atac;
}
void Dama(vector x, int k, int n)
{
    int l=1;
    // l reprezinta linia pe care se incearca asezarea damei k
    while (l<=n)
    {
        x[k]=l;
        if (PotContinua(x,k))
            if (k==n) Scrie(n,x); else Dama(x,k+1,n);
            l++;
    }
}
void main()
{
    vector AsezareDame; int NrDame;
    clrscr(); printf("Problema Damelor\n\n");
    NrSol=0;
    printf("Dati numarul de dame: ");
    scanf("%d",&NrDame);
    Dama(AsezareDame,1,NrDame);
}

```

7. "Problema labirintului". Se dă un labirint memorat sub forma unei matrice de caractere, unde zidurile sunt reprezentate de caracterul '#', iar prin spații locurile de trecere. Un șoricel, aflat în poziția (i_initial, j_initial) trebuie să ajungă la o bucătică de cașcaval, din poziția (j_initial, j_final). El poate să se miște ortogonal, în căsuțele libere alăturate. Să se determine toate drumurile pe care le poate face șoricelul sau să se afișeze *'imposibil'*, când nu există nici un drum.



Aceasta este o problemă clasică rezolvabilă prin metoda "back-tracking" recursivă, în plan. Lungimea drumului șoricelului poate varia, de la o soluție la alta. Drumul șoricelului este memorat în matricea Traseu, cu semnificația Traseu[i][j]=pas dacă șoricelul trece la pasul pas prin căsuța de pe linia i și coloana j, respectiv 0, dacă nu trece pe acolo.

```

#include <stdio.h>
#include <conio.h>
const m=8; const n=10;
typedef int sir[4];
typedef char matrice[m][n];

```

În continuare, tablourile Labirint, oriz și vert sunt declarate și inițializate ca variabile globale.

```

matrice Labirint=
{
    "##### ",
    "### #   ",
    "###    ",

```

```

"      # #####",
"### # #####",
"#      ##",
"  ## #####",
"##### "};

```

Următoarele tablouri indică direcțiile de deplasare: stânga, jos, sus și dreapta:

```

sir  horiz={-1,0,1, 0};
sir  vert={ 0,1,0,-1};

```

Matricea care va memora traseul șoricelului este:

```

matrice Traseu;
int i,j,i_initial,j_initial,i_final,j_final;
void Scrie()
{
    int i,j;
    for (i=0; i<m; i++)
    {
        for (j=0; j<n; j++)
            if (Traseu[i][j]==0)
                if (Labirint[i][j]==' ')
                    printf("%2c",' '); // 2 spatii
                else printf("[ ]"); // "zid"
            else
                printf("%2d",Traseu[i][j]);
        printf("\n");
    }
    printf("\n");
}

```

Procedura recursivă de determinare a drumului către cașcaval, plecând din punctul de coordonate (i,j), unde ne aflăm la pasul pas:

```

void Drum(int i, int j, int pas) /*
{
    int i_nou, j_nou, varianta;
    Se încearcă oricare din cele 4 direcții (adică variante):
    for (varianta=0; varianta<=3; varianta++)
    {
        i_nou=i+horiz[varianta];
        j_nou=j+vert[varianta];
        if (0<=i_nou && i_nou<m && 0<=j_nou && j_nou<n)
            if (Labirint[i_nou][j_nou]==' '
                && Traseu[i_nou][j_nou]==0)
            {
                Traseu[i_nou][j_nou]=pas;
                if (i_nou==i_final && j_nou==j_final) Scrie();
                else Drum(i_nou,j_nou,pas+1);
                Traseu[i_nou][j_nou]=0;
            }
    }
}
void main()
{
    clrscr();
    // afisam labirintul
    int i,j;

```

```

    for (i=0; i<m; i++)
    {
        for (j=0; j<n; j++)
            if (Labirint[i][j]==' ')
                printf("%2c",' '); // 2 spatii
        else
            printf("[ ]"); // "zid"
        printf("\n");
    }
    printf("\n");
// initializam matricea traseului
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
            Traseu[i][j]=0;
// citim coordonatele soricelului si a cascavalului
    printf("Dati pozitia initiala: ");
    scanf("%d%d",&i_initial,&j_initial);
    i_initial--; j_initial--; // indicii sunt de la 0 !
    printf("Dati pozitia finala: ");
    scanf("%d%d",&i_final,&j_final);
    i_final--; j_final--;
// pornim la drum !
    Traseu[i_initial][j_initial]=1; printf("Solutii:\n");
    Drum(i_initial,j_initial,2); getch();
}

```

Probleme propuse

1. Să se calculeze suma $S(n)=1+3+5+\dots+(2n-1)$, folosind o funcție recursivă.
2. Să se calculeze produsele $P_1(n)=1\times4\times7\times\dots\times(3n-2)$ și $P_2(n)=2\times4\times6\times\dots\times(2n)$, folosind funcții recursive.
3. Să se scrie o funcție recursivă care să returneze inversul unui șir de caractere.
4. Scrieți o funcție recursivă `itoa` pentru a converti un întreg într-un șir de caractere.
5. Folosind recursivitatea, să se verifice dacă un vector conține cel puțin un număr negativ în primele n poziții.
6. Să se afișeze conținutul unui vector, folosind o funcție recursivă.
7. Să se inverseze un vector. Se va folosi recursivitatea.
8. Evaluator de expresii algebrice. Vă propune să realizați un program care să citească expresia oricărei funcții și să o poată tabela, adică, dându-i-se un număr de puncte dintr-un interval, să afișeze valorile funcției în fiecare din acele puncte. Expresia funcției va fi memorată într-un șir de caractere. Ea va putea conține operații și funcții elementare (adunare, scădere, înmulțire, împărțire, ridicare la putere, `exp`, `ln`, `abs`, `radical`, `sin` și `cos`). Pentru a putea utiliza inclusiv funcții "cu acoladă", vom folosi o convenție din limbajul BASIC.
9. Astfel, să considerăm următoarele două funcții:

$$f(x) = \begin{cases} x \cdot \cos(1/x), & \text{dac\ } x \neq 0 \\ 0, & \text{dac\ } x = 0. \end{cases}$$

$$\text{și}$$

$$f(x) = \begin{cases} x^2, & \text{dac\ } x \geq 0 \\ x - \sin(x \cdot 2), & \text{altfel} \end{cases}$$

10. Prima funcție va fi dată sub forma: $(x \neq 0) * (x * \cos(1/x)) + (x = 0) * 0$; iar a doua sub forma: $(x > 0) * (x^2) + (x < 0) * (x - \sin(x * 2))$. Deci se vor folosi simbolurile "<" (mai mic sau egal), ">" (mai mare sau egal), "=" (egal) și "#" (diferit), precum și convenția din limbajul BASIC, anume că expresiile relaționale se evaluează la 1, dacă sunt adevărate, respectiv la 0, dacă sunt false.
11. Scrieți o funcție care să verifice dacă un șir de caractere este sau nu "palindrom", adică coincide cu răsturnatul (oglinzitul) său.
12. Rescrieți funcțiile de lucru cu șiruri de caractere, care se găsesc în <STRING.H>.
13. Să se definească un tip de date pentru numere naturale foarte mari și să se scrie proceduri care să adune și să scadă astfel de numere.
14. Elaborati o funcție care să determine diferența între două momente de timp date prin oră, minute și secunde.
15. Scrieți un program care să tabeleze o funcție, definită în textul programului, adică să afișeze valorile funcției în n puncte dintr-un interval dat [a,b].
16. Îmbunătățiți programul AdmitereLiceu (din secțiunea 3 a acestei lecții) pentru a avea posibilitatea de a șterge, insera un elev sau modifica datele despre un elev. De asemenea, programul să permită listarea tuturor elevilor care au mediile cuprinse între două valori date.
17. Ce afișează programul de mai jos:

```
#include <stdio.h>
int x;
int F(int x) { x=x+1; return x*x; }
void main()
{ x=2; printf("x=%d F(x)=%d x=%d",x,F(x),x); }
```

18. Rezolvă problema mutării a n discuri de pe turnul p pe turnul q, în problema Turnurilor din Hanoi, următoarea procedură? Este ea o procedură definită consistent din punct de vedere al recursivității?

```
void Hanoi2(int n,p,q)
{ int r=6-p-q;
  if (n==1) printf("%d-%d",p,q);
  else { printf("%d-%d",p,r);
        Hanoi2(n-1,p,r); printf("%d-%d",r,q); }
}
```

Semnificația funcției este: pentru a muta n discuri de la turnul p la turnul q se mută discul de deasupra de pe turnul p pe turnul r, apoi se mută, recursiv, cele n-1 discuri de pe turnul p pe turnul auxiliar r, după care se aduce discul de pe q pe r.

19. Scrieți o funcție iterativă care să calculeze, cu ajutorul unei stive proprii, suma $S=1+2+3+\dots+n$, respectiv produsul $P=1\times3\times5\times\dots\times(2n-1)$.
20. Scrieți funcții recursive pentru a determina: a) cel mai mare divizor comun a două numere întregi a și b ; b) coeficientul binomial C_n^k , din dezvoltarea binomului lui Newton; c) produsul scalar a doi vectori cu câte n componente reale.
21. Scrieți o funcție recursivă pentru a așeza n piese pe o tablă de șah $n\times n$, câte una pe fiecare coloană, astfel încât piesele să nu se atace. Fiecare piesă atacă precum: a) un nebun; b) o tură; c) un cai; d) un cal și un nebun simultan; e) un nebun și o tură simultan; f) un cal și o tură simultan.
22. Scrieți o funcție recursivă care să genereze permutările unei mulțimi cu n elemente.
23. Să se genereze recursiv submulțimile cu k elemente ale unei mulțimi cu n elemente.
24. Să se genereze un șir de lungime n format numai din literele a , b și c , astfel încât să nu existe două subșiruri alăturate identice.
25. O matrice dreptunghiulară cu m linii și n coloane reprezintă o imagine fotografică (alb/negru) a unui obiect. Elementele matricei sunt valorile binare 0 și 1. Valoarea 0 corespunde fondului, iar valoarea 1 prezenței obiectului. Să se determine numărul de componente ale obiectului, considerând că două puncte pot fi vecine atât ortogonal, cât și diagonal.
26. Să se parcurgă, cu un cal, o tablă de șah cu n linii și n coloane, în așa fel încât prin fiecare căsuță să se treacă exact o singură dată.
27. Să se așeze $2n-2$ nebuni pe o tablă de șah cu n^2 pătrate astfel încât nici o pereche de nebuni să nu se amenințe.
28. Să se așeze pe o tablă de șah cu n^2 pătrate cât mai multe dame care să nu se atace între ele.
29. Scrieți o funcție recursivă și una iterativă pentru a căuta un cuvânt într-un vector de cuvinte, care sunt puse în ordine alfabetică.
30. Se dă o bucată dreptunghiulară de tablă cu lungimea L și lățimea H , având N găuri, de coordonate numere întregi. Să se decupeze din ea o bucată de arie maximă care nu prezintă găuri. Sunt permise doar tăieturi verticale și orizontale. Se va utiliza metoda "*divide et impera*".
31. Un țăran primește o bucată dreptunghiulară de pământ pe care dorește să planteze o livadă. Pentru aceasta, el va împărți bucata de pământ în $m\times n$ pătrate, având dimensiunile egale, iar în fiecare pătrat va planta un singur pom din cele patru soiuri pe care le are la dispoziție. Să se afișeze toate variantele de a alcătui livada respectând următoarele condiții: a) nu trebuie să existe doi pomi de același soi în două căsuțe învecinate ortogonal sau diagonal; b) fiecare pom va fi înconjurat de cel puțin un pom din toate celelalte trei soiuri. Observație: țăranul are la dispoziție suficienți pomi de fiecare soi.
32. Un teren muntos are forma unei matrice cu $m\times n$ zone, fiecare zonă având o înălțime. Un alpinist pleacă dintr-o anumită zonă și trebuie să ajungă într-o zonă maximă în altitudine. Dintr-o zonă, alpinistul se poate

deplasa diagonal sau ortogonal, într-una din zonele (căsuțele) alăturate, doar urcând sau mergând la același nivel. Poate el ajunge într-unul din vârfuri? Dacă da, arătați toate soluțiile problemei.

33. Attila și regele. Un cal și un rege se află pe o tablă de șah. Unele câmpuri sunt “arse”, pozițiile lor fiind cunoscute. Calul nu poate călca pe câmpuri “arse”, iar orice mișcare a calului “arde” câmpul pe care se duce. Să se afle dacă există o succesiune de mutări permise (cu restricțiile de mai sus) prin care calul să ajungă la rege și să revină la poziția inițială. Poziția inițială a calului, precum și poziția regelui sunt considerate nearse.
34. Care din următoarele funcții C evaluează n factorial?
 - `int fact(int n) { if (n<=1) return 1; else return (n*fact(n-1)); }`
 - `int fact(int n) { if (n==1) return 1; else { n--; return (n+1)*fact(n); }`
35. Pete de culoare. Se consideră o matrice de dimensiunea 30×40 ale cărei elemente sunt numere naturale cuprinse între 1 și 9, reprezentând diferite culori. Se definește mulțimea conexă a unui element ca fiind mulțimea elementelor matricei în care se poate ajunge din elementul respectiv prin deplasări succesive pe linie sau coloană, cu păstrarea culorii. Să se determine culoarea și numărul de elemente al mulțimii conexe cu numărul maxim de elemente. Dacă există mai multe soluții, se va preciza una dintre ele.
36. Scrieți în limbajul C *funcția recursivă a lui Ackermann*, în care $A(m, n) = n + 1$, dacă $m = 0$, $A(m, n) = A(m - 1, 1)$, dacă $n = 0$ și $A(m, n) = A(m - 1, A(m, n - 1))$, în alte cazuri.

Capitolul 4. Structuri și tipuri definite de programator

- structuri • accesarea câmpurilor • variabile pointeri • variabile referință •
 - pointeri la structuri • reuniuni • liste • arbori •
- sortare rapidă • programarea mouse-ului în mod text •

Probleme rezolvate

1. Să se definească un tip de date pentru reprezentarea numerelor complexe și să se scrie funcții pentru diferite operații cu astfel de numere.



De remarcat că în `<complex.h>` sunt declarate toate elementele necesare lucrului cu numerele complexe, dar noi vom scrie propriile funcții. Astfel, de pildă, vom folosi funcția `sqrt` din `<math.h>` pentru a determina modulul unui număr complex.

```
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
typedef struct {
    double re, im;
} complex;

void scrie(complex z, char nume)
{
    printf("Numarul complex %c este: (%7.31f,%7.31f)\n",
        nume,z.re,z.im);
}

void citeste(complex *pz, char nume)
{
    double a, b;
    printf("Dati %c.re: ",nume); scanf("%lf",&a);
    printf("Dati %c.im: ",nume); scanf("%lf",&b);
    /* doua modalitati de a referi campurile variabilei dinamice
    indicate de pz */
    (*pz).re=a; pz->im=b;
}

void citestel(complex & z, char nume)
{
    double a,b;
    printf("Dati %c.re: ",nume); scanf("%lf",&a);
    printf("Dati %c.im: ",nume); scanf("%lf",&b);
    z.re=a; z.im=b;
}

double modul(complex z)
{
    return sqrt(z.re*z.re+z.im*z.im);
}

complex suma(complex z1, complex z2)
{
```

```

    complex z;
    z.re=z1.re+z2.re; z.im=z1.im+z2.im;
    return z;
}
complex produs(complex z1, complex z2)
{
    complex z;
    z.re=z1.re*z2.re-z1.im*z2.im;
    z.im=z1.re*z2.im+z1.im*z2.re;
    return z;
}
complex *suma1(complex z1, complex z2)
{
    complex z, *pz;
    pz=(complex *)malloc(sizeof(complex));
    z.re=z1.re+z2.re; z.im=z1.im+z2.im;
    *pz=z;
    return pz;
}
complex *suma2(complex z1, complex z2)
{
    complex *pz;
    pz=(complex *)malloc(sizeof(complex));
    pz->re = z1.re + z2.re;
    pz->im = z1.im + z2.im;
    return pz;
}
complex *suma3(complex *pz1, complex *pz2)
{
    complex *pz;
    pz=(complex *)malloc(sizeof(complex));
    pz->re = pz1->re+pz2->re; pz->im = pz1->im + pz2->im;
    return pz;
}
void main()
{
    clrscr(); complex q; citeste1(q,'q');
    printf("Modulului lui q: |q|=%lf.\n",modul(q));
    complex a,b,c,*pa,*pb,*pc;
    citeste(&a,'a'); scrie(a,'a');
    citeste(&b,'b'); scrie(b,'b'); printf("\n");
    c=suma(a,b); scrie(c,'S');
    c=produs(a,b); scrie(c,'P');
    pc=suma1(a,b); scrie(*pc,'S');
    free(pc);
    pc=suma2(a,b); scrie(*pc,'S'); free(pc);
    pa=(complex *)malloc(sizeof(a));
    *pa=a; pb=&b;
    pc=suma3(pa,pb); scrie(*pc,'S');
    free(pc); free(pa);
    getch();
}

```

Exemplul de față, deși simplu, în esența sa, ne permite prezentarea unor elemente ale limbajului foarte importante, referitoare la definirea tipurilor, structuri, pointeri, variabile dinamice, variabile referință.

Mai întâi, să observăm că am definit un nou tip de date, numit `complex`, ca fiind o înregistrare (vezi `record` din *Pascal*!) cu două câmpuri. Modul de referire a câmpurilor se vede în funcțiile `scrie`, `modul`, `sumă` și `produs`.



În limbajul C putem avea atât variabile definite în zona de memorie a programului, cât și variabile dinamice, generate prin program și alocate în timpul execuției programului. Acestea, însă, trebuie indicate (pointate) de către niște **variabile pointeri**, definite în program.

Un pointer `pv` către o variabilă `v` de tipul `tip` se definește prin `tip *pv`. Aceasta simbolizează faptul că `pv` va conține adresa de memorie a unei variabile anonime, generate dinamic. Zona în care se pastrează variabilele dinamice se numește *heap* ("grămadă"). Pentru a obține o variabilă acolo, trebuie ca acestea să i se aloce memorie corespunzătoare măririi în octeți cerute de tipul sau. De acest lucru se ocupă funcția `malloc`, care se găsește atât în `<stdlib.h>`, cât și în `<alloc.h>`, unde se găsesc și alte funcții asemănătoare. După ce a fost folosită, variabila anonimă dinamică poate fi abandonată, folosind funcția `free`.



Fiecărei variabile dinamice `i` se pune în corespondență cel puțin un pointer, care să o refere. Dacă `p` este un pointer către un tip de date `tip`, atunci crearea unei variabile dinamice de tip `t`, care să fie indicată de `p` se poate face prin:

```
tip *p;
p = (tip *)malloc(sizeof(tip));
```

În acest moment, putem să inițializăm variabila dinamică prin:

```
*p = valoare;
```

Observăm cum am apelat funcția `malloc`:

- se face o conversie către un pointer la `tip`, căci `malloc` returnează un pointer la `void` (un pointer general);
- argumentul lui `malloc` este numărul de octeți necesari variabilei dinamice; acesta poate fi dat de funcția `sizeof`, care poate fi aplicată atât numelui tipului, cât și unei variabile oarecare de acel tip:

```
tip v, *p;
p = (tip *)malloc(sizeof(v));
```

De pildă, pentru o variabilă de tip `complex`, mărimea este de 16 octeți, deoarece fiecare din cei doi `double` ai structurii este de 8 octeți.

Dupa ce variabila dinamică indicată de `p` a fost folosită, zona de memorie din heap ocupată de ea poate fi eliberată prin: `free(p)`;

Să considerăm, acum, că `p` este un pointer către o structură ca în cazul programului anterior (vezi funcțiile `citeste` și `suma2`). Pentru a ne referi la câmpul `camp` al variabilei dinamice indicate de `p` va trebui să scriem fie `(*p).camp1`, fie `p->camp`.



O extensie binevenită în urma trecerii de la C la C++ a fost introducerea variabilelor referință, definite cu ajutorul lui "&".

Asfel, dacă avem declarațiile: `tip v, *pv`, putem scrie `pv=&v`, prin aceasta înțelegând că `pv` va "pointa" (indica) către zona de memorie a variabilei `v`, adică `pv` va primi valoarea adresei de memorie a variabilei `v` (care, firește, are deja alocată memorie, în partea statică a programului). (Acest lucru se întâlnește în funcția `main`, în cazul lui `pb` și `b`.)

Un apel `free(pv)` este lipsit de sens, deoarece variabila `pv` nu indică în heap.



Noutatea adusă de C++ constă și în aceea că argumentele funcțiilor pot fi **variabile referință**, ceea ce înseamnă un plus de flexibilitate în privința apelurilor. A se vedea în acest sens funcția `citestel`.



În continuare să luăm pe rând altele din funcțiile programului anterior și să le analizăm:

a. Funcția `suma1` primește ca argumente două numere complexe și returnează un pointer către un număr complex. Rezultatul se pastrează, pe parcursul funcției în două locuri: atât în variabila locală `z`, cât și în variabila dinamică `*pz`, indicată de pointerul `pz`, căreia i se alocă memorie chiar în corpul funcției.

O variantă greșită a lui `suma1` este cea în care ar lipsi alocarea de memorie și am face alocare în cadrul lui `main()` pentru `pc`. O altă variantă greșită a lui `suma1` este:

```
complex *suma1(complex z1, complex z2)
{
    complex z, *pz;
    pz=(complex *)malloc(sizeof(complex));
    z.re=z1.re+z2.re; z.im=z1.im+z2.im;
    *pz=z; return pz;
}
```

Eroarea constă în faptul că atribuirea `*pz=z` se face înainte ca `z` să primească vreo valoare.

b. Funcția `suma2` lucrează întocmai ca și funcția `suma1`, dar se renunță la variabila `z`, lucrul cu pointerul `pz` fiind mai eficient din punct de vedere a memoriei ocupate (a se compara, de pildă cu funcția `suma`).

c. Când privește funcția `suma3`, aceasta, spre deosebire de toate celelalte, are drept argumente pointeri la complecși, de fapt, variabile dinamice.

d. Funcția principală apelează toate funcțiile definite, după o anumită logică, astfel:

```
void main()
{
```

```
    clrscr(); complex q;
```

Se citește valoarea numărului complex `q`, apelul fiind prin referință!

```
    citestel(q, 'q');
```

Se afișează modulul lui q ca double.

```
printf("Modulului lui q: |q|=%lf.\n",modul(q));
```

Se definesc numerele complexe a , b , c și trei pointeri la complex; pa , pb , pc .

```
complex a,b,c,*pa,*pb,*pc;
```

Se citesc valorile lui a și b și se afișează. Apelul este tot prin referință, însă se trimit ca argumente adresele variabilelor!

```
citeste(&a,'a'); scrie(a,'a');
```

```
citeste(&b,'b'); scrie(b,'b'); printf("\n");
```

Se calculează și se afișează suma și produsul $a+b$, $a*b$, folosind funcțiile

simple, fără pointeri:

```
c=suma(a,b); scrie(c,'S'); c=produs(a,b); scrie(c,'P');
```

Se apelează funcția `suma1` pentru a calcula, prin pointeri, $a+b$ și se afișează valoarea variabilei dinamice `*pc` ce păstrează rezultatul: `pc=suma1(a,b); scrie(*pc,'S');` Apoi această variabilă dispăre (memoria s-a alocat în cadrul lui `suma1`):

```
free(pc);
```

Se procedează la fel folosind funcția `suma2` `pc=suma2(a,b); scrie(*pc,'S');` `free(pc);` Se alocă memorie pentru `pa`: `pa=(complex *)malloc(sizeof(a));`

Se copiază valoarea complexului a în heap, în zona referită de `pa`: `*pa=a;` În pointerul `pb` se depune valoarea adresei variabilei b :

```
pb=&b;
```

Se calculează, folosind `suma3`, suma acelorași numere complexe și se afișează. Se va observa că în toate cele patru cazuri valoarea sumei este aceeași.

```
pc=suma3(pa,pb); scrie(*pc,'S');
```

Se eliberează memoria alocată variabilelor dinamice indicate de `pc` și

 `pa: free(pc); free(pa); getch(); }` Lucrul cu pointeri oferă foarte multe posibilități de lucru programatorilor experimentați, mai ales când se dorește crearea de structuri de date înlănțuite (definite recursiv), precum listele și arborii.

Există și avantaje legate de memorie. Astfel, dacă o structura de tip complex ocupă 16 octeți, un pointer către o astfel de structură ocupa doar 4 octeți, restul de 16 fiind în *heap*. De fapt, orice pointer ocupă doar 4 octeți, fiindcă reprezintă o adresă de memorie.



Programatorul trebuie să fie foarte atent la utilizarea pointerilor, mai ales la alocarea de memorie pentru variabilele dinamice pe care le indică, deoarece aceștia creează multe probleme.

De pildă, dacă în funcția `main` nu se alocă memorie pentru `pc` puteau apărea diverse probleme, pe când lipsa apelului lui `free(pc)` ar fi păstrat în memorie "gunoaie".

Astfel de cazuri și, de asemenea, altele, precum o asignare prost concepută poate genera mesajul “*null pointer assignment*” chiar și după terminarea programului!

2. Se dă o listă de persoane și punctajele obținute de acestea la un concurs. Să se ordoneze descrescător persoanele în funcție de punctaj și să se afișeze lista ordonată.



Pentru a memora datele a 10 persoane dintr-un șir și a două persoane oarecare p și q putem scrie ceva de genul:

```
typedef struct {
    char nume[20];
    int punctaj;
} persoana;
```

```
persoana x[10], p, q;
```

deci definind un nou tip de date, *persoana*, și declarând un tablou unidimensional cu 10 persoane (al căror nume nu depășește 30 de caractere.).

O altă posibilitate este definirea directă a unei variabile $x[10]$ cu același înțeles ca mai sus, precum și a variabilelor p și q este:

```
struct persoana {
    char nume[20];
    int punctaj;
} x[10], p;
```

Nu, nu l-am uitat pe q , ci doar am vrut să precizăm ca el putea fi declarat atât alături de $x[10]$ și p , dar poate fi declarat ulterior prin:

```
struct persoana q;
```

Cele doua feluri de a declara variabile de tip înregistrare sunt echivalente. Pentru a accesa elementele unei structuri, se folosește selectorul ‘.’ De exemplu, prin $p.punctaj=10$ se acordă nota 10 lui p , iar prin $strcpy(x[0].nume, 'Ionescu')$ se denumește primul concurent ‘*Ionescu*’.



Observație: nu e nevoie ca să declarăm neapărat niște structuri între “}” și “;”. De pildă, în exemplul de mai jos, vectorul de persoane este local funcției *main*, chiar dacă declarația de *struct* este globală.

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
#define max 20
struct persoana {
    char nume[max]; int punctaj;
};
int Citeste(struct persoana p[])
{
    int i,n;
    printf("Dati numarul de persoane: "); scanf("%d",&n);
    for (i=0; i<n; i++)
    {
        fflush(stdin); // !! se poate folosi si flushall() !!
        printf("Dati numele persoanei nr. %d: ",i+1);
```

```

        gets(p[i].nume); p[i].nume[max]='\0';
        printf("Dati punctajul lui %s: ",p[i].nume);
        scanf("%d",&p[i].punctaj); fflush(stdin);
    }
    return n;
}
void Schimba(struct persoana *a, struct persoana *b)
{
    struct persoana aux; aux = *a; *a = *b; *b = aux;
}
int Ordoneaza(struct persoana p[], int n)
{
    int i, j, ord=1;
    for (i=0; i<n-1 && ord; i++)
        if (p[i].punctaj<p[i+1].punctaj) ord = 0;
    if (!ord)
    {
        for (i=0; i<n-1; i++)
            for (j=i+1; j<n; j++)
                if (p[i].punctaj<p[j].punctaj) Schimba(&p[i],&p[j]);
    }
    return (!ord);
}
void Afiseaza(struct persoana *x, int n)
{
    int i;
    for (i=0; i<n; i++)
        printf("%d. %20s -> %4d.\n",i+1,x[i].nume,x[i].punctaj);
}
void main()
{
    struct persoana x[max]; // sirul persoanelor
    int n; // numarul de persoane
    clrscr();
    printf("Ordonare persoane dupa punctaj\n\n");
    if ((n = Citeste(x)) == 0) printf("\nNu exista persoane.");
    else
    {
        if (Ordoneaza(x,n)) printf("\nAm ordonat persoanele.\n");
        else printf("\nPersoanele erau deja ordonate.\n");
        Afiseaza(x,n);
    }
    getch();
}

```

Funcția Citeste returnează numărul de persoane citite. În mod similar, **funcția Ordoneaza** returnează 1 dacă șirul persoanelor era neordonat și a fost ordonat, respectiv 0 dacă el era ordonat de la bun început.



Observație!! De remarcat că dacă după ce s-a citit un număr urmează să se citească un șir de caractere e bine să se golească buffer-ul. Altfel compilatorul interpretează caracterul <Enter> apăsăat după introducerea numărului ca fiind șirul de caractere ce se dorește a fi citit.

Prin `fflush(f)` se golește fișierul (*stream-ul*) `f`. Deci, prin `fflush(stdin)` golim buffer-ul intrării standard, care în cazul de față este tastatură. Se poate folosi și `flushall()`, care golește toate bufferele asociate stream-urilor de intrare.

Observatie. În conformitate cu cele spuse în exemplul anterior, aproape de funcția `citeste1`, și aici putem să înlocuim funcția `Schimba` cu funcția de mai jos:

```
void Schimba1(struct persoana &a, struct persoana &b)
{
    struct persoana aux; aux = a; a = b; b = aux;
}
```

Firește, apelul `Schimba(&p[i], &p[j])` va trebui înlocuit, în acest caz, cu `Schimba1(p[i], p[j])`.



Propunem cititorului să rescrie procedura de ordonare aplicând algoritmul “quick-sort” de sortare rapidă.

3. Se dă o listă de persoane și punctajele obținute de acestea la un concurs. Să se ordoneze descrescător persoanele în funcție de punctaj și să se afișeze lista ordonată.



Vom proceda că în exemplul precedent, dar de data aceasta vom apela la funcția `qsort` (definită în `<stdlib.h>` și în `<search.h>` care implementează algoritmul de sortare “quick sort”.

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define max 20
struct persoana {
    char nume[max];
    int punctaj;
};
int Citeste(struct persoana p[])
{
    int i, n;
    printf("Dati numarul de persoane: "); scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        fflush(stdin); // !! se poate folosi si flushall() !!
        printf("Dati numele persoanei nr. %d: ", i+1);
        gets(p[i].nume); p[i].nume[max]='\0';
        printf("Dati punctajul lui %s: ", p[i].nume);
        scanf("%d", &p[i].punctaj);
        fflush(stdin);
    }
    return n;
}
void Schimba(struct persoana *a, struct persoana *b)
{
```



```

    struct persoana aux;
    aux = *a; *a = *b; *b = aux;
}
int CompararePersoane(const void *a, const void *b)
{
    struct persoana *pa, *pb;
    pa=(struct persoana *)a; pb=(struct persoana *)b;
    return (pa->punctaj < pb->punctaj);
}
void Ordoneaza(struct persoana p[], int n)
{
    qsort((void *) p, n, sizeof(struct persoana),
        CompararePersoane);
}
void Afiseaza(struct persoana *x, int n)
{
    int i;
    for (i=0; i<n; i++)
        printf("%d. %20s -> %4d.\n",i+1,x[i].nume,x[i].punctaj);
}
void main()
{
    struct persoana x[max]; // sirul persoanelor
    int n; // numarul de persoane
    clrscr();
    printf("Ordonare persoane dupa punctaj\n\n");
    if ((n = Citeste(x)) == 0) printf("\nNu exista persoane.");
    else
    {
        Ordoneaza(x,n);
        Afiseaza(x,n);
    }
    getch();
}

```



Exemplul ne pune în evidență, apropo de utilizarea funcției `qsort`, următoarele:

- posibilitatea de a defini parametri constanți (ca în antetul funcției `CompararePersoane`);
- posibilitatea de a face conversie între pointeri, ca în cazul: `pa=(struct persoana *)a;` din funcția `CompararePersoane` sau `qsort((void *) p, ...)` din funcția `Ordoneaza`;
- posibilitatea de a folosi numele unei funcții pe post de argument în altă funcție, ca în cazul apelului lui `qsort` din funcția `Ordoneaza`.

4. Se dă o listă de persoane și punctajele obținute de acestea la un concurs. Să se ordoneze descrescător persoanele în funcție de punctaj și să se afișeze lista ordonată.



Vom proceda ca în exemplul precedent, dar vom lucra cu un tablou de pointeri la structuri, în locul unui tablou de structuri. Aceasta prezintă două avantaje principale:

- se face economie de spațiu din zona programului;
- dacă am estimat gresit numărul maxim de persoane, alegându-l mult mai mare decât cel real, necesar, penalizarea este mai mică, proporțional cu spațiul ocupat de o adresă (4 octeți) și nu de valoarea unei structuri de tip persoană (care este max+2 octeți);
- există facilitatea de a defini structuri generice de date.

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

#define maxp 100
#define maxn 20
struct persoana {
    char nume[maxn];
    int punctaj;
};
typedef persoana *ppers;
typedef ppers multime_pers[maxp];
int Citeste(multime_pers p)
{
    int i,n; struct persoana op;
    char numele[maxn]; int punctajul;
    printf("Dati numarul de persoane: "); scanf("%d",&n);
    for (i=0; i<n; i++)
    {
        fflush(stdin); // !! se poate folosi si fflush() !!
        printf("Dati numele persoanei nr. %d: ",i+1);
        gets(numele); numele[maxn]='\0';
        printf("Dati punctajul lui %s: ",numele);
        scanf("%d",&punctajul);
        strcpy(op.nume,numele); op.punctaj=punctajul;
        p[i]=(ppers)malloc(sizeof(struct persoana));
        *p[i]=op; fflush(stdin);
    }
    return n;
}
void Schimba(ppers a, ppers b)
{
    struct persoana aux;
    aux = *a; *a = *b; *b = aux;
}
int Ordoneaza(multime_pers p, int n)
{
    int i, j, ord=1;
    for (i=0; i<n-1 && ord; i++)
        if (p[i]->punctaj<p[i+1]->punctaj) ord = 0;
    if (!ord)
    {
```

```

        for (i=0; i<n-1; i++)
            for (j=i+1; j<n; j++)
                if (p[i]->punctaj<p[j]->punctaj)
                    Schimba(p[i],p[j]);
    }
    return (!ord);
}
void Afiseaza(multime_pers p, int n)
{
    int i;
    for (i=0; i<n; i++)
    {
        printf("%d. %20s -> %4d.\n",i+1,p[i]->nume,p[i]->punctaj);
        free(p[i]);
    }
}
void main()
{
    multime_pers x; // sirul pointerilor catre persoane
    int n; // numarul de persoane
    clrscr();
    printf("Ordonare persoane dupa punctaj\n\n");
    if ((n = Citeste(x)) == 0) printf("\nNu exista persoane.");
    else
    {
        if (Ordoneaza(x,n)) printf("\nAm ordonat persoanele.\n");
        else printf("\nPersoanele erau deja ordonate.\n");
        Afiseaza(x,n);
    }
    getch();
}

```



Observație: Alocarea memoriei pentru șirul de variabile dinamice de tip înregistrare se face în procedura `Citeste`, iar eliberarea memoriei ocupate în funcția `Afiseaza`.

5. Scrieți un program care să deseneze dreptunghiuri în modul text, folosind mouse-ul. Se apasă butonul din stânga și se marchează un colț, apoi se mișcă mouse-ul și se dă drumul la buton în momentul în care s-a fixat cel de-al doilea colț (colțurile sunt diagonal opuse).



Firește, în primul rând, trebuie să dispunem de funcții pentru lucrul cu mouse-ul. Acestea sunt scrise în fișierul "mouset.h":

```

void MOUSE(int &a1, int &a2, int &a3, int &a4)
{
    struct REGPACK regs;
    regs.r_ax = a1; regs.r_bx = a2;
    regs.r_cx = a3; regs.r_dx = a4;
    intr(0x33, &regs);
    a1 = regs.r_ax; a2 = regs.r_bx; a3 = regs.r_cx; a4 = regs.r_dx;
}

```

```

void MouseInit(void)
{
    int a=1,b,c,d; MOUSE(a,b,c,d);
}
void MouseHide(void)
{
    int a=2,b,c,d; MOUSE(a,b,c,d);
}
void MouseShow(void)
{
    int a=1,b,d,c; MOUSE(a,b,c,d);
}
void MouseTData(int& but, int& x, int& y)
{
    int a=3, b, c, d; MOUSE(a,b,c,d);
    but=b; x=c/8+1; y=d/8+1;
}
void MouseTMove(int x, int y)
{
    int a=4, b, c=8*x-1, d=8*y-1; MOUSE(a,b,c,d);
}

```

MouseInit() va inițiază mouse-ul. MouseShow() îl afișează pe ecran MouseHide() îl ascunde, MouseTData(b,x,y) preia în variabilele întregi b, x și y datele despre mouse în mod text: b = butonul de mouse apăsăat

- b=0 înseamnă că nici un buton nu s-a apăsăat;
- b=1 butonul din stânga, b=2 butonul din dreapta, b=3 ambele butoane;
- x, y = coloana, respectiv linia unde se află mouse-ul;

MouseTMove(x,y) mută cursorul de mouse în căsuța de pe coloana x și linia y. De remarcat apelul lui MouseData(b,x,y) prin referință.

În continuare, să scriem programul principal:

```

#include "mouset.h"
#include <conio.h>

#include <dos.h> /* aceasta nu neaparat, caci deja e inclusa in
"mouset.h" */
// deseneaza un chenar de la (x1,y1) la (x2,y2) de caracterul c
void Dreptunghi(int x1, int y1, int x2, int y2, char c)
{
    int aux,x,y;
    if (x1>x2) { aux = x1; x1 = x2; x2 = aux; }
    if (y1>y2) { aux = y1; y1 = y2; y2 = aux; }
    for (x=x1; x<=x2; x++)
    {
        gotoxy(x,y1); putchar(c);
        gotoxy(x,y2); putchar(c);
    }
    for (y=y1; y<=y2; y++)
    {
        gotoxy(x1,y); putchar(c);
        gotoxy(x2,y); putchar(c);
    }
}

```



```

// deseneaza un dreptunghi umplut de la (x1,y1) la (x2,y2)
void DreptunghiPlin(int x1, int y1, int x2, int y2)
{
    int aux,x,y;
    if (x1>x2) { aux = x1; x1 = x2; x2 = aux; }
    if (y1>y2) { aux = y1; y1 = y2; y2 = aux; }
    for (x=x1; x<=x2; x++)
        for (y=y1; y<=y2; y++)
        {
            gotoxy(x,y);
            putchar(219);
        }
}

```



```

void main()
{
    int b,x,y,px,py;
    clrscr();
    // se initializeaza si afiseaza mouse-ul in centrul ecranului
    MouseInit();
    MouseShow();
    // se determina primul punct de coordonate (px,py)
    do {
        MouseTData(b,px,py);
    } while (b!=1);
    /* pentru a nu avea neaplaceri in afisarea dreptunghiurilor
    partiale, se ascunde mouse-ul */
    MouseHide();
    do {
        MouseTData(b,x,y);
        /* x,y sunt coordonatele provizorii
        ale celui de al doilea punct */
        Dreptunghi(px,py,x,y,178);
        MouseShow(); delay(50); MouseHide();
        Dreptunghi(px,py,x,y,32);
        /* intai se deseneaza dreptunghiul, apoi se afiseaza
        putin mouse-ul, apoi se sterge dreptunghiul */
    } while (b==1); // pana se apasa butonul de mouse
    DreptunghiPlin(px,py,x,y);
    // se deseneaza dreptunghiul plin final
    MouseShow; getch();
}

```



Observații:

1. Să considerăm funcția de bază a lui "mouse.h":

```

void MOUSE(int &a1, int &a2, int &a3, int &a4)
{
    struct REGPACK regs;
    regs.r_ax = a1; regs.r_bx = a2; regs.r_cx = a3; regs.r_dx =
a4;
    intr(0x33, &regs);
}

```

```

    a1 = regs.r_ax; a2 = regs.r_bx; a3 = regs.r_cx; a4 =
regs.r_dx;
}

```

Această funcție folosește structura predefinită în <dos.h> cu numele REGPACK. Practic, se pot accesa regiștrii interni ai calculatorului. Funcția de mai sus folosește intreruperea DOS 33 (hexazecimal) care gestionează mouse-ul.



Structura REGPACK conține mai multe câmpuri ce reprezintă valorile pasate sau returnate de un apel al funcției “intr” (care se află tot în <dos.h>) și care permite apelul de intreruperi DOS pentru procesorul 8086:

```

struct REGPACK {
    unsigned r_ax, r_bx, r_cx, r_dx;
    unsigned r_bp, r_si, r_di;
    unsigned r_ds, r_es, r_flags;
};

```

Înrudită cu “intr” sunt “int86” și “int86x”, acestea lucrând cu reuniuni REGS



2. Propunem utilizatorului să utilizeze mouse-ul în propriile aplicații ce lucrează în mod text.

6. Să se exemplifice folosirea reuniunilor (union) pentru citirea și afișarea datelor despre mai multe persoane de ambele sexe.

Vom folosi o structură pentru a memora numele, vârsta și sexul unei persoane, indiferent dacă aceasta este femeie sau bărbat. Dar, în funcție de sex, persoană respectivă va avea un soț (dacă e femeie), respectiv o soție (dacă e bărbat). O înregistrare de tip union va fi folosită, așadar, pentru a memora partenerul unei persoane oarecare.



O înregistrare union este similară cu o înregistrare de tip struct, cu excepția faptului că union va permite să definiți variabile (câmpuri) care să-și împartă același spațiu de memorare.

De exemplu, fie declarația

```

union int_or_long {
    int    i;
    long   l;
} n;

```

Borland C++ va alocă destul spațiu de memorare în variabila n pentru a memora cel mai mare element dintr-un union. Spre deosebire de struct, variabilele n.i și n.l ocupă aceeași locație de memorie. Astfel, scriind într-una din ele, și cealaltă va fi afectată.

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
union partener {
    char sotie[20];
    char sot[20];
};

```

```

struct persoana {
    char nume[20];
    char sex;
    int varsta;
    union partener p;
};

void Citeste(int nr_p, struct persoana *x)
{
    int vs; char sx;
    printf("Se citesc datele persoanei a %d-a:\n",nr_p);
    fflush(stdin);
    printf("Dati numele: ");
    gets(x->nume);
    printf("Dati varsta: ");
    scanf("%d",&vs);
    x->varsta=vs; fflush(stdin);
    printf("Dati sexul: "); scanf("%c",&sx);
    x->sex=sx; fflush(stdin);
    if (x->sex=='F' || x->sex=='f')
    {
        printf("Dati numele sotului: "); gets(x->p.sot);
    }
    else
    {
        printf("Dati numele sotiei: "); gets(x->p.sotie);
    }
}

void Scrie(int nr_p, struct persoana x)
{
    if (wherey()==20) { printf("\nApasati o tasta...\n"); getch(); }

    printf("Datele persoanei a %d-a:\n",nr_p);
    printf("Numele: %s.\n",x.nume);
    printf("Varsta: %d.\n",x.varsta);
    printf("Sexul: %c.\n",x.sex);
    if (x.sex=='F' || x.sex=='f')
        printf("Numele sotului: %s.\n",x.p.sot);
    else
        printf("Numele sotiei: %s.\n",x.p.sotie);
}

void main()
{
    clrscr();
    struct persoana om[20]; int i,n;
    printf("Dati numarul de persoane: "); scanf("%d",&n);
    for (i=0; i<n; i++) Citeste(i+1,&om[i]);
    for (i=0; i<n; i++) Scrie(i+1,om[i]);
    getch();
}

```



Observație. În funcția de afișare, apare linia:

```

if (wherey()==20) { printf("\nApasati o tasta...\n"); getch(); }

```

În acest fel se așteaptă apăsarea unei taste în momentul în care afișarea a adus cursorul pe linia 20. Funcția `wherex()` lucrează la fel ca și `wherey()`, dar pentru coloane. Ambele sunt definite în `<conio.h>`.

7. Să se scrie o structură de date eficientă pentru lucrul cu listele dublu înălțuite. Să se proiecteze o aplicație simplă care să prelucreze o listă dublu înălțuită.



Ne propunem să realizăm o structură de listă dublu înălțuită de numere întregi (de fapt date de tipul `tip`).



Lista este definită printr-o înregistrare care conține:

- un întreg reprezentând lungimea listei (cea utilizată efectiv);
- un pointer către elementul curent al listei;
- un pointer către santinela `inceput` a listei și unul către santinela `sfarsit` a listei: acestea sunt două celule exterioare listei, care facilitează lucrul cu lista.

O celulă a listei este o înregistrare cu următoarea structură:

- o informație de tipul `tip`;
- un pointer către celula anterioară din lista;
- un pointer către celula următoare din lista.

Funcțiile ce vor fi implementate sunt cele de inițializare a listei, adăugare, inserare și ștergere a unei liste, de deplasare în cadrul listei, de testare dacă lista este vidă și de afișare.

Unele funcții returnează 1 sau 0 și anume. Se returnează 1, când funcția face ceva anume, deci reușește, respectiv 0, în alte cazuri (nu are sens operația sau nu este posibilă alocarea de memorie etc.).



Acest fel de a scrie aplicații se numește **“stilul defensiv de programare”**.

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
typedef int tip; // tipul elementelor din lista
#define format_afis1 "%d "
#define format_afis2 "%d"
#define format_cit "%d"
// o celula a listei
struct celula {
    tip info;
    struct celula *prec;
    struct celula *urm;
};
typedef struct celula * pcelula; // pentru inlantuirii
// o structura de lista
typedef struct {
    int lung;
    pcelula inceput;
```



```

        pcelula curent;
        pcelula sfarsit;
} lista;
int Init(lista *pl)
// initializeaza lista, creand santinelele si facand legaturile
{
    pl->lung=0;
    pl->inceput = (pcelula) malloc(sizeof(celula));
    if (!pl->inceput) return 0;
    pl->sfarsit = (pcelula) malloc(sizeof(celula));
    if (!pl->sfarsit) return 0;
    pl->inceput->prec = NULL;
    pl->inceput->urm = pl->sfarsit;
    pl->sfarsit->prec = pl->inceput;
    pl->curent=pl->sfarsit;
    pl->inceput->info = -1;
    pl->sfarsit->info = -1;
    return 1;
}
int EsteVida(lista l)
// testeaza daca lista contine macar un element
{
    return (l.lung==0);
}
int Insereaza(lista *pl, tip el)
/* insereaza un element in fata celui curent; elementul inserat
devine curent */
{
    pcelula p, a, b;
    p=(pcelula) malloc(sizeof(celula));
    if (!p) return 0;
    p->info = el;
    a=pl->curent->prec; b=pl->curent;
    a->urm=p; p->prec=a; p->urm=b; b->prec=p;
    pl->lung++; pl->curent=p;
    return 1;
}
int Sterge(lista *pl)
/* sterge elementul curent din lista; elementul curent va fi
succesorul
celui sters */
{
    pcelula p, a, b;
    if (!pl->lung) return 0;
    else
    {
        a=pl->curent->prec; b=pl->curent->urm;
        p=pl->curent;
        free(p);
        a->urm=b; b->prec=a;
        pl->lung--;
        pl->curent=b;
        if ((b==pl->sfarsit) && (!EsteVida(*pl)))
            pl->curent=pl->curent->prec;
        return 1;
    }
}

```

```

        }
    }
void Distruge(lista *pl)
// elibereaza zonele de memorie ocupata de o lista
{
    while (!EsteVida(*pl)) Sterge(pl);
}
int Inainte(lista *pl)
/* inainteaza pe p->curent cu o pozitie, cel mult pina la
precedentul lui p->sfirsit, daca lista nu e vida, respectiv
p->sfirsit, daca lista e vida */
{
    if (EsteVida(*pl) || (pl->curent->urm==pl->sfarsit)) return 0;
    else
        { pl->curent=pl->curent->urm; return 1; }
}
int Inapoi(lista *pl)
/* aduce pl->curent inapoi cu o pozitie, cel mult pina la
succesorul lui pl->inceput */
{
    if (pl->curent!=pl->inceput->urm)
        { pl->curent=pl->curent->prec; return 1; }
    else return 0;
}
int Adauga(lista *pl, tip el)
// adauga un element dupa ultimul element al listei
{
    while (Inainte(pl));
    if (!EsteVida(*pl)) pl->curent=pl->curent->urm;
    return Insereaza(pl,el);
}
void Afiseaza(lista l)
// afiseaza continutul listei
{
    pcelula p;
    if (EsteVida(l))
        { printf("Lista vida...\n"); return; }
    p=l.inceput->urm;
    printf("Lista este:\n");
    while (p!=l.sfarsit)
    {
        if (l.curent!=p)
            printf(format_afis1,p->info);
        else
        {
            printf("["); printf(format_afis2,p->info);
            printf("] ");
        }
        p=p->urm;
    }
    printf("\n");
}

```

Funcția principală `main()` care exploatează structura de listă și operațiile definite mai sus este prezentată în continuare. Ieșirea se realizează cu tasta 'o'.

```
void main()
{
    lista o_lista;
    Init(&o_lista); Afiseaza(o_lista);
    tip un_element;
    char tasta;
    do
    {
        printf("[A]daugare, [S]tergere, [I]nserare,");
        printf(" Ina[p]oi, I[n]ainte, [O]prire\n");
        tasta=getche(); printf("\n");
        switch(tasta) {
            case 'a': {
                printf("Dati elementul: ");
                scanf(format_cit,&un_element);
                Adauga(&o_lista, un_element);
            } break;
            case 'i': {
                printf("Dati elementul: ");
                scanf(format_cit,&un_element);
                Insereaza(&o_lista, un_element);
            } break;
            case 's': Sterge(&o_lista); break;
            case 'p': Inapoi(&o_lista); break;
            case 'n': Inainte(&o_lista);
        }
        Afiseaza(o_lista);
    } while (tasta !='o');
    Distruge(&o_lista);
}
```



Observație. În afară de funcțiile precizate, există și funcția `DistrugeLista`. Ea este apelată în finalul programului, pentru a dealoca memoria ocupată de listă în timpul programului. Funcționarea ei este simplă: cât timp lista nu s-a golit, să se șteargă câte un element din ea; s-ar fi putut scrie și astfel: cât timp se mai pot șterge elemente din listă, să se șteargă. Adică:

```
void Distruge(lista *pl)
// elibereaza zonele de memorie ocupata de o lista
{
    while (Sterge(pl));
}
```



Atenție la apelurile funcțiilor! Observați că la cele care modifică conținutul listei în vreun fel sau altul, s-au folosit pointeri drept parametri, iar în rest parametrii erau elemente simple de tip listă.

8. Să se scrie o structură de date eficientă pentru lucrul cu arborii binari (conținând în noduri numere întregi). Să se proiecteze o aplicație simplă care să prelucreze arborii binari



Arborii binari care îi vom putea construi cu funcțiile din acest exemplu se bucură de o anumită proprietate. Fiecare nod conține un element care este mai mare decât elementul din oricare nod al subarborului stâng (daca există) și mai mic sau egal cu orice element din subarboarele drepte (dacă există). Pentru a crea astfel de arbori, e de ajuns să scriem o funcție de adăugare a unui nou element într-un arbore binar, funcție recursivă care să îndeplinească următoarele condiții:

- dacă arborele este `NULL`, atunci se creează un nod în care se pune acest element;
- dacă arborele nu este `NULL`, atunci:
 - dacă elementul din rădăcină este $>$ elementul nou sosit, acesta se adaugă în subarboarele din stângă;
 - dacă nu, el se adaugă în subarboarele din dreapta.

Adăugările sunt, firește, recursive.

După ce vom crea funcția de adăugare a unui element în arbore, vom scrie și trei funcții care să afișeze în *preordine*, *inordine* și *postordine* un astfel de arbore. Se observă imediat că afișarea în *inordine* a arborelui creat cu niște elemente oarecare va duce la o afișare ordonată crescător a acestor elemente.



Observație: Cele trei feluri de a parcurge un arbore sunt toate recursive. Afișarea în preordine a unui arbore înseamnă afișarea informației din rădăcină (*R*), apoi a subarborului stâng (*S*), apoi a celui drept (*D*). Afișarea în postordine înseamnă ordinea *S D R*, iar în inordine: *S R D*. În fine, exemplul conține și o funcție ce verifică dacă există sau nu un element într-un arbore. Aceste fiind spuse, să trecem la atac!

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
typedef int tip; // tipul elementelor din nodurile arborelui
#define format_afis "%d "
#define format_cit "%d"
// un nod al arborelui
struct nod {
    tip info;
    struct nod *st;
    struct nod *dr;
};
// un arbore binar e definit ca un pointer catre o inregistrare
de tip nod
typedef nod * arbore;
int EsteNul(arbore pa)
// testeaza daca arborele a este NULL
{
```

```

        return (pa==NULL);
    }
void Init(arbore *pa)
// initializeaza arborele, adica il face NULL
{
    if (!EsteNul(*pa)) *pa=NULL;
}
int Adauga(arbore *pa, tip el)
// adauga un element el in arborele a
{
    if (EsteNul(*pa))
    {
        *pa=(arbore)malloc(sizeof(nod));
        if (!*pa) return 0;
        (*pa)->info = el;
        (*pa)->st = NULL; (*pa)->dr = NULL;
        return 1;
    }
    else
        if (el < (*pa)->info)
            return Adauga(&(*pa)->st, el);
        else
            return Adauga(&(*pa)->dr, el);
}
int Exista(arbore a, tip el)
// verifica existenta elementului el in arborele a
{
    if (!EsteNul(a)) // se poate scrie simplu if (a) !!
    {
        if (a->info==el) return 1;
        else
            if (el<a->info) return Exista(a->st, el);
            else return Exista(a->dr, el);
    }
    else return 0;
}
void Preordine(arbore a)
{
    if (!EsteNul(a))
    {
        printf(format_afis,a->info);
        Preordine(a->st); Preordine(a->dr);
    }
}
void Postordine(arbore a)
{
    if (!EsteNul(a))
    {
        Postordine(a->st); Postordine(a->dr);
        printf(format_afis,a->info);
    }
}
void Inordine(arbore a)
{
    if (!EsteNul(a))

```

```

    {
        Inordine(a->st);
        printf(format_afis,a->info);
        Inordine(a->dr);
    }
}

```

Funcția principală `main()` care exploatează structură de lista și operațiile definite mai sus este prezentat în continuare. Ieșirea se realizează cu tasta 'o'.

```

void main()
{
    arbore un_arbore;
    Init(&un_arbore);
    tip un_element;
    char tasta;
    do
    {
        printf("\n[A]daugare, [P]reordine, [I]nordine, ");
        printf("Po[s]tordine, [O]prire, [C]autare\n");
        tasta=getche(); printf("\n");
        switch(tasta) {
            case 'a': {
                printf("Dati elementul: ");
                scanf(format_cit, &un_element);
                Adauga(&un_arbore, un_element);
            } break;
            case 'i': Inordine(un_arbore); break;
            case 's': Postordine(un_arbore); break;
            case 'p': Preordine(un_arbore);
            case 'c': {
                printf("Dati elementul: ");
                scanf(format_cit, &un_element);
                if (Exista(un_arbore, un_element))
                    printf("Elementul exista.\n");
                else
                    printf("Elementul nu exista.\n");
            }
        }
    } while (tasta != 'o');
}

```

Propunem cititorului care cunoaște elemente de grafică să realizeze o procedură recursivă ce să afișeze arborele creat prin apeluri succesive ale lui `Adauga`.

Probleme propuse

1. Scrieți declarații de tip sau structuri pentru a specifica: a) timpul, în ore, minute și secunde; b) data curentă sub forma: zi/luna/an; c) adresa unei persoane sub forma: oraș/strada/nr/scara/etaj/apartament/judet/cod; d) un număr complex: parte reală / parte imaginară sau modul / argument;

- e) un șir cu maxim 20 de puncte în spațiul euclidian, date prin coordonate carteziane.
2. Scrieți declarații de tip sau structuri pentru a specifica: a) un număr rațional sub forma numărător / numitor; b) un articol din fișierul unei biblioteci sub forma: cotă / autor/ titlu / an apariție / număr pagini / ISBN; c) factura telefonică sub form: antet / număr factură / număr de telefon / valoare abonament / chirie terminal / diverse / cost impuls / număr de impulsuri / total sumă de plată.
 3. Să se scrie un program pentru admiterea la un liceu care să permită:
 - inițializarea bazei de date (a vectorului de înregistrări de tip elev);
 - adăugarea unui elev în baza de date;
 - înlocuirea unui elev cu alt elev;
 - inserarea unui elev pe o anumită poziție în baza de date;
 - eliminarea unui elev de pe o anumită poziție din baza de date;
 - eliminarea din baza de date a unui elev având un anumit nume;
 - căutarea unui elev după nume;
 - calcularea mediilor;
 - listarea alfabetică a elevilor;
 - listarea elevilor în ordinea descrescătoare a mediilor;
 - listarea tuturor elevilor cu medii peste o valoare dată;
 - amestecarea elevilor din baza de date;
 - eliminarea ultimului elev din baza de date.
 4. Să se realizeze deplasarea unei ferestre pe ecranul text, folosind mouse-ul și, de asemenea, redimensionarea sa, precum în mediul *Borland C++*.
 5. Să se scrie un program care să administreze un parc de automobile. Informațiile relative la un automobil sunt: numărul de locuri, puterea (în cai putere), marca, numărul de înmatriculare, tipul de carburant (benzină sau motorină), natura (berlină, break sau coupe). Programul să permită intrarea unei mașini, ieșirea unei mașini, înlocuirea unei mașini cu alta de același model (având alt număr de înmatriculare).
 6. Scrieți un program care să citească temperaturile măsurate din oră în oră, precum și cantitățile zilnice de precipitații dintr-o lună a anului. Apoi programul să permită afișarea temperaturii maxime (împreună cu ziua și ora asociată), temperatura minimă (cu ziua și ora asociată), lista zilelor, ordonată descrescător în funcție de cantitatea de precipitații, media precipitațiilor zilnice din respectiva lună a anului. Eventual să se realizeze histogramme la diferite date statistice.
 7. Pentru un număr complex z și un număr natural $n \geq 1$, scrieți funcții/programe pentru a evalua următoarele funcții complexe:

$$\exp(z) = 1 + z/1! + z^2/2! + \dots + z^n/n!;$$

$$\sin(z) = z - z^3/3! + z^5/5! - \dots + (-1)^n z^{2n+1}/(2n+1)!;$$

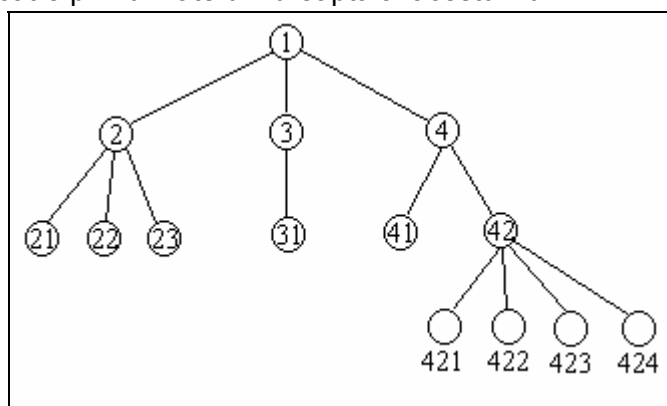
$$\cos(z) = 1 - z^2/2! + z^4/4! - \dots + (-1)^n z^{2n}/(2n)!;$$

$$\ln(1+z) = z - z^2/2 + z^3/3 - \dots + (-1)^n z^n/n, \text{ dacă } |z| < 1;$$

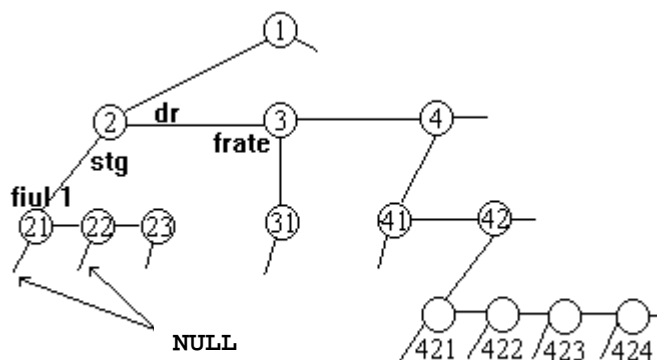
$$\arctg(z) = z - z^3/3 + z^5/5 - \dots + (-1)^n z^{2n+1}/(2n+1), \text{ dacă } |z| < 1.$$

8. Scrieți declarații pentru notațiile unui număr complex în coordonate polare și în coordonate carteziane, apoi funcții care să realizeze conversia între o notație și alta.
9. Scrieți un program C care să rezolve o ecuație de gradul II cu coeficienți complecși.
10. Fiind date două fracții p și q , să se afișeze forma ireductibilă a fracțiilor $p+q$, $p-q$, p/q , $p \times q$, punând în evidență numărătorul și numitorul acestora.
11. Să se determine printr-un program de câte ori o persoană născută în anul $A > 19000$, luna L , ziua Z , având vârsta V și-a aniversat ziua de naștere în aceeași zi a săptămânii cu cea în care s-a născut.
12. Să se verifice dacă un număr p dat este "deosebi" sau nu. Spunem că p este "deosebit" dacă există q astfel încât $p = q + s(q)$, în care $s(q)$ este suma cifrelor lui q .
13. Să se elimine dintr-o listă de numere întregi (creată dinamic) acele numere care împărțite la 13 dau un rest divizibil prin 3.
14. Să se creeze o listă a unor numere întregi, citite de la tastatură. Apoi să se creeze și afișeze lista răstrurnatelor (oginditelor) acestor numere.
15. Să se elimine dintr-o listă de numere reale acele numere care au partea zecimală egală cu zero.
16. Să se elimine dintr-o listă de numere întregi acele numere care sunt pare.
17. Se dă o listă de numere întregi creată dinamic. Să se creeze dinamic două liste, una cuprinzând numerele prime, iar alta cuprinzând numerele care nu sunt prime, dar sunt impare.
18. Se citesc n numere de la tastatură. Să se creeze o listă circulară a lor.
19. a) Să se rezolve problema damelor, memorând soluțiile într-o listă dinamică.
b) Să se genereze permutările unei mulțimi de elemente, memorate într-o listă creată dinamic.
20. Implementați algoritmi "quick-sort" (sortare rapidă) și "merge-sort" (sortare prin interclasare), pe baza tehnicii "*divide et impera*" și realizați programe în care să-i folosiți.
21. Coliere de mărgel. Se consideră n casete, fiecare conținând un număr precizat de mărgel bicolore (combinații de roșu, galben, albastru, verde, maro), culorile fiind cunoscute. Mărgelile din fiecare casetă sunt identice. Să se scrie un program interactiv pentru generarea tuturor colierelor circulare distincte de lungime maximă, cu condiția ca mărgelile alăturate să vină în contact cu aceeași culoare. Se consideră colier un șirag de minimum două mărgel.
22. Masa rotundă. Un număr de $2n+1$ persoane participă la discuții la o masă rotundă, care durează n zile. Să se găsească variantele de așezare la masă astfel încât o persoană să nu aibă în două zile diferite același vecin.

23. Pentru o expresie aritmetică conținând paranteze, operatorii $+$, $-$, $/$ și $*$ și operanzi numerici, să se determine valoarea sa. (Se vor folosi două stive, una a operanzilor, iar alta a operatorilor).
24. Se cere să se scrie un program care să deriveze (formal) o expresie. Se va folosi faptul că orice expresie aritmetică poate fi memorată sub forma unui arbore binar.
25. Se consideră un arbore oarecare și se cere memorarea sa sub forma unui arbore binar, folosind legături de tip *fiu-frate*. Un exemplu este dat în figura următoare. Observăm că putem lega rădăcina 1 de nodul 2, iar apoi, nodul 2 poate fi legat de primul fiu al său (21) și de următorul fiu al rădăcinii, deci 3, despre care se spune că este un *frate* a lui 2. Procedând astfel pentru toate nodurile din arbore, vom obține un arbore binar, cu două legături: cea din stânga este către primul fiu, iar cea din dreapta către primul frate din dreapta al acestui fiu.

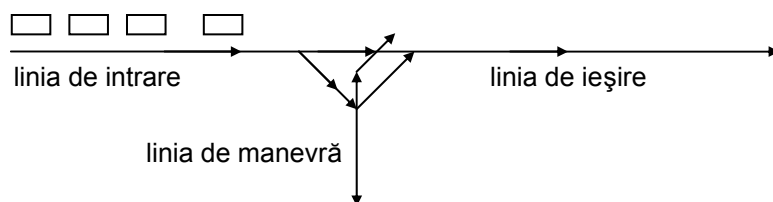


Astfel, arborele din figura anterioară se va memora în arborele binar de mai jos:



26. Se citesc numere întregi de la tastatură, până la întâlnirea numărului 0. Se cere să se creeze două liste dublu înlănțuite, una a numerelor negative, iar alta a numerelor pozitive prime.
27. Se dă o listă dublu înlănțuită de numere întregi pozitive. Să se creeze o listă dublu înlănțuită care să conțină doar numerele pare, apoi să se concateneze cele două liste.

28. Se dau doi arbori binari. Se cere să se înlocuiască fiecare nod al primului cu cel de al doilea arbore.
29. Se dă un arbore oarecare, informațiile din noduri fiind șiruri de caractere. Să se construiască o listă dublu înlănțuită care să conțină toate șirurile din nodurile arborilor, care au lungimile pare, apoi să se ordoneze această listă.
30. Scrieți o funcție care verifică dacă elementele unei liste simplu înlănțuite cuprinzând date de tip `char` sunt sau nu ordonate.
31. Scrieți o funcție care să calculeze suma elementelor pozitive dintr-o listă cuprinzând doar numere reale. Scrieți o funcție care să determine lungimea listei.
32. Într-o listă circulară dublu înlănțuită să se înlocuiască fiecare apariție a unui caracter care este vocală cu cea mai apropiată consoană din cadrul listei. Aceeași problemă în cazul unei liste circulare simplu înlănțuite.
33. Într-o listă circulară simplu înlănțuită să se înlocuiască fiecare apariție a unui caracter care este vocală cu cea mai apropiată consoană din alfabet. Aceeași problemă când lista este o coadă (necirculară).
34. Scrieți funcții/programe pentru a calcula suma, diferența, produsul și pentru a efectua împărțirea cu cât și rest a două polinoame, ale căror coeficienți (reali) sunt memorați în liste create dinamic.
35. Implementați structuri de date eficiente pentru a memora arbori oarecare. Scrieți apoi funcții care să parcurgă un arbore oarecare în lățime și, respectiv, în înălțime.
36. Fie F_k al k -lea termen din șirul lui Fibonacci. Un *arbore Fibonacci* de ordin k are $F_{k-1}-1$ vârfuri interne (notate cu $1, 2, \dots, F_{k+1}-1$) și F_{k+1} frunze (notate cu $0, -1, -2, \dots - (F_{k+1}+1)$) și se construiește după cum urmează:
 - pentru $k=0$ și $k=1$ arborele este $[1]$.;
 - pentru $k \geq 2$, rădăcina este notată cu F_k , subarboarele stâng este arbore Fibonacci de ordin $k-1$, iar subarboarele drept este arbore Fibonacci de ordin $k-2$, în care valorile vârfurilor sunt mărite cu F_k .
 - a) Să se scrie o funcție (recursivă) pentru a construi un arbore Fibonacci de ordin n .
 - b) Să se parcurgă arborele creat în ordine, listând doar informația din nodurile interne.
37. Fiind dat un arbore de sortare, scrieți o funcție recursivă pentru a șterge un nod astfel ca după ștergere, arborele rămas să fie în continuare arbore de sortare.
38. Se citesc întregii pozitivi n și m , ambii mai mici decât 100. Se cere să se afișeze reprezentarea ca fracție zecimală a numărului $r=m/n$. Eventuala perioadă se va afișa între paranteze.
39. Într-un triaj există o linie de cale ferată pentru manevre ca în figura de mai jos. Pe linia de intrare sunt n vagoane, numerotate de la 1 la n . Deplasările sunt permise doar conform sensurilor săgeților din figură.



- a) Cunoscându-se ordinea vagoanelor la intrare și ordinea dorită pe linia de ieșire, se cere să se stabilească dacă problema are soluție, iar în caz afirmativ să se afișeze manevrele necesare pentru a deplasa vagoanele de pe linia de intrare pe cea de ieșire;
- b) Aceeași problemă, dar se consideră că pe linia de intrare deplasarea este permisă în ambele sensuri;
- c) În aceleași condiții ca la (a), să se stabilească toate posibilitățile de a deplasa vagoanele de pe linia de intrare pe cea de ieșire.
- d) În aceleași condiții ca la (b), să se stabilească toate posibilitățile de a deplasa vagoanele de pe linia de intrare pe cea de ieșire.
40. Sortare topologică. Se citește un set de relații de forma $x < y$, cu semnificația că în definirea unui termen y se folosește termenul x , dintr-un dicționar. Se cere să se afișeze toți termenii din dicționar într-o anumită ordine astfel încât să nu se definească la un moment dat un termen care necesită cunoașterea unui alt termen definit mai târziu în cadrul dicționarului.
41. Se citesc relații despre mai multe mulțimi, de forma $A \text{ in } B$, $A \text{ nd } B$ și $A \text{ dj } B$, cu semnificațiile: mulțimea A este inclusă în mulțimea B , mulțimile A și B nu sunt disjuncte, respectiv mulțimile A și B sunt disjuncte. Să se verifice dacă un astfel de set de relații este sau nu consistent, în sensul că nu există o contradicție între relații. În caz de consistență, să se elimine relațiile redundante. (De exemplu, pentru seul de relații: $X \text{ in } Y$, $Y \text{ in } Z$ și $X \text{ dj } Z$ avem contradicție.
42. n bile colorate în cel mult k culori, sunt etichetate cu numere între 1 și n . Bilele se află pe o andrea verticală. Se cere să se mute pe alte k andree, pe fiecare andrea punând bilele de aceeași culoare. Fiecare andrea are un capăt blocat (de o gămălie), iar celălalt este liber și permite intrarea bilelor.
43. Se dau două liste alocate dinamic, fiecare cuprinzând cuvinte ordonate alfabetic. Se cere să se realizeze lista tuturor cuvintelor în ordine alfabetică.
44. Determinați o structură de date eficientă pentru memorarea matricelor rare (care au dimensiuni mari și foarte multe elemente de zero). Realizați adunarea și înmulțirea a două astfel de matrice.
45. Se consideră un arbore care cuprinde strămoșii unei persoane, al cărei nume figurează în rădăcinile arborelui. Nodul care figurează în stânga cuprinde numele tatălui, iar cel din dreapta numele mamei. Fiind dat numele unei persoane oarecare din familie să se afișeze numele bunicilor, dacă aceștia figurează în arbore.

Capitolul 5. Exploatarea fișierelor

- moduri de deschidere a fișierelor • citiri și scrieri din/în fișiere •
 - poziționarea în cadrul fișierului • numărarea articolelor •
 - sortarea componentelor dintr-un fișier cu structuri •
 - utilizarea argumentelor din linia de comandă •

Probleme rezolvate


1. Mai multe persoane au participat la un concurs și au obținut puncte. Să se realizeze o aplicație care să permită stocarea datelor persoanelor într-o bază de date (un fișier cu structuri), să poată adaugă noi persoane, să caute și să modifice punctajul unei persoane și să ordoneze persoanele descrescător după punctaj.

Vom folosi funcțiile standard ale limbajului C pentru lucrul cu fișiere, care sunt declarate în `<stdio.h>`. De remarcat că funcțiile `fopen`, `fread`, `fwrite` etc. la care ne referim pot prelucra orice gen de fișiere, atât fișiere text, cât și fișiere binare sau fișiere având anumite structuri de același tip, una după alta, cum e cazul bazelor de date din exemplul nostru. Însă, trebuie avută mare grijă la modul de folosire a acestor funcții, deoarece ele pot crea mari probleme programatorului amator, sau cel obișnuit cu procedurile și funcțiile similare din *Pascal*!

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct persoana {
    char nume[20];
    int punctaj;
};
void Citeste(struct persoana *p)
{
    fflush(stdin);
    printf("Dati numele: ");
    gets((*p).nume);
    printf("Dati punctajul lui %s: ", (*p).nume);
    scanf("%d", &(*p).punctaj);
    fflush(stdin);
}
void Schimba(struct persoana *a, struct persoana *b)
{
    struct persoana aux;
    aux = *a; *a = *b; *b = aux;
}
void Adauga(char nume_fis[13])
{
    /* Se compara numele fisierului cu sirul vid.
    In caz ca numele fisierului este vid, nu se executa nimic. */
    if (!strcmp(nume_fis, "")) return;
    FILE *f;
    struct persoana p;
```

```

/* se deschide fisierul f pentru "a"daugare la sfarsit, in mod
"b"inar */
if ((f = fopen(nume_fis,"ab")) == NULL)
{
    printf("\nNu se poate deschide fisierul %s.\n",nume_fis);
    return;
}
Citeste(&p);
fwrite(&p, sizeof(p), 1, f);
fclose(f);
}
void CautaSiModifica(char nume_fis[13])
{
    /* Se compara numele fisierului cu sirul vid.
    In caz ca numele fisierului este vid, nu se executa nimic. */
    if (!strcmp(nume_fis,"")) return;
    FILE *f;
    struct persoana p, q;
    // se deschide f pentru actualizare ("r+") in mod "b"inar
    if ((f = fopen(nume_fis,"r+b")) == NULL)
    {
        printf("\nNu se poate deschide fisierul %s.\n",nume_fis);
        return;
    }
    Citeste(&p);
    int gasit=0; long poz;
    while (!feof(f) && (!gasit))
    {
        poz=ftell(f);
        fread(&q,sizeof(struct persoana),1,f);
        if (!strcmp(p.num, q.num))
        {
            gasit=1; fseek(f,poz,SEEK_SET);
            fwrite(&p,sizeof(p),1,f);
        }
    }
    fclose(f);
    if (!gasit) printf("\nPersoana inexistentă.\n");
    else
        printf("\nPersoana avea punctajul: %d si acum are: %d.\n",
            q.punctaj,p.punctaj);
}


long FileSize(FILE *f)
// returneaza numarul de octeti ale unui fisier oarecare f !!
{
    long poz_curenta, lungime;
    poz_curenta = ftell(f); fseek(f, 0L, SEEK_END);
    lungime = ftell(f); fseek(f, poz_curenta, SEEK_SET);
    return lungime;
}
void Listeaza(char nume_fis[13])
{
    /* Se compara numele fisierului cu sirul vid.

```

```

In caz ca numele fisierului este vid, nu se executa nimic. */
if (!strcmp(numa_fis,"")) return;
struct persoana p;
FILE *f;
if ((f = fopen(numa_fis,"rb")) == NULL)
{
    printf("\nNu se poate deschide fisierul.\n");
    return;
}
long i=0;
while (!feof(f))
{
    fread(&p,sizeof(struct persoana),1,f);
    if (!feof(f)) // atentie !!
        printf("%ld. %s -> %d\n",++i,p.nume,p.punctaj);
    if (wherex()==20)
    {
        printf("\nApasati o tasta pentru continuare...\n");
        getch(); clrscr();
    }
}
printf("\nIn total: %ld persoane.\n",
        FileSize(f)/sizeof(struct persoana));
fclose(f);
}

```



```

void Sorteaza(char nume_fis[13])
/* sorteaza inregistrările din baza de date curenta, după
punctaj */
{
    if (!strcmp(numa_fis,"")) return;
    struct persoana p, q;
    FILE *f;
    if ((f = fopen(numa_fis,"r+b")) == NULL)
    {
        printf("\nNu se poate deschide fisierul.\n");
        return;
    }
    size_t lung_pers = sizeof(struct persoana); // 22 octeti
    long lungime_fisier=FileSize(f)/lung_pers;
    long poz, i;
    int ordonat;
    do {
        ordonat=1; rewind(f);
        for (i=1; i<=lungime_fisier-1; i++)
        {
            poz=ftell(f);
            fread(&p,sizeof(struct persoana),1,f);
            fread(&q,sizeof(struct persoana),1,f);
            if (p.punctaj<q.punctaj)
            {
                fseek(f,poz,SEEK_SET);
                fwrite(&q,sizeof(q),1,f);
                fwrite(&p,sizeof(p),1,f);
            }
        }
    } while (!ordonat);
}

```

```

        ordonat=0;
    }
    fseek(f,poz+lung_pers,SEEK_SET);
}
} while (!ordonat);
fclose(f);
}
void main(void)
{
    FILE *f; char nf[13]; char c;
    struct persoana p;
    strcpy(nf,"");
    do {
        clrscr();
        printf("\nAGENDA TELEFONICA -> %s\n\n",nf);
        printf("\n[D]enumire fisier, [A]daugare, [S]orteaza");
        printf("\n[C]autare/modificare, [L]istare, [O]prire\n");
        c=getche(); printf("\n");
        switch(c) {
            case 'd': printf("Dati numele agendei: ");
                       gets(nf); break;
            case 'a': Adauga(nf); getch(); break;
            case 'c': CautaSiModifica(nf); getch(); break;
            case 'l': Listeaza(nf); getch(); break;
            case 's': Sorteaza(nf); Listeaza(nf); getch();
            }
        } while (c!='o');
    }
}

```

Pentru a înțelege mai bine funcțiile de lucru cu fișiere în limbajul C/C++, vom comenta mai jos funcția care ordonează descrescător o bază de date, după punctaj.



Metoda folosită este **“sortarea prin bule (bubble-sort)”**, care poate fi descrisă foarte simplu astfel:

```

repetă
{
    ordonat=adevarat;
    parcurge sirul elementelor, de la primul la penultimul si
    dacă elementul curent < elementul de pe pozitia urmatoare atunci
    {
        interschimba cele doua elemente;
        ordonat=fals;
    }
}
pana cand ordonat=adevarat.

```



Funcția începe cu linia:

```
if (!strcmp(ume_fis,"")) return;
```

care face o verificare asupra numelui fișierului corespunzător bazei de date curente. Astfel, se folosește funcția `strcmp(s1,s2)` care returnează:

- 0 dacă cele două fișiere sunt identice;
- < 0, dacă primul șir e mai mic (alfabetic) decât al doilea;
- > 0, dacă cel de al doilea e mai mic decât primul.

Apoi se declară două variabile p și q , ce vor fi folosite în sortarea bazei de date.

```
struct persoana p, q;
```

Se declară o variabilă:

```
FILE *f;
```

care se asociază fișierului extern cu numele `nume_fis`.

Se încearcă deschiderea fișierului asociat lui f , prin funcția `fopen`, în modul binar, pentru citiri și scrieri ("r+b").

Dacă operația nu reușește, se afișează un mesaj de eroare:

```
if ((f = fopen(nume_fis, "r+b")) == NULL)
```

```
{  
    printf("\nNu se poate deschide fisierul.\n"); return;  
}
```

Altfel, se determină mărimea unei înregistrări din baza de date:

```
size_t lung_pers = sizeof(struct persoana); // 22 octeti și
```

numărul de astfel de înregistrări, adică numărul de persoane.

```
long lungime_fisier=FileSize(f)/lung_pers;
```

S-a folosit o funcție proprie, `FileSize(f)` care dă numărul de octeți ai unui fișier binar oarecare f , dar care trebuie să fie deschis. Vom comenta și această funcție, mai târziu.

Variabila `poz` va reține poziția elementului curent (al i -lea) din fișier, adică octetul la care începe el în fișier.

```
long poz, i; Variabila ordonat indică dacă la o anumită parcurgere a  
bazei de date a avut loc o interschimbare (0) sau nu (1).
```

```
int ordonat; Într-un ciclu corespunzător metodei de sortare  
descrie...
```

```
do {  
    ordonat=1;
```



... ne întoarcem la începutul fișierului f cu `rewind(f)`:

```
rewind(f); Parcurgem baza de date pentru fiecare pereche (p,q) de  
structuri din ea:
```

```
for (i=1; i<=lungime_fisier-1; i++)  
{
```



Vedem unde ne aflăm în fișier, înainte de a citi cele două înregistrări:

```
poz=ftell(f);
```



Observație. Valoarea returnată reprezintă octetul curent, nu înregistrarea curentă!



Citim cele două înregistrări, folosind `fread`, care are sintaxa:

```
size_t fread(void *p, size_t m, size_t n, FILE *f);
```

Această funcție citește un număr specificat prin n de itemi de mărime egale (m) dintr-un fișier dat f , rezultatul depunându-se într-un bloc de la adresa memorată în p . Numărul total de octeți citiți este $n \times m$.



Observație. Valoarea returnată este numărul de itemi citiți (nu octeți!), iar în caz de eroare sau sfârșit de fișier returnează, de obicei 0.

```
fread(&p, sizeof(struct                                persoana), 1, f);
fread(&q, sizeof(struct persoana), 1, f);
```

Apoi comparăm punctajele celor două înregistrări:

```
if (p.punctaj < q.punctaj)
{
```

În caz că cea de a doua (q =succesoarea lui p) este cu punctaj mai bun, folosind `fseek` și `fwrite` le inversăm:

```
fseek(f, poz, SEEK_SET);
fwrite(&q, sizeof(q), 1, f);
fwrite(&p, sizeof(p), 1, f);
ordonat=0;
}
```

Trecem la următoarea înregistrare:

```
fseek(f, poz+lung_pers, SEEK_SET);
```

} Procesul se repetă până în momentul în care `ordonat` își păstrează valoarea 1. } while (!ordonat);

... după care se închide fișierul

```
fclose(f);
```



} Funcția de poziționare în cadrul fișierului este `fseek`, dar ea lucrează la nivel de octeți! Sintaxa sa este: `int fseek(FILE *f, long l, int unde)`. Aici f este fișierul, l este diferența în octeți între poziția `unde` și noua poziție. Pentru fișierele deschise ca fișiere text, l ar putea fi 0 sau o valoare returnată de `ftell`.

În sfârșit, noutatea adusă față de *Pascal* este parametrul `unde`, care poate avea una din valorile:

```
SEEK_SET = 0, semnificând că se pleacă de la începutul fișierului,
SEEK_CUR = 1, semnificând o distanță de la poziția curentă în fișier,
SEEK_END = 2, semnificând de la sfârșitul fișierului
```

Dacă funcția reușește (deci indicatorul în fișier s-a mutat cu succes), `fseek` returnează 0, în caz de eșec `fseek` returnează un întreg diferit de zero, eventual un cod de eroare, în cazul în care, de pildă, fișierul nu ar fi fost deschis.



În continuare, ne ocupăm de funcția `FileSize`. Ea returnează numărul de octeți al unui fișier oarecare f , deschis:

```
long FileSize(FILE *f)
{
    long poz_curenta, lungime;
    // memorăm locul unde ne aflăm
    poz_curenta = ftell(f);
    // ne ducem la sfârșitul fișierului
    fseek(f, 0L, SEEK_END); // 0L semnifica numărul 0 vazut ca
    long int !
    // locul unde ne aflăm acum este chiar lungimea fișierului
```

```

    lungime = ftell(f);
// ne intoarcem de unde am plecat
    fseek(f, poz_curenta, SEEK_SET);
    return lungime;
}

```

2. Scrieți un program care să afișeze toți parametrii din linia de comandă și variabilele de mediu.

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
void main(int nr_de_p, char *param[], char *var_mediu[])
{
    int i;
    clrscr();
    for(i=0; i<nr_de_p; i++)
        printf("Parametrul[%d] = %s\n",i,param[i]);
    for(i=0; env[i] != NULL ; i++)
        printf("Variabila de mediu[%d] = %s\n",i,var_mediu[i]);
}

```



Ce face acest program? Este primul program în care funcția `main` are și parametri. Aceștia sunt:

- `nr_p` = un număr întreg reprezentând numărul de parametri din linia de comandă, adică lungimea vectorului `param`;
- `param` = un vector de șiruri de caractere, în care:
 - `param[0]` = numele programului pornit (cu toată calea sa);
 - `param[1]` = primul argument al programului pornit (dacă există);
 - `param[2]` = al doilea argument al programului pornit (dacă există);
 - ...
- `env[0] ... env[...]` = variabile de mediu, înțelegând prin aceasta diferite date, precum:
 - directorul fișierelor temporare;
 - directorul de *boot*-are (încarcare a sistemului de operare);
 - fișierul interpretor de comenzi (`COMMAND.COM`);
 - căile de căutare (stabilite cu `PATH`);
 - alte variabile stabilite cu `SET`;
 - prompterul sistemului;
 - ultima linie de comandă dată din sistemul de operare.

Toate aceste informații pot fi folosite în programe ce lucrează cu fișiere, ca în exemplul următor, sau în diverse programe profesionale. Succes!

3. Implementați comanda `copy` din *MS-DOS*.



Vom folosi facilitățile descrise în exemplul anterior. Astfel, denumind acest fișier `my_copy.cpp`, se va crea, în urma compilării, executabilul `my_copy.exe` care se va putea apela cu:

my_copy <fisier_sursa> <fisier_destinatie> Însă, dacă lipsesc
unul sau ambii parametri, funcția AnalizaParametri va cere parametri lipsă.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#define BUFLNG 1000 //lungimea buffer-ului folosit la copiere
#define FTMP "my_copy.tmp" //fisier de lucru temporar
char sursa[13], dest[13];
void AnalizaParametri(int nr_p, char *param[])
{
    strcpy(sursa,param[1]);
    strcpy(dest,param[2]);
    switch (nr_p)
    {
        case 1: printf("\nFisier sursa: "); gets(sursa);
        case 2: printf("\nFisier destinatie:"); gets(dest);
        case 3: strcpy(sursa,sursa); strcpy(dest,dest); break;
        default: printf("\nPrea multi parametri.\n");
                exit(1);
    }
}
void Copiere(FILE *f1, FILE *f2)
{
    char c,*buf_cit;
    int nb_cit, nb_scr;
    if ((f1= fopen(from,"rb"))== NULL )
    {
        printf("\nFisier de intrare eronat.\n");
        exit(1);
    }
    if ((f2 = fopen(dest,"rb")) != NULL)
    {
        printf("\nFisier destinatie existent.");
        printf(" Suprascriem [d/n]? ");
        c = getche();
        if (c=='N' || c=='n') strcpy(dest,FTMP);
    }
    if ((f2 = fopen(to,"wb"))== NULL)
    {
        printf("\n\aEroare creare fisier destinatie.\n");
        exit(1);
    }
    if ((buf_cit = (char *)malloc(BUFLNG))== NULL)
    {
        printf("\n\aMemorie insuficienta.\n");
        exit(1);
    }
    while ((nb_cit=fread(buf_cit,1,BUFLNG,f1)) !=0 &&
           (nb_scr=fwrite(buf_cit,1,nb_cit,f2)) == nb_cit);
    fclose(f1); fclose(f2);
    printf("\nCopiere");
    if (nb_cit != nb_scr) printf("\a in");
    printf("completa.\n");
}
```

```
void main(int argc, char *argv[])
{
    FILE *pfrom, *pto;
    AnalizaParametri(argc, argv); Copiere(pfrom, pto);
}
```



Observații.

1. Mesajul afișat poate fi și copiere incompletă. De ce? Lăsăm în seama cititorului să schimbe programul pentru a funcționa în orice caz.
2. Afișarea lui '\a' înseamnă producerea unui sunet în difuzor.

4. Scrieți un program care să afișeze un fișier text astfel: dacă fișierul conține texte între acolade '{' și '}', acelea să fie afișate în alb deschis, iar restul textului în alb închis. De asemenea, se cere ca literele să fie afișate "cu incetinitorul".

Vom deschide un fișier în mod binar, pentru a nu avea neplăceri cauzate de caracterul *Enter* și citind caracter cu caracter, îl vom analiza și îl vom afișa. Excepțiile sunt date de cele două caractere speciale, care vor controla culorile.

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#define pauza 20
void main(void)
{
    FILE *fis;
    char nume_fis[13];
    char linie[255];
    char c;
    textcolor(7); textbackground(0); clrscr();
    printf("Dati numele fisierului: "); gets(nume_fis);
    if ((fis = fopen(nume_fis, "rb")) == NULL)
    {
        printf("Nu se poate deschide fisierul.\n");
        return;
    }
    while (!feof(fis))
    {
        c=fgetc(fis); // fgetc preia un caracter dintr-un fisier
        if (c=='{') textcolor(15);
        else
            if (c=='}') textcolor(7);
        else
            if (c==9) { putch(' '); putch(' '); putch(' '); }
            else { putch(c); delay(pauza); }
    }
    fclose(fis);
}
```



Observație. Afișarea pe ecran a caracterului citit se face cu `putch`, care ține cont de culori. De asemenea, se observă că *TAB* (caracterul cu codul

ASCII egal cu 9) este înlocuit cu trei spații. Propunem cititorului perfecționarea programului, folosind eventual fișiere text.

5. Scrieți un program care să facă o copie unui fișier text dat.

```
#include <stdio.h>
void main(void)
{
    FILE *in, *out; char sursa[13], dest[13];
    fflush(stdin);
    printf("Dati sursa: "); scanf("%s",&sursa); fflush(stdin);
    printf("Dati destinatia: "); scanf("%s",&dest);
    if ((in = fopen(sursa, "rt")) == NULL)
    {
        printf("Nu se poate deschide fisierul sursa.\n");
        return;
    }
    if ((out = fopen(dest, "wt")) == NULL)
    {
        printf("Nu se poate crea fisierul destinatie.\n");
        return;
    }
    // se copiaza caracterele din "in" in "out"
    while (!feof(in))
        fputc(fgetc(in), out);
    fclose(in); fclose(out);
}
```

Probleme propuse

1. Să se scrie un program C care afișează conținutul directorului curent.
2. Scrieți un program care să elimine toate comentariile dintr-un program C. Nu uitați să maneveați adecvat șirurile dintre ghilimele și constantele de caractere!
3. Scrieți un program care citește de la tastatură câte trei numere reale, apoi le scrie într-un fișier text 'IN.TXT'. Apoi, citind aceste trei numere, reprezentând laturile unui triunghi, scrie în fișierul text 'OUT.TXT' tipul triunghiului format.
4. Scrieți un program C care concatenează două fișiere text într-unul singur (primul dintre ele):
5. Scrieți un program pentru a verifica un program C din punct de vedere al erorilor de sintaxă rudimentare, ca de exemplu: paranteze neperechi. Nu uitați ghilimelele, atât cele simple, cât și cele duble și comentariile.
6. Să se scrie un program care sortează un fișier text în el însuși; fișierul conține cel mult 100 de linii.
7. Să se realizeze o aplicație complexă care să gestioneze mai multe seturi de date referitoare la elevii participanți la un concurs (de admitere, de exemplu). Programul va realiza următoarele:
 - selectarea numelui fișierului de lucru (nf), adică a bazei de date;

- adăugarea unui elev în baza de date, precum și ștergerea unui elev din baza de date (ștergera se va face în funcție de numele elevului sau de poziția sa în cadrul fișierului);
- căutarea unui elev în fișier;
- ordonarea alfabetică a elevilor din baza de date, precum și ordonarea descrescătoare după medii a elevilor din baza de date;
- listarea conținutului fișierului (la un moment dat);

Toate aceste operații vor fi realizate cu ajutorul unor proceduri, selectabile dintr-un meniu. Activarea unei opțiuni din meniu se va face cu ajutorul tastelor de săgeți sus și jos. Selectarea propriu-zisă a ei se va realiza acționând tasta `Enter`. Programul se va termina când se va alege ultima opțiune sau se va acționa tasta `Escape` (în meniu). Ca variantă propunem utilizarea mouse-ului.

8. Fie un fișier creat prin apeluri succesive ale procedurii de adăugarea unui elev din programul realizat la problema anterioară. Scrieți un program separat care să afișeze conținutul fișierului într-un fișier text. Adăugați, apoi, acest program, programului de la problema anterioară, sub forma unei noi opțiuni/funcții.
9. Un fișier text conține cel puțin 100000 linii de forma:

```
nume prenume nota1 nota2
```

Fiecare linie cuprinde informații referitoare la un elev care dă un anumit examenul. Se cere să se creeze un fișier cu tip, al elevilor care obțin cel puțin media 5. Fiecare articol al fișierului creat va conține numele și prenumele elevului, precum și media.
10. În condițiile problemelor anterioare, se cere crearea unui fișier al elevilor neadmiși (cu medii sub 5).
11. Se dă un fișier text cu foarte multe linii. Se cere să se ordoneze.
12. Se consideră un fișier `F` cuprinzând datele mai multor elevi (nume, punctaj). Se cere să se creeze trei fișiere text `G1`, `G2` și `G3`, fiecare cuprinzând câte o treime din elevii lui `F`, după ce `F` a fost ordonat după punctaj. Liniile acestor fișiere vor conține numele și punctajele elevilor. Primul fișier va cuprinde prima treime de elevi, fișierul `G2` va cuprinde următoarea treime din elevi, iar `G3` va conține elevii rămași.
13. Aceeași problemă ca mai înainte, dar fără a ordona mai întâi fișierul `F`. În plus, se cere să se ordoneze fișierele `G1` și `G2`, apoi să se interclaseze fișierele `G1` și `G2` rezultând un fișier `G`.
14. Scrieți un program cuprinzând diferite funcții de prelucrare a fișierelor cuprinzând numere întregi: ordonare, căutare, adăugare sau ștergere de articole etc.. Realizați în funcția `main` diferite apeluri ale acestor funcții.
15. Să se scrie un program care să realizeze un pian electronic. Fiecărei taste `i` se va atașa o notă muzicală. Tastele funcționale (`F1` . . `F12`) vor fi folosite pentru diferite operații speciale: salvarea melodiei curente într-un fișier, restaurarea și cântarea unei melodii dintr-un fișier etc.

16. Să se scrie un program care tipărește distribuția frecvenței lungimii cuvintelor aflate într-un fișier text. Cuvintele sunt separate prin spații.
17. Să se scrie un program care compară, linie cu linie, conținutul a două fișiere text, afișând numărul de ordine al celor neidentice.
18. Se dau două șiruri de caractere S și T și un fișier text F . Să se scrie funcția care copiază tot conținutul lui F într-un alt fișier text G , însă înlocuiește fiecare apariție a șirului S cu șirul T .
19. Să se concateneze două fișiere oarecare într-un al treilea fișier. Numele celor trei fișiere se vor citi din linia de comandă a programului.
20. Se consideră un fișier cu informații despre locul nașterii unor persoane. Să se determine pentru fiecare localitate, numărul de persoane născute în localitatea respectivă.
21. Se consideră un fișier cu tip cuprinzând mai multe numere reale, memorate pe 10 poziții caracter, din care 3 zecimale. Să se scrie într-un fișier text, câte 10 pe linie, acele numere care sunt și întregi și sunt și prime.
22. Să se realizeze un program pentru o agendă telefonică care să permită gestionarea unor persoane, a adreselor lor și a numerelor lor de telefon. Operațiile care vor fi realizate sunt: adăugare persoană, eliminare persoană, căutare număr de telefon a unei persoane, căutarea unei persoane a cărui număr de telefon se cunoaște, modificare număr de telefon persoană, ordonare alfabetică, listarea persoanelor într-un fișier text.
23. Să se scrie un program cu fișiere care să permită administrarea unui top al preferințelor dumneavoastră muzicale.
24. Scrieți un program pentru evidența împrumuturilor de cărți la o bibliotecă. Programul va permite editarea unui raport lunar ce va cuprinde: data raportului; numărul de împrumuturi; numărul de cărți împrumutate; numărul de cărți înapoiate; numărul de cărți pierdute; împrumuturile restante.
25. Să se afișeze conținutul unui fișier text într-o fereastră ecran astfel încât textul să fie aliniat: a) la stânga; b) la dreapta; c) pe mijloc.
26. Aceeași problemă ca cea anterioară, dar alinierea textului să se facă "din stânga în dreapta", în sensul că se vor pune spații suplimentare între cuvinte, pentru ca textul să fie aliniat în ambele părți.
27. Să se scrie un program cu o interfață cât mai prietenoasă, care să realizeze evidența aprovizionărilor/vânzărilor efectuate într-un magazin.

Capitolul 6. Algoritmi de teoria grafurilor

- arbori oarecare • parcurgeri în lăţime şi adâncime •
- arbori parţiali de cost minim •

Probleme rezolvate

1. Să se definească o structură adecvată pentru memorarea arborilor oarecare. Să se scrie funcţii care să parcurgă astfel de arbori în lăţime şi în adâncime.

Vom defini un nod al arborelui ca fiind o structură caracterizată de o informaţie (aici un număr întreg) şi un număr de fii, care vor fi pointeri către alte structuri de acelaşi fel.

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
typedef int tip; // tipul elementelor din nodurile arborelui
// un nod al arborelui
struct nod {
    tip info; // informatia din nod
    int nf; // numarul de fii
    struct nod *fiu[20]; // fii = pointeri la alte noduri
};
typedef nod * arb;
int Creeaza(arb *pa, int nr, int parinte)
{
    tip el;
    *pa=(arb)malloc(sizeof(nod));
    if (!*pa) return 0;
    printf("Dati info pt. fiul %d al lui %d: ",nr,parinte);
    scanf("%d",&el); (*pa)->info = el;
    if (el)
    {
        printf("Dati nr. de fii pt. %d: ",el);
        scanf("%d",&(*pa)->nf);
        for (int i=0; i<(*pa)->nf; i++)
            Creeaza(&(*pa)->fiu[i], i+1, el);
    }
    return 1;
}
void Adancime(arb a)
{
    if (a)
    {
        printf("%d,",a->info);
        for (int i=0; i<a->nf; i++) Adancime(a->fiu[i]);
    }
}
void Latime(arb a, int e_radacina)
{
    if (a)
```



```

    {
        int i;
        if (e_radacina) printf("%d", a->info);
        for (i=0; i<a->nf; i++) printf("%d", a->fiu[i]->info);
        for (i=0; i<a->nf; i++) Latime(a->fiu[i], 0);
    }
}
void main()
{
    arb p; clrscr(); printf("\nCrearea arborelui:\n");
    if (!Creeaza(&p, 0, 0))
    {
        printf("Memorie insuficienta!"); return;
    }
    printf("\nParcurearea in adincime:\n"); Adancime(p);
    printf("\nParcurearea in latime:\n"); Latime(p, 1);
    getch();
}

```



Creearea arborelui se realizează astfel: se cere informația nodului curent, apoi numărul de fii ai acestuia, iar după aceasta se apelează aceeași procedură recursiv. Parcurearea în lățime și cea în lărgime sunt ambele proceduri recursive.

La parcurearea în lățime se explorează mai întâi rădăcina, apoi recursiv, fiecare fiu al ei. Parcurearea în lățime înseamnă parcurearea rădăcinii, apoi a fiilor acesteia, apoi a fiilor fiilor ș.a.m.d. pe toate nivelele.

2. Pentru construirea unei rețele interne de comunicație între secțiunile unei întreprinderi s-a întocmit un proiect în care au fost trecute toate legăturile ce se pot realiza între secțiunile întreprinderii. În vederea definitivării proiectului și întocmirii necesarului de materiale etc., se cere să se determine un sistem de legături ce trebuie construit, astfel încât orice secție să fie racordată la această rețea de comunicație, iar cheltuielile de construcție să fie minime.

Problema e una clasică de teoria grafurilor și se reduce la determinarea unui arbore parțial de cost minim al unui graf cu n noduri, dat prin matricea costurilor C , unde $C[i][j] = \infty$ (infinit, o valoare foarte mare), când i și j nu sunt noduri vecine în graf. Problema se poate rezolva fie cu *algoritmul lui Prim*, fie cu cel al lui *Kruskal*.



În **algoritmul lui Prim** se folosește un vector *Vecin*. În fiecare pas al algoritmului, $Vecin[i] = 0$ dacă i este în arborele parțial deja construit, respectiv $Vecin[i] = k$, k fiind vârful cu proprietatea că (i, k) este muchia de cost minim printre toate muchiile cu o extremitate i , în afara arborelui, și o alta în arborele deja construit. Astfel, se folosește **metoda "greedy"**.

```

#include <stdio.h>
#include <conio.h>
#define max 10
#define infinit 1000;

```

```

void main()
{
// nodurile sunt numerotate de la 1 la n !
int C[1+max][1+max];
int n,i,j,k,pas, Cost=0, min;
int Vecin[1+max];
clrscr(); printf("Algoritmul lui Prim\n");
printf("Dati n: "); scanf("%d",&n);
for (i=1; i<=n-1; i++)
{
for (j=i+1; j<=n; j++)
{
printf("C[%d,%d]=" ,i,j); scanf("%d",&C[i][j]);
if (C[i][j]==0) C[i][j]=infinite;
C[j][i]=C[i][j];
}
C[i][i]=0;
}
for (i=1; i<=n; i++) Vecin[i]=1; Vecin[1]=0;
for (pas=2; pas<=n; pas++)
{
min=infinite;
for (i=1; i<=n; i++)
if (Vecin[i]!=0)
if (C[i][Vecin[i]]<min)
{
min=C[i][Vecin[i]];
k=i;
}
printf("%d--%d\n",k,Vecin[k]);
Cost=Cost+C[k][Vecin[k]];
Vecin[k]=0;
for (i=1; i<=n; i++)
if (Vecin[i]!=0)
if (C[i][Vecin[i]]>C[i][k]) Vecin[i]=k;
}
printf("Cost = %d",Cost);
getch();
}

```

3. Să se determine arborele parțial de cost minim al unui graf cu n varfuri dat prin muchiile si costurile atașate lor, folosind *algoritmul lui Kruskal*.



În implementarea **algoritmului lui Kruskal** se foloseste tot metoda "*greedy*". Mai întâi se alege muchia de cost minim, apoi se adaugă, în mod repetat (de $n-1$ ori) muchia de cost minim printre cele nealese încă și cu proprietatea că adăugarea ei nu duce la formarea unui ciclu în arborele deja creat.

Pentru a soluționa problema apariției unui eventual ciclu prin adăugarea unei muchii, pentru fiecare nod vom memora nodul său tată, adică

rădăcina arborelui din care face parte respectivul nod, folosind un vector `Tata`. Se va pleca cu `n` arbori, fiecare format din câte un nod, care este și rădăcina sa: `Tata[i]=-1` la început, dar pe parcurs:

`Tata[i]=-numărul de noduri din arborele respectiv.`

În momentul în care o muchie $i=(u,v)$ devine candidată (muchiiile se iau în ordinea crescătoare a costurilor sale, de la citirea lor!), se verifică cărui arbore aparțin u și v , fie aceștia `arb1` și `arb2`. Dacă `arb1` și `arb2` sunt identici, înseamnă că se formează ciclul, dar dacă nu, se adaugă această muchie și apoi se reunesc cei doi arbori, legându-l pe cel mai mic la cel mai mare.

```
#include <stdio.h>
#include <conio.h>
#define max 10
#define infinit 1000;
int Muchie[max+1][3]; int m,n,i, Cost=0;
int arb1, arb2; int Tata[max+1];
int Componenta(int v)
{
    while (Tata[v]>0) v=Tata[v];
    return v;
}
void Reuneste(int a, int b)
{
    int suma=Tata[a]+Tata[b];
    if (Tata[a]<Tata[b])
    {
        Tata[a]=suma; Tata[b]=a;
    }
    else
    {
        Tata[b]=suma; Tata[a]=b;
    }
}
void main()
{
    clrscr(); printf("Algoritmul lui Kruskal\n");
    printf("Dati nr. de noduri: "); scanf("%d",&n);
    printf("Dati nr. de muchii: "); scanf("%d",&m);
    printf("Dati muchiile in ordine crescatoare a costurilor\n");
    for (i=1; i<=m; i++)
    {
        printf("Muchia nr. %d:\n",i);
        printf("prima extremitate ="); scanf("%d",&Muchie[i][1]);
        printf("a doua extremitate="); scanf("%d",&Muchie[i][2]);
        printf("costul muchiei      ="); scanf("%d",&Muchie[i][0]);
    }
    for (i=1; i<=n; i++) Tata[i]=-1;
    Tata[1]=0;
    for (i=1; i<=m; i++)
    {
        arb1=Componenta(Muchie[i][1]);
        arb2=Componenta(Muchie[i][2]);
        if (arb1!=arb2)
        {
```

```

        Reuneste(arb1,arb2);
        Cost+=Muchie[i][0];
        printf("%d--%d\n",Muchie[i][1],Muchie[i][2]);
    }
}
printf("Cost=%d\n",Cost); getch();
}

```



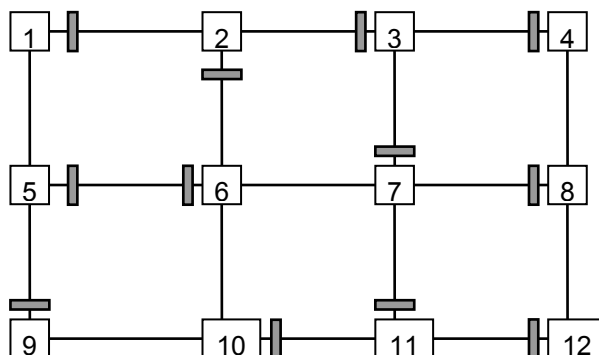
Propunem cititorului să îmbunătățească programul astfel încât să permită introducerea muchiilor în orice ordine.

Probleme propuse

1. Scrieți proceduri iterative pentru parcurgerea grafurilor a) în lățime, b) în adâncime.
2. Scrieți o procedură recursivă pentru parcurgerea unui graf orientat în adâncime.
3. Dându-se n numere pozitive d_1, d_2, \dots, d_n , astfel încât suma lor este $2n-2$, să se construiască un arbore cu n noduri, ale căror grade sunt aceste n numere.
4. Secvență grafică. Spunem că o secvență de numere d_1, d_2, \dots, d_n , este *secvență grafică* dacă există un graf cu n noduri pentru care d_1, d_2, \dots, d_n să fie gradele nodurilor sale. Dându-se o secvență de n numere, se cere să se determine dacă ea este o secvență grafică sau nu.
5. Se dă un graf prin matricea sa de adiacență. Să se verifice dacă este sau nu conex.
6. Realizați colorarea unui graf cu n noduri folosind m culori, astfel încât două vârfuri vecine să aibe culori diferite.
7. Să se afișeze componentele conexe ale unui graf neorientat, dat prin matricea sa de adiacență.
8. Pentru un grup de n persoane să se determine *celebritatea*, adică acea persoană care este cunoscută de toată lumea, dar nu cunoaște pe nimeni (dacă există).
9. Se consideră un graf neorientat. Să se verifice dacă el are sau nu un circuit de lungime a) 3; b) 4.
10. Să se verifice dacă un graf orientat aciclic conține sau nu un drum hamiltonian. Puteți găsi un algoritm liniar?
11. Să se găsească, folosind un algoritm liniar, dacă există, un circuit într-un graf conex care conține două noduri date a și b , dar nu conține nodul c .
12. Scrieți un program care să determine (dacă există) un nod al unui graf conex prin dispariția căruia graful rămâne conex. Se consideră că o dată cu dispariția nodului respectiv, dispar și arcele incidente lui.
13. Implementați *algoritmul lui Prim* pentru determinarea arborelui parțial de cost minim într-un graf, considerând graful dat sub forma listei muchiilor, în ordinea descrescătoare a costurilor acestora.

14. Bariere. Un șoricel se află situat într-un nod al unei rețele dreptunghiulare de dimensiune $m \times n$, având forma și numerotarea nodurilor conform figurii (în care $m=4$, $n=3$):

Fiecare nod v al rețelei are exact o barieră pe o muchie vw ,



care blochează trecerea șoricelului de la v la w , dar și de la w la v . (Pentru exemplul din figura anterioară, în nodul 2 avem o barieră către nodul 6, care împiedică trecerea șoricelului de la nodul 2 la nodul 6, dar și de la nodul 6 la nodul 2.). Șoricelul trebuie să ajungă la o bucătică de cașcaval, situată într-un alt nod al rețelei, parcurgând rețeaua pe drumul de cost minim, respectând următoarele reguli:

- Șoricelul, aflat în nodul v , poate trece la nodul w , dacă nu există nici o barieră pe muchia vw ; această trecere îl costă 1\$.
 - Șoricelul poate schimba poziția barierei din nodul curent, ceea ce îl costă tot 1\$. Pentru exemplul din figura anterioară, șoricelul (presupus a fi inițial în nodul 2) poate ajunge în nodul 7, în mai multe moduri, de exemplu: a) mută bariera din 2 (așezând-o către nodul 1), se deplasează apoi în nodul 6, apoi în 7 (costul: 3\$); b) mută bariera din 2 (așezând-o către nodul 1), se deplasează apoi în nodul 6, apoi în nodul 10, unde pune bariera către nodul 9, apoi se duce în 11, pune bariera de aici către nodul 10 și, în sfârșit, se deplasează în nodul 7 (costul: 7\$). Se cere să se determine un astfel de drum de cost minim al șoricelului către cașcaval.
15. Scrieți un algoritm liniar pentru a determina numărul celor mai scurte drumuri (nu neapărat disjuncte) între două vârfuri x și y dintr-un graf orientat.
16. Se dă o schemă a transportului în comun dintr-un oraș, care conține stațiile 1, 2, ..., n . Există m linii între aceste stații și se știe că între oricare două stații există legătură, eventual cu schimbarea mijlocului de transport. Trebuie să determinați dacă există cel puțin o linie directă prin blocarea căreia legătura (directă sau indirectă) între cel puțin două stații se întrerupe. Dacă astfel de linii există, să se propună înființarea unui număr cât mai mic de linii directe între stațiile existente, astfel încât prin blocarea unei singure linii directe, oricare ar fi aceasta, circulația între

oricare două stații să fie posibilă; se alege soluția pentru care suma ocolurilor pe traseele variantă (măsurate în număr de linii directe) să fie cât mai mică.

17. Fie n persoane P_1, \dots, P_n , care doresc fiecare să transmită propria bârfă celorlalte persoane. Numim "instrucțiune" o pereche (i, j) având următorul efect: persoana P_i transmite persoanei P_j propria sa bârfă, dar și eventualele bârfe primite anterior prin instrucțiuni de la alte persoane. Din păcate, anumite perechi de persoane, citite de la tastatură, se dușmănesc și nu comunică între ele. Să se determine, dacă este posibil, o secvență de instrucțiuni prin care fiecare persoană să cunoască bârfele tuturor celorlalte persoane.
18. Scrieți un program C care să implementeze algoritmul de ordonare cu ansambluri (heap-sort).
19. a) Scrieți o funcție de ștergere a unui arbore binar
b) Scrieți o funcție de duplicare a unui arbore binar.
20. Scrieți o funcție iterativă de parcurgere în postordine a unui arbore binar.

Capitolul 7. Grafică în C/C++

- utilizarea mouse-ului în mod grafic • joc cu strategie câștigătoare •
- bioritmul • utilizarea fonturilor •

Probleme rezolvate

1. Formarea imaginilor în lentilele convergente.

Ne propunem să realizăm un program didactic pentru orele de fizică (optică), care să evidențieze formarea imaginilor în lentilele convergente, în oricare din cele două cazuri: când obiectul se află între lentilă și distanța focală, respectiv după distanța focală. Modul de utilizare a programului este simplu: se fixează mai întâi abscisa lentilei (`lentila`), apoi se mișcă mouse-ul până se stabilește înălțimea dorită a lentilei. Mouse-ul este coborât apoi forțat pe axă și se stabilește focarul din stânga al lentilei (abscisa sa este `focar` = distanța focală).



În sfârșit se stabilește și abscisa obiectului și înălțimea sa: (`obiect`, `h1`). Se determină apoi distanța `x2` a imaginii față de lentilă, folosind *legea lentilelor*:

$$1/f = 1/x1 - 1/x2,$$

- unde f este distanța focală,
- $x1$ este distanța obiectului față de lentilă,
- iar $x2$ este distanța imaginii față de lentilă (chiar dacă aceasta este reală (negativă) sau imaginară (pozitivă)).

Pe măsură ce se stabilește înălțimea lentilei (`h1`), programul reprezintă grafic și imaginea formată.

```
#include <graphics.h>
#include <dos.h>
#include <stdlib.h>
#include "mouse.h"
void OpenGraph()
{
    int gd=0, gm; initgraph(&gd, &gm, "c:\\bc\\bgi");
}
int lentila, h1; int focar, f;
int obiect, x1, x2; int h1, h2;
void StabilesteLentila()
{
    int b, x, y;
    do {
        MouseData(b, x, y); MouseMove(x, 240);
    } while (b!=1);
    lentila=x;
    delay(500); MouseHide(); setwriteMode(1);
    do {
        MouseData(b, x, y); MouseMove(lentila, y);
        h1=240-y; //
        line(lentila, 240-h1, lentila, 240+h1);
```

```

        MouseShow(); delay(50); MouseHide();
        line(lentila,240-h1,lentila,240+h1);
    } while (b!=1);
    setwritemode(0);
    line(x,240-h1,x,240+h1); line(x,240-h1,x-4,240-h1+4);
    line(x,240-h1,x+4,240-h1+4); line(x,240+h1,x-4,240+h1-4);
    line(x,240+h1,x+4,240+h1-4); MouseShow();
}
void StabilesteFocar()
{
    int b,x,y;
    do {
        MouseData(b,x,y); MouseMove(x,240);
    } while (b!=1);
    focar=x; f=abs(lentila-focar); focar=lentila-f;
    MouseHide(); circle(focar,240,2); circle(lentila+f,240,2);
    MouseShow();
}
void StabilesteObiect()
{
    int b,x,y;
    do {
        MouseData(b,x,y); MouseMove(x,240);
    } while (b!=1);
    obiect=x; x1=abs(lentila-obiect);
    obiect=lentila-x1; delay(500);
    MouseHide(); setwritemode(1);
    do {
        MouseData(b,x,y); MouseMove(obiect,y);
        setcolor(14); line(obiect,240,obiect,y);
        h1 = -y+240; x2 = f*x1/(x1-f);
        h2 = h1*x2/x1; setcolor(13);
        line(obiect,240-h1,lentila,240-h1);
        line(lentila,240-h1,lentila+x2,240+h2);
        line(obiect,240-h1,lentila+x2,240+h2);
        setcolor(10);
        line(lentila+x2,240+h2,lentila+x2,240);
        MouseShow(); delay(50); MouseHide();
        setcolor(14); line(obiect,240,obiect,y);
        setcolor(13);
        line(obiect,240-h1,lentila,240-h1);
        line(lentila,240-h1,lentila+x2,240+h2);
        line(obiect,240-h1,lentila+x2,240+h2);
        setcolor(10);
        line(lentila+x2,240+h2,lentila+x2,240);
    } while (b!=1);
    h1=-y+240; setwritemode(0);
    line(obiect,240,obiect,240-h1); MouseShow();
}
void DeseneazaImaginea()
{
    x2 = f*x1/(x1-f); h2 = h1*x2/x1;
    MouseHide(); setcolor(13);
    line(obiect,240-h1,lentila,240-h1);
    line(lentila,240-h1,lentila+x2,240+h2);

```



```

        line(obiect,240-h1,lentila+x2,240+h2);
        setcolor(10); line(lentila+x2,240+h2,lentila+x2,240);
        MouseShow();
    }
    void main()
    {
        int b,x,y;
        OpenGraph();  MouseInit(); setcolor(11);
        do {
            sound(300); delay(100); nosound();
            MouseHide();
            cleardevice(); setcolor(15);
            line(0,240,639,240); MouseShow(); delay(500);
            StabilesteLentila(); delay(500);
            StabilesteFocar(); delay(500);
            StabilesteObiect(); DeseneazaImaginea();
            sound(300); delay(100); nosound();
            do { MouseData(b,x,y); } while (b==0);
            if (b==2) { closegraph(); exit(1); }
        } while (1);
    }

```



Programul folosește fișierul “mouse.h” pentru lucrul cu mouse-ul în modul grafic. Acest fișier este descris în continuare. Observați asemănările și deosebirile față de fișierul “mouset.h” descris într-un capitol anterior și care se referea la utilizarea mouse-ului în modul text.

```

#include <dos.h>
void MOUSE(int &a1, int &a2, int &a3, int &a4)
{struct REGPACK regs;
  regs.r_ax = a1; regs.r_bx = a2; regs.r_cx = a3; regs.r_dx = a4;
  intr(0x33, &regs); a1 = regs.r_ax; a2 = regs.r_bx;
  a3 = regs.r_cx; a4 = regs.r_dx;
}
void MouseInit(void)
{
    int a=1,b,c,d; MOUSE(a,b,c,d);
}

void MouseHide(void)
{
    int a,b,c,d; a=2; MOUSE(a,b,c,d);
}
void MouseShow(void)
{
    int a=1,b,d,c; MOUSE(a,b,c,d);
}
void MouseData(int& but, int& x, int& y)
{
    int a=3, b, c, d; MOUSE(a,b,c,d);
    but=b; x=c; y=d;
}
void MouseMove(int x, int y)
{
    int a=4, b, c=x, d=y; MOUSE(a,b,c,d);
}

```

}

2. Un joc cu strategie câștigătoare



“**Jocul pătratelor alunecătoare**”, pe care vi-l prezentăm în continuare, poate constitui o problemă interesantă de programare pentru orice elev.

Pe o tablă cu $n \times n$ pătrățele (n întreg impar) sunt așezate, alternativ, piese galbene și roșii, lăsând pătrățelul din mijlocul tablei neocupat (v. fig. A). Cei doi jucători mută, alternativ, ortogonal, câte o piesă proprie alăturată spațiului liber, în spațiul liber. Pierde jucătorul care nu mai poate muta (deci spațiul liber este înconjurat numai de piese ale adversarului, eventual și de marginea tablei).

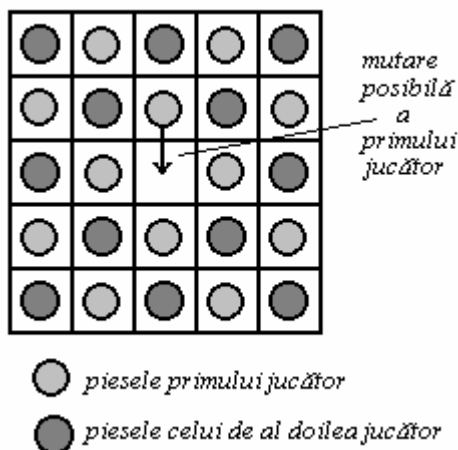


Figura A



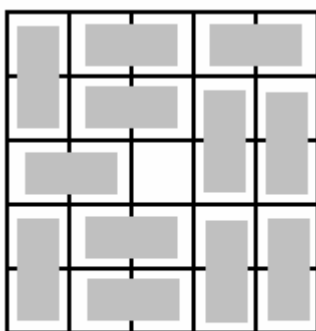
Se poate arăta ușor că al doilea jucător are o strategie sigură de câștig. Astfel, să acoperim tabla de joc cu dominouri (dreptunghiuri formate din două pătrățele elementare), lăsând căsuța din mijloc neacoperită (v. fig. B). Vom observa că primul jucător, după prima mutare, va ocupa centrul și va elibera unul din capetele unui domino.

Astfel, cel de al doilea jucător va putea să mute piesa din celălalt capăt al dominoului. Deci, în orice moment, cel de al doilea jucător poate muta. Partida nu se poate termina remiză, deci primul jucător va pierde.

O metodă generală de aranjare a dominourilor, pe o tablă oarecare $n \times n$, cu n impar, este prezentată în figura C: dominourile se așază în spirală, pornind din colțul stânga-sus și ajungând în centrul tablei.

Această aranjare a dominourilor pe tablă va determina un vector $Opus$, cu semnificația: $Opus[i] = j \Rightarrow$ dacă pătrățelul i este liber, atunci jucătorul al doilea va muta piesa din pătrățelul j .

Pătrățelele sunt numerotate de sus în jos, de la stânga la dreapta. Astfel, căsuța de pe linia i și coloana j va avea numărul $(n-1) \cdot i + j$. De observat că și $Opus[j] = i$.



*Un mod de aranjare a
dominourilor
pentru $n=5$.*

Figura B

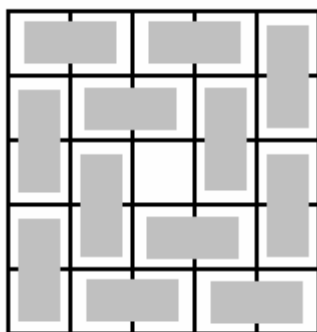


Figura C

Aranjarea în spirală a dominourilor ($n=5$).

Programul următor implementează această strategie de joc pentru calculator, care este întotdeauna al doilea jucător și va câștiga întotdeauna.

```
/* JoculPatratelorAlunecatoare */
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <dos.h>
typedef unsigned char byte;
typedef unsigned int word;
byte Leg[26]={0,6,7,8,5,4,1,2,3,10,9,12,11,
              0,15,14,21,22,19,18,25,16,17,24,23,20};
byte spatiu=0, lat=60, x0=110, y0=50;
byte is, js, Tabla[7][7];
void OpenGraph()
{
    int gd=DETECT,gm; initgraph(&gd,&gm,"c:\\bc\\bgi");
}
byte CasutaLibera()
{
    return (5*(is-1)+js);
}
```

```

}
void InitTabla()
{
    byte i,j; is=3; js=3;
    for (i=1; i<=5; i++)
        for (j=1; j<=5; j++)
            if ((i+j) % 2 != 0) Tabla[i][j]=LIGHTRED;
            else Tabla[i][j]=YELLOW;
    Tabla[is][js]=spatiu;
    for (i=0; i<=6; i++)
        {
            Tabla[0][i]=YELLOW; Tabla[6][i]=YELLOW;
            Tabla[i][0]=YELLOW; Tabla[i][6]=YELLOW;
        }
}

void DesPiesa(int i, int j)
{
    if ((i!=is) || (j!=js))
    {
        setcolor(WHITE);
        rectangle(x0+lat*i, y0+lat*j, x0+lat*i+lat, y0+lat*j+lat);
        setfillstyle(SOLID_FILL,Tabla[i][j]);
        fillellipse(x0+lat*i+lat/2, y0+lat*j+lat/2,lat/4, lat/4);
    }
    else
    {
        setfillstyle(SOLID_FILL,BLACK);
        bar(x0+lat*i, y0+lat*j, x0+lat*i+lat, y0+lat*j+lat);
        setcolor(WHITE);
        rectangle(x0+lat*i,y0+lat*j, x0+lat*i+lat, y0+lat*j+lat);
    }
}

void DesTabla()
{
    byte i,j; setfillstyle(SOLID_FILL,MAGENTA);
    bar(x0+lat+4,y0+lat+4,x0+lat*6+4,y0+lat*6+4);
    for (i=1; i<=5; i++)
        for (j=1; j<=5; j++)
            {
                setfillstyle(SOLID_FILL,0);
                bar(x0+lat*i,y0+lat*j,x0+lat*i+lat,y0+lat*j+lat);
                DesPiesa(i,j);
            }
}

void MutaCalc()
{
    byte isv,jsv,cas; isv=is; jsv=js;
    cas=Leg[CasutaLibera()];
    is=(cas-1)/5+1; js=(cas-1)%5+1;
    Tabla[is][js]=spatiu; DesPiesa(is,js);
    Tabla[isv][jsv]=YELLOW; DesPiesa(isv,jsv);
}

void MutaOm()
{

```

```

int tasta, mut;
do {
    tasta=getch(); if (tasta==0) tasta=getch();
    mut=0;
    switch (tasta) {
        case 75: if (is<5)
            if (Tabla[is+1][js]==LIGHTRED)
            {
                is++; mut=1;
                Tabla[is][js]=spatiu; DesPiesa(is,js);
                Tabla[is-1][js]=LIGHTRED; DesPiesa(is-1,js);
            }
            break;
        case 77: if (is>1)
            if (Tabla[is-1][js]==LIGHTRED)
            {
                is--; mut=1;
                Tabla[is][js]=spatiu; DesPiesa(is,js);
                Tabla[is+1][js]=LIGHTRED; DesPiesa(is+1,js);
            }
            break;
        case 80: if (js>1)
            if (Tabla[is][js-1]==LIGHTRED)
            {
                js--; mut=1;
                Tabla[is][js]=spatiu; DesPiesa(is,js);
                Tabla[is][js+1]=LIGHTRED; DesPiesa(is,js+1);
            }
            break;
        case 72: if (js<5)
            if (Tabla[is][js+1]==LIGHTRED)
            {
                js++; mut=1;
                Tabla[is][js]=spatiu; DesPiesa(is,js);
                Tabla[is][js-1]=LIGHTRED; DesPiesa(is,js-1);
            }
            break;
        case 27: {
            closegraph();
            printf("Ati abandonat...");
            exit(1);
        }
    }
    if (!mut)
        { sound(200); delay(50); nosound();
        }
} while (!mut);
}
int EsteGata()
{
    return ((Tabla[is-1][js]==YELLOW) && (Tabla[is+1][js]==YELLOW)
        && (Tabla[is][js-1]==YELLOW) && Tabla[is][js+1]==YELLOW));
}
void Sunet()
{

```

```

        byte i;
        for (i=1; i<=25; i++) { sound(300+20*i); delay(30); }
        nosound();
    }
    void main()
    {
        OpenGraph(); setbkcolor(BLUE); setfillstyle(9,LIGHTGREEN);
        bar(30,65,getmaxx()-30,getmaxy()-30);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,4);
        settextjustify(CENTER_TEXT,CENTER_TEXT);
        setcolor(CYAN); outtextxy(320,35,"ALUNECATOARELE");
        InitTabla(); DesTabla();
        do {
            MutaOm(); delay(200); MutaCalc();
        } while (!EsteGata());
        Sunet(); getch(); closegraph();
        printf("Iarasi ati pierdut ...");
    }

```

3. Trasarea grafică a bioritmului unei persoane



Specialiști din diferite domenii ale științelor care au ca obiect de studiu omul, printre care și medicina, au ajuns la concluzia că puterea fizică, psihică și intelectuală a oricărei persoane evoluează funcțional sub forma unor sinusoide, ale căror perioade sunt de 23, 28 și 33 de zile, iar începutul sinusoidelor depind de data nașterii; pentru o persoană născută pe 1 ianuarie în anul 1, se consideră că bioritmul său pornește cu cele trei curbe de la 0 (medie), în sens crescător.

Pornind de la ideile de mai sus, prezentăm în continuare un simplu program de trasare grafică a bioritmului pentru o anumită persoană. Programul, scris în *Borland C* (dar se poate adapta ușor în orice versiune de C, care are o interfață grafică pentru programare), cere utilizatorului să introducă data sa de naștere, numele său, precum și data de referință. Bioritmul va fi trasat pe o perioadă de 32 de zile, începând cu data de referință stabilită.

```

// program bioritm
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include <dos.h>
#define pi 3.1415926
void OpenGraph()
{
    int gd=0, gm; initgraph(&gd,&gm,"C:\\\\BC\\\\BGI");
}
int NrZile(int d, int m, int y)
{
    int n;
    const int nn[12]={0,31,59,90,120,151,181,212,243,273,304,334};
    n=nn[m-1]; n=n+365*y+y/4 + d + 1; n=n-y/100 + y/400;
}

```

```

if (!( (y%4!=0) || (y%400==0) || (y%100==0) || (m>2) )) n--;
return n;
}
void Inputt(char *sir, int *nr)
{
    printf("%s",sir); scanf("%d",nr);
}
void main()
{
    int p[3] = {23,28,33}; int q[3];
    int m1,d1,y1,m2,d2,y2,i,n,nz,se,sen; float ur,sr;
    char nume[40];
    clrscr(); fflush(stdin);
    printf("* BIORITM *\n\n");
    printf("Dati numele : ");
    gets(nume);
    printf("Data nasterii:\n");
    Inputt("ziua : ",&d1);
    Inputt("luna : ",&m1);
    Inputt("anul : ",&y1);
    printf("Data de referinta :\n");
    Inputt("ziua : ",&d2);
    Inputt("luna : ",&m2);
    Inputt("anul : ",&y2);
    nz=NrZile(d1,m1,y1); nz=NrZile(d2,m2,y2)-nz;
    OpenGraph();
    outtextxy(100,20,"Bioritm pentru: "); outtextxy(250,20,nume);
    for (i=0; i<3; i++)
    {
        setcolor(9+i); line(10,405+18*i,100,405+18*i);
        if (i==0) outtextxy(100,400+18*i,"fizic");
        else if (i==1) outtextxy(100,400+18*i,"psihic");
        else outtextxy(100,400+18*i,"intelect");
        q[i]=nz % p[i]; ur=2*pi*q[i]/p[i];
        sr=sin(ur); se=(sr+1)*80;
        for (n=nz; n<=nz+31; n++)
        {
            q[i] = n % p[i]; ur = 2*pi*q[i]/p[i];
            sr=sin(ur); sen=(sr+1)*160;
            line(24*(n-nz-1),480-(100+se),24*(n-nz),480-(100+sen));
            se=sen; delay(30);
        }
    }
    setcolor(7); char nr_zi[2];
    for (i=0; i<=31; i++)
    {
        itoa(i, nr_zi, 10); line(20*i,50,20*i,390);
        outtextxy(20*i+2,55,nr_zi); delay(30);
    }
    line(getmaxx(),50,getmaxx(),390); setcolor(15);
    line(0,220,getmaxx(),220); rectangle(0,50,getmaxx(),390);
    getch();
    closegraph();
}

```

Funcția NrZile determină numărul de zile scurse de la 1 ianuarie anul 1, până la o anumită dată. Astfel se poate determina numărul de zile între data de naștere a persoanei și data de referință.



După cum am spus, programul poate fi rescris în orice altă implementare de C și pe orice alt tip de calculator, însă de fiecare dată trebuie să se țină cont de rezoluția ecranului și de orientarea axei O_Y . În cazul nostru, s-a utilizat Borland C++, rezoluția a fost de 640×480 pixeli, iar orientarea axei O_Y a fost de sus în jos. Deși calculatorul nu greșește, e posibil ca oamenii de știință care au inventat bioritmul să fi greșit de la bun început, așa încât rămâne la latitudinea dumneavoastră dacă veți lua în considerare rezultatele furnizate de acest program sau nu, tratând programul ca un simplu amuzament.

4. Folosirea fișierelor *CHR* ale firmei *Borland International*



Formatul fonturilor *CHR*

Programatorul de grafică dornic să realizeze un afișaj pe oblică, folosind seturile de caractere (fonturile) din fișierele *CHR* va trebui să cunoască, înainte de toate, ideea care stă la baza acestor fonturi, precum și formatul fișierelor corespunzătoare.

Fonturile din fișierele *CHR* (numite și "încondeiate" (*stroke fonts*, în engleză)) definesc caracterele ca o secvență de linii (trăsături), în opoziție cu fonturile de tip *bitmap*, care definesc caracterele ca o matrice de puncte.

Avantajul fonturilor încondeiate este acela că ele pot fi scalate la mărimi arbitrare și să-și mențină rezoluția lor. Fonturile de tip *bitmap* sunt făcute pentru o anumită mărime a punctului și nu pot fi scalate prea bine. De exemplu, dacă aveți un font de tip *bitmap* pentru o matrice de 72 puncte pe inch (DPI) și mărim de patru ori în înălțime și în lățime punctele pentru a utiliza o imprimantă laser cu 300 DPI, atunci pot apărea margini prea mari (largi) în fontul 72 DPI.

Pe de altă parte, fonturile de tip *stroke*, nu sunt convertite în puncte, pînă cînd rezoluția perifericului de ieșire nu este cunoscută. Astfel, dacă vreți să scrieți cu fonturi de tip *stroke* pe o imprimantă laser cu 300 DPI, punctele care trasează liniile caracterelor vor fi tipărite la 300 DPI.

Acestea, fiind spuse, să trecem la prezentarea structurii unui fișier de tip *CHR*, exemplificînd pe fișierul "TRIP.CHR", care este același pentru toate mediile de programare *Borland*.

; offset 0h este un header specific firmei Borland:

HeaderSize	de ex.	080h	
DataSize	de ex.	(mărimea fișierului)	; lungimea fiș. CHR
descr	de ex.	"Triplex font"	; descriere
fname	de ex.	"TRIP"	; numele fișierului
MajorVersion	de ex.	1	; versiunea fontului
MinorVersion	de ex.	0	; subversiunea
db	'PK',8,8		


```

db  'BGI ',descr,' V'
db  MajorVersion+'0'
db  (MinorVersion / 10)+'0',(MinorVersion mod 10)+'0'
db  ' - 19 October 1987',0DH,0AH
db  'Copyright (c) 1987 Borland International', 0dh,0ah      ; (c)
db  0,1ah                                           ; null & ctrl-Z = end
dw  HeaderSize                                     ; mărimea header-ului
db  fname                                           ; numele fontului
dw  DataSize                                         ; mărimea fiș. font
db  MajorVersion,MinorVersion                       ; numărul de versiune
db  1,0                                              ; nre. de versiune min.
db  (HeaderSize - $) DUP (0)                       ; tampon

```

La offset-ul 80h încep datele pentru fișier, după cum urmează:

```

;      80h  '+'   indicatoare pentru tipul fișierului stroke
;      81h-82h  numărul de caractere în fișierul font (n)
;      83h      nedefinit
;      84h      valoarea ASCII a primului caracter din fișier
;      85h-86h  offset pt. definițiile stroke (8+3n)
;      87h      scan flag (normal 0)
;      88h      distanța de la origine la vârful literei mari
;      89h      distanța de la origine la linia de bază
;      90h      distanța de la origine la cea mai coborâtă linie
;      91h-95h  nedefinit
;      96h      offset-uri pentru definiții individuale de caractere
;      96h+2n   tabela grosimilor (un word pentru fiecare caracter)
;      96h+3n   startul definițiilor caracter

```

Definițiile individuale de caractere consistă dintr-un număr variabil de cuvinte (word) (16 bits), descriind operația necesară în desenarea caracterului. Fiecare word consistă dintr-o perche de coordonate (x,y) și două coduri de operații pe doi biți. Cuvintele sunt codificate după cum urmează:

```

Byte 1      7 6 5 4 3 2 1 0  bit #
            op1 <coord. X pe 7 biti cu semn>
Byte 2      7 6 5 4 3 2 1 0  bit #
            op2 <coord. Y pe 7 biti cu semn>

```

Codurile de operații sunt:

```

op1=0 op2=0 Sfârșitul definiției de caracter.
op1=1 op2=0 Mută pointerul grafic în punctul de coordonate (x,y).
op1=1 op2=1 Trasează o linie de la punctul curent la punctul de
coordonate (x,y).

```

Descrierea în limbajul C a structurii unui font CHR este dată mai jos:

/*

FONT.H - informațiile din header-ul unui fișier de fonturi CHR

Copyright (c) 1988,1989 Borland International

*/

```
#define Prefix_Size      0x80
#define Major_Version    1
#define Minor_Version    0
#define SIGNATURE '+'
enum OP_CODES {
    END_OF_CHAR = 0, DO_SCAN      = 1, MOVE = 2, DRAW = 3
};
typedef struct {
    char sig;           /* SIGNATURE byte */
    int nrchrs;         /* numarul de caractere in fisier */
    char mystery;       /* nedefinit inca */
    char first;         /* primul caracter din fisier */
    int cdefs;          /* offset la definitiile de caractere */
    char scan_flag;     /* True daca setul de caractere este scanabil */
    char org_to_cap;    /* inaltimea de la origine la vârful literei mari */
    char org_to_base;   /* inaltimea de la origine la linia de baza */
    char org_to_dec;    /* inaltimea de la origine la linia inferioara */
    char fntname[4];    /* patru caractere pentru numele fontului */
    char unused;        /* nedefinit inca */
} HEADER;
typedef struct {
    char opcode;        /* Byte pentru codul de operatie */
    int x;              /* Offset relativ pe directia x */
    int y;              /* Offset relativ pe directia y */
} STROKE;
typedef struct {
    unsigned int header_size; /* Versiunea 2.0 a formatului de header */
    unsigned char font_name[4]; /* Numele intern al fontului */
    unsigned int font_size; /* Marimea in bytes a fisierului */
    unsigned char font_major, font_minor; /* Info. de versiune a driverului */
    unsigned char min_major, min_minor; /* Informatii de revizie pentru BGI */
} FHEADER;
```

Rutine speciale de afișare. Pornind de la informațiile prezentate anterior, se pot dezvolta proceduri de încărcare a fonturilor și de afișaj în orice direcție.

// Program pentru afisari speciale

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <graphics.h>
```

```

#include <conio.h>
#include <math.h>
#include <string.h>
#include "font.h"
FILE *ffile;
char          *Font;
char          Prefix[Prefix_Size];
HEADER        Header;
int           Offset[256];
char          Char_Width[256];
int POZX = 25, POZY = 25, i;
int decode( unsigned int *iptr, int *x, int *y );
/*      Aceasta functie decodifica formatul fisierului,
        punind-o intr-o structura interna mai convenabila      */
int unpack( char *buf, int index, STROKE **neww )
{
    unsigned int *pb;
    STROKE *po;
    int num_ops = 0;
    int jx, jy, opcode, i, opc;
    pb = (unsigned int *) (buf + index);
    while( FOREVER ){
        num_ops += 1;
        opcode = decode( pb++, &jx, &jy );
        if( opcode == END_OF_CHAR ) break;
    }
    po = (*neww = (STROKE*)calloc( num_ops, sizeof(STROKE) ));
    if( !po ) exit(100)
    }
    pb = (unsigned int *) (buf + index);
    for( i=0 ; i<num_ops ; ++i ){
        opc = decode( pb++, &po->x, &po->y );
        po->opcode = opc; po++;
    }
    return( num_ops );
}

```

Funcția următoare decodifică un singur cuvânt din fișier punându-l într-o structură internă mai convenabilă, de tip "stroke":

```

int decode( unsigned int *iptr, int *x, int *y )
{
    struct DECODE {
        signed   int xoff   : 7;
        unsigned int flag1  : 1;
        signed   int yoff   : 7;
        unsigned int flag2  : 1;
    } cword;
    cword = *(struct DECODE *)iptr;
    *x = cword.xoff;
    *y = cword.yoff;
    return( (cword.flag1 << 1) + cword.flag2 );
}

```

Pentru afișarea unui caracter litera, la poziția de coordonate (x1,y1) față de punctul (x0,y0), rotit sub unghiul alfa se poate folosi funcția:

```

void WriteChar(char litera, float alfa, int x0, int y0, int xl,
int yl)
{
    STROKE *sptr;
    int j, i = litera;
    int xx, yy, xxr, yyr;
    int Caracter = unpack( Font, Offset[i], &sptr );
    y0 = getmaxy() - y0; yl = getmaxy() - yl;
    for( j=0 ; j<Caracter ; ++j, ++sptr ){
        xx = xl+sptr->x; yy = yl+sptr->y;
        xxr = (xx-x0)*cos(alfa) - (yy-y0)*sin(alfa) + x0;
        yyr = (xx-x0)*sin(alfa) + (yy-y0)*cos(alfa) + y0;
        if (sptr->opcode == 2) moveto(xxr, getmaxy()-yyr);
        if (sptr->opcode == 3) lineto(xxr, getmaxy()-yyr);
    }
}

```

Apelând de mai multe ori funcția WriteChar obținem funcția următoare, ce permite afișarea unui șir de caractere cuvint sub unghiul alfa, în punctul de coordonate (xl,yl), rotit față de punctul de coordonate (x0,y0):

```

void WriteStr(char * cuvint, float alfa, int x0, int y0, int xl,
int yl)
{
    POZX = xl; POZY = yl;
    for (int i=0; i < strlen(cuvint); i++)
    {
        WriteChar(cuvint[i], alfa, x0, y0, POZX, POZY);
        POZX += Char_Width[cuvint[i]] * cos(alfa);
        POZY -= Char_Width[cuvint[i]] * sin(alfa);
        x0    += Char_Width[cuvint[i]] * cos(alfa);
        y0    -= Char_Width[cuvint[i]] * sin(alfa);
    }
}

```

Funcția de mai jos scrie un șir de caractere pe un cerc de rază și coordonate ale centrului date:

```

void CircleStr(char * cuvint, int x0, int y0, int raza)
{
    float alfa; int l = strlen(cuvint); int xx, yy;
    y0 = getmaxy() - y0;
    for (int i=0; i<l; i++)
    {
        alfa = -2*M_PI*i/l;
        xx = x0+raza * cos(alfa);
        yy = getmaxy()-y0+raza * sin(alfa);
        WriteChar(cuvint[i], M_PI/2-alfa, xx, yy, xx, yy);
    }
}

```

Se pot scrie cuvinte sub forma unei sinusoide, folosind funcția:

```

void SinWrite(char * cuvint, int x, int y, int raza)
{
    float alfa; int l = strlen(cuvint);
    for (int i=0; i<l; i++)
    {
        alfa = -4*M_PI*i/l;
        y = y + raza * sin(alfa);
    }
}

```

```

        WriteChar(cuvint[i], 0, x, y, x, y);
        x += Char_Width[cuvint[i]];
    }
}

```

Iată un exemplu de utilizare a funcțiilor descrise mai sus:

```

void main()
{
    long length, current;
    char *cptr;
    STROKE *sptr;
    ffile = fopen( "scri.chr", "rb" );
    if( NULL == ffile ){
        exit( 1 );
    }
    fread(Prefix, Prefix_Size, 1, ffile);
    cptr = Prefix;
    while( 0x1a != *cptr ) ++cptr;
    *cptr = '\0';
    fread(&Header, sizeof(HEADER), 1, ffile);
    fread( &Offset[Header.first], Header.nchrs,
        sizeof(int), ffile );
    fread( &Char_Width[Header.first], Header.nchrs,
        sizeof(char), ffile );
    current = ftell( ffile );
    fseek( ffile, 0, SEEK_END );
    length = ftell( ffile );
    fseek( ffile, current, SEEK_SET );
    Font = (char *) malloc( (int) length );
    if( NULL == Font )
    {
        fprintf( stderr, "Memorie insuficienta.\n\n" );
        exit( 1 );
    }
    fread( Font, (int)length, 1 , ffile );
    fclose(ffile);
    int gd, gm;
    gd = 0; initgraph(&gd, &gm, "c:\\bc\\bgi");
    rectangle(0,0,getmaxx(),getmaxy());
    for (i=-3; i<3; i++)
    {
        setcolor(i);
        if (!i) setcolor(CYAN);
        WriteStr("      Try the", -M_PI*i/3, 320, 200, 320, 200);
    }
    setcolor(WHITE);
    CircleStr("HICS ! * BEST GRAP", 320, 200, 80);
    setcolor(LIGHTCYAN);
    SinWrite("Author Bogdan Patrut, tel. 034/123175", 10, 470, 10);
    getch(); closegraph();
}

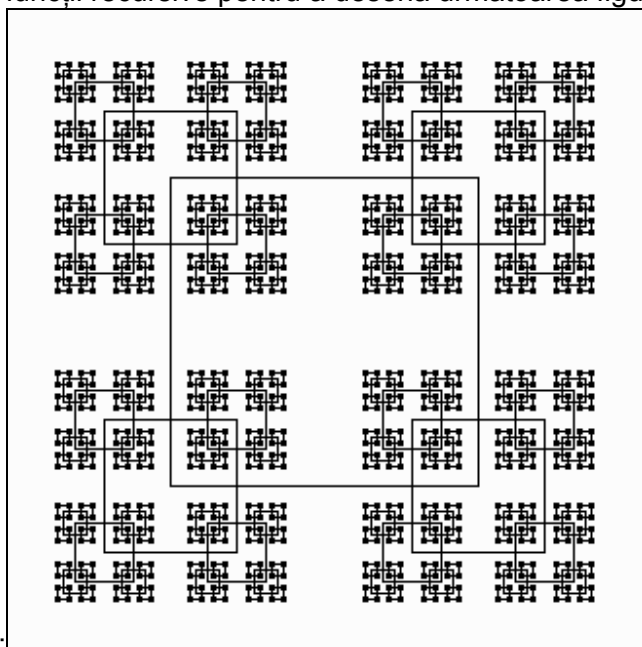
```

Probleme propuse

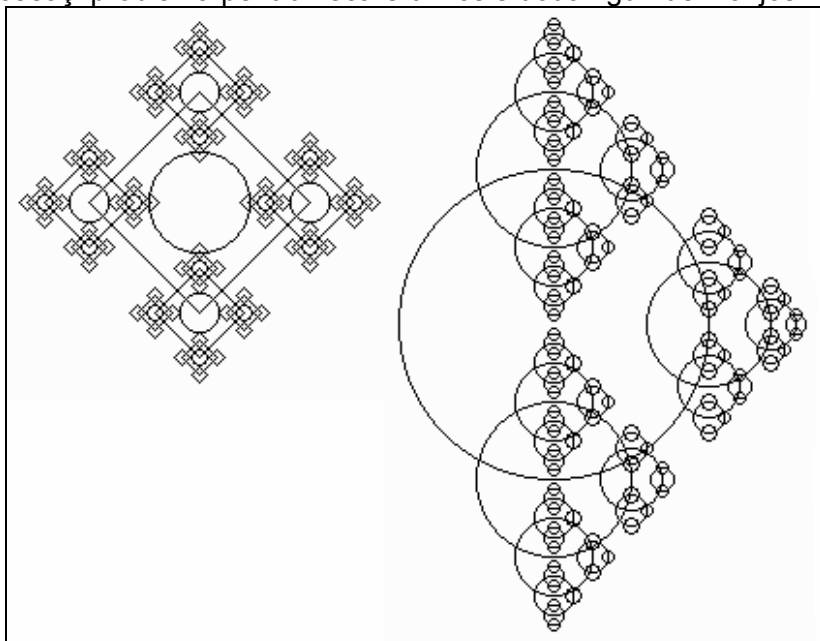
1. Scrieți un program (care utilizează mouse-ul) pentru simularea grafică a formării imaginilor în lentilele divergente.
2. Scrieți un program (care utilizează mouse-ul) pentru simularea grafică a formării imaginilor în oglinzile convergente.
3. Scrieți un program (care utilizează mouse-ul) pentru simularea grafică a formării imaginilor în oglinzile divergente.
4. Realizați o implementare a jocului pătratelor alunecătoare în care numărul (impar) n de pătrate ale tablei să fie dat de la tastatură, generând, astfel, așezarea imaginară a pieselor de domino în spirală.
5. Scrieți un program care să deseneze n dreptunghiuri concentrice, în mijlocul ecranului.
6. Scriți o procedură care să deseneze un poligon regulat cu n laturi.
7. Scrieți o procedură care să deseneze o stea cu n colțuri, folosind două poligoane regulate cu n laturi, imaginare, concentrice și de "raze" inegale.
8. Scrieți un program care să rezolve ecuația de gradul II. Intrările și ieșirile se vor face în mod grafic.
9. Scrieți funcții pentru introducerea, respectiv afișarea de date numerice și alfanumerice în mod grafic.
10. Rescrieți aplicațiile complexe din capitulul 5, astfel încât intrările și ieșirile de la/pe ecran să se realizeze în mod grafic.
11. Dintr-un fișier cu tip se citesc numele unor elevi, precum și notele acestora la 14 discipline. Să se realizeze un program în modul grafic, care să permită următoarele reprezentări grafice:
 - pentru fiecare elev, histograma notelor sale;
 - pentru toți elevii, histograma mediilor lor.
12. Scrieți un program care să reprezinte grafic, pe tot ecranul, funcția $\sin(x)$, pentru x variind între $-\pi$ și π .
13. Realizați un program care să permită reprezentarea bioritmului a două persoane în două zone de ecran, care să poată fi redimensionate și mutate cu ajutorul mouse-ului.
14. Aceeași problemă ca mai înainte, cu posibilitatea de a folosi tastatura în loc de mouse.
15. Realizați un program / o funcție pentru reprezentarea grafică a unei funcții.
16. Scrieți un interpretor pentru expresii de funcții, care să recunoască simbolul x , numere, operatorii $+$, $-$, $*$ și $/$ și funcțiile elementare \sin , \cos , \ln , \exp , $\sqrt{}$ etc.. Astfel, dacă o expresie este corectă, să reprezinte grafic funcția respectivă pe un interval real precizat.
17. Să se realizeze un program care să reprezinte grafic patru funcții în patru zone diferite ale ecranului.
18. Aceeași problemă ca și la problema anterioară, dar cele patru zone să poată fi mutate cu ajutorul mouse-ului.
19. Realizați un program de desenare de tip *PaintBrush*. Se vor scrie proceduri pentru selectarea de culori, pentru trasarea de elipse, linii și dreptunghiuri. De asemenea, se vor scrie proceduri pentru salvarea și

restaurarea imaginilor. Desenarea figurilor geometrice, precum și accesarea opțiunilor din meniu se va realiza folosind mouse-ul.

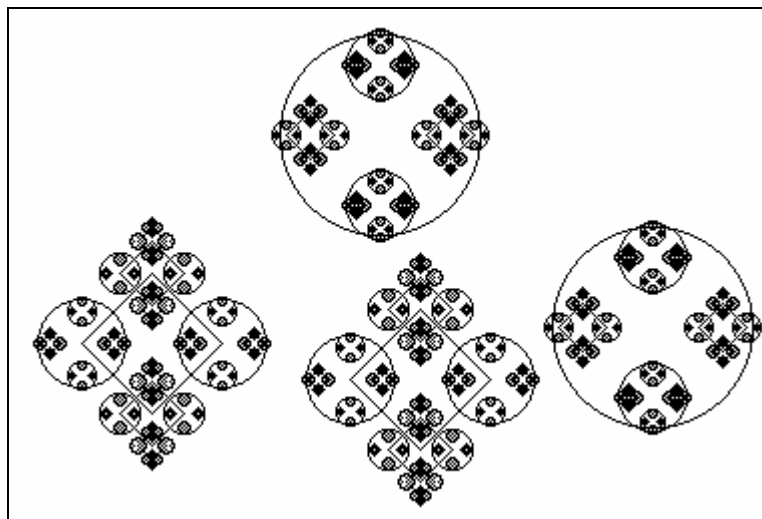
20. Realizați funcții recursive pentru a desena următoarea figură "recursivă"



21. Aceeași problemă pentru fiecare din cele două figuri de mai jos:



22. Folosind mouse-ul și funcții recursive, realizați un program care, prin acționarea celor două butoane de mouse să amplaseze diferite figuri recursive pe ecran, ca în figura următoare:



23. Scrieți o funcție recursivă care să umple o zonă de ecran mărginită de o anumită curbă colorată.
24. Scrieți un program care să traseze un cerc, folosind ecuațiile sale parametrice.
25. Scrieți un algoritm pentru trasarea unui cerc, care să utilizeze doar adunări și scăderi.
26. Aceeași problemă pentru trasarea unei elipse.
27. Scrieți un algoritm pentru trasarea unui segment de dreaptă între două puncte ale căror coordonate ecran se cunosc.
28. Realizați jocul "turnurilor din hanoi" cu animație în mod grafic.
29. Realizați o implementare grafică a problemei damelor.
30. Realizați o implementare grafică a problemei colorării grafurilor.

Capitolul 8.

Programare orientată pe obiecte în *Borland C++*

- definirea claselor • constructori și destructori • metode •
- funcții virtuale (pure) • moștenire •

Probleme rezolvate

1. Un joc de animație, orientat pe obiecte



Orientat obiect, programul pe care îl prezentăm și îl comentăm în continuare realizează o animație a unor obiecte grafice de diferite tipuri și forme. Este vorba despre un joc în care trebuie să prindeți mai mulți fluturi (numărul lor este introdus de la tastatură, la începutul jocului), cu ajutorul unui omuleț care are într-o mână o plasă de prins fluturi. Totul se desfășoară într-un cadru natural, reprezentat de un soare care strălucește și doi nori care se deplasează pe cer. Fluturii zboară aleator, iar omul care îi prinde este deplasat de către jucător. Există două variante ale jocului, una în care deplasarea se face cu ajutorul tastelor de cursor și alta în care deplasarea se realizează cu ajutorul mouse-ului.

Ideea de bază în realizarea programului este relativ simplă. Se definește o clasă abstractă *Obiect*, din care apoi se derivează clasele corespunzătoare tuturor obiectelor grafice care apar pe ecran. Clasa abstractă este caracterizată de două câmpuri care reprezintă coordonatele obiectului, plus încă două reprezentând deplasările pe orizontală și pe verticală. De asemenea există trei metode pure, implementate la nivelul claselor derivate, concrete: una pentru deplasarea unui obiect, care apelează altele două: una care șterge obiectul (din poziția veche) și alta care îl redesenează (în poziția nouă).

O imagine din timpul desfășurării acestui joc și listingul comentat al programului vă vor documenta mai bine în legătură cu algoritmi și bazele de date folosite.

```
// Program C++ pentru prezentarea animatiei unor obiecte grafice
// de tipuri (clase) diferite, derivate dintr-o clasa abstracta
// si cu rescrierea metodelor virtuale pure.
```

```
#include <iostream.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
// definirea clasei abstracte Obiect
class Obiect {
// x si y = coordonatele obiectului
// dx si dy = deplasările (-1,0,1)
```

```

// aceste 4 cimpuri vor fi recunoscute de urmasi (clasele
derivate)
    protected:
        int x, y, dx, dy;
// metode publice
    public:
// constructori
        Obiect() { };
        Obiect(int x0, int y0);
// metode de desenare, mutare si stergere a obiectelor
// sint pure si virtuale, deci vor fi definite in cadrul
claselor derivate
        virtual void Deseneaza()=0;
        virtual void Muta()=0;
        virtual void Sterge()=0;
};
// clasa Soare este derivata din clasa Obiect, in mod public
class Soare: public Obiect {
    private:
// un soare se poate afla in doua ipostaze,
// in functie de cum sint dispuse razele
        int ipostaza;
    public:
// constructorul clasei Soare va primi coordonatele initiale
// precum si ipostaza initiala a soarelui
        Soare(int x0, int y0, int ipostaza0);
// cele trei metode vor fi redefinite
        void Deseneaza();
        void Muta();
        void Sterge();
};

// derivare similara si in cazul clasei Nor
class Nor: public Obiect {
    public:
        Nor(int x0, int y0);
        void Deseneaza();
        void Muta();
        void Sterge();
};
// ... si in cazul clasei Fluture
class Fluture: public Obiect {
    public:
// un fluture poate fi vizibil (cind zboara)
// sau invizibil, dupa ce a fost capturat
        int vizibil;
        Fluture(); // constructorul, fara parametri
        int GetX(); // functie ce returneaza coordonata x a
fluturelui
        int GetY(); // returneaza coordonata y
        void Deseneaza();
        void Muta();
        void Sterge();
} F[10]; // am declarat un vector F de maxim 10 fluturi
int NrInitFluturi;

```

```

// numarul initial de fluturi, care va fi dat de la tastatura
int NrFluturi;
// numarul curent de fluturi,
// care scade de fiecare data cind se prinde un fluture
// clasa Om, derivata public din clasa Obiect
class Om: public Obiect {
public:
    Om(int x0, int y0);
    int GetPX(); // coordonata x a paletii de prins fluturi
    int GetPY(); // coordonata y a paletii
    void Deseneaza();
    void Muta();
    void Sterge();
};
// clasa abstracta Obiect: definire constructor
Obiect::Obiect(int x0, int y0)
{ x=x0; y=y0; }

// clasa Soare: definire metode
// constructorul
Soare::Soare(int x0, int y0, int ipostaza0)
{
    x=x0; y=y0;
    ipostaza=ipostaza0;
}
// functia de desenare
void Soare::Deseneaza()
{
    circle(x,y,20); // un cerc de raza 20 pixeli
    // daca se afla in ipostaza 1, atunci se deseneaza
    // 4 raze, verticale si orizontale
    if (ipostaza==1)
    {
        line(x-40,y,x-20,y);
        line(x+20,y,x+40,y);
        line(x,y-40,x,y-20);
        line(x,y+20,x,y+40);
    }
    else // altfel, in ipostaza 2, se deseneaza raze oblice
    {
        line(x-14,y-14,x-34,y-34);
        line(x-14,y+14,x-34,y+34);
        line(x+14,y-14,x+34,y-34);
        line(x+14,y+14,x+34,y+34);
    }
}
// stergerea soarelui - este, de fapt, o redesenare cu negru
void Soare::Sterge()
{
    setcolor(BLACK);
    Deseneaza();
    setcolor(WHITE);
}
// functia de mutare a soarelui;

```

```

// nu este o mutare propriu-zisa, ci doar o reafisare in noua
ipostaza
void Soare::Muta()
{
    Sterge();
    ipostaza=3-ipostaza; // schimbarea ipostazei 1 <-> 2
    Deseneaza();
}
// constructorul clasei Nor; deplasarea initiala este spre
stinga
Nor::Nor(int x0, int y0)
{ x=x0; y=y0; dx=-1; }
// desenarea unui nor = o elipsa cu centrul in (x,y)
void Nor::Deseneaza()
{
    ellipse(x,y,0,360,70,30);
}
// functia de stergere a unui obiect din clasa Nor
void Nor::Sterge()
{
    setcolor(BLACK); Deseneaza(); setcolor(WHITE);
}
// functia de mutare a unui obiect din clasa Nor
// daca obiectul ajunge cu centrul (x,y) in capete,
// se schimba sensul deplasarii pe orizontala
void Nor::Muta()
{
    Sterge();
    x += 10*dx;
    if (x < 0) { x = 0; dx = -dx; }
    if (x > getmaxx()) { x = getmaxx(); dx = -dx; }
    Deseneaza();
}
// constructorul clasei Fluture initializeaza automat x si y,
// intr-o zona dreptunghiulara a ecranului;
// dx si dy se initializeaza cu una din valorile -1,0 si 1
// la inceput fluturele este vizibil
Fluture::Fluture()
{
    x = 300 + random(200); y = 300 + random(100);
    dx = random(3)-1; dy = random(3)-1; vizibil = 1;
}
// metoda de desenare a unui obiect din clasa Fluture
void Fluture::Deseneaza()
{
    // aripa din stinga
    line(x,y,x-5,y-5); line(x,y,x-5,y+5);
    line(x-5,y-5,x-5,y+5);
    // aripa din dreapta
    line(x,y,x+5,y-5); line(x,y,x+5,y+5);
    line(x+5,y-5,x+5,y+5);
    // corpul
    line(x,y-10,x,y+7);
    // capul
    circle(x,y-10,2);
    // antenele

```

```

        line(x,y-10,x-3,y-13); line(x,y-10,x+3,y-13);
    }
    // metoda de stergere a unui Fluture
    void Fluture::Sterge()
    {
        setcolor(BLACK); Deseneaza(); setcolor(WHITE);
    }
    // mutarea unui Fluture se realizeaza doar daca el este vizibil
    void Fluture::Muta()
    {
        if (vizibil)
        {
            Sterge();
            // se genereaza noile deplasari
            dx = random(3)-1; dy = random(3)-1;
            // se actualizeaza coordonatele x si y
            x += 10*dx; y += 10*dy;
            // fluturile nu poate depasi o anumita zona dreptunghiulara
            if (x < 0) x = 0;
            if (y < 180) y = 180;
            if (x > getmaxx()) x = getmaxx();
            if (y > 380) y = 380;
            Deseneaza();
        }
    }
    // urmeaza functiile care returneaza coordonatele unui Fluture
    int Fluture::GetX()
    { return x; }
    int Fluture::GetY()
    { return y; }
    // clasa Om: definire metode
    // constructor
    Om::Om(int x0, int y0)
    { x=x0; y=y0; }
    // functie de desenare
    void Om::Deseneaza()
    {
        circle(x,y,10); // cap
        line(x,y+10,x,y+40); line(x-20,y+15,x+20,y+15);
        line(x+20,y+15,x+20,y-15);
        circle(x+20,y-25,10); // paleta
        line(x,y+40,x-10,y+80); line(x,y+40,x+10,y+80);
    }
    // functiile care returneaza coordonatele paletei omului
    int Om::GetPX()
    { return x+20; }
    int Om::GetPY()
    { return y-25; }
    // stergerea unui om, in vederea mutarii sale in alta parte
    void Om::Sterge()
    {
        setcolor(BLACK); Deseneaza(); setcolor(WHITE);
    }
    // functie care stabileste daca fluturile F este prins
    // de paleta omului
    int PrindeFluture(Om O, Fluture F)

```

```

{
    int eps = 7;
    // precizia de prindere in paleta a fluturului
    return ((abs(F.GetX()-O.GetPX()) < eps) &&
        (abs(F.GetY()-O.GetPY()) < eps) && (F.vizibil == 1));
}
// metoda care muta un obiect din clasa Om
void Om::Muta()
{
    int tasta; // codul tastei apasata
    int i;      // i = indice de fluturi in vectorul F
    int gasit;
    // gasit = 1 daca s-a prins fluturile F[i] in paleta
    dx = 0; dy = 0; // valori de deplasare a omului implicit 0
    if (kbhit()) // daca se apasa o tasta
        { tasta = getch(); // se citește codul ei
          if (tasta == 0) tasta = getch();
        }
    // daca acesta este 0,
    // atunci se mai citește o data codul; acest lucru se
    // folosește pentru codurile tastelor de cursor
    switch(tasta)
    // se modifica dx si dy, in functie de tasta
        { case 72: dy = -1; break; // cursor sus
          case 80: dy = +1; break; // cursor jos
          case 75: dx = -1; break; // cursor stinga
          case 77: dx = +1; break; // cursor dreapta
        }
    Sterge(); // se sterge omul din pozitia veche
    x += 10*dx; // se actualizeaza coordonatele sale
    y += 10*dy;
    i = 0; gasit = 0; // se cauta secvential primul
    // fluturi F[i], prins in paleta
    while ((i < NrInitFluturi) && (! gasit))
        {
            if (PrindeFluturi(*this,F[i]))
                // daca s-a prins F[i] in paleta, atunci...
                { F[i].Sterge(); // se sterge
                  F[i].vizibil = 0; // devine invizibil
                  gasit = 1; // se va termina ciclul while
                  NrFluturi--; // nr de fluturi scade cu 1
                }
            else i++; //daca nu, trecem la alt fluturi
        }
    Deseneaza(); // se deseneaza omul in noua pozitie
}
}
// programul principal
void main()
{
    cout << "Dati numarul de fluturi : ";
    // se afiseaza acest text
    cin >> NrInitFluturi; // si se citește aceasta variabila
    // s-au folosit obiectele standard cout si cin din iostream.h
    /* precum si operatorii << si >>, redefiniti pentru aceste
    obiecte */
    NrFluturi = NrInitFluturi; // initializarea nr. de fluturi
    int i, gm, gd = DETECT; // initializarea grafica

```

```

initgraph(&gd,&gm,"c:\\borlandc\\bgi");
line(0,200,getmaxx(),200); // linia orizontului
for (i=0; i<1000; i++) // cimpia
    putpixel(random(getmaxx()),200+random(getmaxy()-200),15);
Nor N1(100,50); N1.Deseneaza(); // cei doi nori
Nor N2(400,80); N2.Deseneaza();
Soare S(550,60,1); S.Deseneaza(); // soarele
Om O(200,300); O.Deseneaza(); // omul
for (i=0; i<NrInitFluturi; i++) // fluturii
    F[i].Deseneaza();
// urmeaza jocul propriu-zis:
do
{
    N1.Muta(); // mutarea celor doi nori
    N2.Muta();
    S.Muta(); // rotirea soarelui
    O.Muta(); // mutarea omului, cu taste
    line(0,200,getmaxx(),200);
    // redesenarea orizontului
    for (i=0; i<NrInitFluturi; i++)
        F[i].Muta(); //se muta toti fluturii, pe rind
    if (kbhit())//se forta terminarea jocului
        { if (getch() == char(27)) // daca se apasa
            { closegraph(); exit(1); } // Escape
        }
}
while (NrFluturi > 0);
// jocul se termina cind se termina fluturii
closegraph(); // se iese din modul grafic
}

```

2. Se știe că în calculator, orice număr real este reprezentat printr-un număr rațional, deoarece nu se pot reprezenta decât un număr finit de zecimale ale oricărui număr. Prin urmare, fiecare număr rațional corespunde mai multor valori reale, dintr-un anumit interval care reprezintă o vecinătate pentru un număr rațional. Acestea considerente ne conduc la ideea realizării unei clase interval și a operațiilor necesare operării cu ele, ca în exemplul următor:



```

// * Implementarea clasei INTERVAL si redefinirea operatorilor
// + , - , * si / pentru aceasta clasa *
#include <iostream.h>
#include <conio.h>
class interval
{
public :
    float inf; float sup;
    interval(float a, float b) { inf=a, sup=b;};
    friend interval operator+(interval, interval);
    friend interval operator-(interval, interval);
    friend interval operator*(interval, interval);

```

```

        friend interval operator/(interval, interval);
};
interval operator+(interval interv_1, interval interv_2)
{
    return interval(interv_1.inf+interv_2.inf,
                    interv_1.sup+interv_2.sup);
}
interval operator-(interval interv_1, interval interv_2)
{
    return interval(interv_1.inf-interv_2.sup,
                    interv_1.sup-interv_2.inf);
}

interval operator*(interval interv_1, interval interv_2)
{
    float p[4], min , max;
    p[0] = interv_1.inf * interv_2.inf;
    p[1] = interv_1.inf * interv_2.sup;
    p[2] = interv_1.sup * interv_2.inf;
    p[3] = interv_1.sup * interv_2.sup;
    min = p[0]; max = min;
    for (int i=0; i<=3; i++)
        { if (p[i] < min) min = p[i];
          if (p[i] > max) max = p[i];
        }
    return interval (min, max);
}
interval operator/(interval interv_1, interval interv_2)
{
    float p[4], min , max;
    // * Eroare : impartire prin zero ! *
    if (interv_2.inf <= 0 && interv_2.sup >=0)
        { return interval(0, 0) ;}
    else
    {
        p[0] = interv_1.inf / interv_2.inf;
        p[1] = interv_1.inf / interv_2.sup;
        p[2] = interv_1.sup / interv_2.inf;
        p[3] = interv_1.sup / interv_2.sup;
        min = p[0]; max = min;
        for (int i=0; i<=3; i++)
            { if (p[i] < min) min = p[i];
              if (p[i] > max) max = p[i];
            }
        return interval (min, max);
    }
}

void main(void)
{
    float a, b, c, d;
    char op;
    clrscr();
    cout<<" Dati intervalul 1 ! \n";
    cout<<" - inf : ";
    cin>>a;

```



```

cout<<" - sup : ";
cin>>b;
interval interv_1(a,b);
cout<<" Dati intervalul 2 ! \n";
cout<<" - inf : ";
cin>>c;
cout<<" - sup : ";
cin>>d;
interval interv_2(c,d);
interval int_sum = interv_1 + interv_2;
interval int_dif = interv_1 - interv_2;
interval int_prd = interv_1 * interv_2;
interval int_cit = interv_1 / interv_2;
cout<<" Suma : [ "<<int_sum.inf<<" , "<<int_sum.sup<<" ] \n";
cout<<" Diferenta : [ "<<int_dif.inf<<" , "<<
    int_dif.sup<<" ] \n";
cout<<" Produsul : [ "<<int_prd.inf<<" , "<<
    int_prd.sup<<" ] \n";
cout<<" Citul : [ "<<int_cit.inf<<" , "<<int_cit.sup<<" ] \n";
getch();
}

```

3. Să se definească mai multe clase, derivate dintr-o clasă abstractă și să creeze o listă eterogenă, cuprinzând obiecte de diferite clase.

Un exemplu este dat în programul următor. Țiruri de caracter, numere, mașini și persoane se află împreună în lista eterogenă p, pentru care metoda vizualizeaza se comportă diferit de la un caz la altul. Inițializarea datelor obiectelor din lista eterogenă se face cu ajutorul metodei set. Programul pune în evidență și utilizarea funcțiilor cu argumente implicite.

```

/* LISTA ETEROGENA */
#include <iostream.h>
#include <string.h>
#include <conio.h>
class ORICE {                                     // * CLASA ABSTRACTA
*
    public :
        virtual void vizualizeaza(void)=0;      // * VIRTUALIZARE SI
                                                // * FUNCTIE PURA *
};
class NR_REAL : public ORICE {    // * MOSTENIRE *
    float valoare;
    public :
        // * INITIALIZARE FOLOSIND
        NR_REAL(float a) { valoare = a; }      // UN CONSTRUCTOR *
        void vizualizeaza(void) // * REDEFINIRE METODA *
        {
            cout<<"NR_REAL cu valoarea = "<<valoare<<"\n";
        }
        // * TIPARIRE CU "COUT"
};
class NR_INTREG : public ORICE {
    int valoare;
    public :
        NR_INTREG(int a) { valoare = a; }
        void vizualizeaza(void)
        {
            cout<<"NR_INTREG cu valoarea = "<<valoare<<"\n";
        }
}

```

```

};
class SIR : public ORICE {
    char* continut;
public :
    SIR(char* a)
    { continut=new char[10]; // * ALOCARE DINAMICA DE MEMORIE *
      strcpy(continut,a); }
    void vizualizeaza(void)
    {
        cout<<"SIR avind continutul = "<<continut<<"\n";
    }
};
class PUNCT : public ORICE {
    int x,y;
public :
    PUNCT(int a, int b) { x = a; y = b; }
    void vizualizeaza(void)
    {
        cout<<"PUNCT cu abscisa = "<<x<<" si ordonata = "<<y<<"\n";
    }
};
class MASINA : public ORICE {
    char* marca;
    char* tara;
    int viteza;
public : // * INITIALIZARE PRIN METODA *
    void denumeste(char* m="Dacia 1300",
                   char* t="Romania", int v=170)
    { marca = new char[10];
      strcpy(marca,m); tara = new char[10];
      strcpy(tara,t); viteza=v ;}
    void vizualizeaza(void)
    {
        cout<<"MASINA cu marca "<<marca<<
          " produsa in "<<tara<<
          ", cu viteza maxima = "<<viteza<<" km/h\n";
    }
};
class PERSOANA : public ORICE {
    protected :
        char* nume;
        char* prenume;
        char sex;
        int virsta;
    public : // * FUNCTIE CU PARAMETRI IMPLICITI *
        void denumeste(char* p="Bogdan",char* n="Patrut",
                       char sx='M',int v=28)
        { nume = new char[15];
          strcpy(nume,n); prenume = new char[20];
          strcpy(prenume,p); sex = sx; virsta = v; }
        void vizualizeaza(void)
        {
            cout<<"PERSOANA numita = "<<prenume<<" "<<nume<<
              " de sex "<<sex<<" si avind "<<virsta<<" ani\n";
        }
};

// * FOLOSIREA OPERATORULUI DE DOMENIU "::" *
class LISTA ETEROGENA { // * CREAREA UNEI LISTE ETEROGENE DE
    ORICE* pB; // OBIECTE , TOATE DERIVATE DINTR-O
public : // CLASA ABSTRACTA "ORICE" *
    void set(ORICE* p) { pB = p; }
    void vizualizeaza(void)

```

```

        { pB->vizualizeaza(); }
};
void main(void)
{
    SIR s("abcdef"); NR_REAL r1(-3.5), r2(0.001);
    NR_INTREG k(10); PUNCT pct(50,75);
    MASINA m[2]; PERSOANA pers;
    LISTA_ETEROGENA p[7];
    m[0].denumeste("Oltcit");
    m[1].denumeste("Mercedes","Germania");
    pers.denumeste();
    p[0].set(&s); p[1].set(&r1); p[2].set(&m[0]);
    p[3].set(&k); p[4].set(&r2); p[5].set(&pct);
    p[6].set(&pers);
    clrscr();
    cout<<" * LISTA ETEROGENA * \n\n";
    for (int i=0;i<7;i++) // * FOLOSIREA MECANISMULUI
        { cout<<" "<<(i+1)<<" : "; // "LATE BINDING" PENTRU
            p[i].vizualizeaza(); // FOLOSIREA METODEI IN FUNCTIE
        }; // DE INSTANTA *
    getch();
}

```

Rezultatul programului este afișarea următoarelor:

```

* LISTA ETEROGENA *
1 : SIR avind continutul = abcdef
2 : NR_REAL cu valoarea = -3.5
3 : MASINA cu marca Oltcit produsa in Romania, cu viteza maxima = 170 km/h
4 : NR_INTREG cu valoarea = 10
5 : NR_REAL cu valoarea = 0.001
6 : PUNCT cu abscisa = 50 si ordonata = 75
7 : PERSOANA numita = Bogdan Patrut de sex M si avind 28 ani

```

Probleme propuse

1. Ce afișează următorul program, când se răspunde: a) cu f , b) cu n .

```

#include <stdio.h>
struct fairy_tale
{
    virtual void act1() {
        printf("Printesa intilneste broscoiul\n");act2();
    }
    void act2() {
        printf("Printesa saruta broscoiul\n");act3();
    }
    virtual void act3() {
        printf("Broscoiul se transforma in print\n");act4();
    }
    virtual void act4() {
        printf("Si au trait multi ani fericiti\n");act5();
    }
    void act5() {
        printf("Sfirsit\n");
    }
};
struct unhappy_tale:fairy_tale
{
    void act3() {
        printf("Broscoiul ramine broscoi\n");act4();
    }
    void act4() {
        printf("Printesa fuge ingrozita\n");act5();
    }
}

```

```

        void act5() {
            printf("Sfirsit nu prea fericit\n");
        }
};
main()
{
    char c;
    fairy_tale* tale;
    printf("Care poveste doriti ? ([f]ericita/[n]efericita)");
    scanf("%c",&c);
    if (c=='f') tale = new fairy_tale;
    else tale = new unhappy_tale;
    tale->act1();
    delete tale;
}

```

2. Scrieți o clasă C++ pentru liste dublu înălțuite și realizați operațiile de adăugare, inserare, căutare și stergere de elemente ca metode ale acestei clase.
3. Scrieți o clasă C++ pentru stive și implementați operațiile push și pop (de introducere, extragere a elementelor din stivă. Folosiți `template` pentru a crea stive cu orice gen de conținut sau folosiți ideea de listă eterogenă.
4. Implementați structura de arbore binar de căutare și principalele operații ce se execută cu astfel de arbori.
5. Implementați o clasă pentru numere mari și realizați operații aritmetice cu astfel de numere.
6. Definiți o clasă și scrieți principalele metode pentru a opera cu numere complexe: a) sub formă algebrică; b) sub formă trigonometrică.
7. Implementați o clasă pentru lucrul cu triunghiuri (cu lungimile laturilor cunoscute). Se vor crea metode pentru calculul ariei, perimetrului, razei cercului circumscris sau înscris în triunghi, mediane, unghiuri, dar și pentru testarea dacă două triunghiuri sunt congruente sau asemenea.
8. Implementați obiectual structura de tablou unidimensional de numere întregi și operații cu astfel de obiecte: a) citire; b) afișare; c) inversare; d) suma / produsul elementelor; e) ordonare crescătoare etc.
9. Aceeași problemă ca și mai înainte, dar pentru orice gen de tablouri. Se vor folosi posibilitățile oferite de `template`!
10. O matrice cu dimensiuni mari și numărul de elemente nule foarte mare se numește matrice rară. Se cere să se dea o metodă generală de memorare a matricelor rare, precum și o implementare în limbajul C++ a unor astfel de structuri de date, sub forma unei clase numită `matrice`. De asemenea, se cere să se implementeze și metode de calcul cu matrice rare, care să beneficieze de proporția mare de elemente nule. De fapt, se va face memorarea doar a elementelor nenule și, de asemenea, se vor executa operații doar între aceste elemente ale matricelor, ceea ce va reprezenta economisirea și de spațiu de memorie, dar și de timp de calcul. Pentru a exemplifica un mod de memorare a matricelor rare, să considerăm matricea următoare, în care în loc de elementul A_{ij} , vom nota doar ij :

		Locația						
		1	2	3	4	5	6	
0	0	0	0	0	0	0	0	VALA 23 25 31 32 35 ... (valorile elementelor)
A= 0	0	23	0	25	0	25	3	ICOL 3 5 1 2 5 ... (indicele pe coloană)

31 32 0 0 35 IPEL 1 1 3 6 (pt. a identifica linia)

Memorarea matricei A se face conform schemei de dreapta sus, în care $IPEL(i)$ furnizează adresa în zona VALA și ICOL a primului element nenul din linia i a matricei A; $IPEL(i+1) - IPEL(i) = \text{numărul de elemente nenule din linia } i$; IPEL conține $m+1$ elemente (pentru cele m linii ale lui A), iar $IPEL(m+1) = NZ+1$, în care $NZ = \text{numărul de elemente nenule ale matricei A}$. Dacă linia j din A este nulă, atunci $IPEL(j) = IPEL(j+1)$.

11. Realizați o implementare a unei clase `fereastră`, care să permită gestionarea mai multor ferestre pe ecranul grafic. Se vor implementa operații de redimensionare și mutare a ferestrei, închidere, minimizare, restaurare, maximizare, precum și metode pentru lucrul cu bare de rulare orizontale sau verticale ("scroll bars"), ajutor ("help") etc.
12. Realizați o aplicație orientată obiect pentru gestionarea datelor referitoare la elevii dintr-o clasă de liceu (nume, prenume, data nașterii, domiciliu, note la toate disciplinele etc.).
13. Definiți o clasă `Graf` pentru a lucra cu grafuri neorientate. Se vor implementa metode pentru desenarea grafului, colorarea vârfurilor sale, determinarea componentelor conexe, a circuitelor hamiltoniene etc. Se cere, de asemenea, ca aplicația să lucreze în mod grafic.
14. Definiți o clasă `Functie` pentru a lucra cu funcții reale. Se vor implementa metode pentru tabelarea funcției, reprezentarea graficului ei, determinarea minimului și maximului pe un interval dat, determinarea aproximativă a rădăcinilor, calculul formal al derivatei, calculul derivatei într-un anumit punct, calculul integralei numerice pe un anumit domeniu. Aplicația va rula în mod grafic.
15. Realizați o aplicație orientată obiect care să reprezinte un joc de tip "*Invadatorii*": din partea de sus a ecranului grafic, mai multe șiruri de nave dușmane se apropie de sol. Aici se află nava proprie, deplasabilă stânga dreapta, cu ajutorul căreia se pot nimici navele adverse (prin tragerea unui glonț). Jocul este câștigat dacă navele adverse sunt nimicite înainte de a ajunge la sol. Altfel, jocul este pierdut.

Capitolul 9.

Probleme recapitulative

- instrucțiuni de control • tipuri de date simple • vectori •
- matrice • funcții recursive • structuri • fișiere •

1. Probleme cu instrucțiuni de control și tipuri de date simple

1. Să se afișeze toate modurile în care se poate descompune un număr natural n ca sumă de numere naturale consecutive.

2. Să se descompună în factori primi un număr natural n .

3. Se citește numărul natural n de la tastatură. Să se afișeze triunghiul de numere:

```
1
1 2
1 2 3
.....
1 2 3 .... n
```

Rezolvare:

```
#include <stdio.h>
```

```
void main()
{
    int n,i,j;
    scanf("%d",&n);
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=i; j++)
            printf("%2d",j);
        printf("\n");
    }
}
```

4. Se citește numărul natural n de la tastatură. Să se afișeze triunghiul de numere:

```
1
1 3
1 3 5
.....
1 3 5 .... 2n-1
```

Rezolvare:

```
#include <stdio.h>

void main()
{
    int n,i,j;
    scanf("%d",&n);
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=i; j++)
            printf("%2d",2*j-1);
        printf("\n");
    }
}
```

5. Se citește numărul natural n de la tastatură. Să se afișeze triunghiul de numere:

```
1 2 3 .... n
.....
1 2 3
1 2
1
```

Rezolvare:

```
#include <stdio.h>

void main()
{
    int n,i,j;
    scanf("%d",&n);
    for (i=n; i>=1; i--)
    {
        for (j=1; j<=i; j++)
            printf("%2d",j);
        printf("\n");
    }
}
```

6. Se citește numărul natural n de la tastatură. Să se afișeze triunghiul de numere:

```
      1
     1 2
    1 2 3
   1 2 3 4
  .....
1 2 ..... n
```

Rezolvare:

```
#include <stdio.h>
void main()
{
    int n,i,j;
    scanf("%d",&n);
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n-i; j++)
            printf(" ");
        for (j=1; j<=i; j++)
            printf("%4d",j);
        printf("\n");
    }
}
```

7. Se citește numărul natural n de la tastatură. Să se afișeze triunghiul de numere:

```
n n-1 ..... 3 2 1
n n-1 ..... 2 1
.....
n n-1
n
```

Rezolvare:

```
#include <stdio.h>
void main()
{
    int n,i,j;
    scanf("%d",&n);
    for (i=n; i>=1; i--)
    {
        for (j=n; j>=n-i+1; j--)
            printf("%2d",j);
        printf("\n");
    }
}
```

8. Se citește numărul natural n de la tastatură. Să se afișeze triunghiul de numere:

```
n  n-1 ..... 3 2 1
n-1 n-2 ..... 2 1
n-2 n-3 ..... 1
.....
3 2 1
2 1
1
```


Rezolvare:

```
#include <stdio.h>

void main()
{
    int n,i,j;
    scanf("%d",&n);
    for (i=n; i>=1; i--)
    {
        for (j=i; j>=1; j--)
            printf("%2d",j);
        printf("\n");
    }
}
```

9. Se citește numărul natural n de la tastatură. Să se calculeze, folosind toate instrucțiunile repetitive cunoscute, expresiile:

- a) $S = 1 + 2 + 3 + \dots + n$; b) $P = 1 \times 2 \times 3 \times \dots \times n$;
c) $S = 1^3 + 2^3 + 3^3 + \dots + n^3$; d) $P = 1^2 \times 2^2 \times \dots \times n^2$;
e) $S = 1 - 2 + 3 - 4 + \dots \pm n$; f) $S = 1 \times 2 + 2 \times 3 + 3 \times 4 + \dots + n \times (n+1)$;
g) $S = 1 + 1/2 + 3 + 1/4 + \dots$ (n termeni); h) $P = 1 \times 3 \times 5 \times \dots \times (2n-1)$;
i) $S = 1 + 3^2 + 5 + 7^2 + \dots$ (n termeni); j) $S = 1 - 4 + 7 - 10 + \dots$ (n termeni);
k) $S = 1^2 - 1/2 + 3^2 - 1/4 + \dots$ (n termeni); l) $P = 1 \times 4 \times 7 \times 10 \times \dots$ (n factori).

Rezolvare:

```
#include <stdio.h>

void main()
{
    int i,n,semn;
    long int suma, produs;
    float suma_r;
    printf("\nDati n:");
    scanf("%d",&n);
    printf("\nRezultate:\n");
    // punctul a
    for (suma=0, i=1; i<=n; i++)
        suma+=i;
    printf("a) %ld\n",suma);
    // punctul b
    for (produs=1, i=1; i<=n; i++)
        produs*=i;
    printf("b) %ld\n",produs);
    // punctul c
    for (suma=0, i=1; i<=n; i++)
        suma+=i*i*i;
    printf("c) %ld\n",suma);
    // punctul d
```

```

        for (produs=1, i=1; i<=n; i++)
            produs+=i*i;
        printf("d) %ld\n",produs);
// punctul e
        for (suma=0, semn=1, i=1; i<=n; i++, semn=-semn)
            suma+=semn*i;
        printf("e) %ld\n",suma);
// punctul f
        for (suma=0, i=1; i<=n; i++)
            suma+=i*(i+1);
        printf("f) %ld\n",suma);
// punctul g
        for (suma_r=0, i=1; i<=n; i++)
            if (i % 2 == 1)
                suma_r=suma_r+i;
            else
                suma_r=suma_r+(float)1/i;
        printf("g) %0.2f\n",suma_r);
// punctul h
        for (produs=1, i=1; i<=n; i++)
            produs*=(2*i-1);
        printf("h) %ld",produs);
}

```

10. Se citește numărul natural n de la tastatură. Să se calculeze, folosind toate instrucțiunile repetitive cunoscute, expresiile:

- a) $S = 1 + 1 \times 2 + 1 \times 2 \times 3 + \dots + 1 \times 2 \times \dots \times n$;
- b) $S = 1 - 1 \times 3 + 1 \times 3 \times 5 - 1 \times 3 \times 5 \times 7 + \dots$ (n termeni)
- c) $S = 1^2 - 1/(1 \times 2) + 3^2 - 1/(1 \times 2 \times 3 \times 4) + 5^2 - 1/(1 \times 2 \times 3 \times 4 \times 5 \times 6) + \dots$ (n termeni)
- d) $S = 1^2 - 1 \times 4^2 + 1 \times 4 \times 7^2 - 1 \times 4 \times 7 \times 10^2 + \dots$ (n termeni)

Rezolvare:

```

#include <stdio.h>

void main()
{
    int i,n,semn;
    long int suma, produs;
    float suma_r;
    printf("\nDati n:");
    scanf("%d",&n);
    printf("\nRezultate:\n");
// punctul a
    for (suma=0, produs=1, i=1; i<=n; i++)
        { produs*=i; suma+=produs; }
    printf("a) %ld\n",suma);
// punctul b
    for (suma=0, produs=1, semn=1, i=1; i<=n; i++, semn=-semn)
        {
            produs*=(2*i-1);

```

```

        suma+=semn*produs;
    }
    printf("b) %ld\n",suma);
// punctul c
    for (suma_r=0, produs=1, i=1; i<=n; i++)
    {
        produs*=i;
        if (i % 2 == 1)
            suma_r = suma_r + i*i;
        else
            suma_r = suma_r - (float)1/produs;
    }
    printf("c) %0.2f",suma_r);
}

```

11. Se citește numărul natural n de la tastatură. Să se determine suma primelor n numere prime.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

long int prim(long int m)
{
    for (long int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

void main()
{
    long int n, suma, a, i;
    scanf("%ld",&n);
    for (suma=2, a=3, i=1; i<n; a+=2)
        if (prim(a))
            { suma+=a; i++; }
    printf("%ld",suma);
}

```

12. Se citește numărul natural n de la tastatură. Să se determine al n -lea termen din șirul lui Fibonacci și suma primilor n termeni din acest șir. Șirul lui Fibonacci este definit astfel: primii doi termeni sunt 1, iar oricare alt termen se calculează ca fiind suma celor doi termeni imediat anteriori: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 etc.

Rezolvare:

```

#include <stdio.h>

```

```

void main()
{
    long int t1, t2, t3, i, n, suma;
    printf("\nDati n:"); scanf("%ld",&n);
    t1=0; t2=1; t3=1;
    if (n==1) suma=1; else suma=2;
    for (i=2; i<n; i++)
    {
        t1=t2; t2=t3; t3=t1+t2;
        suma+=t3;
    }
    printf("Termenul=%ld, suma=%ld", t3, suma);
}

```

13. Pentru un număr n natural citit de la tastatură să se determine suma divizorilor săi pozitivi.

Rezolvare:

```

#include <stdio.h>

void main()
{
    long int n, i, suma;
    scanf("%ld", &n);
    for (i=1, suma=0; i<=n; i++)
        if (n % i == 0)
            suma+=i;
    printf("%ld",suma);
}

```

14. Pentru un număr n natural citit de la tastatură să se determine suma divizorilor săi pozitivi, care sunt numere impare.

Rezolvare:

```

#include <stdio.h>

void main()
{
    long int n, i, suma;
    scanf("%ld", &n);
    for (i=1, suma=0; i<=n; i++)
        if ((n % i == 0) & (i % 2 == 1))
            suma+=i;
    printf("%ld",suma);
}

```

15. Pentru un număr n natural citit de la tastatură să se determine suma divizorilor săi pozitivi, care sunt numere prime.

16. Pentru un număr n natural citit de la tastatură să se calculeze suma numerelor prime mai mici decât n .

17. Pentru un număr n natural citit de la tastatură să se calculeze suma pătratelor perfecte mai mici decât n .

Rezolvare:

```
#include <stdio.h>
#include <math.h>

void main()
{
    long int n, suma, a;
    printf("\nDati n:");
    scanf("%ld",&n);
    for (suma=0, a=1; a<=n; a++)
        if (sqrt(a)==(long int)sqrt(a))
            suma = suma + a;
    printf("Suma este:%ld",suma);
}
```

18. Pentru un număr n natural citit de la tastatură să se calculeze suma cuburilor perfecte mai mici decât n .

19. Pentru un șir de n numere naturale citite de la tastatură determinați câte din ele sunt câte impare, câte prime și câte cuburi perfecte.

Rezolvare:

```
#include <stdio.h>
#include <math.h>

long int este_prim(long int m)
{
    for (long int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

int este_cub_perfect(long double m)
{
    for (long int i=1; i<=m; i++)
        if (i*i*i==m) return 1;
    return 0;
}

void main()
{
    int n, ncub, nprime, nimp, i;
    long int a;
```

```

printf("\nDati n:"); scanf("%d",&n);
nimp=0; ncub=0; nprime=0;
for (i=0; i<n; i++)
{
    printf("Dati un numar:");
    scanf("%ld",&a);
    if (a % 2 == 1) nimp++;
    if (este_cub_perfect(a)) ncub++;
    if (este_prim(a)) nprime++;
}
printf("Numarul elementelor impare   = %d\n", nimp);
printf("Numarul elementelor prime    = %d\n", nprime);
printf("Numarul elementelor cuburi   = %d\n", ncub);
}

```

20. Pentru un șir de n numere citite de la tastatură determinați numărul cel mai mare din șir, cel mai mic număr negativ din șir.

Rezolvare:

```

#include <stdio.h>

long int suma_cifre(long int m)
{
    long int s=0;
    while (m!=0)
        { s += m % 10; m/=10; }
    return s;
}

void main()
{
    int i, n;
    float a, max, min_neg;
    printf("\nDati n:"); scanf("%d",&n);
    printf("\nDati un numar: "); scanf("%f",&a);
    max=a;
    min_neg=a;
    for (i=1; i<n; i++)
    {
        printf("\nDati un numar: ");
        scanf("%f",&a);
        if (a>max) max=a;
        if ((a<min_neg) && (a<0)) min_neg=a;
    }
    printf("\nMaximul este: %0.2f",max);
    if (min_neg<0)
        printf("\nMinimul negativ este: %0.2f",min_neg);
    else
        printf("\nNu exista un numar negativ in sir");
}

```

21. Pentru un șir de n numere citite de la tastatură, care nu sunt divizibile prin 10, determinați numărul a cărui oglindit este cel mai mic.

22. Pentru un șir de n numere citite de la tastatură, determinați câte din ele sunt identice cu oglinditele lor și câte au suma cifrelor un pătrat perfect.

23. Să se simuleze mișcarea unei bile de sus în jos, pe coloana C a ecranului.

Rezolvare:

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

void main()
{
    int c,i;
    clrscr();
    printf("Dati coloana: "); scanf("%d",&c);
    clrscr();
    for (i=2; i<=24; i++)
    {
        gotoxy(c,i-1); printf(" ");
        gotoxy(c,i); printf("O");
        delay(100);
    }
}
```

24. Să se simuleze mișcarea unei bile de jos în sus, pe coloana C a ecranului.

25. Să se simuleze mișcarea unei bile de la stânga la dreapta și apoi de la dreapta la stânga, în mod repetat, pe linia L a ecranului, până la acționarea unei taste.

26. Să se simuleze pe ecran mișcarea a două bile de la stânga la dreapta și invers și de sus în jos și invers, simultan, până la acționarea unei taste.

27. Să se simuleze pe ecran mișcarea unei bile pe diagonală, sub unghiul de 45 de grade, care să se comporte ca pe o masă de biliard, până la acționarea unei taste.

Rezolvare:

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

void main()
{
```

```

int x,y,dx,dy;
clrscr();
dx=1; dy=1; x=40; y=12;
do {
    gotoxy(x,y); printf(" ");
    if ((x==1) || (x==80)) dx=-dx;
    if ((y==1) || (y==24)) dy=-dy;
    x+=dx; y+=dy;
    gotoxy(x,y); printf("O");
    gotoxy(1,1);
    delay(50);
} while (!kbhit());
}

```

2. Probleme cu vectori (tablouri unidimensionale)

1. Se citesc numere întregi de la tastatură, până la întâlnirea unui număr negativ. Să se memoreze toate numerele pare citite, într-un vector x.

Rezolvare:

```
#include <stdio.h>.
```

```

void main()
{
    int a,n=0;
    int x[100];
    do {
        scanf("%d",&a);
        if ((a>=0) && (a%2 == 1)) x[n++]=a;
    } while (a>=0);
    for (int i=0; i<n; i++)
        printf("%d",x[i]);
}

```

2. Se citesc numere întregi de la tastatură, până la întâlnirea unui număr prim. Să se memoreze într-un vector sumele cifrelor tuturor numerelor citite.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

int prim(int m)
{
    for (int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

int suma_cifre(int m)
{
    int s=0;

```



```

        while (m!=0)
            { s += m % 10; m/=10; }
        return s;
    }

void main()
{
    int a,n=0;
    int x[100];
    do {
        scanf("%d",&a);
        if (!prim(a)) x[n++]=suma_cifre(a);
    } while (!prim(a));
    for (int i=0; i<n; i++)
        printf("%d",x[i]);
}

```

3. Să se elimine dintr-un vector x de numere întregi elementul de pe poziția p.

Rezolvare:

```

#include <stdio.h>

void main()
{
    int x[100],n,i,p;
    scanf("%d",&n);
    for (i=0; i<n; i++) scanf("%d",&x[i]);
    scanf("%d",&p);
    for (i=p; i<n-1; i++) x[i]=x[i+1];
    n--;
    for (i=0; i<n; i++) printf("%d",x[i]);
}

```

4. Să se elimine dintr-un vector x de numere întregi toate elementele negative.

Rezolvare:

```

#include <stdio.h>

void main()
{
    int x[100],n,i,j;
    scanf("%d",&n);
    for (i=0; i<n; i++) scanf("%d",&x[i]);
    for (i=0; i<n; )
        if (x[i]<0)
        {
            for (j=i; j<n-1; j++) x[j]=x[j+1];
            n--;
        }
        else i++;
    for (i=0; i<n; i++) printf("%d",x[i]);
}

```

```
}
```

5. Să se elimine dintr-un vector x de numere întregi toate elementele de pe poziții divizibile prin 3.

6. Să se insereze, pe poziția p, un element a în vectorul x.

Rezolvare:

```
#include <stdio.h>

void main()
{
    int x[100], n, i, p, a;
    scanf("%d", &n);
    for (i=0; i<n; i++) scanf("%d", &x[i]);
    scanf("%d", &p);
    scanf("%d", &a);
    for (i=n; i>p; i--) x[i]=x[i-1];
    x[p]=a;
    n++;
    for (i=0; i<n; i++) printf("%d", x[i]);
}
```

7. Să se insereze, între oricare două elemente dintr-un vector x de numere întregi, suma lor.

Rezolvare:

```
#include <stdio.h>

void main()
{
    int x[100], y[200], n, i, j, a;
    scanf("%d", &n);
    for (i=0; i<n; i++) scanf("%d", &x[i]);
    for (i=0; i<n; i++)
    {
        y[2*i]=x[i];
        y[2*i+1]=x[i]+x[i+1];
    }
    for (i=0; i<2*n-1; i++)
    {
        x[i]=y[i];
        printf("%d", x[i]);
    }
}
```

8. Să se insereze, după fiecare element par dintr-un vector x de numere întregi suma cifrelor impare ale acelui element.

9. Să se insereze, după fiecare element negativ dintr-un vector x de numere întregi pătratul elementului respectiv.

10. Să se înlocuiască fiecare element negativ și impar dintr-un vector x de numere întregi cu modulul acelui element.

Rezolvare:

```
#include <stdio.h>
#include <math.h>

void main()
{
    int x[100], y[200], n, i, j, a;
    scanf("%d", &n);
    for (i=0; i<n; i++) scanf("%d", &x[i]);
    for (i=0; i<n; i++)
        if ((x[i]<0) && (x[i] % 2 == -1)) x[i]=abs(x[i]);
    for (i=0; i<n; i++) printf("%d", x[i]);
}
```

11. Să se înlocuiască fiecare element prim dintr-un vector x de numere întregi cu suma cifrelor acelui element.

12. Să se înlocuiască fiecare element a cărui sumă a cifrelor este număr prim, dintr-un vector x de numere întregi, cu inversul acelui element.

13. Se consideră un vector de numere întregi x. Se cere să se determine un vector y cu același număr de componente întregi, astfel încât elementele lui y să fie suma cifrelor elementelor din x.

Rezolvare:

```
#include <stdio.h>

int suma_cifre(int m)
{
    int s=0;
    while (m!=0)
        { s += m % 10; m/=10; }
    return s;
}

void main()
```

```

{
    int i,j,n,aux;
    int x[100], y[100];
    scanf("%d",&n);
    for (i=0; i<n; i++)
        scanf("%d",&x[i]);
    for (i=0; i<n; i++)
        y[i]=suma_cifre(x[i]);
    for (i=0; i<n; i++)
        printf("%d,",y[i]);
}

```

14. Se consideră un vector de numere întregi x. Se cere să se determine un vector y cu același număr de componente întregi, astfel încât elementele lui y modulele elementelor lui x, pentru elementele prime, respectiv pătratele elementelor lui x, pentru elementele care nu sunt prime.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

int prim(int m)
{
    for (int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

void main()
{
    int i,j,n,aux;
    int x[100], y[100];
    scanf("%d",&n);
    for (i=0; i<n; i++)
        scanf("%d",&x[i]);
    for (i=0; i<n; i++)
        if (prim(x[i]))
            y[i]=2*x[i];
        else
            y[i]=x[i]*x[i];
    for (i=0; i<n; i++)
        printf("%d,",y[i]);
}

```

15. Se consideră un vector de numere întregi x. Se cere să se determine un vector y cu același număr de componente întregi, astfel încât elementele lui y să fie cuburile sumelor cifrelor impare ale elementelor lui x.

16. Se consideră un vector de numere întregi x. Se cere să se determine un vector y cu același număr de componente întregi, astfel încât elementele lui y

să fie suma dintre produsele cifrelor impare și suma cifrelor pare ale elementelor din x.

Rezolvare:

```
#include <stdio.h>

int suma_cifre_pare(int m)
{
    int s=0, uc;
    while (m!=0)
    {
        uc = m % 10;
        if (uc % 2 == 0) s += uc;
        m/=10;
    }
    return s;
}

int produs_cifre_impere(int m)
{
    int p=1, uc;
    while (m!=0)
    {
        uc = m % 10;
        if (uc % 2 == 1) p *= uc;
        m/=10;
    }
    return p;
}

void main()
{
    int i,n;
    int x[100], y[100];
    scanf("%d",&n);
    for (i=0; i<n; i++)
        scanf("%d",&x[i]);
    for (i=0; i<n; i++)
        y[i]=(suma_cifre_pare(x[i])+produs_cifre_impere(x[i]));
    for (i=0; i<n; i++)
        printf("%d",y[i]);
}
```

17. Se consideră un vector de numere întregi x. Se cere să se determine un vector y cu același număr de componente întregi, astfel încât elementele lui y să fie suma dintre elementele lui x și suma cifrelor acestora.

18. Se consideră un vector de numere întregi x, de lungime n, n fiind par. Se cere să se determine un vector y, de lungime n/2, în care $y_1 = x_1 + x_n$, $y_2 = x_2 + x_{n-1}$ ș.a.m.d.

Rezolvare:

```
#include <stdio.h>

void main()
{
    int i,n;
    int x[100], y[100];
    scanf("%d",&n);
    for (i=1; i<=n; i++)
        scanf("%d",&x[i]);
    for (i=1; i<=n/2; i++)
        y[i]=x[i]+x[n-i+1];
    for (i=1; i<=n/2; i++)
        printf("%d",y[i]);
}
```

19. Se consideră un vector de numere întregi x , de lungime n , n fiind par. Se cere să se determine un vector y , de lungime $n/2$, în care $y_1 = \min(x_1, x_n)$, $y_2 = \min(x_2, x_{n-1})$ ș.a.m.d.

20. Se consideră un vector de numere întregi x , de lungime n , n fiind par. Se cere să se determine un vector y , de lungime $n/2$, în care $y_1 = \max(x_1^2, s_n)$, $y_2 = \max(x_2^2, s_{n-1})$ ș.a.m.d., unde s_i = suma cifrelor pare ale lui $x_i^2 - 1$.

21. Se consideră un vector de numere întregi x , de lungime n , n fiind par. Se cere să se determine un vector y , de lungime $n/2$, în care $y_1 = \max(x_1^2, s_n)$, $y_2 = \max(x_2^2, s_{n-1})$ ș.a.m.d., unde s_i = suma cifrelor prime ale lui x_i .

Rezolvare:

```
#include <stdio.h>

int suma_cifre(int m)
{
    int s=0;
    while (m!=0)
        { s += m % 10; m/=10; }
    return s;
}

void main()
{
    int i,n;
    int x[100], y[100];
    scanf("%d",&n);
    for (i=1; i<=n; i++)
        scanf("%d",&x[i]);
    for (i=1; i<=n/2; i++)
        if (x[i]*x[i]>suma_cifre(x[n-i+1]))
            y[i]=x[i]*x[i];
}
```

```

        else
            y[i]=suma_cifre(x[n-i+1]);
    for (i=1; i<=n/2; i++)
        printf("%d,",y[i]);
}

```

22. Se consideră un vector de numere întregi x , de lungime n , n fiind par. Se cere să se determine un vector y , de lungime $n/2$, în care $y_1 = \text{cmmdc}(x_1^2, x_n)$, $y_2 = \text{cmmdc}(x_2^2, x_{n-1})$ ș.a.m.d.

23. Se consideră doi vectori de numere întregi x și y , de lungime n . Se cere să se determine un vector z , de lungime n , în care $z_1 = \text{cmmdc}(x_1, s_1^2)$, $z_2 = \text{cmmdc}(x_2, s_2^2)$ ș.a.m.d., în care s_i este suma cifrelor pare ale lui y_i .

Rezolvare:

```

#include <stdio.h>

int suma_cifre_pare(int m)
{
    int s=0, uc;
    while (m!=0)
    {
        uc = m % 10;
        if (uc % 2 == 0) s += uc;
        m/=10;
    }
    return s;
}

int cmmdc(int a, int b)
{
    if ((a==0) || (b==0)) return 0;
    if (a>b) return cmmdc(a-b,b);
    if (a<b) return cmmdc(a,b-a);
    return a;
}

void main()
{
    int i,n;
    int x[100],y[100],z[100];
    scanf("%d",&n);
    for (i=1; i<=n; i++)
        scanf("%d",&x[i]);
    for (i=1; i<=n; i++)
        scanf("%d",&y[i]);
    for (i=1; i<=n; i++)
        x[i]=cmmdc(x[i],
            suma_cifre_pare(y[i])*suma_cifre_pare(y[i]));
    for (i=1; i<=n; i++)

```

```
        printf("%d", x[i]);
    }
}
```

24. Se consideră doi vectori de numere întregi x și y , de lungime n . Se cere să se determine un vector z , de lungime n , în care $z_1 = \text{cmmdc}(x_1^2, s_1^2)$, $z_2 = \text{cmmdc}(x_2^2, s_2^2)$ ș.a.m.d., în care s_i este suma cifrelor lui $\min(x_i, y_i^2 - 1)$.

25. Se consideră doi vectori de numere întregi x și y , de lungime n . Se cere să se determine un vector z , de lungime n , în care $z_1 = \max(x_1^2, s_1)$, $z_2 = \max(x_2^2, s_2)$ ș.a.m.d., în care s_i este suma cifrelor impare a celui mai mare divizor comun pentru x_i și y_{n+1-i} .

26. Se consideră doi vectori de numere întregi x și y , de lungime n . Se cere să se determine un vector z , de lungime n , în care $z_1 = \min(s_1^2, t_1^3)$, $z_2 = \max(s_2^2, t_2^3)$, $z_3 = \min(s_3^2, t_3^3)$, $z_4 = \max(s_4^2, t_4^3)$ ș.a.m.d., în care s_i este suma cifrelor pare alui lui $x_i + y_i$, iar t_i este produsul cifrelor divizibile prin 3 ale lui x_i .

27. Se consideră doi vectori de numere întregi x și y , de lungime n . Se cere să se determine un vector z , de lungime n , în care $z_1 = s_1 + t_1$, $z_2 = s_2 + t_2$, $z_3 = s_3 + t_3$, $z_4 = s_4 + t_4$ ș.a.m.d., în care s_i este suma cifrelor impare alui lui $x_i - y_i$, iar t_i este produsul cifrelor divizibile prin 3 ale lui s_i .

28. Să se memoreze în componentele unui vector primele n numere naturale prime.

Rezolvare:

```
#include <stdio.h>
#include <math.h>

int prim(int m)
{
    for (int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

void main()
{
    int x[100], n, i, a;
    scanf("%d", &n);
    // Un for interesant!
    for (i=0, a=2; i<n; a++)
        if (prim(a)) x[i++] = a;
    for (i=0; i<n; i++)
        printf("%d", x[i]);
}
```


29. Să se determine componentele unui vector x astfel încât $x_i = 1 \times 2 \times \dots \times i$.

Rezolvare:

```
#include <stdio.h>

void main()
{
    int n, i, x[100];
    scanf("%d", &n);
    x[1]=1;
    for (i=2; i<=n; i++)
        x[i]=x[i-1]*i;
    for (i=1; i<=n; i++)
        printf("%d, ", x[i]);
}
```

30. Să se determine componentele unui vector x astfel încât $x_i = 1^2 + 2^2 + \dots + i^2$.

31. Să se memoreze în componentele unui vector primele n numere naturale prime, care citite invers sunt tot numere prime.

Rezolvare:

```
#include <stdio.h>
#include <math.h>

long int prim(int m)
{
    for (long int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

long int oglindit(int a)
{
    int b=0;
    while (a)
    {
        b=b*10+a%10;
        a/=10;
    }
    return b;
}

void main()
{
    long int n, i, a, x[100];
```

```

scanf("%ld",&n);
x[0]=2;
for (i=0, a=3; i<n; a+=2)
    if (prim(a) && prim(oglindit(a)))
        x[++i]=a;
for (i=0; i<n; i++)
    printf("%d,",x[i]);
}

```

32. Să se memoreze în componentele unui vector primele n numere naturale prime a căror sumă a cifrelor este tot număr prim.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

long int prim(long int m)
{
    for (long int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

long int suma_cifre(long int m)
{
    long int s=0;
    while (m!=0)
        { s += m % 10; m/=10; }
    return s;
}

void main()
{
    long int n, i, a, x[100];
    scanf("%ld",&n);
    x[0]=2;
    for (i=0, a=3; i<n; a+=2)
        if (prim(a) && prim(suma_cifre(a)))
            x[++i]=a;
    for (i=0; i<n; i++)
        printf("%d,",x[i]);
}

```

33. Să se elimine dintr-un vector x de numere naturale toate numerele prime a căror sumă a cifrelor este număr prim.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

```

```

long int prim(long int m)
{
    for (long int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

long int suma_cifre(long int m)
{
    long int s=0;
    while (m!=0)
        { s += m % 10; m/=10; }
    return s;
}

void main()
{
    long int x[100],n,i,j;
    scanf("%ld",&n);
    for (i=0; i<n; i++)
        scanf("%ld",&x[i]);
    for (i=0; i<n; )
        if (prim(x[i]) && prim(suma_cifre(x[i])))
            {
                for (j=i; j<n-1; j++) x[j]=x[j+1];
                n--;
            }
        else i++;
    for (i=0; i<n; i++) printf("%ld,",x[i]);
}

```

34. Să se elimine dintr-un vector x de numere naturale toate numerele prime care au produsul cifrelor divizibil prin 3.

35. Se consideră un vector x cu n componente întregi. Să se calculeze $S = x_1s_1 + x_2s_2 + x_3s_3 + \dots + x_4s_4$, unde s_i = suma cifrelor divizibile prin 3 ale lui x_i .

Rezolvare:

```

#include <stdio.h>
#include <math.h>

long int suma_cifre_div3(long int m)
{
    long int s=0; int cifra;
    while (m!=0)
        {
            cifra = m % 10;
            if (cifra % 3 == 0)
                s += cifra;
            m/=10;
        }
    return s;
}

```

```

}

void main()
{
    long int x[100],n,i,suma;
    scanf("%ld",&n);
    for (i=0; i<n; i++)
        scanf("%ld",&x[i]);
    for (i=0, suma=0; i<n ; i++)
        suma += x[i]*suma_cifre_div3(x[i]);
    printf("%ld",suma);
}

```

36. Se consideră un vector x cu n componente întregi. Să se calculeze $S = 1 \times x_1^2 + 2 \times x_2^2 + 3 \times x_3^2 + \dots + n \times x_n^2$.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

void main()
{
    long int x[100],n,i,suma;
    scanf("%ld",&n);
    for (i=1; i<=n; i++)
        scanf("%ld",&x[i]);
    for (i=1, suma=0; i<=n ; i++)
        suma += i*x[i]*x[i];
    printf("%ld",suma);
}

```

37. Se consideră un vector x cu n componente reale. Să se determine în vectorul y , de lungime n , părțile întregi ale componentelor din x .

38. Se consideră un vector x cu n componente reale. Să se determine în vectorul y , de lungime n , părțile fracționare ale componentelor din x .

39. Se consideră un vector x cu n componente reale. Să se memoreze în vectorul y acele elemente din x care sunt și numere întregi.

Rezolvare:

```

#include <stdio.h>

void main()
{
    float x[100];
    int y[100];
    int n,i,j;
    scanf("%d",&n);

```

```

    for (i=0; i<n; i++) scanf("%f",&x[i]);
    for (i=0, j=0; i<n; i++)
        if (x[i]==(int)(x[i]))
            y[j++]=x[i];
    for (i=0; i<j; i++)
        printf("%d",y[i]);
}

```

40. Se consideră un vector x cu n componente întregi. Să se memoreze în vectorul y acele elemente din x care sunt pătrate perfecte.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

void main()
{
    int x[100], y[100];
    int n,i,j;
    scanf("%d",&n);
    for (i=0; i<n; i++) scanf("%d",&x[i]);
    for (i=0, j=0; i<n; i++)
        if (sqrt(x[i])== (int)(sqrt(x[i])))
            y[j++]=x[i];
    for (i=0; i<j; i++)
        printf("%d",y[i]);
}

```

3. Probleme cu tipul caracter și șiruri de caractere

1. Se consideră un vector x cu n componente caractere. Să se determine în vectorul y, de lungime n, codurile ASCII ale componentelor din x.
2. Se consideră un vector x cu n componente naturale, din intervalul 65..90. Să se determine în vectorul y, de lungime n, caracterele având codurile ASCII egale cu componentele din x.
3. Se citesc n șiruri de caractere. Să se determine câte din acestea încep cu litera 'h', câte se termină cu o vocală, câte au lungimea mai mare decât 4, câte au lungimea un pătrat perfect și câte conțin un număr impar de consoane.

Rezolvare:

```

#include <stdio.h>
#include <string.h>
#include <math.h>

```

```

void main()
{
    char s[20]; int i,n;
    printf("Dati n: ");
    scanf("%d",&n);
    int nh=0, nvoc=0, n4=0, npp=0, nimpc=0;
    int cons,j;
    char c, t[2];
    for (i=1; i<=n; i++)
    {
        printf("Dati sirul nr. %d:",i);
        scanf("%s",&s);
        if (s[0]=='h') nh++;
        c=s[strlen(s)-1];
        if ((c=='a') || (c=='e') ||
            (c=='i') || (c=='o') || (c=='u'))
            nvoc++;
        if (strlen(s)>4) n4++;
        if ((int)sqrt(strlen(s))==sqrt(strlen(s)))
            npp++;
        cons=0;
        for (j=0; j<strlen(s); j++)
        {
            c=s[j]; t[0]=c; t[1]='\0';
            if (!strstr("aeiou",t)) cons++;
        }
        if (cons%2==1) nimpc++;
    }
    printf("Nr. de siruri ce incep cu 'h': %d\n",nh);
    printf("Nr. de siruri ce se termina cu o vocala: %d\n",
        nvoc);
    printf("Nr. de siruri ce au lungimea > 4: %d\n",n4);
    printf("Nr. de siruri cu lungimea patrat perfect: %d\n",
        npp);
    printf("Nr. de siruri cu un nr. impar de consoane: %d\n",
        nimpc);
}

```

4. Se citesc n șiruri de caractere. Să se determine câte din acestea încep și se termină cu aceeași literă, câte au cel puțin două vocale, câte sunt cifre și câte sunt cifre impare.

5. Se citesc n șiruri de caractere. Să se memoreze într-un vector de șiruri de caractere.

6. Se citesc n șiruri de caractere. Să se memoreze într-un vector de șiruri de caractere acele șiruri care încep și se termină cu o vocală.

7. Se citesc n șiruri de caractere. Să se memoreze într-un vector de șiruri de caractere acele șiruri care au lungimea un număr prim.

8. Se citesc n șiruri de caractere. Să se memoreze într-un vector de șiruri de caractere oglinditele șirurilor citite.

9. Se citesc n șiruri de caractere. Să se memoreze într-un vector de caractere primele litere ale șirurilor citite.

Rezolvare:

```
#include <stdio.h>
#include <string.h>
#include <math.h>

void main()
{
    char s[20], x[20];
    int i,n;
    printf("Dati n: ");
    scanf("%d",&n);
    int nh=0, nvoc=0, n4=0, npp=0, nimpc=0;
    int cons,j;
    char c, t[2];
    for (i=1; i<=n; i++)
    {
        printf("Dati sirul nr. %d:",i);
        scanf("%s",&s);
        x[i]=s[0];
        printf(" -> %c\n",x[i]);
    }
}
```

10. Se citesc n șiruri de caractere. Să se memoreze într-un vector de caractere, pentru fiecare șir în parte, prima consoană depistată în șir, iar dacă nu este să se memoreze simbolul dolarului.

11. Se citesc n șiruri de caractere. Să se memoreze aceste șiruri într-un vector de șiruri de caractere, în ordine inversă.

12. Se citesc n șiruri de caractere, care au lungimile cel puțin 2. Pentru fiecare șir să se elimine primul și ultimul caracter, iar ceea ce rămâne să se memoreze într-un vector de șiruri de caractere.

13. Se dă un vector x de n șiruri de caractere. Să se elimine din vector elementele care au mai puțin de trei vocale.

14. Se dă un vector x de n șiruri de caractere. Să se elimine din vector elementele care au sunt identice cu oglinditele lor.

15. Se dă un vector x de n șiruri de caractere. Să se creeze un vector y de numere întregi în care să se păstreze lungimile elementelor din x.

Rezolvare:

```
#include <stdio.h>
#include <string.h>
#include <math.h>

void main()
{
    char x[20][20];
    int y[20];
    int i,n;
    printf("Dati n: ");
    scanf("%d",&n);
    int nh=0, nvoc=0, n4=0, npp=0, nimp=0;
    int cons,j;
    char c, t[2];
    for (i=0; i<n; i++)
    {
        printf("Dati sirul nr. %d:",i);
        scanf("%s",&x[i]);
    }
    for (i=0; i<n; i++)
    {
        y[i]=strlen(x[i]);
        printf("%d",y[i]);
    }
}
```

16. Se dă un vector x de n șiruri de caractere. Să se creeze un vector y de numere întregi în care să se păstreze numărul de consoane depistate în elementele din x.

17. Se dă un vector x de n șiruri de caractere. Să se creeze un vector y de numere întregi în care să se păstreze numărul de vocale depistate în elementele din x, care au cel puțin două consoane.

18. Se consideră un șir de caractere, reprezentând o propoziție. Să se despartă în cuvinte, acestea memorându-se într-un vector.

19. Se consideră un șir de caractere, reprezentând o propoziție. Să se memoreze într-un vector cuvintele care conțin măcar două vocale.

20. Se consideră un șir de caractere, reprezentând o propoziție. Să se afișeze cuvintele identice cu oglinditele lor.

21. Se consideră un șir de caractere, reprezentând o propoziție. Să se precizeze care sunt cuvintele ce apar în propoziție și de câte ori apare fiecare cuvânt în propoziție.
22. Se consideră un șir de caractere, reprezentând o propoziție. Să se rearanjeze cuvintele în șir astfel încât ele să fie în ordine alfabetică.
23. Se dă un șir de caractere. Să se transforme toate literele mari în litere mici, iar toate literele mici în litere mari, restul caracterelor rămânând neschimbate.
24. Se dă un șir de caractere. Să se înlocuiască fiecare vocală cu vocala imediat următoare în alfabet.
25. Se dă un șir de caractere. Să se înlocuiască fiecare vocală cu litera imediat următoare în alfabet.
26. Se dă un șir de caractere. Să se înlocuiască fiecare apariție multiplă consecutivă a unui caracter cu o singură apariție a aceluși caracter și numărul de apariții. De exemplu, dacă șirul dat este "BBAMEEEZZB", atunci se va obține șirul "B2AME3Z2B".
27. Se dă un șir de caractere. Să se determine un cel mai lung subșir care este identic cu oglinditul său.
28. Se dă un șir de caractere. Să se memoreze conținutul său într-un vector de caractere, apoi să se ordoneze invers alfabetic acest șir.
29. Se dă un șir de caractere. Să se calculeze numărul de apariții a unei consoane între două vocale.
30. Se dă un șir de caractere. Să se afișeze toate prefixele acestuia. De exemplu, pentru șirul "ABCDE" se vor afișa șirurile "A", "AB", "ABC", "ABCD" și "ABCDE".
31. Se dă un șir de caractere. Să se afișeze toate sufixele acestuia. De exemplu, pentru șirul "ABCDE" se vor afișa șirurile "E", "DE", "CDE", "BCDE" și "ABCDE".
32. Se dă un șir de caractere de lungime pară. Să se afișeze toate infixele acestuia, centrate în mijlocul șirului. De exemplu, pentru șirul "ABCDE" se vor afișa șirurile "C", "BCD" și "ABCDE".

4. Probleme cu matrice (tablouri bidimensionale)

1. Se consideră o matrice A de numere întregi. Să se determine cel mai mic element, cel mai mare element par, cel mai mic număr prim și cel mai mare pătrat perfect din A.

Rezolvare:

```
#include <stdio.h>
#include <math.h>
#include <values.h>

int esteprim(int n)
{
    int d;
    if (n<2) return 0;
    if (n==2) return 1;
    if (n%2==0) return 0;
    for (d=3; d<=sqrt(n); d++)
        if (n%d==0) return 0;
    return 1;
}

int estepatratperfect(int n)
{
    if (n<0) return 0;
    float q=sqrt(n);
    if (q==(int)q) return 1;
    else return 0;
}

void main()
{
    int a[10][10];
    int m,n,i,j;
    int min,maxpar,minprim,maxpp;
    printf("Dati m,n:"); scanf("%d%d",&m,&n);
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            printf("Dati a[%d,%d]:",i,j);
            scanf("%d",&a[i][j]);
        }
    min=a[0][0];
    maxpar=-MAXINT;
    minprim=MAXINT;
    maxpp=0;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            if (a[i][j]<min) min=a[i][j];
            if ((a[i][j]>maxpar) && (a[i][j]%2==0))
                maxpar=a[i][j];
        }
}
```

```

        if (esteprim(a[i][j]) &&
            (a[i][j]<minprim))
            minprim=a[i][j];
        if (estepatratperfect(a[i][j]) &&
            (a[i][j]>maxpp))
            maxpp=a[i][j];
    }
    printf("min = %d\n",min);
    printf("max par = %d\n",maxpar);
    printf("min prim = %d\n",minprim);
    printf("max patrat perfect = %d\n",maxpp);
}

```

2. Se consideră o matrice A de numere întregi. Să se determine suma componentelor pare, produsul componentelor negative, numărul componentelor prime și numărul componentelor care sunt divizibile prin 5.

Rezolvare:

```

#include <stdio.h>
#include <math.h>

int esteprim(int n)
{
    int d;
    if (n<2) return 0;
    if (n==2) return 1;
    if (n%2==0) return 0;
    for (d=3; d<=sqrt(n); d++)
        if (n%d==0) return 0;
    return 1;
}

void main()
{
    int a[10][10];
    int m,n,i,j;
    int sumapoz,prodneg,nrprime,ndiv5;
    printf("Dati m,n:"); scanf("%d%d",&m,&n);
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            printf("Dati a[%d,%d]:",i,j);
            scanf("%d",&a[i][j]);
        }
    sumapoz=0; prodneg=1;
    nrprime=0; ndiv5=0;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            if (a[i][j]>0) sumapoz+=a[i][j];
            if (a[i][j]<0) prodneg*=a[i][j];
            if (esteprim(a[i][j])) nrprime++;
        }
}

```

```

        if (a[i][j]%5==0) ndiv5++;
    }
    printf("Suma pozitivelor: %d\n",sumapoz);
    printf("Produsul negativelor: %d\n",prodneg);
    printf("Numarul numerelor prime: %d\n",nrprime);
    printf("Numarul celor divizibile cu 5: %d\n",ndiv5);
}

```

3. Se consideră o matrice A de numere întregi. Să se determine câte coloane conțin doar numere pozitive, câte coloane conțin cel puțin un număr divizibil prin numărul de linii și câte coloane conțin măcar două numere pare.

Rezolvare:

```

#include <stdio.h>

void main()
{
    int a[10][10];
    int m,n,i,j;
    int colpoz,coldiv,colpare;
    printf("Dati m,n:"); scanf("%d%d",&m,&n);
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            printf("Dati a[%d,%d]:",i,j);
            scanf("%d",&a[i][j]);
        }
    colpoz=0; coldiv=0; colpare=0;
    int doar_pozitive, diviz, npare;
    for (j=0; j<n; j++)
    {
        doar_pozitive=1; diviz=0; npare=0;
        for (i=0; i<m; i++)
        {
            if (a[i][j]<=0) doar_pozitive=0;
            if (a[i][j] % m == 0) diviz=1;
            if (a[i][j] % 2 == 0) npare++;
        }
        if (doar_pozitive) colpoz++;
        if (diviz) coldiv++;
        if (npare>=2) colpare++;
    }
    printf("Exista %d coloane doar cu elemente pozitive\n",
        colpoz);
    printf("Exista %d col. cu macar un nr. diviz. prin m\n",
        coldiv);
    printf("Exista %d col. cu macar doua elemente pare\n",
        colpare);
}

```

4. Se consideră o matrice A de numere întregi. Să se determine câte elemente au cel puțin o cifră de 2, câte au un număr par de cifre impare, câte elemente negative au mai mult de două cifre și câte elemente pozitive au mai mult de două cifre de 3 sau 5.
5. Se consideră o matrice A de numere întregi. Să se determine câte linii conțin cel puțin două numere prime, câte linii conțin cel puțin un număr având mai mult de două cifre de 4 și câte linii au suma elementelor pară.
6. Se consideră o matrice A de numere întregi. Să se determine câte linii au suma elementelor egală cu pătratul numărului de coloane și câte coloane au produsul elementelor egal cu pătratul diferenței dintre coloane și linii.
7. Se consideră o matrice A de numere întregi. Să se determine câte elemente sunt identice cu elementul maxim din matrice și câte cu elementul minim din matrice.
8. Se consideră o matrice A de numere întregi. Să se determine câte elemente pare sunt situate pe linii cu număr de ordin impar și câte linii conțin elemente care au un număr par de cifre impare.
9. Se consideră o matrice A de numere întregi. Să se elimine din matrice coloana C și linia L.

Rezolvare:

```
#include <stdio.h>

void main()
{
    int a[10][10];
    int m,n,i,j,C,L;
    printf("Dati m,n:"); scanf("%d%d",&m,&n);
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            printf("Dati a[%d,%d]:",i,j);
            scanf("%d",&a[i][j]);
        }
    for (i=0; i<m; i++)
    {
        for (j=0; j<n; j++)
            printf("%3d",a[i][j]);
        printf("\n");
    }
    printf("Dati numarul coloanei si al liniei\n");
    printf("C="); scanf("%d",&C);
    printf("L="); scanf("%d",&L);
}
```

```

for (i=L; i<m-1; i++)
    for (j=0; j<n; j++)
        a[i][j]=a[i+1][j];
m--;
for (j=C; j<n-1; j++)
    for (i=0; i<m; i++)
        a[i][j]=a[i][j+1];
n--;
for (i=0; i<m; i++)
{
    for (j=0; j<n; j++)
        printf("%3d",a[i][j]);
    printf("\n");
}
}

```

10. Se consideră o matrice A de numere întregi. Să se introducă înaintea liniei L o linie cu elementele 1, 2, ... n, unde n este numărul de coloane ale lui A.

Rezolvare:

```

#include <stdio.h>

void main()
{
    int a[10][10];
    int m,n,i,j,L;
    printf("Dati m,n:"); scanf("%d%d",&m,&n);
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            printf("Dati a[%d,%d]:",i,j);
            scanf("%d",&a[i][j]);
        }
    for (i=0; i<m; i++)
    {
        for (j=0; j<n; j++)
            printf("%3d",a[i][j]);
        printf("\n");
    }
    printf("Dati numarul liniei\n");
    printf("L="); scanf("%d",&L);
    for (i=m; i>L; i--)
        for (j=0; j<n; j++)
            a[i][j]=a[i-1][j];
    m++;
    for (j=0; j<n; j++)
        a[L][j]=j+1;
    for (i=0; i<m; i++)
    {
        for (j=0; j<n; j++)
            printf("%3d",a[i][j]);
        printf("\n");
    }
}

```

}

11. Se consideră o matrice A de numere întregi. Să se introducă înaintea coloanei C o coloană cu elementele $m, m-1, \dots, 3, 2, 1$, unde m este numărul de linii ale lui A.
12. Se consideră o matrice pătratică A de numere întregi. Să se determine dacă suma elementelor de sub diagonală principală este număr prim.
13. Se consideră o matrice pătratică A de numere întregi. Să se determine cel mai mare divizor comun și cel mai mic multiplu comun dintre suma elementelor de sub diagonală principală și produsul elementelor de pe diagonală secundară.
14. Se consideră o matrice A de numere reale. Să se adauge la matricea A o coloană care să conțină sumele elementelor de pe linii.
15. Se consideră o matrice A de numere reale. Să se adauge la matricea A o linie care să conțină sumele elementelor pare și negative de pe coloane.
16. Se consideră o matrice A de numere reale. Să se elimine din A liniile care conțin elemente negative mai mic decât -5.
17. Se consideră o matrice A de numere reale. Să se elimine din A coloanele care conțin cel puțin un element nul.
18. Se consideră o matrice A de numere reale. Să se elimine din A liniile care conțin cel puțin trei numere întregi.
19. Se consideră o matrice A de caractere. Să se determine câte din caracterele lui A sunt vocale.
20. Se consideră o matrice A de caractere. Să se determine câte din caracterele lui A sunt cifre.

Rezolvare:

```
#include <stdio.h>

void main()
{
    char a[10][10];
    int ncifre;
```

```

int m,n,i,j;
printf("Dati m,n:"); scanf("%d%d",&m,&n);
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
    {
        printf("Dati a[%d,%d]:",i,j);
        scanf("%1s",&a[i][j]);
    }
ncifre=0;
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
        if ((a[i][j]>='0') && (a[i][j]<='9'))
ncifre++;
printf("Numarul de cifre depistate in matrice:
%d",ncifre);
}

```

21. Se consideră o matrice A de șiruri de caractere. Să se determine câte elemente încep cu o vocală și câte încep cu o consoană.

22. Se consideră o matrice A de șiruri de caractere. Să se determine câte elemente încep și se termină cu aceeași literă.

23. Se consideră o matrice A de șiruri de caractere. Să se înlocuiască fiecare apariție a unui șir palindrom (adică identic cu oglinditul său) cu șirul din colțul dreapta-sus al lui A.

24. Se consideră o matrice A de șiruri de caractere. Să se elimine din A toate liniile și apoi toate coloanele care conțin mai puțin de 4 șiruri de lungime pară.

25. Se consideră o matrice A de șiruri de caractere. Să se creeze o matrice B de numere întregi, care să conțină lungimile șirurilor de caractere din A.

26. Se consideră o matrice A de caractere. Să se formeze și să se afișeze șirul reprezentând concatenarea elementelor de pe diagonala secundară.

27. Se consideră o matrice A de caractere. Să se formeze și să se afișeze șirul reprezentând concatenarea tuturor vocalelor întâlnite în matrice, la o parcurgere pe verticală și de sus în jos a sa.

28. Se consideră o matrice A de șiruri de caractere. Să se formeze o matrice B de șiruri de caractere care să conțină oglinditele elementelor din A.

29. Se consideră o matrice A de șiruri de caractere. Să se formeze o matrice A de șiruri de caractere care să conțină șirurile din A, după ce s-au eliminat

primul și ultimul caracter din A (se presupune că elementele din A au cel puțin două caractere).

30. Se consideră o matrice pătratică A de șiruri de caractere. Să se obțină transpusa acestei matrice și să se afișeze ambele matrici.

Rezolvare:

```
#include <stdio.h>

void main()
{
    char a[10][10];
    int n,i,j;
    printf("Dati n:"); scanf("%d",&n);
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
        {
            printf("Dati a[%d,%d]:",i,j);
            scanf("%ls",&a[i][j]);
        }
    printf("Matricea initiala:\n");
    for (i=0; i<n; i++)
    {
        for (j=0; j<n; j++)
            printf("%3c",a[i][j]);
        printf("\n");
    }
    int aux;
    for (i=0; i<n; i++)
        for (j=0; j<i; j++)
        {
            aux=a[i][j];
            a[i][j]=a[j][i];
            a[j][i]=aux;
        }
    printf("Matricea finala:\n");
    for (i=0; i<n; i++)
    {
        for (j=0; j<n; j++)
            printf("%3c",a[i][j]);
        printf("\n");
    }
}
```

31. Se consideră o matrice pătratică A de șiruri de caractere, de dimensiuni pare. Se împarte matricea în patru zone pătratice identice. Să se determine dacă un anumit șir de caractere se găsește în prima zonă, nu se găsește în a doua, iar oglinditul său se găsește într-una din celelalte două zone.

32. Se consideră o matrice pătratică A de numere întregi, de dimensiuni pare. Se împarte matricea în patru zone pătratice identice. Să se determine dacă suma elementelor din prima zonă este număr par, dacă produsul elementelor din zona a doua este mai mic decât maximul dintre suma elementelor pozitive din zona a treia și numărul de elemente impare negative din zona a patra.

33. Se consideră o matrice A de numere reale, de dimensiuni m, n și o matrice B de numere reale, de dimensiuni n, p. Se cere să se determine matricea C de dimensiuni m, p, care să reprezinte produsul lui A cu B.

Rezolvare:

```
#include <stdio.h>

typedef      int matrice[10][10];

void citeste_matricea(char nume, matrice a,
                      int * nlinii, int * ncoloane)
{
    int i,j;
    printf("Dati numarul de linii ale matricei %c: ",nume);
    scanf("%d",nlinii);
    printf("Dati numarul de coloane ale matricei  %c: ",nume);
    scanf("%d",ncoloane);
    for (i=0; i<*nlinii; i++)
        for (j=0; j<*ncoloane; j++)
            {
                printf("Dati %c[%d,%d]: ",nume,i,j);
                scanf("%d",&a[i][j]);
            }
}

void scrie_matricea(char nume, matrice a, int nl, int nc)
{
    printf("Matricea %c este:\n",nume);
    int i,j;
    for (i=0; i<nl; i++)
        {
            for (j=0; j<nc; j++)
                printf("%3d",a[i][j]);
            printf("\n");
        }
}

void main()
{
    int m,n,p,i,j,k;
    matrice a,b,c;
    citeste_matricea('a',a,&m,&n);
    scrie_matricea('a',a,m,n);
    citeste_matricea('b',b,&n,&p);
    scrie_matricea('b',b,n,p);
}
```

```

/* inmultirea matricilor */
for (i=0; i<m; i++)
    for (j=0; j<p; j++)
    {
        c[i][j]=0;
        for (k=0; k<n; k++)
            c[i][j]+=a[i][k]*b[k][j];
    }
scrie_matricea('c',c,m,p);
}

```

34. Se consideră două matrice de numere reale și se cere să se determine suma și diferența acestora.

35. Se consideră două matrice de numere reale. Să se determine o matrice în care fiecare element este minimul dintre elementele corespunzătoare din cele două matrice.

36. Se consideră o matrice de numere reale. Să se formeze două matrice de numere reale, în prima memorându-se părțile întregi, iar în a doua părțile fracționare ale elementelor corespunzătoare din matricea dată.

37. Se consideră o matrice pătratică de numere reale. Să se formeze două matrice de numere reale, în prima păstrându-se pătratele elementelor corespunzătoare din matricea dată, iar în a doua suma dintre matricea dată și transpusa matricei pătratelor.

38. Să se memoreze într-o matrice cu m linii și n coloane primele $m \times n$ numere prime, care citite invers sunt tot numere prime. Memorarea se va face de la stânga la dreapta și de sus în jos.

39. Se citesc $n \times n$ elemente și se cere să se așeze într-o matrice pătratică A cu n linii și n coloane, în spirală, în sensul acelor de ceasornic, pornind din colțul stânga sus către centru.

40. Se citesc $n \times n$ elemente și se cere să se așeze într-o matrice pătratică A cu n linii și n coloane, în spirală, în sensul invers acelor de ceasornic, pornind din colțul stânga sus către centru.

41. Se citesc $n \times n$ elemente și se cere să se așeze într-o matrice pătratică A cu n linii și n coloane, în spirală, în sensul acelor de ceasornic, pornind din centru către margine.

42. Se citesc $n \times n$ elemente și se cere să se așeze într-o matrice pătratică A cu n linii și n coloane, în spirală, în sensul invers acelor de ceasornic, pornind din centru către margine.

43. Se citesc $n \times n$ elemente și se cere să se așeze într-o matrice pătratică A cu n linii și n coloane, de la stânga la dreapta, pe liniile impare de sus în jos, iar pe cele pare de jos în sus.

5. Probleme de ordonare

1. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât elementele pare să fie înaintea celor impare, dar ordinea să se păstreze atât la cele pare, cât și la cele impare.

2. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât elementele pare să își păstreze ordinea, iar cele impare să fie în ordine crescătoare.

3. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât ele să fie în ordinea descrescătoare a sumelor cifrelor lor.

Rezolvare:

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int x[100],n,i,j,a,aux;
```

```
    scanf("%d",&n);
```

```
    for (i=0; i<n; i++) scanf("%d",&x[i]);
```

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

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

```
            if ((x[i] % 2 == 1) && (x[j] % 2 == 1) &&  
                (x[i] > x[j]))
```

```
                { aux = x[i]; x[i] = x[j] ; x[j] = aux; }
```

```
    for (i=0; i<n; i++) printf("%d",x[i]);
```

```
}
```

4. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât elementele prime să fie pe aceleași poziții, iar celelalte elemente să fie în ordinea crescătoare a produsului cifrelor lor.

5. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât elementele a căror sumă a cifrelor sunt numere prime să fie înaintea celorlalte elemente, în ordine crescătoare.

Rezolvare:

```
#include <stdio.h>
#include <math.h>

int prim(int m)
{
    for (int i=2; i<=sqrt(m); i++)
        if (m%i==0) return 0;
    return 1;
}

int suma_cifre(int m)
{
    int s=0;
    while (m!=0)
        { s += m % 10; m/=10; }
    return s;
}

void main()
{
    int i,j,n,p,q,aux;
    int x[100], y[100];
    scanf("%d",&n);
    for (i=0; i<n; i++)
        scanf("%d",&x[i]);
    p=0; q=n-1;
    for (i=0; i<n; i++)
        if (prim(suma_cifre(x[i])))
            y[p++]=x[i];
        else
            y[q--]=x[i];
    /* Obs. elementele cu suma cifrelor neprima apar
       in ordine inversa */
    for (i=0; i<p-1; i++)
        for (j=i+1; j<p; j++)
            if (y[j]<y[i])
                { aux=y[i]; y[i]=y[j]; y[j]=aux; }
    for (i=0; i<n; i++)
        x[i]=y[i];
    for (i=0; i<n; i++)
        printf("%d,",x[i]);
}
```

6. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât elementele pare să fie înaintea celor impare, în ordine crescătoare, iar elementele impare să fie în ordine descrescătoare.

Rezolvare:

```
#include <stdio.h>
```

```

void schimba(int* a, int* b)
{
    int aux = *a;
    *a = *b;
    *b = aux;
}

void main()
{
    int aa=20; int bb=30;
    schimba(&aa,&bb);
    printf("%d,%d",aa,bb);
    int i,j,n,p,q,aux;
    int x[100];
    scanf("%d",&n);
    for (i=0; i<n; i++)
        scanf("%d",&x[i]);
    for (i=0; i<n-1; i++)
        for (j=i+1; j<n; j++)
            if ((x[i]%2==0) && (x[j]%2==0) && (x[i]>x[j]))
                schimba(&x[i],&x[j]);
            else
                if ((x[i]%2==1) && (x[j]%2==1) && (x[i]<x[j]))
                    schimba(&x[i],&x[j]);
            else
                if ((x[i]%2==1) && (x[j]%2==0))
                    schimba(&x[i],&x[j]);
    for (i=0; i<n; i++)
        printf("%d,",x[i]);
}

```

7. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât elementele prime să fie înaintea celorlalte, în ordine descrescătoare, iar elementele rămase să fie în ordinea crescătoare a produsului cifrelor impare ale lor.

8. Să se rearanjeze elementele unui vector x de numere întregi, astfel încât elementele pozitive să fie înaintea celor negative, acestea din urmă fiind puse în ordine descrescătoare a numărului lor de cifre.

9. Se consideră o matrice A de numere reale. Să se rearanjeze coloanele în ordinea crescătoare a numărului de elemente nule pe care le conțin.

Rezolvare:

```

#include <stdio.h>

void main()
{
    int a[10][10], nnule[10];
    int m,n,i,j;

```

```

printf("Dati m,n:"); scanf("%d%d",&m,&n);
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
    {
        printf("Dati a[%d,%d]:",i,j);
        scanf("%d",&a[i][j]);
    }
printf("Matricea initiala:\n");
for (i=0; i<m; i++)
{
    for (j=0; j<n; j++)
        printf("%3d",a[i][j]);
    printf("\n");
}
for (j=0; j<n; j++)
{
    nnule[j]=0;
    for (i=0; i<m; i++)
        if (a[i][j]==0) nnule[j]++;
}
int k, aux;
for (j=0; j<n-1; j++)
    for (k=j+1; k<n; k++)
        if (nnule[k]<nnule[j])
            for (i=0; i<m; i++)
            { aux=a[i][j];
              a[i][j]=a[i][k];
              a[i][k]=aux; }
printf("Matricea finala:\n");
for (i=0; i<m; i++)
{
    for (j=0; j<n; j++)
        printf("%3d",a[i][j]);
    printf("\n");
}
}

```

10. Se consideră o matrice A de numere reale. Să se rearanjeze coloanele în ordinea crescătoare a numărului de elemente cu suma cifrelor divizibilă prin 3.

11. Se consideră o matrice A de numere reale. Să se rearanjeze liniile în ordinea descrescătoare a numărului de elemente identice cu elementul cel mai mic din matrice.

12. Se consideră o matrice A de numere reale. Să se rearanjeze liniile în ordinea descrescătoare a numărului de elemente pozitive impare pe care le conțin.

13. Se consideră o matrice A de numere reale. Să se rearanjeze elementele matricii astfel încât ele să fie dispuse în ordine crescătoare de la stânga la dreapta și de sus în jos.

14. Se consideră o matrice A de șiruri de caractere. Să se formeze o matrice B de șiruri de caractere în care se regăsesc elementele din A așezate în ordine alfabetică, de la stânga la dreapta și de sus în jos.

15. Se consideră o matrice A de șiruri de caractere. Să se formeze o matrice B de șiruri de caractere în care se regăsesc elementele din B așezate în ordine inversă a lungimilor lor, de la stânga la dreapta și de sus în jos.

6. Probleme cu structuri

1. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, nota1, nota2 și media de tip real. Presupunând cunoscute notele, se cere să se calculeze mediile și să se ordoneze elevii descrescător după medii.

2. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, nota1, nota2 și media de tip real. Presupunând cunoscute notele, se cere să se calculeze mediile și să se ordoneze elevii în ordinea alfabetică a numelor. Pentru nume identice, se vor ordona alfabetic după prenume.

3. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, nota1, nota2 și media de tip real. Presupunând cunoscute prenumele elevilor, să se determine sexul fiecărui elev după regula: dacă prenumele se termină în "a", atunci sexul este feminin, cu excepția numelor precum "Mihnea", "Luca", "Horia", "Mircea", iar dacă prenumele nu se termină în "a", atunci sexul este masculin, cu excepția numelor precum "Carmen", "Alice", "Beatrice".

4. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, nota1, nota2 și media de tip real. Presupunând cunoscute mediile să se ordoneze elevii descrescător după medii, iar pentru medii egale, alfabetic după nume și prenume.

5. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, nota1, nota2 și media de tip real. Presupunând că se cunoaște sexul fiecărui elev, să se listeze mai întâi fetele în ordinea invers alfabetică a prenumelor, apoi băieții în ordinea alfabetică a numelor.

6. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Presupunând cunoscute mediile să se modifice câmpul admis astfel încât doar primii m elevi să fie considerați admiși.

7. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Presupunând cunoscute mediile să se modifice câmpul admis astfel încât doar elevii având medii peste o valoare citită de la tastatură să fie considerați admiși.

8. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se modifice câmpurile admis astfel încât să fie considerați admiși toți băieții cu medii peste 8 și toate fetele cu medii peste 7.

9. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se modifice câmpurile admis astfel încât să fie considerați admiși toți elevii a căror nume începe cu o consoană, iar media este peste 8.

10. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se modifice câmpurile admis astfel încât să fie considerați admiși doar prima jumătate dintre elevi, după o aranjare alfabetică a lor (după nume și prenume).

11. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se elimine din lista elevilor elevii neadmiși și elevii care au numele mai mare de 8 caractere.

12. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se elimine din lista elevilor fetele.

13. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se elimine din lista elevilor băieții având prenumele din trei litere.

14. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se obțină o listă a fetelor și una a băieților, apoi să se afișeze cele două liste pe două coloane paralele, pe ecran.

15. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se elimine din lista elevilor băieții care sunt cuprinși, în listă, între două fete.

16. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se elimine din lista elevilor fetele peste 18 ani și să se calculeze media de vârstă a fetelor rămase.

17. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se elimine din lista elevilor băieții cuprinși între două fete având peste 18 ani și să se calculeze media de vârstă a elevilor rămași.

18. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se insereze în listă, între oricare două persoane, o nouă persoană având numele persoanei dinainte și prenumele persoanei de după, iar media egală cu a unei persoane aleasă la întâmplare din listă.

19. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se insereze în listă, după fiecare persoană, o nouă persoană care să aibă prenumele "Ionescu" și prenumele persoanei dinainte, dacă aceasta este o fată, respectiv "Popescu" și prenumele persoanei dinainte, dacă aceasta este un băiat.

20. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se interschimbe prenumele elevilor astfel: al primului cu al ultimului, celui de al doilea cu al penultimului ș.a.m.d..

21. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se insereze în listă, între oricare două persoane, o nouă persoană având un număr de ani ales la întâmplare între 15 și 20, precum și prenumele persoanei dinainte și numele "Popescu", apoi să se determine media de vârstă a tuturor persoanelor din lista modificată.

22. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se adauge listei o persoană a cărei date să fie alese la întâmplare din datele celorlalte persoane din listă (de exemplu numele de la o persoană, prenumele de la alta, vârsta de la alta ș.a.m.d.). Să se verifice dacă numele noii persoane este compatibil cu sexul ei. Se va ține cont de ultima literă din prenume și de faptul că "Horia", "Mircea", "Mihnea" sau "Luca" sunt nume de băieți, pe când "Alice", "Beatrice" sau "Carmen" nume de fete.

23. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se determine media de vârstă a fetelor și numărul băieților cu medii mai mari decât 8.

24. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se determine media de vârstă a băieților a căror prenume începe cu o literă din prima jumătate a alfabetului, precum și numele celor mai tinere fete.

25. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se determine câte fete se numesc "Laura", "Alina" sau "Mihaela" și câte din acestea au lângă ele un băiat mai mare de 16 ani.

26. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se determine care valoarea mediei mediilor tuturor elevilor, apoi să se înlocuiască toate mediile mai mici decât valoarea rezultată cu aceasta.

7. Probleme cu fișiere

1. Să se copieze conținutul unui fișier text în alt fișier text.
2. Să se afișeze conținutului unui fișier text pe ecran.
3. Să se afișeze, pagină cu pagină, conținutul unui fișier text, pe ecran. Trecerea la următoarea pagină se va realiza apăsând o tastă.
4. Se dă un fișier text. Să se creeze un alt fișier text care să conțină toate liniile din primul fișier, care au cel puțin două cuvinte (cuvintele se consideră separate de spații sau unul din simbolurile ".", ",", ";").
5. Se dă un fișier text. Să se creeze un alt fișier text care să conțină toate liniile din primul fișier, mai puțin cele de lungime pară care încep cu o vocală.
6. Se dă un fișier text. Să se creeze un alt fișier text care să conțină toate liniile din primul fișier, mai puțin cele care conțin un număr impar de consoane.
7. Se dă un fișier text. Să se elimine din el toate liniile de ordin impar.
8. Se dă un fișier text. Să se elimine din el toate liniile de ordin par, care conțin un număr impar de vocale.
9. Se dă un fișier text. Să se elimine din el toate liniile care conțin un număr impar de litere "A" și care încep cu litera "B".
10. Se dau două fișiere text. Să se verifice dacă cele două fișiere au conținuturi identice.
11. Se dă un fișier text, cu maxim 100 de linii. Să se ordoneze alfabetic liniile în acest fișier.
12. Se dă un fișier text. Se cere să se înlocuiască fiecare linie cu oglindita sa.

13. Se dă un fișier text, conținând numere, separate prin spații. Se cere să se determine dacă există linii care să conțină numere prime în ele, iar dacă da, câte.

14. Se consideră un fișier text care conține m linii, pe fiecare linie câte n numere întregi. Să se citească acestea într-o matrice A cu m linii și n coloane, apoi să se afișeze matricea A pe ecran.

15. Se dau două fișiere text. Să se alipească cele două fișiere într-un al treilea.

16. Se consideră o matrice A de numere întregi, citită de la tastatură. Să se scrie conținutul matricei și al transpusei matricei într-un fișier text.

17. Se dă un număr n natural. Să se scrie într-un fișier text următorul triunghi de numere:

```
1
1 2
1 2 3
.....
1 2 3 ... n
```

18. Se dă un fișier text cu maxim 100 linii. Să se creeze un alt fișier text care să conțină liniile din fișierul inițial, dispuse în ordine inversă.

19. Se dă un fișier text. Să se preia liniile acestui fișier într-un vector de șiruri de caractere, apoi să se ordoneze vectorul, alfabetic și să se adauge în fișierul text dat.

20. Să se înlocuiască fiecare apariție a unui șir de caractere S într-un fișier text cu alt șir T .

21. Să se elimine dintr-un fișier text toate aparițiile unui șir de caractere S .

22. Se consideră o structură pentru stocarea datelor despre n elevi, având câmpurile: nume și prenume de tip șir de caractere, sex de tip caracter, vârstă de tip întreg, media de tip real și admis de tip logic (sau întreg). Să se scrie un program care să execute operații de ștergere, adăugare, modificare și consultare a datelor despre acești elevi. De asemenea, se cere ca lista de elevi să poată fi salvată într-un fișier text și preluată de acolo.

23. Să se numere de câte ori apare un șir S , mărginit de spații sau început/sfârșit de rând, în cadrul unui fișier text.

24. Se dă un fișier text. Să se înlocuiască fiecare apariție a unui șir care începe cu literă mare și este mărginit cu spații sau început/sfârșit de rând cu oglinditul șirului respectiv.

8. Probleme cu funcții recursive

1. Să se scrie o funcție recursivă care să caute (secvențial) un element a într-un vector x cu n componente.
2. Să se scrie o funcție recursivă care să caute binar un element a într-un vector x , între indicii s și d .
3. Să se scrie o funcție recursivă care să determine suma elementelor unui vector x de n numere întregi.
4. Să se scrie o funcție recursivă care să determine produsul elementelor pare dintr-un vector x de n numere întregi.
5. Să se scrie o funcție recursivă care să determine suma elementelor negative dintr-un vector x de n numere întregi.
6. Să se scrie o funcție recursivă care să determine suma pătratelor elementelor de pe poziții impare dintr-un vector x de n numere întregi.
7. Să se scrie o funcție recursivă care să verifice dacă într-un vector x de n numere întregi există măcar un element prim.
8. Să se scrie o funcție recursivă care să verifice dacă într-un vector x de n numere întregi există măcar un element pozitiv divizibil prin 3.
9. Să se scrie o funcție recursivă care să verifice dacă într-un vector x de n numere întregi există măcar un element cu suma cifrelor divizibilă prin 3.
10. Să se scrie o funcție recursivă pentru a determina suma elementelor negative impare dintr-un vector x de n numere întregi.
11. Să se scrie o funcție recursivă pentru a determina cel mai mare divizor comun al elementelor dintr-un vector x cu n numere întregi pozitive.

12. Să se scrie o funcție recursivă care să determine cel mai mare element dintr-un vector.
13. Să se scrie o funcție recursivă pentru a determina cel mai mic element negativ impar dintr-un vector de numere întregi.
14. Să se scrie o funcție recursivă pentru a determina cel mai mic element în valoare absolută dintr-un vector de numere întregi.
15. Să se scrie o funcție recursivă pentru a determina dacă există un element negativ cu suma cifrelor pozitive impară într-un vector de numere întregi.

Bibliografie

- Bogdan Pătruț - *Aplicații în C și C++*, Editura Teora, București, 1998-2004
- Herbert Schildt - *C - manual complet*, Editura Teora, București, 2002
- K. Jamsa, L. Klauder - *Totul despre C și C++, manual fundamental de programare în C și C++*, Editura Teora, București, 2003
- Liviu Negrescu - *Introducere în limbajul C*, Editura Mictoinformatica, Cluj-Napoca
- Grigore Albeanu - *Programarea în Pascal și Turbo Pascal. Culegere de probleme*, Editura Tehnică, București, 1994
- Valeriu Iorga, Eugenia Kalisz, Cristian ăăpuș - *Concursuri de programare. Probleme și soluții*, Editura Teora, București, 1997
- Dorel Lucanu - *Proiectarea algoritmilor. Tehnici elementare*, Editura Universității "Al. I. Cuza" Iași, 1993
- Emanuela Mateescu, Ioan Maxim - *Arbori*, Editura ăara Fagilor, Suceava, 1996
- Victor Mitrană - *Provocarea algoritmilor. Probleme pentru concursurile de informatică*, Editura Agni, București, 1994
- Doina Rancea - *Limbajul Turbo Pascal*, Editura Libris, Cluj, 1994
- Dennis M. Ritchie, Brian. W. Kernighan - *The C Programming Language*, Prentice-Hall, Inc., Eaglewood Cliffs, New Jersey, 1978
- Bogdan Pătruț - *Învățați limbajul Pascal în 12 lecții*, Editura Teora, București 1997
- Bogdan Pătruț - *Aplicații în Visual Basic*, Editura Teora, București, 1998-2004
- Bogdan Pătruț - *Aplicații în Delphi*, Editura Teora, București, 2001-2004
- Sorin Tudor - *Tehnici de programare*, Editura Teora, București, 1994
- * * * - *Colecția Gazeta de matematică*, 1996-1997, Editura Libris, Cluj

CUPRINS

Introducere	3
Capitolul 0. Scurtă introducere în limbajul C	5
Capitolul 1. Elemente de bază ale limbajului C/C++	32
Capitolul 2. Tablouri și pointeri	56
Capitolul 3. Recursivitate	74
Capitolul 4. Structuri și tipuri definite de programator	89
Capitolul 5. Exploatarea fișierelor	116
Capitolul 6. Algoritmi de teoria grafurilor	128
Capitolul 7. Grafică în C/C++	135
Capitolul 8. Programare orientată pe obiecte în <i>Borland C++</i>	153
Capitolul 9. Probleme recapitulative	166
Bibliografie	216

Vizitați www.edusoft.ro !

Cărți și software educațional

Carti, soft educational - EduSoft - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites Media Print

Address <http://www.edusoft.ro/> Go

EduSoft Carti, manuale, auxiliare didactice
Software educational
Jocuri de strategie

600065 Bacau, str. 9 Mai, nr. 82, sc. C, ap. 13
Tel. 0234/206090, 0723187198, 0741638182
www.edusoft.ro, contact@edusoft.ro

Cautare: Cautare Avansata Vezi Cosul **Publicare carte** Ajutor

Cautare

Subiectul cautarii:

START!

Publicam cartea ta pe banii nostri!

Domenii

- Toate produsele
- CD/Carti de Arta
- CD/Carti de Biologie,
- Chimie
 - CD/Carti pentru copii
 - CD/Carti de Drept
 - CD/Carti de Economie
 - Enciclopedii
 - CD/Carti de Geografie
 - CD/Carti de Informatica - programare
 - CD/Carti de Informatica - teorie
 - CD/Carti de Informatica - utilizare
- Jocuri
- CD/Carti de Matematica
- CD/Carti de Stiinte socio-umane
- Alte domenii

carti, manuale

<p>20 aplicatii in Delphi si Visual Basic Bogdan Patrut, 15 lei</p>	<p>Analiza numerica in Pascal si C Mihai Talmadu, Alina Mihaela Patriciu 10 lei</p>	<p>Aplicatii PowerPoint educationale Bogdan Patrut, Monica Patrut 11 lei</p>	<p>Carte romaneasca de invatatura a lui Varlaam. Studiu lingvistic - Fonetica Luminita Druga, 10 lei</p>
<p>Chimie anorganica de la A la Z. Sinteze, algoritmi, teste de evaluare Lidia Minza, 11 lei</p>	<p>Constructia compilatoarelor folosind Flex si Bison Anthony A. Aaby, Dan Popa 8 lei</p>	<p>Dictionar explicativ scolar de termeni istorici si arhaici Ioan Murariu, 10 lei</p>	<p>Geografia fizica a Romaniei. Sinteze pentru Teste nationale / Capacitate Sorin Gutunoi, 15 lei</p>

Error on page.

Start Total Command... NC Left c_cpp_vc.DOC... Inbox - Microsof... Carti, soft edu... Internet

8:09 AM