## Metoda bulelor

1. sortbul(int tab[], int m){ int i, j, temp;
2. for(i=1; i<m; i++)
3. for(j=i; j>0; j--)
4. if(tab[j]<tab[j-1]){
5. temp= tab[j];
6. tab[j]= tab[j-1];
7. tab[j-1]=temp; }}

## Sortarea prin inserție

1. sortins(int tab[], int m){ int i, j, temp;
2. for(i=0; i<m-1; i++)
3. for(j=0; j>m-i; j--)
4. if(tab[i+j]<tab[i]){
5. temp= tab[i];
6. tab[i]= tab[i+j];
7. tab[i+j]=temp;}}

## Sortarea prin selecție

Se caută minimul și se pune pe prima poziție. Al doilea minim pe poziția a doua. Se repetă acțiunea…

1. sortselectie(int tab[], int m){ int i, j, temp, min, locmin;
2. for(i=0; i<m-1; i++){
3. min = tab[i];
4. locmin= i;
5. for(j=i+1; j<m; j++)
6. if(tab[j]< min){
7. min = tab[j];
8. locmin= j;
9. tab[locmin]= tab[i];
10. tab[i]= min; } }}

## Algoritmul Shell

1. sortshell(int tab[], int n){ int i, j, pas, temp;
2. for(pas=n/2; pas>0; pas/=2)
3. for(i=pas; i<n; i++)
4. for(j=i-pas; j>=0&& tab[j]>tab[j+pas];j-=pas){
5. temp= tab[j];
6. tab[j]= tab[j+pas];
7. tab[j+pas]=temp;} return;
8. }

## Algoritmul QuickSort

1. voidquicksort(float tab[], int st, intdr)
2. { int i, j;
3. floattemp, pivot;
4. pivot = tab[st];
5. i = st;
6. j =dr;
7. while(i < j)
8. {
9. while(tab[i]<= pivot && i <dr) i++;
10. while(tab[j]> pivot) j--;
11. if(i < j){
12. temp= tab[i];
13. tab[i]= tab[j];
14. tab[j]=temp;
15. } }
16. if(j > st){
17. tab[st]= tab[j];
18. tab[j]= pivot;}
19. if(st < j-1)quicksort(tab, st, j-1);
20. if(j+1<dr)quicksort(tab, j+1, dr);}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

int sort\_function( const void \*a, const void \*b);

char list[7][4] = { "cat", "car", "cab",

"cap", "can" ,"big", "bag"};

int main(void)

{ int x;

clrscr();

// apelam la qsort

qsort((void \*)list,7,sizeof(list[0]),

sort\_function);

//afisam tabloul sortat

for (x = 0; x < 7; x++)

printf("%s\n", list[x]);

getch();

return 0;}

// Functia utilizatorului

int sort\_function( const void \*a, const void \*b)

{return( strcmp((char \*)a,(char \*)b) );}

1) Să se realizeze un program care citeşte folosind elemetele până acum introduse, caracter cu caracter de la tastatură până la întâlnirea caracterului ‘$’ şi le trece într-un fişier al cărui nume se trece de la tastatură sau se specifică în linia parametrilor de comandă. (scriere in fisier)

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{**

FILE **\***pf**;** *//am declarat variabila pointer de fisier*

char c**,\***nume**;**

printf**(**"Dati numele fisierului de lucru: "**);**

nume**=(**char**\*)**malloc**(**20**\***sizeof**(**char**));** *//in C standard nu era necesar sa scriu tipul, adica char\**

gets**(**nume**);**

if **((**pf**=**fopen**(**"C:\\test.txt"**,** "w"**))==**NULL**)** *//deschidem sau cream un fisier pt. scriere (eventual suprascriere)*

*//cu a in loc de w am fi creat o adaugare la continutul existent la un moment dat in fisier*

**{** printf**(**"Nu se poate deschide fisierul.\n"**);**

getch**();**

exit**(**1**);** *//iesire fortata din program cu cerinta nesatisfacuta*

**}**

puts**(**"Scrieti un text. Cu caracterul '$' se incheie:"**);**

do **{**

c**=**getchar**();** *//in c se introduce un caracter citit de la tastatura*

putc**(**c**,**pf**);** *//se putea si fputc(c,pf);*

**}** while **(**c **!=**'$'**);**

getchar**();**

fclose**(**pf**);}**

2) Scrierea intr-un fisier.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** char c**;**

FILE **\***stream**;** *//declarare stream (flux de date)*

stream **=** fopen**(**"C:\\test.txt"**,** "w"**);** *//deschidere fisier*

puts**(**"Scrieti un text. Cu caracterul '$' se incheie:"**);**

do **{**

c**=**getchar**();** *//in c se introduce un caracter citit de la tastatura*

putc**(**c**,**stream**);** *//se putea si fputc(c,pf);*

**}** while **(**c **!=**'$'**);** getchar**();** fclose**(**stream**); }**

3) Să se scrie un program care să citească dintr-un fisier orice caracter ASCII şi să afişeze pe ecran conţinutul acestuia.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** FILE **\***pf**;** *//declarare stream (flux de date)*

pf **=** fopen**(**"C:\\test.txt"**,** "r"**);** *//deschidere fisier*

char c**,\***nume**;**

if**(**pf **==** NULL**)**

**{** printf **(**"Eroare"**);**

getch**();**

exit**(**1**); }**

printf**(**"Dati numele fisierului de lucru: "**);**

nume**=(**char**\*)**malloc**(**20**\***sizeof**(**char**));**

gets**(**nume**);**

c**=**fgetc**(**pf**);** *//citeste un caracter din fisier sau c=getc(pf)*

while **(**c**!=**EOF**)** *// test daca s-a ajuns la sfarsitul fisierului*

**{** putchar**(**c**);** *//afiseaza un caracter pe ecran*

c**=**fgetc**(**pf**); }**

fclose**(**pf**);** getchar**(); }**

4) Citire dintr-un fisier a unui sir de caractere cu functia fread/ fgets.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** char s**;**

FILE **\***stream**;** *//declarare stream (flux de date)*

stream **=** fopen**(**"D:/test.txt"**,** "r"**);**

*//deschidere fisier*

if**(**stream **==** NULL**)**

**{** printf **(**"Nu se poate deschide fisierul.\n"**);**

getch**();**

exit**(**1**); }**

fread**(&**s**,** sizeof**(**char**),**100**,**stream**);** *//sau fgets(&s,100,stream);*

printf**(**"%s"**, &**s**);**

getchar**();**  fclose**(**stream**); }**

5) Să se realizeze un program care copiază conţinutul unui fişier de pe disc într-altul al cărui nume este specificat în program.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** char sir**[**256**];**

FILE **\***citire**, \***scriere**;**

citire **=** fopen**(**"C:\\test.txt"**,** "r"**);**

scriere **=** fopen**(**"D:/test.txt"**,** "w"**);**

if**(**citire **==** NULL**)**

**{** printf **(**"Nu se poate deschide fisierul de citire.\n"**);**

getch**();**

exit**(**1**);}**

fgets**(**sir**,**sizeof**(**sir**),**citire**);** */\* citeste un sir pana se ajunge la lungimea de caractere specificata din fisierul de citire \*/*

fputs**(**sir**,**scriere**);** */\*scrie sirul de caractere in fisierul de scriere\*/*

printf**(**"Textul copiat este: %s"**,** sir**);**

fclose**(**citire**);**  fclose**(**scriere**);**  getchar**(); }**

6) Indicând un număr de octet, să se afişeze conţinutul octetului specificat şi restul conţinutului fişierului de la octetul specificat.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** Char s**;**

FILE **\***stream**;** *//declarare stream (flux de date)*

stream **=** fopen**(**"D:/test.txt"**,** "r"**) //***deschidere fisier*

fseek **(**stream**, (**3**)-**1**,** SEEK\_SET**);**

if**(**stream **==** NULL**)**

**{** printf **(**"Nu se poate deschide fisierul.\n"**);**

getch**();**

exit**(**1**); }**

printf**(**"\nElementul de la pozitia 3 este: %c.\n" **,** fgetc**(**stream**));**

puts**(**"\nRestul fisierului de la pozitia mai sus mentionata este:"**);**

while **(!**feof**(**stream**))** *//se testeaza daca s-a ajuns la sf. de fisier*

**{** printf**(**"%c"**,**fgetc**(**stream**)); }**

getchar**();**

fclose**(**stream**);** *//inchidere*

**}**

7) Sa se scrie un program care face calcule si le afiseaza intr-un fisier, folosind functia fprintf.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** FILE **\***pf**;**

pf **=** fopen**(**"D:/test.txt"**,** "r"**);**

int cant**=**20**;**

float pret**=**2378.64**;**

if**(**pf **==** NULL**)**

**{** printf **(**"Eroare"**);**

getch**();**

exit**(**1**); }**

fprintf**(**pf**,** "\tValoare totala:\n"**);** *//afiseaza in fisier*

fprintf**(**pf**,** "\t\tLei: %14.2f\n"**,**cant**\***pret**);** *//afiseaza in fisier*

printf**(**"\tValoare totala:\n"**);***//afiseaza pe ecran*

printf**(**"\t\tLei: %14.2f\n"**,**cant**\***pret**);***//afiseaza pe ecran*

fclose**(**pf**);** getchar**(); }**

8) In fisierul “D:/test.txt” se aflã mai multe numere despãrtite intre ele prin spatiu. Sã se sorteze elementele acestui vector folosind metoda inteschimbãrii (selectiei directe sau a paharelor), folosind un fisier auxiliar a.i. primul fisier va fi sters iar noul fisier va primi numele celui vechi.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** int n**,**i**=**0**,**j**=**0**,**temp**,**v**[**50**];**

FILE **\***f**,\***g**;**

if **((**f**=**fopen**(**"D:\\test.txt"**,** "r"**))==**NULL**)**

printf**(**"Eroare la deschiderea fisierului de citire"**);**

else

if **((**g**=**fopen**(**"C:\\numere.txt"**,**"w"**))==**NULL**)**

**{** printf**(**"Eroare la deschiderea fisierului de scriere.txt"**);**

fclose**(**f**); }**

else

**{** while**(!**feof**(**f**))**

**{** fscanf**(**f**,**"%d"**,&**v**[**i**]);**

i**++;}**

n**=**i**;** *// valoarea finalã a lui i reprezintã nr. de elem. ale vect.*

*//sortarea elementelor din fisier*

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

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

if **(**v**[**j**]<**v**[**i**])**

**{** temp**=**v**[**i**];**

v**[**i**]=**v**[**j**];**

v**[**j**]=**temp**;}**

*//afisarea elementelor din fisier*

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

fprintf**(**g**,**"%5d"**,**v**[**i**]);** fclose**(**f**);**

remove**(**"D:\\test.txt"**);** fclose**(**g**);**

rename**(**"C:\\numere.txt"**,** "C:\\test.txt"**);**

printf**(**"\nOperatia s-a realizat"**);**

**}**

getchar**(); }**

9) Se citeşte un fişier text ce conţine numere întregi despărţite prin spaţiu sau ENTER. Să se afişeze minimul si maximul dintre valorile existente în acel fişier, precum si ampliturinea sirului de nr (diferenta absoluta intre extreme).

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<limits.h>

void main**()**

**{** FILE **\***stream**;** */\*declarare stream (flux de date)\*/*

stream **=** fopen**(**"D:\\test.txt"**,** "r"**);** */\*deschidere fisier\*/\*

if**(**stream **==** NULL**)**

**{** printf **(**"Eroare"**);**

getch**();**

exit**(**1**); }**

int x**;** *//variabila in care retinem numerele citite*

int maxim**=-**INT\_MAX**;** *//consideram maximul cel mai mic numar intreg*

int minim**=**INT\_MAX**;***//consideram minimul cel mai mare numar intreg*

while**(!**feof**(**stream**))** *//atata timp cat nu e sf de fisier*

**{**

fscanf**(**stream**,**"%d"**,&**x**);** *//citeste sirul de nr si le compara cu x*

if **(**x**>**maxim**)**

maxim**=**x**;**

else

if **(**x**<**minim**)**

minim**=**x**;}**

fclose**(**stream**);**

printf**(**"Maximul din fisier este: %d\n"**,**maxim**);**

printf**(**"Minimul din fisier este: %d\n"**,**minim**);**

printf**(**"Amplitudinea din fisier este: %d\n"**,**maxim**-**minim**);**

getchar**();}**

10) Se citeşte un fişier text ce conţine numere întregi despărţite prin spaţiu sau ENTER. Să se afişeze suma numerelor pare si suma numerelor impare dintre valorile existente în acel fişier.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main**()**

**{** FILE **\***stream**;** */\*declarare stream (flux de date)\*/*

stream **=** fopen**(**"D:\\test.txt"**,** "r"**);** */\*deschidere fisier\*/*

if**(**stream **==** NULL**)**

**{** printf **(**"Eroare"**);**

getch**();**

exit**(**1**); }**

int x**;** *//variabila in care retinem numerele citite*

int sumaImpar**=**0**;**

int sumaPar**=**0**;**

while**(!**feof**(**stream**))** *//atata timp cat nu e sf de fisier*

**{**fscanf**(**stream**,**"%d"**,&**x**);***//citeste sirul de nr si le compara cu x*

if**(**x **%** 2 **==** 0**)**

sumaPar**=**sumaPar **+** x**;**

if**(**x **%** 2 **!=** 0**)**

sumaImpar**=**sumaImpar **+** x**; }**

fclose**(**stream**);**

printf**(**"sumaImpar este: %d\n"**,**sumaImpar**);**

printf**(**"sumaPar este: %d\n"**,**sumaPar**);**

getchar**(); }**

11) Se presupune un fişier text ("vocale.txt") ce conţine o inşiruire de caractere. Să se contorizeze apariţia fiecărei vocale în fişier.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<limits.h>

void main**()**

**{** char c**;**

int a**=**0**,**e**=**0**,**i**=**0**,**o**=**0**,**u**=**0**;** *//a=nr. de aparitii a vocalei "a", s.a.m.d.*

FILE **\***stream**;** */\*declarare stream (flux de date)\*/*

stream **=** fopen**(**"D:\\vocale.txt "**,** "r"**);** */\*deschidere fisier\*/*

if**(**stream **==** NULL**)**

**{** printf **(**"Eroare"**);**

getch**();**

exit**(**1**); }**

while**(!**feof**(**stream**))** *//atata timp cat nu e sf de fisier*

**{**fscanf**(**stream**,**"%c"**,&**c**);** *//sau c=getc(f); sau c=fgetc(f);*

*switch****(****c****)***

**{** case 'a'**:** a**++;** break**;**

case 'e'**:** e**++;** break**;**

case 'i'**:** i**++;** break**;**

case 'o'**:** o**++;** break**;**

case 'u'**:** u**++; }; }**

fclose**(**stream**);**

printf**(**"Vocala \"a\" aparare de %d ori.\n"**,**a**);**

printf**(**"Vocala \"e\" aparare de %d ori.\n"**,**e**);**

printf**(**"Vocala \"i\" aparare de %d ori.\n"**,**i**);**

printf**(**"Vocala \"o\" aparare de %d ori.\n"**,**o**);**

printf**(**"Vocala \"u\" aparare de %d ori.\n"**,**u**);**

getchar**(); }**

**Citirea unui sir de la sfirsit la inceput :**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

typedef struct st

{ char ch;

struct st \*ps;

} stack;

int main()

{ stack \*p,\*q; char a;

p=NULL; // initializarea pointerului

puts("Introdu textul,finisat cu ’1’");

// ciclu pentru implementarea stivei

do {

a=getchar();

q=(struct st \*)malloc(sizeof(stack));

q->ps=p; p=q; q->ch=a; }

while (a!='1');

puts("Textul inversat:");

do { //Schimbam adresa pointerului p

p=q->ps;

//eliberam memoria la care indică q

free(q); q=p;

// q şi p acum indică aceeaşi adresă

printf("%c",p->ch); // afişăm

} while (p->ps!=NULL);

getch(); }

**Inserarea-adaugarea unui elemet in coada .**struct Nod

{ int val;

Nod\* next; };

Nod\* put(Nod\* c, int v)

{ Nod\* nou = new Nod;

nou->val = v;

nou->next = NULL;

if(!c) return nou;

else

{ Nod\* t=c;

while(t->next)

t=t->next; t->next=nou; return c; } }

**Extragerea unui elemnt din coada.**

Nod\* get( Nod\* c, int \*v)

{ if (c)

{ \*v = c->val;

Nod\* t = c;

c = c->next;

delete t; return c; }

return c; }

**In program se verifică dacă două fişiere sunt identice. După numărul, suma şi ordinea elementelor în fişiere.Pentru verificarea cerută sunt utilizate două stive.**

...

struct Elm {

int inf;

struct Elm \*lnk;};

struct Elm \*top1, \*top2;

int i = 1, l = 0, s = 0;

FILE \*ptr;

void pushStack(int, int); // Prototipul functiei

void main() { clrscr();

top1 = top2 = NULL;// initializam pointerii

if ((ptr = fopen("l1.txt", "r")) = = NULL)

{ printf("Error 1");//fisierul nu exista

return 0; } else {

fscanf(ptr,"%d", &i);// citim date din fisier

while (!feof(ptr)) { pushStack(i, 0);

// apelam la functia crearii stivei

fscanf(ptr,"%d", &i);

//continuăm citirea din fişier

s += i; //suma el. citite din fişier

l++; //numarul elementelor în stivă

} fclose(ptr); }

if ((ptr = fopen("l2.txt", "r")) == NULL) { printf("Error 2"); return 0; }

else { fscanf(ptr,"%d", &i);

while (!feof(ptr)) { pushStack(i, 1);

fscanf(ptr,"%d", &i); s -= i; l--; }

fclose(ptr); }

if ((l == 0) && (s == 0)) { // controlăm identitatea stivelor create

int difCount = 0; struct Elm \*tmp; while (top1 != NULL)

{ tmp = top2; while (tmp != NULL) {

if (tmp->inf == top1->inf)

goto ret; tmp = tmp->lnk; }

difCount++; ret: top1 = top1->lnk;

} if (difCount != l)

printf("Stivele difera difera cu %d

elemente \n", difCount);

else

printf("Stivele sunt asemenea\n");

}

else

printf("Stivele difera [lung./sumele]\n");

return 0;}

void pushStack(int n, int st)

{ struct Elm \*p;

p = (Elm\*) malloc(sizeof(top1));

//crearea noului element in stiva

p->inf = n; if (st == 0)

{ p->lnk = top1; top1 = p;

} else { p->lnk = top2;

top2 = p; }}

**Căutarea într-un arbore binar ordonat**

*// cãutare nerecursivã în arbore ordonat*

***tnod \* find ( tnod \* r, int x)***

***{*** *while (r!=NULL) // cat timp se mai poate cobora in arbore*

*{ if (x == rval)* ***return r****; // x gasit la adresa r*

*if ( x < rval)*

*r=rst; // cauta spre stanga*

*else*

*r=rdr; // cauta spre dreapta*

*}* ***return NULL****;* ***}***

**Căutarea într-un arbore binar ordonat**

*// cautare recursiva în arbore ordonat*

**tnod \* find ( tnod \* r, int x)**

**{**  if (r==NULL) return NULL; // x negasit in arbore

if (x == rval) return r; // x gasit in nodul r

if ( x < rval)

return find (rst,x); // cauta in subarb stanga

else

return find (rdr,x); // cauta in subarb. dreapta

**}**

**CĂUTARE SECVENTIALĂ**

*Se compara pe rind elementele tabloului cu cheiea* ***key*** *pina cind,*

*fie se gaseste egalitatea* ***v[i]=key****, fie s-a ajuns la sfirsitul tabloului.*

*Exemplu de cod functiei de cautare liniară/secvențială in C/C++*

***int caut\_secv( int key, int v[], int n )***

*{ int i=0;*

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

*if (v[i] == key )* ***return i;***

***return -1;***

*}*

Un alt exemplu de implementare funției de cautare liniară

int LinearSearch(int \*x, int k, int key){

int i = 0;

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

if ( x[i] == key )

break;

return i < k ? i : -1;

}

**Cautare binară**

int main()

{ int c, st, dr, m, n, key, a[100];

printf(“Introdu numarul de of elemente\n");

scanf("%d", &n);

printf(“Introdu %d de elemente\n", n);

for (c = 0; c < n; c++) scanf("%d", &a[c]);

printf(“Introdu elemental de cautat\n");

scanf("%d", &key); st = 0;

dr = n - 1; m = (st + dr) / 2;

while (st <= dr)

{ if (a[m] < key) st = m + 1;

else if (a[m] == key)

{ printf(“elem. key %d este plasat in pozitia %d.\n",

key, m+1); break; }

else

dr = m - 1; m = (st + dr)/2; }

if (st > dr)

printf(“Elementul key= %d nu apartine sirului !\n", key);

return 0; }

**Cautarea prin interpolare**

Dacă se cunoaşte că elementul cautat **key** poate fi găsit între elementele unui vector **a[l]** și **a[r] ( l**- stînga, **r**-dreata, **a[l]< key < a[r]**) poate fi utilizată cautarea prin interpolare. Aceasta este similară cu cautarea binară, dar foloseşte o altă formulă pentru calculul lui "m", si anume*:*

**m = l + (k - a[l]) \* (r - l) / (a[l] - a[r])**

ceea ce conduce la o delimitare mai rapida a zonei din tablou in care s-ar putea gasi **key**.

*int InterpolSearch ( int a[], int n, int key)  
{  
 int m, l=0, r=n-1;  
 while (a[l]<=key && a[r]>=key)  
 { m=l+((key-a[l])\*(r-l))/(a[r]-a[l]);  
 if (a[m]<key) l=m+1;  
 else if (a[m]>key) r=m-1;  
 else return m; }  
 if (A[l]==key) return left;  
 else return -1; }*

**Arbori:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAXWORD 100

**struct tnode**

{ char \*word;

int count;

struct tnode \*left;

struct tnode \*right;};

**struct tnode** \* addtree( struct tnode \*, char \* );

**void treeprint**( struct tnode \* );

**int main()**{ struct tnode \*root;

char word[MAXWORD];

FILE \*fp;

root = NULL;

fp = fopen("text.dat", "rt");

if( fp == NULL) {

fprintf(stderr, "Nu pot deschide fisierul\n\b");

return 1; }

do

{ fscanf(fp, "%s", word);

if( word[0] != '\0' )

**root = addtree(root, word);**

} while(!feof(fp));

**treeprint( root );**

return 0; }

**struct tnode \* addtree( struct tnode \*p, char \*w )**

{ int cond; if( p == NULL)

{ p = malloc( sizeof(struct tnode) );

p->word = strdup(w);

p->count = 1;

p->left = p->right = NULL; }

else if( (cond = strcmp(w, p->word)) == 0 )

p->count++;

else if( cond < 0 )

p->left = addtree( p->left, w);

else

p->right = addtree( p->right, w);

return p;}

**void treeprint( struct tnode \* p)**

{ if( p != NULL ) {

treeprint( p->left );

printf("%s %d\n", p->word, p->count );

treeprint( p->right ); }}

**Exemplu 2 de cod pentru cautare valorii minime in arbore binar ordonat**

#include <stdio.h>

#include <conio.h>

#include <alloc.h>

struct Node

{ int d;

Node \*left;

Node \*right; };

Node \* insert(Node \*root,int d);

Node \* tree\_min(Node \*p);

Node \* root, \*pf, \*pv, \*p;

int main() {

*int b[] = { 12, 4, 22, 6, 9, 1, 2, 3 } ;*

*clrscr();*

*printf("%d\n",b[0]); root=NULL;*

*pf = insert( root, b[0] );*

*p=pf;*

*for (int i=1;i<8;i++) p=insert(pf, b[i]);*

*p = tree\_min( pf );*

*printf("min=%d", p->d);*

*getch(); return 0; }*

*//* *Funcţia creării unui arbore binar ordonat*

*Node \* insert(Node \*root, int d)*

*{ Node \*pv;*

*pv = root;*

*if (root==NULL)*

*{ pv = new Node;*

*pv->d=d; pv->left=0; pv->right=0; }*

*else*

*{ if (d < pv->d) pv->left = insert(pv->left,d);*

*else pv->right = insert(pv->right,d); }*

*return pv; }*

*/\* Funcţia de cautare unui nod cu cheie minimă\*/*

*Node \* tree\_min(Node \*p)*

*{ while (p->left !=NULL) { p=p->left;*

*printf("%d ",p->d); } return p; }*

***Parcurgerea adincime-latime***

#include <stdio.h>

…

**struct nod**

{ int inf;

int dim;

struct nod\* tab[100];};

**struct nod \* crearb();**

/\* Functia pentru parcurgerea in adincime \*/

**void padinc(struct nod \*);**

/\*Functia pentru parcurgerea in latime\*/

**void plat(struct nod \*);**

int main(){ struct nod \*root; clrscr();

/\*Creez arborele\*/

root = crearb();

printf("Parcurgere in adincime:"); padinc(root);

printf("\nParcurgere in latime:"); plat(root);

return 0;}

struct nod \* crearb(){ int inf, n, i;

struct nod \*p;

printf("Inf=");

scanf("%d", &inf);

p = malloc( sizeof(struct nod) );

p->inf = inf;

printf("Cati fii are?");

scanf("%d", &n);

p->dim = n; for(i = 0; i < n; i++)

p->tab[i] = crearb(); return p; }

void padinc(struct nod \*p) { int i;

if( p != NULL ) {

printf("%d", p->inf);

for( i = 0; i < p->dim; i++)

padinc( p->tab[i] ); } }

void plat(struct nod \*p) { struct nod \*queue[100];

struct nod \*t; int front = 0, back = 0; int i;

queue[ front++ ] = p;

while ( ! (front == back) ) {

t = queue[ back ]; back++;

printf("%d", t->inf);

for(i = 0; i < t->dim; i++)

queue[ front++ ] = t->tab[i]; }}