UNIVERSITATEA TEHNICĂ A MOLDOVEI

STRUCTURI DE DATE ŞI ALGORITMI

CULEGERE DE PROBLEME

pentru executarea lucrărilor de laborator şi de control

Chişinău
2014

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Facultatea Calculatoare, Informatică şi Microelectronică
Catedra Informatică Aplicată

STRUCTURI DE DATE ŞI ALGORITMI

CULEGERE DE PROBLEME

pentru executarea lucrărilor de laborator şi de control

Chişinău
Editura „Tehnica-UTM”
2014

Îndrumarul metodic conține exemple de programe şi lucrări individuale pentru executarea lucrărilor de laborator la cursul „Structuri de date şi algoritmi”.

Scopul principal al executării lucrărilor de laborator propuse este însuşirea principalelor metode de utilizare şi prelucrare a tipurilor structurate de date şi a celor mai simple structuri dinamice de date.

Este destinat studenților facultății Calculatoare, Informatică şi Microelectronică, care studiază cursul „Structuri de date şi algoritmi”, precum şi studenților altor facultăți ale U.T.M.

Autori: conf. univ., dr. L. Luchianova
conf. univ., dr. L. Stadler

Redacto responsabil: conf. univ., dr. V. Moraru
Recenzent: conf. univ. L. Carcea

Redactor: E.Gheorghişteanu

Bun de tipar 11.11.14 Formatul hârtiei 60x84 1/16
Hârtie ofset. Tipar RISO Tirajul 60 ex.
Coli de tipar 2,0 Comanda nr. 99

MD – 2004, UTM, Chişinău, bd. Ştefan cel Mare, 168
Editura „Tehnica-UTM”
MD – 2068, Chişinău, str. Studenților, 9/9

CUPRINS

1. Lucrarea de laborator nr.1.
Structuri de date. Fişiere şi înregistrări..... 4

2. Lucrarea de laborator nr.2.
Şiruri de caractere……. 10

3. Lucrarea de laborator nr.3.
Recursivitate……. 16

4. Lucrarea de laborator nr.4.
Structuri dinamice de date…… 22

5. Lucrarea de laborator nr.5.
Algoritmi de sortare şi căutare…….. 27

Lucrarea de laborator nr.1
Tema: *Structuri de date. Fişiere şi înregistrări*

Exemplu de realizare:

Să se elaboreze un program, care i-ar permite managerului pe credit să obțină lista clienților cu sold nul (clienților, care nu sunt datori bani), a clienților cu sold pozitiv (cărora compania le este datoare o oarecare sumă) şi a clienților cu sold negativ (care sunt datori companiei bani pentru mărfurile şi serviciile deja prestate). În program să se prevadă crearea fişierului. Informația referitor la fiecare client este prezentată prin numărul contului, nume şi bilanț.

Programul:

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

struct info{ int account;
           char name[30];
           double balance;};
int main()
{  info clt;
   int request=0;
   int f;
   FILE *cfPtr;

   clrscr();
   if ((cfPtr = fopen("clients.dat","w"))==NULL)
       printf("Fişierul nu poate fi deschis\n");
   else{
      printf("Introduceți numărul contului,
              numele şi bilanțul.\n");
      printf("Introduceți EOF pentru a finaliza
              introducerea.\n");
      printf("? ");
```

```
scanf("%d%s%lf",&clt.account,clt.name,&clt.balanc
e);
while ( !feof(stdin) )
  {  fprintf(cfPtr,"%d%s%.2lf\n",clt.account
     clt.name,clt.balance);
     printf("?");scanf("%d%s%lf",&clt.account,clt
       .name,&clt.balance);
   }
      fclose( cfPtr );
}
if ((cfPtr = fopen("clients.dat","r"))==NULL)
      printf("Fişierul nu poate fi deschis\n");
else
 {
    while (request!=4)
      {
       clrscr();
       printf("Introduceţi interpelarea\n"
          "1 - Lista clienţilor cu sold nul\n"
          "2 - Lista clienţilor cu sold
               negativ\n"
          "3 - Lista clienţilor cu sold
              pozitiv\n"
          "4 - End run\n");
       scanf("%d",&request);
       fscanf(cfPtr,"%d%s%lf",&clt.account,
clt.name, &clt.balance);

      switch (request)
        { case 1:
           f=1;
           printf("\n Conturi cu  sold nul:\n");
           while (!feof(cfPtr)){
           if (clt.balance==0)
             {printf("%-10d%-13s%7.2f\n",
              clt.account,clt.name,clt.balance);
              f=1;}
```

```
            fscanf(cfPtr,"%d%s%lf",&clt.account,
            clt.name, &clt.balance);
            }
           if(f==0)printf(" nu exista!\n”);
            break;
        case 2:
           f=0;
           printf("\n Conturi cu sold
                  negativ:\n");
           while (!feof(cfPtr))
           {if (clt.balance<0)
              {printf("%-10d%-13s%7.2f\n",
              clt.account,clt.name,clt.balance);
               f=1;}
             fscanf(cfPtr,"%d%s%lf",&clt.account,
                    clt.name, &clt.balance);
            }
           if(f==0)printf(" nu exista!\n”)
           break;
        case 3:
          printf("\nConturi cu sold pozitiv:\n");
          while (!feof(cfPtr))
           { if (clt.balance>0)
              {printf("%-10d%-13s%7.2f\n",
               clt.account,clt.name,clt.balance;
               f=1;}
             fscanf(cfPtr,"%d%s%lf",&clt.account,
                    clt.name,&clt.balance);
           }
          if(f==0)printf("nu exista!\n”)
           break;
       }
    rewind(cfPtr);
    printf("\n");
    }
   printf("Finalizarea lucrului.\n");
   fclose(cfPtr);
 }
getch(); return 0;}
```

Sarcini individuale

Fişierul *student*
Structura înregistrării
- *NPP(40 caractere)*
- *Numărul seriei (1 caracter)*
- *Numărul grupei (1 caracter)*
- *Reuşita – 5 examene în fiecare din cele 10 sesiuni*
- *Forma instruirii (bugetară, contractuală) (1 caracter)*
- *Examen*
  - *denumirea disciplinei (10 caractere)*
  - *nota (1 caracter)*

Înregistrările sunt ordonate conform numărului seriei, în interiorul seriei – conform numărului grupei, în grupă - conform NPP.

Fişierul *ghidul telefonic*
Structura înregistrării
- *NPP(40 caractere)*
- *Adresa*
  - *strada (20 caractere)*
  - *numărul casei (4 caractere)*
  - *apartamentul (3 caractere*)

Înregistrările sunt ordonate conform NPP

Fişierul *registrul fişierelor*
Structura înregistrării
- *Numele fişierului (8 caractere)*
- *Specificarea (3 caractere)*
- *Data creării:*
  - *ziua (2 caractere)*
  - *luna (2 caractere)*
  - *anul (2 caractere)*
- *Numărul blocurilor*

Înregistrările sunt ordonate conform numelor fişierelor.

Fişierul *registrul cărţilor*
Structura înregistrării

- *Autorul (40 caractere)*
- *Denumirea (80 caractere)*
- *Anul ediţiei (4 caractere)*
- *Specialitatea (40 caractere)*

Înregistrările sunt ordonate conform numelor autorilor.

1. Să se creeze fişierul *student*. Introduceţi într-un fişier separat înregistrările din fişierul *student*, reordonându-le conform numărului grupei. În lista fiecărei grupe înregistrările să se ordoneze conform numărului seriei.

2. Să se creeze fişierul *student*. Introduceţi într-un fişier separat înregistrările din fişierul *student*, adăugând notele grupei indicate de la ultima sesiune.

3. Să se creeze fişierul *student*. Introduceţi într-un fişier separat înregistrările din fişierul *student*, adăugând datele despre studenţii noi.

4. Să se creeze fişierul *student*. Introduceţi în fişiere separate înregistrările referitor la studenţii cu diverse forme de studii.

5. Să se creeze fişierul *registrul cărţilor*. Introduceţi într-un fişier separat toate lucrările autorului indicat de utilizator.

6. Să se creeze fişierul *registrul cărţilor*. Să se găsească specialitatea pentru care numărul cărţilor este cel mai mare.

7. Să se creeze fişierul *registrul cărţilor*. Introduceţi într-un fişier separat datele referitor la toate ediţiile pentru specialitatea indicată.

8. Să se creeze fişierul *registrul fişierelor*. Să se transcrie fişierul creat într-un fişier separat, ordonând înregistrările conform numărului blocurilor.

9. Să se creeze fişierul *registrul fişierelor*. Introduceţi într-un fişier separat înregistrările referitor la fişierele create înainte de data indicată.

10. Să se creeze fişierul *registrul fişierelor*. Introduceţi într-un fişier separat înregistrările referitor la fişierele cu specificarea indicată.

11. Să se creeze fişierul *ghidul telefonic*. Să se selecteze şi să se introducă într-un fişier separat datele referitor la abonaţii, posesori de telefoane, numerele cărora încep cu două cifre indicate.

12. Să se creeze fişierul *ghidul telefonic*. Conform NPP să se găsească numerele telefoanelor şi adresele abonaţilor.

13. Să se creeze fişierul *ghidul telefonic*. Conform numărului telefonului să se găsească NPP şi adresa proprietarului.

14. Să se creeze fişierul *ghidul telefonic*. Să se găsească numărul staţiei telefonice (primele două cifre din numărul telefonului) care are cel mai mare număr de abonaţi.

15. Să se creeze fişierul *registrul fişierelor*. Introduceţi într-un fişier separat înregistrările referitor la fişierele, dimensiunile cărora depăşesc numărul indicat de blocuri.

Lucrarea de laborator nr.2
Tema: *Şiruri de caractere*

Exemplu de realizare:

Următorul program demonstrează acțiunea funcțiilor ***strcmp, strncmp, strchr, strcspn, strpbrk, strrchr.***

Programul:

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

void main()
{
int key;
key=0;
while (key!=6)
   {
   clrscr();
   puts("Select the function to explore");//Meniul
   puts("1 - strcmp & strncmp");
   puts("2 - strchr");
   puts("3 - strcspn");
   puts("4 - strpbrk");
   puts("5 - strrchr");
   puts("6 - EXIT");

   printf("\nkey= ");
   scanf("%d",&key);

   switch (key)
   {
   case 1:
{/*demonstrarea acţiunii funcţiilor strcmp şi
strncmp*/
   clrscr();
```

```
   const char *s1="Happy New Year";
   const char *s2="Happy New Year";
   const char *s3="Happy Holidays";

printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n
\n",
        "s1 = ",s1,"s2 = ",s2,"s3 = ",s3,
        "strcmp(s1, s2) = ", strcmp(s1,s2),
        "strcmp(s1, s3) = ", strcmp(s1,s3),
        "strcmp(s3, s1) = ", strcmp(s3,s1));
   printf("%s%2d\n%s%2d\n%s%2d\n",
        "strncmp(s1,s3,6)= ",strncmp(s1,s3,6),
        "strncmp(s1,s3,7)= ",strncmp(s1,s3,7),
        "strncmp(s3,s1,7)= ",strncmp(s3,s1,7));
  getch();
   break;
   } ;
   case 2:
{// demonstrarea acţiunii funcţiei strchr
   clrscr();
   const char *string = "this is a test";
   char character1 = 'a', character2 = 'z';

   if (strchr(string,character1)!=NULL)
      printf("\n'%c\' was found in \"%s\".\n",
           character1,string);
   else  printf("\n'%c\' was not found in
\"%s\".\n",
           character1,string);

   if (strchr(string,character2)!=NULL)
      printf("\n'%c\' was found in \"%s\".\n",
           character2,string);
   else  printf("\n'%c\' was not found in
\"%s\".\n",
           character2,string);
   getch();
   break;
   }
```

```
   case 3:{ /*demonstrarea acţiunii funcţiei
strcspn*/

   clrscr();
   const char *string1 ="The value is 3.14159";
const char *string2 = "1234567890";

printf("%s%s\n%s%s\n\n%s\n%s%u",
      "string 1 = ",string1,"string2 = ",string2,
      "The length of the initial segment of
string1",
      "containing no chsrscters from string2 = ",
      strcspn(string1,string2));
      getch();
      break;
   }
   case 4:{ /*demonstrarea acţiunii funcţiei
strpbrk*/

   clrscr();
   const char *string1="This is a test";
const char *string2="beware";

printf("%s\"%s\"\n'%c'%s\n\"%s\"\n",
      "Of the characters in ",string2,
      *strpbrk(string1,string2),
      "is the first character to appear
in",string1);
      getch();
      break;

   }
   case 5:{ /*demonstrarea acţiunii funcţiei
strrchr */

   clrscr();
   const char *string1="A zoo has many animals"
                "including zebras"    ;
int c='z';
```

```
printf("%s\n%s'%c'%s\"%s\"\n",
      "The number of string1 beginning with the ",
      "last occurrence of character",c,
      " is ", strrchr(string1,c));

   getch();
   break;
   }
   default: break;
   }
   }
}
```

Sarcini individuale

1. Fie date două şiruri *str1* şi *str2*. Să se examineze posibilitatea obținerii din şirul *str1* a şirului *str2* prin permutarea simbolurilor.

2. Să se scrie funcția ***convert*** (a transforma), care primeşte un parametru – şirul ***date*** (data), ce conține valoarea datei calendaristice ll/zz/aa (luna/ziua/anul). De exemplu, 14 decembrie 1960 va avea aspectul 12/14/60. Să se procedeze astfel ca luna sau anul în parametrul ***date*** să poată fi exprimate nu numai prin două cifre, ci, în caz de necesitate, şi printr-o singură cifră. De exemplu, 17 mai 1929 poate fi reprezentată cu acelaşi succes ca 05/17/29 sau ca 5/17/29. Problema funcției ***convert*** constă în transformarea valorii primite la forma 'Luna ziua, anul'. Astfel, rezultatul transformării datei indicate trebuie să fie 'Decembrie 14, 1960' (se presupune că toate datele se referă la secolul XX).

3. Să se elaboreze un program, care va introduce rândurile textului într-un tablou de simboluri s[100], utilizând funcția ***gets***. Să se afişeze rândurile în registrele superior şi inferior.

4. Să se elaboreze un program, care va introduce 4 şiruri care reprezintă valori întregi, transformă şirurile în numere întregi, sumează şi afişează suma celor patru valori.

5. Să se elaboreze un program, care va introduce câteva rânduri de text şi simbolul de căutare şi va utiliza funcţia ***strch*** pentru a determina numărul sumar de includeri ale simbolului în text.

6. Să se elaboreze un program, care va introduce câteva şiruri şi le va afişa pe acele care încep cu litera b.

7. Să se elaboreze un program, care va introduce 4 şiruri, care reprezintă valori cu virgulă mobilă, transformă şirurile, dublând valorile, sumează şi afişează suma celor patru valori.

8. Să se elaboreze un program, care va introduce câteva şiruri şi le va afişa pe acele care se termină cu "ED".

9. Să se scrie funcţia ***splite(name, first, last)***, care din parametrul ***name***, ce păstrează numele, prenumele şi patronimicul persoanei, le extrage în variabilele ***first*** (prenumele) şi ***last*** (numele). Numele şi prenumele sunt despărţite cu un oarecare număr de blancuri. De exemplu, după apelul ***splite('Ion Ursu', str1, str2)*** în ***str1*** trebuie să nimerească cuvântul ***'Ion'***, iar în ***str2*** – ***'Ursu'***. Este necesar, de asemenea, a prevedea depistarea şi prelucrarea datelor incorecte. În particular, dacă în ***name*** în genere nu va fi nici un blanc, procedura trebuie să stabilească în ambii parametri de ieşire valoarea specială ***'error'*** (eroare). De ce situaţii de eroare e necesar a se mai ţine cont?

10. Să se scrie funcţia ***sortmid***, care va sorta o serie din ***n*** şiruri în ordine alfabetică, bazându-se pe litera cu numărul de ordine ***k*** din fiecare şir, unde ***k*** este un parametru, transmis funcţiei ***sortmid***. De exemplu, dacă ***k*** = 3, atunci elementele şirului trebuie să fie sortate în ordine ascendentă conform valorii în cea de a treia literă a fiecărui şir. Dacă lungimea şirului este mai mică decât ***k***, atunci

vom presupune, că în calitate de litera cu numărul de ordine ***k*** al acestuia, real inexistentă, serveşte blancul.

11. Să se scrie funcţia care va efectua sortarea, analogică celei din problema 10, a şirurilor în ordine alfabetică inversă.

12. Fie că textul reprezintă o consecutivitate de şiruri. În fiecare şir este numele, prenumele şi patronimicul. Să se tipărească toate patronimicele în ordine alfabetică, de asemenea, textul ce conţine şiruri, constituite numai din nume şi prenume.

13. Fie că se introduc 10 nume arbitrare. Este necesar a le tipări în ordine alfabetică. A se încerca soluționarea problemei fără a sorta însuşi numele. Deoarece se cere tipărirea simplă a acestora în ordine alfabetică, să se definească un tablou, ce va conține numerele de rând ale numelor. În caz de necesitate a permutării să se efectueze permutarea nu a numelor, ci a numerelor de ordine ale acestora. Această metodă este foarte comodă, când e necesară sortarea unor obiecte complicate şi voluminoase.

14. Să se scrie funcţia ***count***, care primeşte doi parametri ***str1*** şi ***str2*** şi returnează numărul, care indică numărul de includeri a ***str2*** în ***str1***. Funcţia nu trebuie să-şi modifice parametrii. Orice literă în ***str1*** poate fi luată în consideraţie doar într-o singură includere ***str2***. De exemplu, ***count('balalaica', 'ala')*** trebuie să returneze 1, şi nu 2.

15. Să se scrie funcţiile ***encode*** (a codifica) şi ***decode*** (a decodifica), care primesc doi parametri ***str*** şi ***alpha***. Primul parametru reprezintă cuvântul supus codificării (decodificării), cel de-al doilea reprezintă o oarecare permutare a 26 litere ale alfabetului latin. Principiul transformării pentru codificare constă în următoarele. Dacă o oarecare literă în ***str*** este litera cu numărul de ordine ***k*** în alfabetul obişnuit, atunci în locul ei trebuie să fie luată litera din poziţia ***k*** a "noului" alfabet ***alpha***. Pentru funcţia de decodificare se utilizează principiul invers.

Lucrarea de laborator nr.3
Tema: *Recursia*

Exemplu de realizare:

Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int x^m \sin ax = \begin{cases} -\frac{x^m}{a}\cos ax + \frac{m}{a}\int x^{m-1}\cos axdx, m \geq 1 \\ -\frac{\cos ax}{a}, m = 0 \end{cases}$$

Programul:

```
#include  <math.h>
#include  <conio.h>
#include  <stdio.h>
// Prototipul funcţiei integral
float integral(float , float, int, int);
void main()
{
float i,x0,xm;
int m,a;

clrscr();
puts("Enter borders, power value and an
integer");
// Indicarea limitelor de integrare
printf("x0= "); scanf("%f",&x0); printf("xm= ");
scanf("%f",&xm);
// Indicarea puterii
printf("m= "); scanf("%d",&m); printf("a= ");
scanf("%d",&a); // Indicarea coeficientului
```

```
/*apelul funcţiei recursive, pentru calcularea
integralei*/
i=integral(x0,xm,m,a);
// Afişarea rezultatelor pe ecran
printf("I= %.3f",i);
getch();

}

float integral(float a, float b, int n, int k)
 { /*integrala se calculează conform formulei
      Newton-Leibniz*/
if (n>=1) // are loc apelul recursiv al funcţiei
  return (-pow(b,n)/k*cos(k*b)+ n/k*
          integral(a,b,n-1,k))-(-pow(a,n) /k*
          cos(k*b)+n/k*integral(a,b,n-1,k));
else
  if (n==0) // condiţia de ieşire din recursie
  return (- cos(k*b)/k)-(-cos(k*a)/k);
}
```

Sarcini individuale

1. Să se scrie o funcție recursivă, care pentru x real n întreg indicate va calcula valoarea x^n conform formulei:

$$x^n=\begin{cases}1, n=0\\ \dfrac{1}{x^{|n|}}, n<0\\ x * x^{n-1}, n>0\end{cases}$$

2. Să se elaboreze un program de calculare a funcției lui Akkerman pentru toate argumentele nenegative întregi m şi n:

$$A(m,n)=\begin{cases} A(0,n)=n+1 \\ A(m,0)=A(m-1,1), m>0 \\ A(m,n)=A(m-1,A(m,n-1)), m,n>0 \end{cases}$$

3. Să se scrie o funcție recursivă, care va calcula $y=\sqrt[k]{x}$ conform formulei:

$$y_0=1,\ y_{n+1}=y_n+\frac{\left(\frac{x}{y_n^{k-1}}-y_n\right)}{k},$$

n=0,1,2… . În calitate de răspuns să se admită aproximația, pentru care este satisfăcută condiția

$|y_n-y_{n+1}|<\varepsilon$, где $\varepsilon=0{,}0001$.

4. Pentru numărul real a>0 să se calculeze valoarea:

$$\frac{\sqrt[3]{a}\left|-\sqrt[6]{a^2+1}\right.}{1+\sqrt[7]{3+a}}.$$

Rădăcinile $y=\sqrt[k]{x}$ să se calculeze cu exactitatea E=0,00001 conform următoarei formule:

$$y_0=1,\ y_{n+1}=y_n+\frac{\left(\frac{x}{y_n^{k-1}}-y_n\right)}{k}$$

n=0,1,2… . În calitate de răspuns să se admită aproximația y_{n+1}, pentru care este satisfăcută condiția

$|y_n-y_{n+1}|<E$.

5. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int \sin^n x dx = \begin{cases} \underline{-\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n} \int \sin^{n-2} x dx, n > 2,} \\ \frac{x}{2} - \frac{1}{4} \sin 2x, n = 2 \\ -\cos x, n = 1; \end{cases}$$

6. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int \cos^n x dx = \begin{cases} \underline{-\frac{\cos^{n-1} x \sin x}{n} + \frac{n-1}{n} \int \cos^{n-2} x dx, n > 2,} \\ \frac{x}{2} + \frac{1}{4} \sin 2x, n = 2 \\ \sin x, n = 1; \end{cases}$$

7. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int \frac{dx}{\sin^n x} = \begin{cases} \underline{-\frac{1}{n-1} \cdot \frac{\cos x}{\sin^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\sin^{n-2} x}, n \geq 2,} \\ \ln tg \frac{x}{2}, n = 1 \\ x, n = 0; \end{cases}$$

8. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int \frac{dx}{\cos^n x} = \begin{cases} \underline{-\frac{1}{n-1} \cdot \frac{\sin x}{\cos^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\cos^{n-2} x}, n \geq 2,} \\ \ln tg \left(\frac{\pi}{4} + \frac{x}{2} \right), n = 1 \\ x, n = 0; \end{cases}$$

9. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int x^n e^{ax} dx = \begin{cases} \dfrac{x^n e^{ax}}{a} - \dfrac{n}{a}\int x^{n-1} e^{ax}, n > 1, \\ \dfrac{e^{ax}}{a^2}\ ax-1\ , n = 1; \end{cases}$$

10. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int x^n a^{mx} dx = \begin{cases} \dfrac{x^n a^{mx}}{n \ln a} - \dfrac{n}{m \ln a}\int x^{n-1} a^{mx}, n > 1, \\ \dfrac{x a^{mx}}{m \ln a} - \dfrac{a^{mx}}{m\ \ln a^{\,2}}, n = 1; \end{cases}$$

11. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int e^{ax} \cos^n x dx =$$

$$\begin{cases} \dfrac{e^{ax} \cos^{n-1} x\ \ a\cos x + n \sin x}{a^2 + n^2} + \dfrac{n\ \ n-1}{a^2 + n^2}\int e^{ax} \cos^{n-2} x dx, n \geq 2, \\ \dfrac{-e^{ax}\ \ \sin x + a\cos x}{a^2 + n^2}, n = 1; \\ \dfrac{e^{an}}{a}, n = 0; \end{cases}$$

12. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int e^{ax}\sin^n bxdx =$$

$$\begin{cases} \dfrac{e^{ax}\sin^{n-1}bx\ \ a\sin x - nb\cos bx}{a^2+n^2b^2} + \dfrac{n\ \ n-1\ \ b^2}{a^2+n^2b^2}\int e^{ax}\sin^{n-2}bxdx, n\geq 2, \\ \dfrac{-e^{ax}\ \ a\sin bx - b\cos bx}{a^2+b^2}, n=1; \\ \dfrac{e^{an}}{a}, n=0; \end{cases}$$

13. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int \ln^n xdx = \begin{cases} x\ln^n x - n\int \ln^{n-1} xdx, n>1, \\ x\ln x - x, n=1; \end{cases}$$

14. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int x^m \ln^n xdx = \begin{cases} \dfrac{x^{m+1}}{m+1}\ln^n x - \dfrac{n}{m+1}\int x^m \ln^{n-1} xdx, n>1, \\ x^{m+1}\left[\dfrac{\ln x}{m+1} - \dfrac{1}{m+1^{\ 2}}\right], n=1; \end{cases}$$

15. Să se calculeze pentru limitele indicate de integrare [a,b]:

$$\int \frac{dx}{a^2+x^{2\ \ n}} =$$

$$\begin{cases} \dfrac{1}{2\ \ n-1\ \ a^2}\left[\dfrac{x}{a^2+x^{2\ \ n-1}} + \ \ 2n-3\ \ \int \dfrac{dx}{a^2+x^{2\ \ n-1}}\right], n>1, \\ \dfrac{1}{a}arctg\dfrac{x}{a}, n=1; \end{cases}$$

Lucrarea de laborator nr.4
Tema: *Structuri dinamice de date*

Exemplu de realizare:

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

struct list{
      int data;
      struct list *link;
      };

void create_list(list **);
void preview_list(list *);
void clear_memory(list *);


void main()
{ list *p1=NULL,*p2=NULL,*p=NULL;
  list *t1,*t2,*q1,*q2,*tp;

  clrscr();

   create_list(&p);
   puts("Start");
   preview_list(p);

   tp=p;
//Crearea a două liste pe baza celei inițiale
   while (tp){
      if (tp->data<0){
       t1=new(list); t1->data=tp->data; t1-
>link=NULL;
       if (p1==NULL) {p1=t1; q1=t1;}
          else {q1->link=t1; q1=t1;}
```

```
      }
      else{
         t2=new(list); t2->data=tp->data; t2-
>link=NULL;
         if (p2==NULL) {p2=t2; q2=t2;}
            else {q2->link=t2; q2=t2;}
      }
      tp=tp->link;
   }
//Afişarea listei elementelor negative
   puts("\nnegativ");
   preview_list(p1);

// Afişarea listei elementelor pozitive
   puts("\npozitiv");
   preview_list(p2);

// Curăţirea memoriei

clear_memory(p);clear_memory(p1);clear_memory(p2)
;

}

void create_list(list **top)
{  int i,n;
   list *t,*q;

   srand(time(NULL));

   puts("Enter number of elements");
   printf("n= "); scanf("%d",&n);
//Crearea primului element al listei
   t=new(list); t->data=rand()%40-20; t-
>link=NULL;
   *top=t; q=t;

//crearea şi adăugarea elementelor în listă
   for(i=1; i<n; i++){
```

```
        t=new(list); t->data=rand()%40-20; t-
>link=NULL;
        q->link=t; q=t;
    }
}

void preview_list(list *top)
{
    list *t;
    t=top;
//Afişarea listei pe ecran
    while (t)
    {  printf("%d ",t->data);
       t=t->link;
    }
    getch();
}

void clear_memory(list *top)// funcţia pentru
curăţirea memoriei
{ list *t;

t=top;
while(t)
{
top=top->link;
delete(t);
t=top;
}

}
```

Sarcini individuale

1. Să se elaboreze un program, care va efectua concatenarea a două liste înlănțuite de simboluri. Programul trebuie să includă funcția căreia, în calitate de parametri, i se transmit pointerii spre două liste şi aceasta uneşte cea de-a doua listă la prima.

2. Să se elaboreze un program, care va uni două liste ordonate de numere întregi într-o singură listă ordonată. Funcția trebuie să primească, în calitate de parametri, pointerii spre primul nod al fiecărei liste, preconizate pentru unire, şi să returneze pointerul spre primul nod al listei reunite.

3. Fie dată o oarecare listă de numere întregi. Să se calculeze suma acelor elemente ale listei, valorile cărora sunt mai mici decât valorile elementelor care le succed.

4. Să se elaboreze un program, care va insera 25 de valori întregi aleatoare de la 0 până la 100 într-o listă ordonată înlănțuită. Programul trebuie să calculeze suma elementelor şi media aritmetică, care trebuie să fie număr cu virgulă mobilă.

5. Să se elaboreze un program, care va efectua transformarea din sistemul zecimal de numerație în sistemul hexazecimal. Pentru înscrierea resturilor să se utilizeze stiva.

6. Să se elaboreze un program, care va diviza lista indicată de numere reale în două liste separate, selectând din lista inițială elementele care nu depăşesc numărul introdus de la tastatură.

7. Arbitrul e înconjurat de *n* jucători. Începând cu jucătorul *m*, fiecare al *k*-lea jucător este eliminat. Să se elaboreze un program care va afişa ordinea eliminării jucătorilor.

8. Să se elaboreze un program, care, în baza listei indicate, va forma altele două, plasând în prima din ele elementele pozitive, iar în cea de-a doua – elementele negative ale listei inițiale.

9. Să se elaboreze un program, care va utiliza stiva pentru a determina dacă şirul este palindrom (adică se scrie similar pe litere în ordine directă şi inversă). Programul trebuie să ignoreze blancurile şi semnele de punctuație.

10. Să se elaboreze un program, care va efectua compararea a două stive, elementele cărora sunt numere întregi. Perechile de elemente ce nu coincid se vor afişa pe ecran.

11. Fie dată o listă de numere întregi A. Să se introducă în lista B numerele de ordine ale elementelor maximale din lista A.

12. Să se elaboreze un program, care va efectua transformarea din sistemul zecimal de numerație în sistemul binar. Pentru înscrierea resturilor să se utilizeze stiva.

13. Fie dată o listă de numere reale. Pentru fiecare element al listei să se tipărească numărul elementelor negative, ce-l succed.

14. Să se definească o funcție, care formează lista L, inserând în aceasta câte un element, care se conține în lista L1, dar nu se conține în lista L2.

15. Să se elaboreze un program, care în baza listei indicate de simboluri va forma altele două, plasând în prima din ele literele minuscule, iar în cea de-a doua – literele majuscule ale listei inițiale.

Lucrarea de laborator nr.5
Tema: *Algoritmi de sortare şi căutare*

Exemplu de realizare:

*Fie dat un tablou unidimensional din **n** elemente întregi. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda selecţiei directe şi metoda inserţiei. Să se calculeze numărul de permutări şi comparaţii.*

Programul:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#define SIZE 20

int a[SIZE],b[SIZE];

void completation(int [],int [],int);
void print_array(int [], int);
void insertion_sort (int [], int);
void selection_sort(int [],int);

void main()
{
clrscr();

completation(a,b,SIZE);

puts("\n\tArray: \n");
print_array(a,SIZE);

insertion_sort(a,SIZE);
puts("\nSorted array: ");
print_array(a,SIZE);
```

```
selection_sort(b,SIZE);
puts("\nSorted array: ");
print_array(b,SIZE);
getch();
}

//Funcţia de creare a tablourilor pentru sortare
void completation(int x[],int y[],int n )
 { int i;
   srand(time(NULL));

   for (i=1; i<=n; i++)
     x[i]=y[i]=rand()%50-20;
     /*completarea tabloului cu numere întregi
       aleatoare */
 }

//Funcţia de afişare a tabloului pe ecran
void print_array(int x[], int n)
{ int i;

  for(i=1; i<=n; i++)
      {
      printf("%5d",x[i]);
      if ((i)%10==0)printf("\n");
      }
}

//Sortarea prin metoda inserţiei directe
void insertion_sort (int x[], int n)
{ int i,j,k,m=0,c=0;
  for (j=2; j<=n; j++)
   {
     k=x[j];
     i=j-1;
     m++;
     while (k<x[i]&&i>0)
         {
         x[i+1]=x[i];
```

```
        --i;
        m++;
        c++;
        }
    x[i+1]=k;
   }
//Afişarea numărului de comparaţii şi permutări
printf("\ninsertion_sort result\ncomparations=
%d\nmovements= %d",c,m);
getch();
}

// Sortarea prin metoda selecţiei directe
void selection_sort(int *m,int n)
 {
     int x;
     int i,j,k,c=0,mm=0;

     for (i=0; i<=n-2;i++)
          {
             x=m[i]; k=i;
             for (j=i+1;j<=n;j++)
                 {c++;
                  if(m[j]<x)
                      {
                       k=j;
                       x=m[k];
                       mm++;
                      }
                   }
                     m[k]=m[i];
                     m[i]=x;
                }
 // Afişarea numărului de comparaţii şi permutări
 printf("\nselestion_sort result\ncomparations=
        %d\nmovements= %d",c,mm);
 getch();
 };
```

Sarcini individuale

1. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda de sortare prin interschimbare a elementelor vecine şi metoda sortării prin interschimbare Shaker. Să se compare eficiența metodelor în funcție de numărul de comparații şi de numărul de permutări.

2. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda de sortare prin inserție directă şi metoda Shell. Să se compare eficiența metodelor în funcție de numărul de comparații şi de numărul de permutări.

3. Fie dat un tablou unidimensional din *n* caractere. Să se ordoneze elementele tabloului în ordine descendentă, utilizând metoda bulelor şi funcția predefinită metoda sortării prin interschimbare Shaker.

4. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine descendentă, utilizând o metoda de interschimbare şi metoda de sortare prin inserție directă. Să se compare eficiența metodelor în funcție de numărul de comparații şi numărul de permutări.

5. Fie dat un tablou unidimensional din *n* caractere. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda bulelor şi metoda de sortare Shaker. Să se compare eficiența metodelor în funcție de numărul de comparații şi numărul de permutări.

6. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine descendentă, utilizând metoda Shaker şi metoda de sortare rapidă qsort. Să se compare eficiența metodelor în funcție de numărul de comparații şi numărul de permutări.

7. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda Shaker şi metoda de sortare prin selecţie directă. Să se compare eficienţa metodelor în funcţie de numărul de comparaţii şi de numărul de permutări.

8. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda de inserţie directă şi metoda de sortare Shaker. Să se compare eficienţa metodelor în funcţie de numărul de comparaţii şi de numărul de permutări.

9. Fie dat un tablou unidimensional din *n* caractere. Să se ordoneze elementele tabloului în ordine descendentă, utilizând metoda bulelor şi funcţia predefinită *qsort*.

10. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine descendentă, utilizând o metodă de interschimbare şi metoda de inserţie directă. Să se compare eficienţa metodelor în funcţie de numărul de comparaţii şi de numărul de permutări.

11. Fie dat un tablou unidimensional din *n* caractere. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda bulelor şi metoda de sortare Shaker. Să se compare eficienţa metodelor în funcţie de numărul de comparaţii şi de numărul de permutări.

12. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine descendentă, utilizând două metode de sortare prin interschimbare. Să se compare eficienţa metodelor în funcţie de numărul de comparaţii şi de numărul de permutări.

13. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând metoda Shaker şi metoda de selecție directă. Să se compare eficiența metodelor în funcție de numărul de comparații şi numărul de permutări.

14. Fie dat un tablou unidimensional din *n* elemente întregi. Să se ordoneze elementele tabloului în ordine ascendentă, utilizând două metode de inserție directă. Să se compare eficiența metodelor în funcție de numărul de comparații şi numărul de permutări.

15. Fie dat un tablou unidimensional din *n* caractere. Să se ordoneze elementele tabloului în ordine descendentă, utilizând metoda şi funcția predefinită *qsort*.