O‘ZBEKISTON RESPUBLIKASI ALOQA, AXBOROTLASHTIRISH VA TELEKOMMUNIKATSIYA TEXNOLOGIYALARI DAVLAT QO‘MITASI

Toshkent axborot texnologiyalari universiteti

“Dasturiy injiniring” fakulteti

“MA’LUMOTLAR TUZILMASI VA ALGORITMLAR”

fanidan laboratoriya ishlarini bajarish bo‘yicha

USLUBIY KO‘RSATMA

Toshkent 2013

Uslubiy ko‘rsatma “Ma’lumotlar tuzilmasi va algoritmlar” fanidan ta’lim oluvchi talabalarga mo‘ljallangan bo‘lib, mazkur fandan laboratoriya ishlarini bajarish uslubi va topshiriqlar o‘rin olgan. Uslubiy ko‘rsatma talabalarning “Ma’lumotlar tuzilmasi va algoritmlar” fanidan nazariy va amaliy bilimlarini oshirishlariga yordam beradi. Laboratoriya ishlariga mo‘ljallangan barcha mavzular misollar, algoritmlar va ularning C++ dasturlash muhitidagi kodlari bilan keng yoritib berilgan. Har bit laboratoriya ishida ishdan maqsad, qisqacha nazariy qism, topshiriqlar va topshiriqlarni bajarishga namunalar keltirilgan. Uslubiy ko‘rsatma 6 ta laboratoriya ishini bajarishga mo‘ljallangan va birinchi, ikkinchi va uchinchi ishlar 2 soatga, to‘rtinchi, beshinchi va oltinchi ishlar 4 soatga, jami 18 soatga mo‘ljallanib tuzilgan.

Tuzuvchilar: **t.f.n. Akbaraliyev B.B.**
ass. Yusupova Z.Dj.

Taqrizchilar: **prof.,t.f.d. Yaqubov M.C.**
dotsent,t.f.n. Xudayberdiyev M.X.

Uslubiy ko‘rsatma Axborot texnologiyalarining dasturiy ta’minoti kafedrasining 2012 yil 4 dekabrida o‘tkazilgan majlisida ko‘rilgan va tasdiqlangan.

Dasturiy injiniring fakulteti ilmiy-uslubiy kengashi ruxsati bilan chop etildi.

Toshkent axborot texnologiyalari universiteti, 2013 yil

MUNDARIJA

KIRISH

Ushbu uslubiy ko‘rsatma “Informatika va axborot texnologiyalari (sohalar bo‘yicha)” yo‘nalishi 2-bosqich talabalari uchun mo‘ljallangan bo‘lib, “Ma’lumotlar tuzilmasi va algoritmlar” fanidan bilim, malaka va ko‘nikmalarini oshirishda hamda tajriba ishlarini bajarishda foydalanilishi mumkin. Uslubiy ko‘rsatma 6 ta tajriba ishi va foydalanilgan adabiyotlar ro‘yhatidan tashkil topgan.

1-tajriba ishida C++ tilida ma’lumotlarning oddiy sozlangan va keltirilgan toifalari haqida va ularga oid misollar keltirilgan.

2-tajriba ishida yarimstatik ma’lumotlar tuzilmasi navbat, stek va dek haqida qisqacha nazariy bilimlar va ularni C++ tilida e’lon qilish, ular ustida amallar bajarishga oid misollar keltirilgan.

3-tajriba ishida dinamik ma’lumotlar tuzilmasi, ya’ni, bir bog‘lamli ro‘yhatlar, ularni e’lon qilish va ustida amallar bajarishga oid misollar va algoritmlarga mo‘ljallangan.

4-tajriba ishida daraxtsimon ma’lumotlar tuzilmasi, binar daraxtlar va ularni e’lon qilish, uni ustida amal bajarish algoritmlari va misol uchun dastur kodlari keltirilgan.

5-tajriba ishida tuzilmadan biror kalit bo‘yicha elementni qidirish usullari va qidiruvni optimallashtirish yo‘llari va algoritmlar misollar bilan taqdim etiladi.

6-tajriba ishida tuzilmalarni saralash usullaridan ayrimlarining algoritmlari va misollar keltirilgan.

Har bir tajriba ishi oxirida shu mavzuga oid talabalar uchun topshiriq variantlari va topshiriqni bajarishga namuna, unda esa variantlarga o‘xshash bo‘lgan bitta misolning to‘liq dasturi berilgan.

Uslubiy ko‘rsatma oxirida foydalanilgan adabiyotlar ro‘yhati keltirilgan.

TASHKILIY-USLUBIY KO‘RSATMALAR

1. Har bir tajriba ishini bajarishdan oldin, tayyorlanishi lozim bo‘lgan tajriba ishiga oid mavzular bo‘yicha maslahatlar (konsultatsiya) o‘tkaziladi.

2. Har bir tajriba ishi hajmi, uni tayyorlash va bajarish tartibi shunday tuzilganki, barcha talabalar berilgan topshiriqlarni tayyorlashlari va hisobotlarni o‘z vaqtida topshirishlari imkoni e’tiborga olingan.

3. Talabalar navbatdagi tajriba ishini bajarishga oldindan tayyorlanib boradilar.

4. Talabalar 1000 V gacha bo‘lgan tajriba qurilmalarida ishlash uchun texnika xavfsizligini o‘rganishlari majbur.

5. Talaba tajriba ishiga tayyorgarlik ko‘rish davrida mazkur ko‘rsatma va tavsiya etilayotgan adabiyotlardan foydalangan holda kerakli nazariy materiallarni o‘rganishlari, zaruriy hisoblashlarni amalga oshirishlari va nazorat savollariga javob berishlari shart.

6. Tayyor bo‘lmagan talabalarga tajriba ishlarini bajarishiga ruxsat berilmaydi.

7. Mashg‘ulot mobaynida hisobot topshirmagan talabalar, keyinchalik o‘qituvchi belgilagan vaqtda topshiradilar.

8. Tajriba ishlarini o‘z vaqtida topshirmagan talabalar keyinchalik o‘qituvchi bilan vaqtni kelishgan holda topshiradilar.

9. Har bir tajriba ishi talaba tomonidan mustaqil ravishda tayyorlanadi. Har bir talaba shaxsiy tarzda hisobot topshiradi. Hisobotni elektron hujjat ko‘rinishda topshirishga ruxsat etiladi. Hisobotni tayyorlashda tajriba ishini bajarish tartibiga binoan quyidagi ma’lumotlar bo‘lishi kerak:

1. Mavzu
2. Ishdan maqsad
3. Masalaning qo‘yilishi

4. Topshiriqqa oid qisqacha nazariy ma’lumot
5. Masalani yechish algoritmi
6. Dastur kodi
7. Natijaning ekran ko‘rinishi

10. Talabalar bilimlari tajriba mashg‘uloti va hisobot topshirish mobaynida o‘qituvchi tomonidan tekshiriladi.

11. Hisobot topshirish davrida talaba nazorat savollari orqali aniqlanuvchi hajm asosida nazariy bilimlarini hamda bajarilayotgan ishning fizik mohiyati tushunchasini ko‘rsatishi lozim.

1-tajriba ishi. MA’LUMOTLARNING ODDIY SOZLANGAN TOIFALARI

Ishdan maqsad: Ma’lumotlarning oddiy sozlangan va nostandart toifalarini o‘rganish va ularni tadqiq qilish.

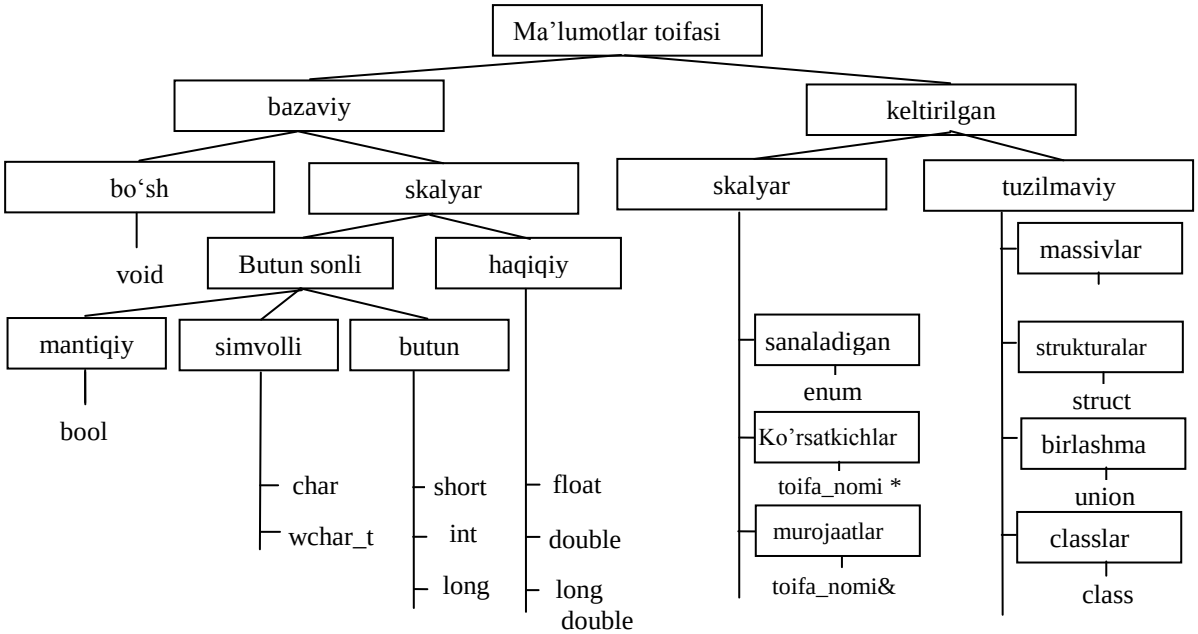
Qo‘yilgan masala: C++ tilida butun, haqiqiy, belgili, mantiqiy toifadagi ma’lumotlarni e’lon qilish, nostandart toifalarni yaratish va ularga doir misollarning dasturini ishlab chiqish.

Ish tartibi:

- Tajriba ishi nazariy ma’lumotlarini o‘rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- C++ dasturlash muhitida dasturni yaratish;
- Natijalarni tekshirish;
- Hisobotni tayyorlash va topshirish.

1.1. Ma’lumotlar toifalari

Ko‘plab dasturlash tillarida ma’lumotlar bazaviy va keltirilgan toifalarga ajratiladi. Ma’lumotlarning toifalarini 1.1-rasmdagidek klassifikatsiyalash mumkin.



1.1-rasm. Toifalar klassifikatsiyasi

Ma’lumotlarning ixtiyoriy toifasi qiymatlar sohasi va ular ustida bajarilishi mumkin bo‘lgan amallar orqali tavsiflanadi. *void* kalit so‘zi hech qanday toifaga ega emaslikni anglatadi. Bunday toifadagi funksiyalar hech qanday qiymatni qaytarmaydi. Lekin asosiy dastur tanasi, ya’ni *main()* funksiyasi *void* toifasiga ega bo‘lolmaydi, u *int* toifasida bo‘lishi kerak.

1.2. Sozlangan toifalar

1.2.1. Butun toifa – int

Mazkur toifa butun sonlar to‘plamining qandaydir qism to‘plami bo‘lib, uning o‘lchami mashina, ya’ni kompyuter konfiguratsiyasiga bog‘liq ravishda o‘zgarib turadi. Mazkur toifaga kiruvchi sonlar ikkiga bo‘linadi: ishorali (signed) va ishorasiz (unsigned). Sonlarmi xotirada tasvirlashda eng chapdagi bit ishora uchun belgilanadi. Toifalarni *signed* (ishorali), *unsigned* (ishorasiz) kalit so‘zlari bilan modifikatsiyalash mumkin. Bunda ishorali toifa uchun ajratilgan joyning eng chap biti ishora uchun, qolgan bitlar qiymatlarni saqlash uchun ishlatiladi, ya’ni 0 – plus, 1 - minus. Ishorasiz toifalarda esa barcha bitlar qiymatlarni saqlash uchun ishlatiladi. Ularning har biri uchun mos ravishda qiymat qabul qilish oralig‘i mavjud:

a) ishorasiz sonlar uchun $(0...2^n-1)$;

b) ishoralilar uchun $(-2^{n-1}\dots 2^{n-1}-1)$.

Butun sonlar ustida turli matematik (+, -, /, *) va solishtirish amallarini bajarish mumkin, ya’ni ==, !=, <, <=, >, >= operatorlar bilan binar amallarni bajarish mumkin. Ammo bu operatsiyalarning natijalari *int* toifasiga kirmaydi, ular *bool* toifasiga kiradi.

Butun qiymat qabul qiluvchi o‘zgaruvchilarni e’lon qilish uchun *int, short int, long int* xizmatchi so‘zlaridan foydalanish mumkin. Butun qiymatli toifalarning barchasi 1.1-jadvalda keltirilgan:

1.1-jadval

Butun toifa shakllari

Toifa ko‘rinishi	Mazkur toifadagi o‘zgaruvchining qabul qiladigan qiymatlar oralig‘i	O‘zgaruvchining kompyuter xotirasidan egallaydigan joyi
short int	signed: -32768 ... 32767 unsigned: 0 ... 65535	2 bayt
int	signed: -2147483648 ... 2147483647 unsigned: 0 ... 4294967295	4 bayt
long int	signed: -2147483648 ... 2147483647 unsigned: 0 ... 4294967295	4 bayt

Bu sanab o‘tilgan toifalar o‘zlarining qiymatlar qabul qilish oralig‘i va xotiradan egallagan joyining katta yoki kichikligi bilan farqlanadi. Shuning uchun, o‘zgaruvchilarning qabul qiladigan qiymatlarini katta yoki kichikligiga qarab, yuqoridagi toifalardan mosini tanlash maqsadga muvofiqdir.

Toifalar uchun xotira hajmining ajratilishi kompyuter konfiguratsiyasiga va kompilyatorga bog‘liq bo‘ladi. Ixtiyoriy bir toifaning xotirada egallaydigan hajmini bilish mumkin. Buning uchun ***sizeof()*** funksiyasini ishlatish mumkin.

```
#include <iostream.h>
using namespace std;
int main()
{
  cout<<sizeof(int);
  system("pause");
}
```

Bu yerda natija baytlarda chiqadi, ya’ni 4. Funksiyaga kirish parametri sifatida toifa nomi beriladi.

Butun toifaning quyidagicha ko‘rinishlari mavjud.

Short	*unsigned*
short int	*unsigned int*
signed short	*long*
signed short int	*long int*
unsigned short	*signed long*
unsigned short int	*signed long int*
int	*unsigned long*
signed int	*unsigned long int*

Berilgan m va n butun sonlari ustida quyidagi arifmetik amallar bajarish dasturini ko‘rib chiqaylik: m+n, m-n, m*n.

```
#include <iostream>
using namespace std;
int main()
{   int m,n;
    cin>>m>>n;
    int k1=m+n;
    int k2=m-n;
    int k3=m*n;
    cout<<k1<<k2<<k3;
    system("PAUSE");
}
```

1.2.2. Haqiqiy toifa

Haqiqiy toifaga kasr qismlari bor chekli sonlar to‘plami kiradi. Haqiqiy sonlar ustida turli matematik amallarni bajarish mumkin. Bu amallarning natijalari ham haqiqiy toifaga kiradi. Bu yerda ham binar amallarga nisbatan masalaning

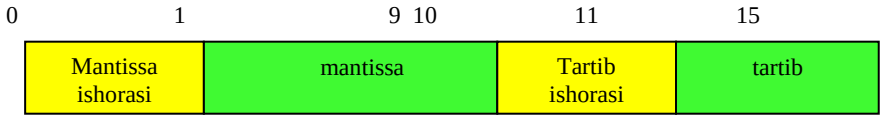
yechimlari mantiqiy toifaga tegishli bo‘ladi.

Kompyuter xotirasida haqiqiy sonlar asosan qo‘zg‘aluvchan nuqta formatida saqlanadi.

$937,56 = 93756 * 10^{-2} = 0,93756 * 10^{3} = 0,93756E3$

$0,002355 = 2,355 * 10^{-3} = 2,355E-3$

Xotiraga haqiqiy sonlar yozilayotganda uning uchun ajratilgan xotira sohasining 1-bitiga *E* simvolidan chapdagi mantissa ishorasi 1 ta bitga, keyin mantissa, undan keyin *E* – ya’ni har doim 10 soniga teng deb olinadigan eksponenta belgisi darajasining ishorasi 1 ta bitga, so‘ngra uning darajasidagi son, ya’ni *E* simvolidan o‘ngdagi son yoziladi (1.2-rasmga qarang).



1.2-rasm. Haqiqiy sonlarni xotiraga yozilish shakli

Haqiqiy (kasr) qiymatli toifaga tegishli o‘zgaruvchilarni e’lon qilish uchun *float, double, long double* xizmatchi so‘zlaridan foydalanish mumkin.

1.2-jadval

Haqiqiy toifa shakllari

Toifa ko‘rinishi	Mazkur toifadagi o‘zgaruvchining qabul qiladigan qiymat oralig‘i	O‘zgaruvchining kompyuter xotirasidan egallaydigan joyi
Float	+/- 3.4E-38 … +/-3.4E+38	4 bayt
Double	+/- 1.7E-308 … +/- 1.7E-308	8 bayt
long double	+/- 1.7E-308 … +/- 1.7E-308	8 bayt

Berilgan ***m*** va ***n*** haqiqiy sonlari ustida quyidagi amallarni bajarish dasturini ko‘rib chiqaylik.

#include <iostream>

```
using namespace std;
int main()
{
float m,n;
   cin>>m>>n;
   float k1=m+n;
   float k2=m-n;
   float k3=m*n;
   cout<<k1<<"  "<<k2<<"  "<<k3;
   system("PAUSE");
}
```

C++ da ushbu toifalarni oldiga *signed* va *unsigned* kalit so‘zlarini qo‘yib toifalarni modifikatsiyalash mumkin. Masalan,

signed float

unsigned float

signed double

unsigned double

signed long double

unsigned long double

1.2.3. Mantiqiy toifa

Mazkur toifa mantiqiy mulohazalarning to‘g‘riligini aniqlash uchun, turli xil dasturlash tillarida turlicha ifodalaniladigan ifodalarni 2 ta ko‘rinishda aniqlaydi. Mantiqiy ma’lumotlar ustida quyidagi mantiqiy operatsiyalarni bajarish mumkin: konyunktsiya (va), dizyunktsiya (yoki) va inkor (yo‘q), hamda qiyinroq bo‘lgan ekvivalentlik, implikatsiya, chiqarib tashlash va boshqa operatsiyalar. Yuqorida keltirilgan ixtiyoriy operatsiyaning natijasi – mantiqiy qiymatga ega bo‘ladi.

Mantiqiy qiymatni xotirada saqlash uchun bitta bit yetarli.

1.3-jadval

Asosiy mantiqiy funksiyalarning chinlik jadvali

A	B	not A	A or B	A and B
1	1	0	1	1
1	0	0	1	0
0	1	1	1	0
0	0	1	0	0

1.4-jadval

Mantiqiy toifa tavsifi

Toifa ko‘rinishi	Mazkur toifadagi o‘zgaruvchining qabul qiladigan qiymat oralig‘i	O‘zgaruvchining kompyuter xotirasidan egallaydigan joyi
Bool	true , false	1 bayt

C++ da *and* mantiqiy amalining yana bir yozilish shakli *&&*, *or* yoki *||*, *not* yoki *!* va “inkor-yoki” amali *xor* kabi yozilishi mumkin.

bool toifasiga bitta misol ko‘rib chiqamiz.

```
#include <iostream>
using namespace std;
int main()
{   bool b=true;
    bool s=false;
    bool d1=not b || s;
    bool d2=b && s;
    bool d3=b xor s;
    cout<<d1<<"  "<<d2<<"  "<<d3;
    system("PAUSE");
}
```

Natija: 0 0 1

1.2.4. Belgili toifa

Belgili toifaga belgilarning chekli to‘plami yoki liter, ularga lotin alifbosidagi harflar va unda yo‘q kirill harflar, o‘nlik raqamlar, matematik va maxsus belgilar kiradi. Belgili ma’lumotlar hisoblash texnikasi bilan inson o‘rtasidagi aloqani o‘rnatishda katta ahamiyatga ega. Belgili toifadagi o‘zgaruvchilar ustida turli matematik amallarni bajarish mumkin. Bunda amallar belgilarning ASCII kodlari ustida bajariladi. Shu sababli, belgili toifalarni taqqoslash ham mumkin va taqqoslashlarning natijalari bool toifasiga kiradi. C++ tilida belgili toifalarning qiymatlari qo‘shtirnoq ichida beriladi va u bitta belgidan iborat bo‘lishi mumkin.

1.5-jadval

Belgili toifa shakllari

Toifa ko‘rinishi	Mazkur toifadagi o‘zgaruvchining qabul qiladigan qiymat oralig‘i	O‘zgaruvchining kompyuter xotirasidan egallaydigan joyi
char(signed char)	-128…127	1 bayt
unsigned char	0…255	1 bayt
wchar_t (kengaytirilgan simvolli tip)	0…65535	2 bayt

Satr (qator) – bu qandaydir belgilar ketma-ketligi bo‘lib, satr bitta, bo‘sh yoki bir nechta belgilar birlashmasidan iborat bo‘lishi mumkin. C++ tilida satrlarni e’lon qilish belgilar massivi shaklida amalga oshiriladi. Bu haqda keyinroq batafsil to‘xtalamiz.

Belgili toifadagi o‘zgaruvchilar ustida o‘zlashtirish, taqqoslash va turli matematik amallarni bajarish mumkin. Bunda agar belgili toifalar ustida matematik amallar bajariladigan bo‘lsa, belgilarning ASCII kodlari olinadi.

Belgilar va qatorlarga doir quyidagi sodda dasturni keltiramiz:

```
#include<iostream.h>
using namespace std;
int main()
{   char x='a';
    char y='b';
    char min;
    cout<<"belgilar yig‘indisi="<<x+y<<endl;//a va b  simvollarni ASCII kodlarini yig‘indisi - 195
    cout<<x<<"  "<<y<<endl;//a  b ni ekranga chiqarish
    if(x>y) min=y;
     else min=x;
    cout<<"min="<<min;// ekranga a chiqadi
system("pause");
}
```

Natija: belgilar yig‘indisi=195

a b

min=a

1.3. Keltirilgan toifalar

1.3.1. Sanaladigan toifa

Bir qancha qiymatlardan birini qabul qila oladigan o‘zgaruvchiga sanaladigan toifadagi o‘zgaruvchilar deyiladi va bunday o‘zgaruvchilarni e’lon qilishda *enum* kalit so‘zi va undan keyin toifa nomi hamda figurali qavs ichida vergullar bilan ajratilgan o‘zgarmas qiymatlar ro‘yhati ishlatiladi.

Masalan:

enum Ranglar{oq, qora, qizil, yashil};

Bu yerda Ranglar nomli sanoqli toifa yaratildi. Ushbu toifaning 4 ta

o‘zgarmas elementlari mavjud va ular dastlab 0 dan boshlab sanaladigan butun sonli qiymatga ega bo‘ladilar. Ayrim hollarda foydalanuvchi tomonidan o‘zgarmaslarga ixtiyoriy sonli qiymat ham o‘zlashtirilishi mumkin. O‘zgarmaslarga qiymatlar o‘sish tartibida berilishi kerak. Masalan,

enum Ranglar{oq=100,qora=200,qizil,yashil=400};

Bu yerda qizil o‘zgarmasning qiymati 201 ga teng bo‘ladi. Endi shu toifadagi birorta o‘zgaruvchini e’lon qilish mumkin.

Ranglar r=qizil;

Endi r o‘zgaruvchi ranglar toifasida aniqlangan o‘zgarmaslardan ixtiyoriy birini qiymat sifatida qabul qila oladi. Masalan:

```
#include<iostream.h>
using namespace std;
int main()
{   enum kunlar{du=1,se,chor};
   kunlar hafta;
   hafta=chor;
   cout<<hafta;//
   int kun;
   cout<<"\nbugun qaysi kun=";
   cin>>kun;
   if(kun==chor) cout<<"\ntalabalar bilan uchrashuvingiz bor";
system("pause");
}
```

Natija: 3

bugun qaysi kun=3

talabalar bilan uchrashuvingiz bor

1.3.2. Ko‘rsatkichli toifa

Ko‘rsatkichlar ma’lumotlarni emas, balki bu ma’lumotlar joylashgan

xotiradagi manzilni o‘zida saqlaydi. Ko‘rsatkichlar xotirada bor yo‘g‘i 4 bayt joyni egallab, u ko‘rsatayotgan ma’lumotlar ancha katta joyni egallagan bo‘lishi mumkin. Ko‘rsatkichlar qanday ishlashini bilish uchun mashina xotirasi tashkil etilishining tayanch prinsiplarini bilish lozim. Mashina xotirasi 16 lik sanoq sistemasida raqamlangan yacheykalar ketma-ketligidan iboratdir. Har bir o‘zgaruvchining qiymati uning adresi deb ataluvchi alohida xotira yacheykasida saqlanadi. Ko‘rsatkichli toifadagi o‘zgaruvchilar o‘zida ana shu kabi o‘zgaruvchilar yoki boshqa ma’lumotlarning xotiradagi adresini saqlaydilar. C++ da o‘zgaruvchini ko‘rsatkichli toifada e’lon qilish uchun o‘zgaruvchi nomidan oldin * belgisi qo‘yiladi. Har bir o‘zgaruvchining toifasi bilan e’lon qilingani kabi ko‘rsatkichli o‘zgaruvchilar ham ma’lum bir toifa bilan e’lon qilinadi. Bunda ko‘rsatkichli o‘zgaruvchining toifasi – shu ko‘rsatkich ko‘rsatayotgan xotira yacheykasidagi ma’lumotning toifasi bilan bir xil bo‘lishi kerak. Masalan, int a=1 bo‘lsin. Ushbu o‘zgaruvchining adresini o‘zida saqlovchi b ko‘rsatkichli o‘zgaruvchini e’lon qilishda ham int toifasi ishlatiladi, ya’ni *int *b*. Endi bunday toifadagi o‘zgaruvchiga a o‘zgaruvchining adresini o‘zlashtirish uchun a ning oldiga & - adres operatorini qo‘yish zarur, ya’ni *b=&a*.

Misol.

```
#include<iostream.h>
using namespace std;
int main()
{  short int a=1234567;
   short int *b;
   b=&a;
   cout<<b; // a o‘zgaruvchining adresi 0x22ff76 ni ekranga chiqaradi
system("pause");
}
```

Ko‘rsatkichli toifalar yordamida fayllarga ham murojaat qilsa bo‘ladi, masalan quyida *f.txt* faylidagi ma’lumotlarni ekranga chiqarish dasturi keltirilgan:

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

```
#include <iostream.h>
using namespace std;
int main()
{
FILE *p;
  char s;int i=0;
  if((p=fopen("f.txt","r"))==NULL)
 i  cout<<"o'xshamadi";
  else cout<<"ulandi\n";
  while(s!=EOF){
        s=fgetc(p);
        if(s=='s')i++;
        cout<<s;
        }
  fclose(p); cout<<"s harfi "<<i<<”marta qatnashgan”;
  system("pause");
}
```

f.txt fayli tarkibi:

c++ tilida fayllar
bilan
ishlash dasturi

Dastur natijasi:

ulandi
c++ tilida fayllar
bilan
ishlash dasturi
s harfi 3 marta qatnashgan

1.3.3. Massivlar

Massiv bu bir toifaga mansub elementlar to‘plami bo‘lib, uning 2 xil ko‘rinishi mavjud: 1 o‘lchovli va 2 o‘lchovli massivlar. 1 o‘lchovli massivda har bir element 1 ta indeksga, 2 o‘lchovli massiv (matritsa) da esa elementlar 2 ta indeksga ega bo‘ladi. 1 o‘lchovli massivda elementlarning indeksi ularning turgan o‘rni, ya’ni tartib raqami bilan belgilanadi. 2 o‘lchovli massivlarda esa elementlarning 1-indeksi uning joylashgan satri va 2-indeksi esa u joylashgan ustun tartib raqami bilan belgilanadi. Har ikkala holatda ham massiv elementlari indekslari 0 dan boshlanadi. C++ dasturlash muhitida massivlarni e’lon qilish uchun ularning oldiga toifasi ko‘rsatilib, massivga nom va [] kvadrat qavs ichida massiv uzunligi, ya’ni elementlar soni ko‘rsatiladi, ya’ni masalan: int a[10]; char b[10][20];

1 o‘lchovli massiv - a[0],a[1],…,a[n]

2 o‘lchovli massiv - a[0][0],a[0][1],…,a[0][m]
a[1][0],a[1][1],…,a[1][m]
…
a[n][0],a[n][1],…,a[n][m]

Ikki o‘lchamli massiv elementiga murojaatni amalga oshirish uchun uning indeksi qiymatlari zarur bo‘ladi. Fizik bosqichda ikki o‘lchamli massiv ham xuddi bir o‘lchamli massiv kabi ko‘rinishga ega bo‘ladi hamda translyatorlar massivni qator yoki ustun ko‘rinishida ifodalaydi.

```
#include <iostream.h>
using namespace std;
int main()
{
  int a[2][3],s=0;
  for(int i=0;i<2;i++)
  for(int j=0;j<3;j++){
        cin>>a[i][j];
```

```
        s+=a[i][j];
        }
  cout<<s;
   system("pause");
}
```

Agar massiv toifasi char bo‘lsa, u holda massiv satr hisoblanadi, ya’ni simvollar ketma-ketligi. Satr belgilariga murojaat xuddi massivdagi kabi indeks bilan amalga oshiriladi. Masalan:

```
char str[10];
str[0]=’q’;
```

Satrlar ustida turli amallarni bajarishga mo‘ljallangan bir qancha funksiyalar mavjud. Satr uzunligini aniqlash *strlen()* bilan amalga oshiriladi. Satrlarga oid misol ko‘ramiz. Berilgan satrdagi unli harflarni ajratib ko‘rsating.

```
#include<iostream.h>
using namespace std;
int main(){
char str[20];
cin>>str;
int l=strlen(str);
for(int i=0;i<l;i++)
     if(str[i]=='a'||str[i]=='o'||str[i]=='i'||str[i]=='e'||str[i]=='u'||
str[i]=='y')
          cout<<str[i];
 system("pause");
}
```

Natija: *dastur*

au

1.3.4. Vektorlar

C++ dasturlash muhitida ma’lumotlarni massivdan tashqari sal boshqacharoq usulda saqlashning yana bir turi mavjud, ya’ni vektorlar. Vektor elementlari ustida massiv elementlari ustida bajariladigan amallarni bajarish mumkin. Ma’lumotlarni massivda saqlashda elementlar soni oldindan ma’lum bo‘lishi kerak. Ayrim paytlarda massivga nechta element kiritilishi ma’lum bo‘lmaydi va o‘shanda dinamik dasturlashdan foydalanish kerak bo‘ladi, ya’ni massivga qo‘shiladigan elementga xotira ajratishga to‘g‘ri keladi. Shunday hollarda *vector* klassidan foydalanish mumkin. Vector klassi o‘zgaruvchan uzunlikdagi massiv yaratishga yordam beradi. Vektor bu elementlari soni oldindan ma’lum bo‘lmagan bir xil toifadagi elementlar ketma-ketligidir. Vektorning massivdan farqi, vector uzunligi oldindan berilmaydi va u dastur bajarilishi mobaynida o‘zgarib turadi. Vektor yaratish uchun <vector> kutubxonasiga ulanish kerak, ya’ni dastur boshida *#include<vector>* qatori bo‘lishi kerak va vektorni e’lon qilishning 2 ta usuli mavjud – vektor uzunligini ko‘rsatib va bo‘sh vektor ko‘rinishida.

vector<toifa_nomi> o‘zgaruvchi_nomi;

Masalan, *vector <int> test;* bu yerda *int* toifali, *test* nomli bo‘sh vektor yaratildi. Vektor elementlariga indeks orqali murojaat qilib bo‘ladi, lekin bu ko‘rinishda vektor yaratilganda vektor elementiga indeks bilan murojaat qilib qiymat berib bo‘lmaydi, ya’ni quyidagi dastur kodi noto‘g‘ri:

```
vector<int> vek;
vek[0]=123;
vek[1]=234;
```

Bu holda vektorga element kiritish quyidagicha amalga oshiriladi:

```
vek.push_back(7);//vector oxiriga yangi element 7 ni kiritish
vek.push_front(17);//vector boshiga yangi element 17 ni kiritish
vek.pop_back();//
```

vektor oxirgi elementini o‘chirish funksiyasi

vek.pop_front();// vektor 1-elementini o‘chirish funksiyasi

Misol:

```
#include <vector>
#include <iostream.h>
using namespace std;
int main(){
vector< string > text;
string word;
    while(word!="0"){
    text.push_back( word );
    cin>>word;
    }
for(int i=0;i<text.size();i++) cout<<text[i]<<"  ";
system("pause");
}
```

Vektor yaratishning 2-usuli xuddi massivga o‘xshash bo‘lib, unda vektor uzunligi oldindan ko‘rsatiladi va berilgan uzunlikka mos barcha elementlarga avtomatik tarzda 0 qiymat beriladi. Vektor elementlariga murojaat xuddi massiv elementlariga murojaat kabi indeks orqali amalga oshiriladi va qiymat berilishi mumkin.

```
#include <vector>
#include <iostream.h>
using namespace std;
int main()
{
 vector< int > ivec(5);
 ivec[0]++; //bunda vektor 0-elementi qiymati bittaga oshirildi
 ivec[1]=11; //vektor 1-elementiga 11 qiymati berildi
 for(int i=0;i<ivec.size();i++) cout<<ivec[i]<<"  ";
 system("pause");
```

```
}
```

Natija: 1 11 0 0 0

Agar bu usulda vektor yaratiladigan bo‘lsa, *push_back()* va *push_front()* funksiyalari vektor uzunligini oshiradi. Misol uchun:

```
#include <vector>
#include <iostream.h>
using namespace std;
int main()
{
vector< int > ivec(5);
ivec[0]++;
ivec[1]=11;
ivec.push_back(123);
for(int i=0;i<ivec.size();i++) cout<<ivec[i]<<"  ";
system("pause");
}
```

Natija: 1 11 0 0 0 123

Vektor ustida quyidagi funksiyalar orqali amal bajarish mumkin:

- *test.at(i)* - test[i] kabi vektor i-elementiga murojaat qilish;
- *test.asign(n,m)* – vektorga m qiymatli n ta element kiritish;
- *test.front()* – vektor ko‘rsatkichini 1-elementga o‘rnatish;
- *test.back()* - vektor ko‘rsatkichini oxirgi elementga o‘rnatish;
- *test.size()* – vektor elementlari sonini aniqlash;
- *test.swap(test2)* – test vektori tarkibi bilan test2 vektori tarkibini almashtirish
- *test.empty()* – vektor bo‘shligini tekshirish;

Vektorga oid misol ko‘ramiz. Quyidagicha masala qo‘yilgan bo‘lsin:

massivning juft qiymatli elementlaridan vektor hosil qiling.

```
#include <vector>
#include <iostream.h>
```

```
using namespace std;
int main(){
 vector< int > avec;
 int n;cout<<"n=";cin>>n;
 int a[n];
 for(int i=0;i<n;i++){ cin>>a[i];
      if(a[i]%2==0) avec.push_back(a[i]);}
cout<<”avec=”;
 for(int i=0;i<avec.size();i++) cout<<avec[i]<<"  ";
system("pause");
}
```

Natija: n=5

1 2 3 4 5

avec= 2 4

1.3.5. Strukturalar

Strukturalar turli toifadagi maydonlardan tashkil topgan yozuv hisoblanadi. Strukturalarni e’lon qilish uchun *struct* kalit so‘zi ishlatiladi. Undan keyin toifaga nom beriladi va {} qavs ichida maydonlar toifalari va nomlari e’lon qilinadi.

```
struct G{
      char ch;
      } talaba, talabalar[10];
```

Yaratilgan toifa bilan e’lon qilingan o‘zgaruvchi *talaba* - yozuv hisoblanadi, massiv esa *talabalar[10]* - jadvalni tashkil etadi. Yozuv va jadval yozuvi maydoniga qiymat berish quyidagicha:

yozuv.maydon_nomi=qiymat;

Masalan:

talaba.ch=’a’;

Agar jadval yozuvi maydoniga qiymat beriladigan bo‘lsa, bunda jadval yozuv massivi shaklida tashkil qilinadi va shu massiv elementiga indeks bilan murojaat orqali amalga oshiriladi:

Jadval_elementi[indeks].maydon_nomi=qiymat;

Ya’ni, *talabalar[i].ch=’a’;*

Misol. Talabalar tartib raqami va ism-familiyasidan iborat jadval tuzib, ma’lumotlarni kiritish va ekranga chiqarish dasturi.

```
#include <iostream.h>
using namespace std;
int main(
{
   struct Guruh{
         int n;
         char fio[30];
         };
   Guruh talaba[5];
  for(int i=0;i<5;i++){
         talaba[i].n=i+1;
         cin>>talaba[i].fio;
         }
  for(int i=0;i<5;i++)
         cout<<talaba[i].n<<"  "<<talaba[i].fio<<endl;
   system("pause");
}
```

Bu yerda *Guruh* nomli nostandart toifa yaratildi va uning 2 ta maydoni mavjud: talabaning tartib raqami *n* va familiyasi, ismi, otasining ismi uchun uzunligi 30 bo‘lgan satrli maydon *fio[30]*.

1.3.6. Birlashmalar (union)

Birlashmalar xuddi strukturalarga o‘xshash toifa hisoblanadi, farqi shuki, birlashmalarda bir vaqtning o‘zida faqat uning bitta elementigagina murojaat qilish mumkin. Birlashma toifasi quyidagicha aniqlanadi:

```
union {   1-elementni tavsiflash;
            ...
            n-elementni tavsiflash;
          };
```

Birlashmalarning asosiy xususiyati shuki, e’lon qilingan har bir element uchun xotiraning bitta hududi ajratiladi, ya’ni ular bir-birini qoplaydi. Bu yerda xotiraning shu qismiga istalgan element bilan murojaat qilsa bo‘ladi, lekin buning uchun element shunday tanlanishi kerakki, olinadigan natija ma’noga ega bo‘lishi kerak. Birlashmaning elementiga murojaat xuddi struktura elementiga murojaat kabi amalga oshiriladi. Birlashmalar qo‘llaniladigan xotira obyektini initsializatsiya qilish maqsadida ishlatiladi, agarda har bir murojaat vaqtida bir qancha obyektlardan faqat bittasi faollashtirilsa.

Birlashma toifasidagi o‘zgaruvchi uchun ajratiladigan xotira hajmi ushbu toifaning eng uzun elementi uchun ketadigan xotira hajmi bilan aniqlanadi. Kichik uzunlikdagi element ishlatilganda, birlashma toifasidagi o‘zgaruvchi uchun ajratilgan xotira sohasining ayrim qismi ishlatilmaydi. Birlashmaning barcha elementi uchun xotiraning bitta adresdan boshlanuvchi bitta sohasi ajratiladi. Masalan:

```
union {   char  fio[30];
          char  adres[80];
          int   yoshi;
          int   telefon;   } inform;
union {   int ax;
          char al[2];     }   ua;
```

Birlashma tipidagi inform obyektini ishlatganda qiymat qabul qilgan elementnigina qayta ishlash mumkin, ya’ni masalan *inform.fio* elementiga qiymat berilgandan keyin boshqa elementlarga murojaat ma’noga ega emas. *ua* birlashmasi *al* elementining kichik *ua.al[0]* va katta *ua.al[1]* baytlariga alohida murojaat qilish mumkin. Birlashma tipiga oid misol ko‘rib chiqamiz.

```
#include <iostream.h>
using namespace std;
int main()
{    union Guruh{
        int n;
        int m;
        };
    Guruh w;
    w.n=12;// w birlashmasining n elementiga qiymat berish
    w.m=23; // w birlashmasining m elementiga qiymat berish
    cout<<w.n<<"  "<<w.m; //bu yerda w  uchun ajratilgan joyga oxirgi marta m uchun 23 qiymati yozilgani sababli ekranga 23  23 javobi chiqariladi.
   system("pause");
}
```

1.3.7. Klasslar

Klass – bu dasturchi tomonidan ixtiyoriy kiritilgan mavjud tiplar asosida yaratilgan strukturalangan toifa hisoblanadi. Klasslar lokal va global o‘zgaruvchilar va ular ustida amal bajaradigan funksiyalar to‘plamidan iborat bo‘lishi mumkin. Klasslar quyidagicha tasvirlanadi:

```
class klass_nomi{
        <lokal va global o‘zgaruvchilar ro‘yhati>;
        <funksiyalar>
};
```

Klasslarga oid misol:

```
#include <iostream.h>
using namespace std;
class daraxt
{
 public:
    unsigned int uzunligi ;
    unsigned int yoshi;
 int o_sish(int i){
    i++;
    return i;
    };
};
int main()
{
   int k=2;
   daraxt olma_daraxt;
   olma_daraxt.uzunligi=5;
   olma_daraxt.yoshi=7;
   cout<<olma_daraxt.o_sish(k);
  system("pause");
}
```

Natija: *3*

Ishni bajarishga namuna

Berilgan topshiriq variantlariga o‘xshash bo‘lgan bitta masalani bajarib ko‘ramiz. Quyidagicha masala qo‘yilgan: Berilgan familiyalardan imlo qoidasiga mos ravishda ismlar hosil qiling.

Algoritm

1. Familiya kiritilishini so‘rash.
2. Kiritilgan familiya uzunligini o‘lchash.
3. Familiya oxirgi va oxiridan 1 ta oldingi simvolini tekshirish, ya’ni familiya oxiri “*ev*” bilan tugasa, satrning oxirgi 3 ta simvolini o‘chirish va 7-qadamga o‘tish, aks holda 4-qadamga o‘tish.
4. Familiya oxirgi va oxiridan 1 ta oldingi simvolini tekshirish, ya’ni familiya oxiri “*ov*” bilan tugasa, satrning oxirgi 2 ta simvolini o‘chirish va 7-qadamga o‘tish, aks holda 5-qadamga o‘tish.
5. Familiya oxirgi va oxiridan 2 ta oldingi simvolini tekshirish, ya’ni familiya oxiri “*eva*” bilan tugasa, satrning oxirgi 4 ta simvolini o‘chirish va 7-qadamga o‘tish, aks holda 6 qadamga o‘tish.
6. Familiya oxirgi va oxiridan 2 ta oldingi simvolini tekshirish, ya’ni familiya oxiri “*ova*” bilan tugasa, satrning oxirgi 3 ta simvolini o‘chirish va 7-qadamga o‘tish.
7. Hosil bo‘lgan ismni ekranga chiqarish.

Talabalar algoritmni so‘z bilan yoki blok-sxema ko‘rinishida ifodalashlari mumkin.

Dastur kodi:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
   int l;
   char a[100];
   cout << " Familiyani kiriting: ";
   gets(a);
   l=strlen(a);
   if(a[l-1] == 'v' && a[l-2] == 'e'){ l = l-3; }
```

```
if(a[l-1] == 'v' && a[l-2] == 'o'){ l = l-2; }
if(a[l-1] == 'a' && a[l-3] == 'e'){ l = l-4; }
if(a[l-1] == 'a' && a[l-3] == 'o'){ l = l-3; }
cout << "\n Natija: \n ";
for(int i = 0; i < l; i++)
        cout<<a[i];
getch();
}
```

Dastur natijasi:

Familiyani kiriting: Axmadaliyev

Axmadali

Nazorat savollari

1. Ma’lumotlar toifasi tushunchasi nima va nima uchun ma’lumotlar toifalanadi?
2. Qanday ma’lumot toifalarini bilasiz?
3. Oddiy va sozlangan toifalarni tushuntiring.
4. Struct va vector toifalarini tushuntiring.
5. Class va massivlar qanday e’lon qilinadi?

Topshiriq

Variantlar:

1. Berilgan sonlar ketma-ketligidagi maksimal va minimal elementlarning o‘rnini almashtiring.
2. Berilgan sonlar ketma-ketligidagi har bir elementni o‘zi, o‘zidan oldingi va o‘zidan keyingi element bilan yig‘indisiga almashtiring.
3. k-darajagacha bo‘lgan Nyuton binomi sonlaridan vektor hosil qiling. Nyuton binomi sonlari quyidagicha aniqlanadi.

1	1
1 2 1	2
1 3 3 1	3
1 4 6 4 1	…
1 5 10 10 5 1	k

4. nxn matritsaning yuqori chap uchburchagidagi elementlaridan vektor hosil qiling

5. nxn matritsaning yuqori o‘ng uchburchagidagi elementlaridan vektor hosil qiling

6. nxn matritsaning pastki o‘ng uchburchagidagi elementlaridan vektor hosil qiling

7. nxn matritsaning pastki chap uchburchagidagi elementlaridan vektor hosil qiling

8. Matritsani matritsaga ko‘paytiring

9. Jadval hosil qiling va unga ma’lumotlarni kiriting, ekranga chiqaring.

10. Talabalar ism-familiyasi, yoshi va ballaridan iborat jadval yarating va talabalarni ism-familiyasini alfavit bo‘yicha tartibga keltiring.

11. 10-variantdagi jadvaldan bali bo‘yicha eng katta va eng kichik talabalarning o‘rnini almashtiring

12. 2 ta bir xil tipdagi jadval berilgan. Ikkala jadvalni o‘zaro solishtiring va aynan bir xil bo‘lgan yozuvlarni o‘chiring.

13. Birlashma tipidagi (tipda 2 ta element e’lon qiling) 2 ta o‘zgaruvchini bir-biridan farqli elementlariga qiymat bering va ularning yig‘indisini ikkala o‘zgaruvchining qiymat berilmagan elementlariga o‘zlashtiring. Har ikkala o‘zgaruvchining barcha elementlarini ekranga chiqaring va natijani tushuntiring.

14. Avtomobil nomli klass yarating va unda turli o‘zgaruvchi va funksiyalar yarating. Ushbu klassdan foydalanib, turli markali avtomobillar uchun o‘zgaruvchilarni yaratib, ular haqida ma’lumotlar kiritib, ekranga chiqaring.

15. Berilgan matnli fayldan simvollarni o‘qib, ekranga chiqaring va raqamlarni ajratib ko‘rsating.

16. Berilgan matnli faylda a harfi necha marta qatnashganini sanang.

17. Berilgan matnli fayldagi satrlar sonini aniqlang.

18. Talabalar ism-familiyasi, yoshi va ballari maydonidan iborat klass yarating va talabalar ro‘yhatini tuzing. So‘ralayotgan talaba ro‘yhatda bor yo‘qligini aniqlang.

19. Oy nomlaridan iborat sanaladigan toifa yarating. So‘ralayotgan oy qaysi faslga tegishliligini aniqlang.

20. Mahsulot nomlaridan iborat elementlar va ularning qiymati sifatida narxlari kiritilgan sanaladigan toifa yarating. So‘ralayotgan narxda qanday mahsulot yoki mahsulotlar to‘plamini xarid qilsa bo‘ladi, shuni aniqlash dasturini tuzing.

21. Kiritilgan ismning harflarini alfavit bo‘yicha tartibga keltiring.

22. Satrli toifadagi vektor berilgan. Bir xil qiymatdagi elementlarni aniqlang, ekranga chiqaring.

23. Berilgan ismlardan imlo qoidasiga mos ravishda familiyalar hosil qiling.

24. Berilgan satrda nechta undosh harflar borligini aniqlang.

25. Berilgan satrdagi sonlar yig‘indisini aniqlang.

26. F faylda berilgan satr necha marta uchrashini aniqlang.

27. G fayldan nusxa ko‘chiring.

28. F fayldan matritsa hosil qiling, ya’ni fayldagi har bir qator matritsaning satri va qatordagi ‘#’ belgisi bilan ajratilgan satrlar ustunlar qilib belgilansin.

29. Matritsaning juft va toq elementlaridan 2 ta vektor hosil qiling.

30. Massivdagi tub sonlarni va indekslarini ekranga chiqaring.

2-tajriba ishi. YARIMSTATIK MA’LUMOTLAR TUZILMASI

Ishdan maqsad: Navbat, stek va dekni o‘rganish hamda ularni tadqiq qilish. Yarimstatik ma’lumotlar tuzilmalari ustida amal bajarish algoritmlarini o‘rganish.

Qo‘yilgan masala: C++ tilida navbat, stek va dekni statik ko‘rinishda e’lon qilish va topshiriq variantiga ko‘ra uning ustida amal bajarish dasturini ishlab chiqish.

Ish tartibi:

- Tajriba ishi nazariy ma’lumotlarini o‘rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- C++ dasturlash muhitida dasturni yaratish;
- Natijalarni tekshirish;
- Hisobotni tayyorlash va topshirish.

2.1. Yarimstatik ma’lumotlar tuzilmasi

Yarimstatik ma’lumotlar tuzilmasini quyidagicha tavsiflash mumkin:

- o‘zgaruvchan uzunlikka ega va uni o‘zgartiruvchi oddiy funksiyalariga ega;
- tuzilmaning uzunligini o‘zgartirish ma’lum bir chegarada, ya’ni qandaydir bir maksimal qiymatdan oshmagan holda amalga oshirilishi mumkin;

Agar yarimstatik tuzilmani mantiqiy jihatdan qaraydigan bo‘lsak, u holda chiziqli ro‘yhat munosabati bilan bog‘langan ma’lumotlar ketma-ketligi tushuniladi. Xotirada yarimstatik ma’lumotlar tuzilmasini fizik jihatdan tasvirlaydigan bo‘lsak, bu xotirada slotlarning oddiy ketma-ketligidir, ya’ni har bir element xotirada navbatdagi slotlarda joylashadi. Yarimstatik MTni fizik tasvirlashning yana bir ko‘rinishi bir tomonlama bog‘langan ro‘yhat (zanjir) ko‘rinishida ifodalash mumkin, ya’ni bunda har bir navbatdagi elementning adresi joriy elementda ko‘rsatiladi. Bunday tasvirlashda tuzilmaning uzunligiga

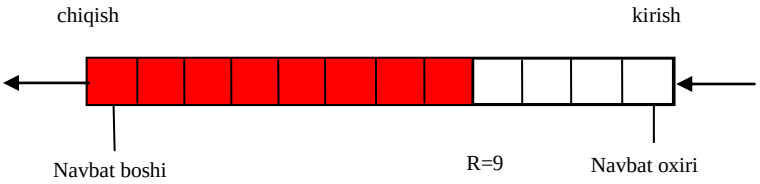
cheklanish unchalik qattiq qo‘yilmaydi. Bunday tuzilmalarga – *navbat, stek, dek* va *satrlar* kiradi.

2.2. Navbat

Navbat bu FIFO (First In - First Out - "birinchi kelgan – birinchi ketadi"), shunday o‘zgaruvchan uzunlikdagi ketma-ketlik, ro‘yhatki, unda tuzilmaga elementlar faqat bir tomondan, ya’ni navbatning oxiridan qo‘shiladi va elementlarni tuzilmadan chiqarish boshqa tomondan, ya’ni navbat boshidan amalga oshiriladi. Navbat ustida bajariladigan asosiy amallar

- yangi elementni qo‘shish,
- elementni chiqarib tashlash,
- uzunligini aniqlash,
- navbatni tozalash.

Navbatni statik xotirada vektor ko‘rinishida ifodalashda 2 ta parametr, ya’ni navbat boshini (navbatning 1-elementini) va oxirini (navbatning oxirgi elementini) ko‘rsatuvchi ko‘rsatkichlar olinadi (2.1-rasm).



2.1-rasm. Navbat tuzilmasi

Navbatga yangi element kiritilayotganda navbat oxiri ko‘rsatkichi ko‘rsatayotgan adresga yoziladi va shundan keyin navbat oxiri ko‘rsatkichi bittaga oshiriladi. Navbatdan elementni o‘chirishda navbat boshi ko‘rsatkichi ko‘rsatayotgan adresdagi element o‘chiriladi va shundan keyin bu ko‘rsatkichning qiymati bittaga oshiriladi. Navbatga elementlar kiritilganda navbat oxiri ko‘rsatkichi shu navbat uchun ajratilgan xotira sohasining oxiriga yetib qoladi. Bunda navbat to‘lgan hisoblanadi.

Agar navbatdan elementlar o‘chiriladigan bo‘lsa, navbat boshida bo‘sh joy ajratiladi. Vaholanki, navbat oxiri ko‘rsatkichi chegaraga yetib qolganligi sababli, navbatga yangi element kiritib bo‘lmaydi. Shu sababli navbatda har safar element o‘chirilganda qolgan barcha elementlar bitta oldinga surilishi kerak bo‘ladi. Natijada navbat oxirida bo‘sh joy ochiladi. Bu holatda navbat boshi ko‘rsatkichiga xojat qolmaydi. Lekin shuni aytish kerakki, bu yondashuv bir muncha noqulay hisoblanadi. Shuning uchun har safar elementlarni surib o‘tirmaslik uchun navbatni halqasimon shaklda tashkil etamiz. Ya’ni bunda xotirada navbat sohasining oxiriga yetib borilganda navbat boshiga o‘tib ketiladi. Ushbu holatda navbat boshi va oxiri ko‘rsatkichlari xotiradagi navbat sohasining boshini ko‘rsatadi. Bu ikkala ko‘rsatkichlarning tengligi navbatning bo‘shligini anglatadi. Halqasimon navbatda element qo‘shish amali o‘chirish amalidan ko‘proq bajarilsa, navbat oxiri ko‘rsatkichi navbat boshi ko‘rsatkichiga “yetib oladi”. Bu holat navbat to‘laligini anglatadi. Halqasimon navbatda elementni o‘chirish ikkala ko‘rsatkich ko‘rsatayotgan bitta adresda amalga oshiriladi. Bunday navbatning uzunligi boshi va oxiri ko‘rsatkichlari farqi bilan aniqlanadi.

***C++ tilida navbatni statik, ya’ni bir o‘lchamli massiv ko‘rinishda amalga oshirishga misol*:**

Navbat uchun 10 ta joy ajratilgan bo‘lsin, navbatni butun sonlardan iborat massiv shaklida ifodalaymiz. Bunda navbat dastlab bo‘shligi sababli, navbat oxiri ko‘rsatkichi R=0 bo‘ladi. Navbatga yangi element qo‘shish va navbatdan elementni chiqarib olish algoritmi, navbat bo‘shligini va to‘laligini tekshirish algoritmlari quyidagi dasturda keltirilgan.

Masala. Butun sonlardan iborat navbatning juft elementlarini o‘chirish dasturini keltiramiz.

Algoritm

1. Agar navbat to‘lmagan bo‘lsa unga element kiritamiz, kiritib bo‘lgach keyingi 2-qadamga o‘tish, aks holda navbat to‘lganligini xabar berib, keyingi 2-qadamga o‘tish.

2. Agar navbat bo‘sh bo‘lmasa 3-qadamga o‘tamiz, aks holda 4-qadamga o‘tamiz.

3. Navbatning chiqishiga kelib turgan elementni olib, juftlikka tekshiramiz. Agar element toq bo‘lsa, uni navbatga kiritamiz. 2-qadamga o‘tish.

4. Navbat bo‘sh bo‘lsa, bu haqda xabar berib keyingi 5-qadamga o‘tamiz.

5. Navbat tarkibini ekranga chiqaramiz.

Dastur kodi

```
#include <iostream>
using namespace std;
int a[10],R=0,n;//bu yerda n navbatga kiritilishi kerak bo'lgan elementlar soni.
int kiritish(int s){
   a[R]=s; R++;
   }
int chiqarish(){
   int t=a[0];
   for(int i=0;i<R-1;i++)
     a[i]=a[i+1];
   R--;
   return t;
   }
 bool isEmpty(){
if(R==0) return true; else return false;
}
bool isFull(){
if(R>=10)return true;else return false;
}
int print(){
   int i;
while(i<R){
```

```
int k=chiqarish();i++;
cout<<k<<"  ";
kiritish(k);}
}
int main(){
int n,s;
   cout<<"n=";cin>>n;
   for(int i=0;i<n;i++){
      if(!isFull()){cin>>s;
      kiritish(s);}
else{cout<<"navbat to'ldi"; n=i;break;}
      }
   cout<<"\nnavbat elementlari: ";
   print();
   for(int i=0;i<n;i++){
      s=chiqarish();
      if(s%2!=0)kiritish(s);
      }
   cout<<"\nnatijaviy navbat elementlari: ";
   print();
   system("PAUSE");
}
```

Dasturning bajarilishi natijasi:

```
n=5
6
7
9
8
11
```

navbat elementlari: 6 7 9 8 11

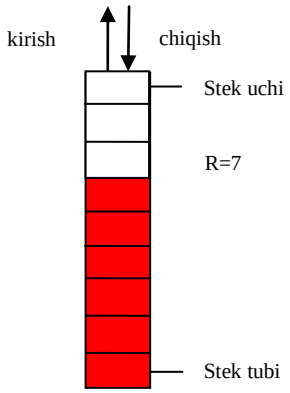
natijaviy navbat elementlari: 7 9 11

2.3. Steklar

Stek bu *LIFO* (Last In - First Out - "oxirgi kelgan – birinchi ketadi"), shunday oʻzgaruvchan uzunlikdagi ketma-ketlik, roʻyhatki, unda tuzilmaga elementlarni kiritish va chiqarish amallari bir tomondan, ya'ni stek uchidan amalga oshiriladi. Stek ustida bajariladigan asosiy amallar:

- yangi elementni qoʻshish;
- elementni oʻchirish;
- stek elementlar sonini aniqlash;
- stekni tozalash.

Stekni statik xotirada vektor koʻrinishida ifodalashda stek uzunligini koʻrsatuvchi koʻrsatkich ishlatiladi. Bu koʻrsatkich stekdagi 1-boʻsh joyni koʻrsatadi. Dastlab hali stek boʻshligida bu koʻrsatkich R=0 boʻladi. Quyidagi rasmda stekda 6 ta element mavjudligi uchun R=7 boʻladi (2.2-rasm).



2.2-rasm. Stek tuzilmasi

Stekka yangi element kiritilayotganda stek koʻrsatkichi (R) koʻrsatayotgan adresga yoziladi va shundan keyin bu koʻrsatkich bittaga oshiriladi. Stekdan elementni oʻchirishda koʻrsatkichning qiymati bittaga kamaytiriladi va shu adresdagi element oʻchiriladi. Stekni tozalash amalini bajarish uchun stek

ko‘rsatkichi R ga stek uchun ajratilgan xotira sohasining boshlang‘ich adresi qiymati beriladi. R stekdagi elementlar sonini bildiradi.

***C++ tilida stekni statik ko‘rinishda, ya’ni bir o‘lchamli massiv ko‘rinishida amalga oshirishga misol*:**

Masalaning qo‘yilishi: Elementlari butun sonlardan iborat stekning juft qiymatli elementlari o‘chirilsin. Aytaylik, stek uchun 10 ta joy ajratilgan bo‘lsin, bunda dastlab stek bo‘shligi sababli R=0 bo‘ladi. Stekga yangi element qo‘shish va chiqarish, stek bo‘shligini va to‘laligini tekshirish funksiyalaridan foydalanib shu masalani yechamiz.

Algoritm

1. Agar stek to‘lmagan bo‘lsa elementlarni kiritamiz. Stekning toq elementlarini saqlab turish uchun yangi **b[]** massiv e’lon qilamiz.

2. Agar stek bo‘sh bo‘lmasa, 3-qadamga o‘tish, aks holda 4-qadamga o‘tish.

3. Stek uchidagi elementni olamiz va juftlikka tekshiramiz. Agar element toq bo‘lsa b massivga joylaymiz. 2-qadamga o‘tish.

4. b massiv elementlarini teskari tartibda stekka joylash.

5. Stek tarkibini ekranga chiqarish.

Dastur kodi

```
#include <iostream>
using namespace std;
int a[10],R=0,n;//bu yerda n stekka kiritilishi kerak bo'lgan elementlar soni.
int kiritish(int s){
   a[R]=s; R++;
   }
int chiqarish(){
   R--;
   return a[R];
   }
  bool isEmpty(){
if(R==0) return true;
```

```
else return false;
}
  bool isFull(){
if(R>=10) return true;else return false;
}
int print(){
    int i=0,c[n];
while(!isEmpty()){
c[i]=chiqarish();
cout<<c[i]<<"  ";i++;}
for(int j=i-1;j>=0;j--) kiritish(c[j]);
}
int main(){
int n,s;
    cout<<"n=";cin>>n;
    for(int i=0;i<n;i++){
       if(!isFull()){
       cin>>s;
       kiritish(s);}
       else{cout<<"stek to'ldi"; n=i;break;}
       }
    cout<<"\nstek elementlari: ";
    print();
int b[n],k=0;
    for(int i=0;i<n;i++){
       s=chiqarish();
       if(s%2!=0) b[k++]=s;
       }
    for(int i=k-1;i>=0;i--) kiritish(b[i]);
```

```
 cout<<"\nnatijaviy stek elementlari: ";
  print();
  system("PAUSE");
}
```

Dasturning bajarilishi natijasi:

n =5

6

7

9

8

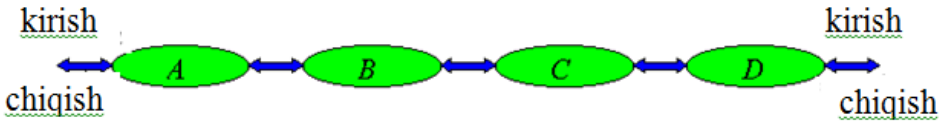
11

stek elementlari: 11 8 9 7 6

natijaviy stek elementlari: 11 9 7

2.4. Deklar

Dek so‘zi (*DEQ* - Double Ended Queue) ingliz tilidan olingan bo‘lib 2 ta chetga ega navbat degan ma’noni bildiradi. Dekning o‘ziga xos xususiyati shuki, unga elementlar har ikkala tomondan – chapdan va o‘ng tomondan kiritilishi va chiqarilishi mumkin (2.3-rasm).



2.3-rasm. Dek tuzilmasi

Dek ustida bajariladigan amallar:

1. Chapdan element kiritish.
2. O‘ngdan element kiritish.
3. Chapdan element chiqarish.

4. O‘ngdan element chiqarish.

5. Dek bo‘shligini tekshirish.

6. Dek to‘laligini tekshirish.

***C++ tilida dekni statik ko‘rinishda, ya’ni bir o‘lchamli massiv ko‘rinishida amalga oshirishga misol*:** Berilayotgan butun sonlar ketma-ketligining 1-yarmini dekning chap tomonidan, qolgan yarmini dekning o‘ng tomonidan kiriting. Dekning elementlarini bir safar chapdan, bir safar o‘ngdan juftlikka tekshirib, toq elementlari o‘chirilsin.

Algoritm

1. Dekka nechta element kiritilishi aniqlanadi – n, i=0.

2. i++; agar i<n bo‘lsa va dek to‘lmagan bo‘lsa, 3-qadamga, aks holda 4-qadamga o‘tiladi.

3. Agar i<n/2 bo‘lsa, navbatdagi element chapdan, aks holda, ya’ni i>n/2 bo‘lsa, dekning o‘ng tomonidan kiritiladi, 2-qadamga o‘tish.

4. Agar dek bo‘sh bo‘lmasa, chapdan element chiqarib olamiz. Agar element juft bo‘lsa, b[] massivga joylaymiz. 5-qadamga o‘tiladi. Agar dek bo‘sh bo‘lsa, 6-qadamga o‘tish.

5. Agar dek bo‘sh bo‘lmasa, o‘ngdan element chiqarib olamiz. Agar element juft bo‘lsa, b[] massivga joylaymiz. 5-qadamga o‘tiladi. Agar dek bo‘sh bo‘lsa, 6-qadamga o‘tish.

6. b[] massiv elementlarini dekka o‘ng tomondan kiritamiz.

7. Dek tarkibini ekranga chiqaramiz.

Dastur kodi

```
#include <cstdlib>
#include <iostream>
using namespace std;
int a[10],n,R=0;
bool isEmpty(){
    if(R==0) return true; else return false;
    }
```

```
bool isFull(){
     if(R>=10) return true; else return false;
     }
int kirit_left(int s){
    if(isFull()){cout<<"\ndek to'ldi";n=R;return EXIT_SUCCESS;}
    for(int i=R;i>0;i--)
    a[i]=a[i-1];
    a[0]=s;R++;
    }
int olish_left(){
    if(isEmpty()){cout<<"\ndek bo'sh";return EXIT_SUCCESS;}
    int t=a[0];
    for(int i=0;i<R-1;i++)
    a[i]=a[i+1];
    R--;
    return t;
    }
int kirit_right(int s){
   if(isFull()){cout<<"\ndek to'ldi";n=R;return EXIT_SUCCESS;}
    a[R]=s;R++;
    }
int olish_right(){
    if(isEmpty()){cout<<"\ndek bo'sh";return EXIT_SUCCESS;}
    R--;
    return a[R];
    }
 int print(){
      cout<<endl<<"dek ele-tlari=";
      for(int i=0;i<R;i++)cout<<a[i]<<"  ";
      cout<<endl;
```

```
    }
int main(int argc, char *argv[])
{ int n,s;cout<<"n="; cin>>n;
   for(int i=0;i<n;i++){
         if(!isFull()){
         cout<<"kirit=";cin>>s;
         if(i>=n/2) kirit_right(s);
         else kirit_left(s);}
         else {cout<<"dek to'ldi\n";break;}
      }
      print();
      int b[n/2],k=0,c[n/2],p=0;
while(!isEmpty()){
            int q=olish_left();
            if(q%2==0) b[k++]=q;
            if(isEmpty()) break;
            int p=olish_right();
            if(p%2==0) b[k++]=p;
            }
            int i=0;
      while(i<k){
             kirit_right(b[i]);
             i++;
             }
     print();
     system("PAUSE");
   return EXIT_SUCCESS;
}
```

Dastur natijasi

n=8

kirit=1

kirit=2

kirit=3

kirit=4

kirit=5

kirit=6

kirit=7

kirit=8

dek ele-tlari=4 3 2 1 5 6 7 8

dek ele-tlari=4 8 2 6

Nazorat savollari

1. Statik va yarimstatik ma’lumotlar tuzilmasi nima va farqini tushuntiring?

2. Navbat tuzilmasini tushuntiring.

3. Yarimstatik ma’lumotlar tuzilmasini dasturda e’lon qilishning qanday usullarini bilasiz?

4. Stek tuzilmasini tushuntiring va misol keltiring.

5. Dek nima va navbat tuzilmasidan farqi nimada?

Topshiriq

Variantlar:

1. Navbatda birinchi va oxirgi elementlar o‘rni almashtirilsin.

2. Navbat o‘rtasidagi element o‘chirib tashlansin. Agar navbat elementlari soni toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.

3. Navbatni juft o‘rinda turgan elementlari o‘chirilsin.

4. Navbat o‘rtasiga '+' belgi joylashtirilsin.

5. Navbat eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.

6. Navbat eng katta elementi topilsin va undan keyin 0 joylashtirilsin.

7. Navbat eng kichik elementi oʻchirilsin.

8. Navbatda birinchi elementga teng barcha elementlar oʻchirilsin.

9. Navbatda oxirgi elementga teng barcha elementlar oʻchirilsin.

10. Navbat eng katta elementi oʻchirilsin.

11. Navbat eng kichik elementi topilsin va uning oʻrniga 0 joylashtirilsin.

12. Stek birinchi va oxirgi elementlari oʻrni almashtirilsin.

13. Stek elementlari teskari tartibda joylashtirib chiqilsin.

14. Stek oʻrtasidagi element oʻchirib tashlansin. Agar stek elementi toq boʻlsa, bitta element, aks holda ikkita element oʻchirilsin.

15. Stekning juft oʻrinda turgan elementlari oʻchirilsin.

16. Stek oʻrtasiga '*' belgi joylashtirilsin.

17. Stek eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.

18. Stek eng katta elementi topilsin va undan keyin 0 joylashtirilsin.

19. Stek eng kichik elementi oʻchirilsin.

20. Stekda birinchi elementga teng barcha elementlar oʻchirilsin.

21. Stek oxirgi elementiga teng barcha elementlar oʻchirilsin.

22. Stek eng katta elementi oʻchirilsin.

23. Stek eng kichik elementi topilsin va uning oʻrniga 0 joylashtirilsin.

24. Dekning har 2 ta elementidan keyin ularning yigʻindisini joylang.

25. Dekning oʻrtasidagi element yoki elementlarni oʻchiring.

26. Dekning juft elementlari yigʻindisini hisoblang.

27. Berilgan soʻzning unli harflarini dekning chap tomonidan, undoshlarini oʻng tomondan kiriting.

28. Dekning toq elementlaridan navbat, juft elementlaridan stek hosil qiling.

29. Dekdagi manfiy sonlarni oʻchiring.

30. Dekni oʻrtasiga “dek” soʻzini kiriting.

3-tajriba ishi. DINAMIK MA’LUMOTLAR TUZILMASINI TADQIQ QILISH

Ishdan maqsad: Chiziqli, bir bog‘lamli ro‘yhatlar tuzilmasini o‘rganish va uni ustida amal bajarish algoritmlarini tadqiq qilish.

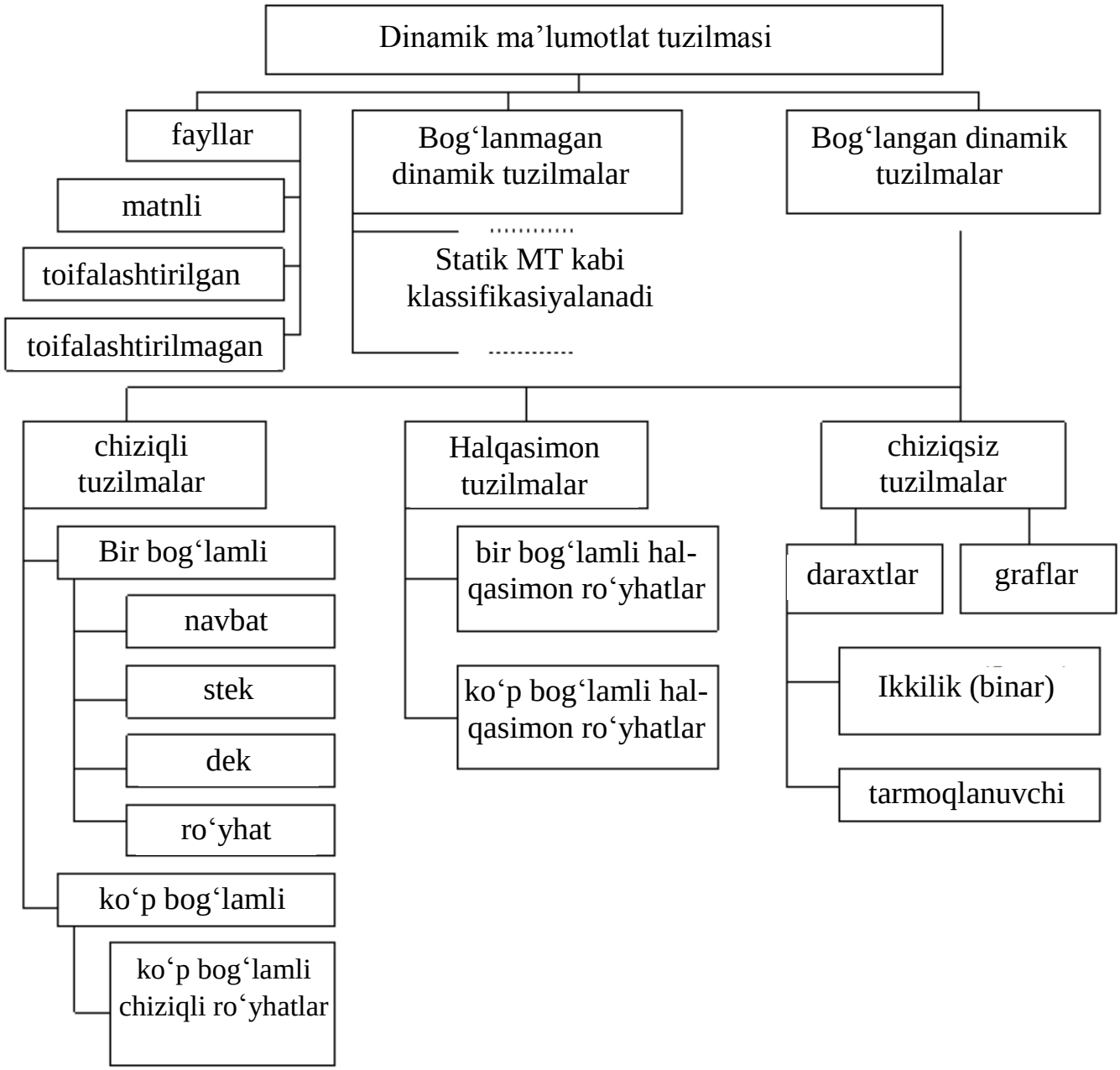
Qo‘yilgan masala: C++ tilida ro‘yhatli tuzilma elementlarini ko‘rsatkichli maydonlar bilan yaratish va dinamik tuzilmani e’lon qilish, uning ustida turli amallar bajarish dasturini ishlab chiqish.

Ish tartibi:

- Tajriba ishi nazariy ma’lumotlarini o‘rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- C++ dasturlash muhitida dasturni yaratish;
- Natijalarni tekshirish;
- Hisobotni tayyorlash va topshirish.

3.1. Dinamik ma’lumotlar tuzilmasi

Statik ma’lumotlar tuzilmasi vaqt o‘tishi bilan o‘z o‘lchamini o‘zgartirmaydi. Biz har doim dastur kodidagi statik ma’lumotlar tuzilmasiga qarab ularning o‘lchamini bilishimiz mumkin. Bunday ma’lumotlarga teskari ravishda dinamik ma’lumotlar tuzilmasi mavjud bo‘lib, bunda dastur bajarilishi davomida dinamik ma’lumotlar tuzilmasi o‘lchamini o‘zgartirishi mumkin**.** *Dinamik ma’lumotlar tuzilmasi* – bu qandaydir bir qonuniyatga asoslanib shakllangan, lekin elementlari soni, o‘zaro joylashuvi va o‘zaro aloqasi dastur bajarilishi davomida shu qonuniyat asosida dinamik o‘zgaruvchan bo‘lgan ma’lumotlar tuzilmasidir. Dinamik ma’lumotlar tuzilmasi 3.1-rasmdagidek klassifikatsiyalanadi.



3.1-rasm. Dinamik ma’lumotlar tuzilmasi klassifikatsiyasi

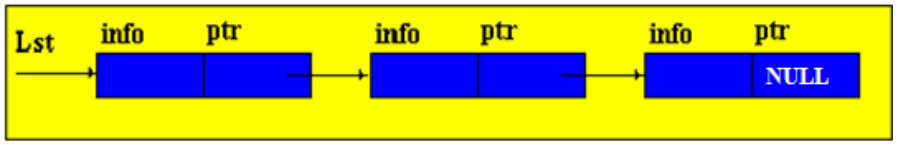
Dasturlarda dinamik ma’lumotlar tuzilmasidan ko‘pincha chiziqli ro‘yhatlar, steklar, navbatlar va binar daraxtlar ishlatiladi. Bu tuzilmalar bir-biridan elementlarning bog‘lanish usuli va ular ustida bajarilishi mumkin bo‘lgan amallari bilan farqlanadi. Dinamik tuzilmalar massiv va yozuvdan farqli ravishda operativ xotirada ketma-ket sohalarda joylashmaydi. Ixtiyoriy dinamik tuzilma elementi 2 ta maydondan tashkil topadi: tuzilma tashkil etilishiga sabab bo‘layotgan

informatsion maydon va elementlarning o‘zaro aloqasini ta’minlovchi *ko‘rsatkichli maydon*. Chiziqli ro‘yhatlarda har bir element o‘zidan keyingisi yoki oldingisi bilan ham bog‘langan bo‘lishi mumkin. Birinchi holatda, ya’ni elementlar o‘zidan keyingi element bilan bog‘langan bo‘lsa, bunday ro‘yhatga *bir bog‘lamli ro‘yhat* deyiladi. Agar har bir element o‘zidan oldingi va o‘zidan keyingi element bilan bog‘langan bo‘lsa, u holda bunday ro‘yhatlarga *2 bog‘lamli ro‘yhatlar* deyiladi. Agar oxirgi element birinchi element ko‘rsatkichi bilan bog‘langan bo‘lsa, bunday ro‘yhatga *halqasimon ro‘yhat* deyiladi. Ro‘yhatning har bir elementi shu elementni identifikatsiyalash uchun *kalit*ga ega bo‘ladi. Kalit odatda butun son yoki satr ko‘rinishida ma’lumotlar maydonining bir qismi sifatida mavjud bo‘ladi. Ro‘yhatlar ustida quyidagi amallarni bajarish mumkin.

- ro‘yhatni shakllantirish (birinchi elementini yaratish);
- ro‘yhat oxiriga yangi element qo‘shish;
- berilgan kalitga mos elementni o‘qish;
- ro‘yhatning ko‘rsatilgan joyiga element qo‘shish (berilgan kalitga mos elementdan oldin yoki keyin)
- berilgan kalitga mos elementni o‘chirish;
- kalit bo‘yicha ro‘yhat elementlarini tartibga keltirish.

Ro‘yhatlar bilan ishlashda dasturda boshlang‘ich elementni ko‘rsatuvchi ko‘rsatkich talab etiladi. Chiziqli bir bog‘lamli ro‘yhatlar ustida turli amallar bajarish algoritmlari va dasturlarini ko‘rib chiqamiz.

3.2. Chiziqli bir tomonlama yo‘nalgan ro‘yhatlar



3.2-rasm. Chiziqli bir bog‘lamli ro‘yhatlar

Bir bog‘lamli ro‘yhat deb elementlarning shunday tartiblangan ketma-ketligiga aytiladiki, har bir element 2 ta maydonga: informatsion maydon *info* va ko‘rsatkich maydoni *ptr* ga ega bo‘ladi (3.2-rasm).

Mazkur ko‘rinishdagi ro‘yhat elementi ko‘rsatkichining o‘ziga xosligi shundan iboratki, u faqatgina ro‘yhatning navbatdagi (o‘zidan keyin keluvchi) elementi adresini ko‘rsatadi. Bir tomonlama yo‘naltirilgan ro‘yhatda eng so‘ngi element ko‘rsatkichi bo‘sh, ya’ni *NULL* bo‘ladi.

Lst – ro‘yhat boshi ko‘rsatkichi. U ro‘yhatni yagona bir butun sifatida ifodalaydi. Ba’zan ro‘yhat bo‘sh bo‘lishi ham mumkin, ya’ni ro‘yhatda bitta ham element bo‘lmasligi mumkin. Bu holda *lst = NULL* bo‘ladi. Hozir chiziqli bir bog‘lamli ro‘yhat hosil qilish dasturini ko‘rib chiqsak. Buning uchun biz foydalanuvchi tomonidan yaratiladigan nostandart toifa yaratib olishimiz kerak. Buning bir qancha usullari mavjud, ya’ni klasslar yoki strukturalar bilan amalga oshirish mumkin. Masalan,

```
class Node{
public://klass ma’lumotlariga tashqaridan bo‘ladigan murojaatga ruxsat berish
    int info;   // informatsion maydon
    Node* next;// ko‘rsatkichli maydon
};
```

Bu yerda biz *Node* nomli toifa yaratdik va ro‘yhatimiz butun sonlardan iborat. Endi ro‘yhat elementlarini shu toifa orqali e’lon qilsak bo‘ladi, ya’ni

```
Node *lst = NULL;// ro‘yhat boshi ko‘rsatkichi
Node *last = NULL;// ro‘yhatga oxirgi kelib tushgan elementning ko‘rsatkichi
```

Endi shu belgilashlar orqali ro‘yhat hosil qilamiz.

```
Node * p = new Node;
int numb = -1;
  cout<<"son kiriting: ";
  cin>>numb;
```

```
p->info = numb;
p->next = NULL;
  if (lst == NULL) {
                    lst = p;
                    last = p;
                     }
 else{  last->next = p;
        last = p; }
```

Bu dasturda yangi element ro‘yhat oxiridan kelib qo‘shiladi.

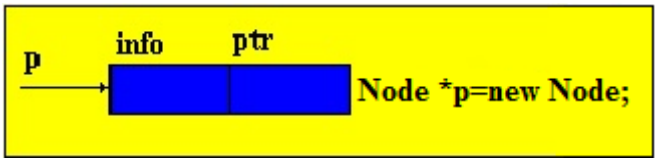
Bir bog‘lamli ro‘yhatlar ustida amallar bajarish algoritmlari

1. Bir bog‘lamli ro‘yhat boshiga element qo‘yish

3.3-rasm. Bir bog‘lamli chiziqli ro‘yhat tuzilishi

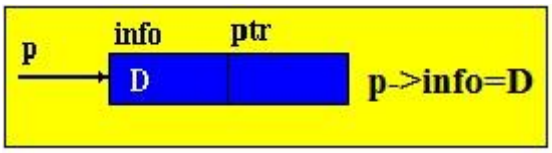
3.3-rasmdagi ro‘yhat boshiga informatsion maydoni D o‘zgaruvchi bo‘lgan element qo‘yamiz. Ushbu ishni amalga oshirish uchun quyidagi amallarni bajarish lozim bo‘ladi:

a) p ko‘rsatkich murojaat qiladigan, bo‘sh element yaratish (3.4-rasm).



3.4-rasm. Yangi element hosil qilish

b) Yaratilgan element informatsion maydoniga D o‘zgaruvchi qiymatini o‘zlashtirish (3.5-rasm).

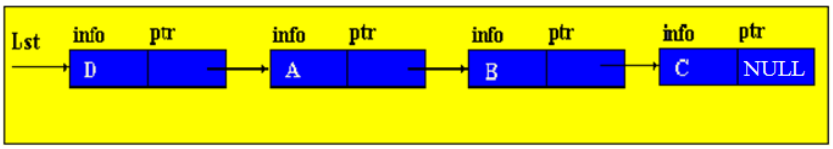


3.5-rasm. Yangi element info maydoniga qiymat kiritish

c) Yangi elementni ro‘yhat bilan bog‘lash: p->ptr=lst; (shu holatda yangi element va lst – ro‘yhat boshini ko‘rsatyapti)

d) lst ko‘rsatkichni ro‘yhat boshiga ko‘chirish (3.6-rasm). lst=p;

Va nihoyat:



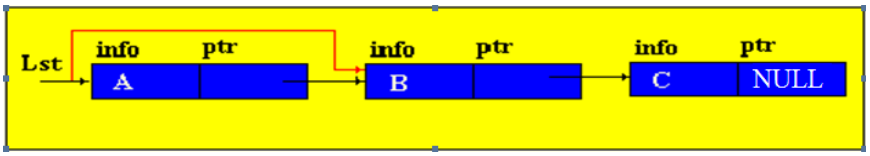
3.6-rasm. Ro‘yhat boshiga element qo‘shish

Endi shu algoritmni C++ tilidagi realizatsiyasini ko‘rib chiqamiz.

```
Node * p = new Node;
        int numb = -1;
        cout<<"son kiriting: ";
        cin>>numb;
        p->info = numb;
       if (lst ==NULL){
              p->next = NULL;
              lst = p; }
       else { p->next = lst;
              lst = p;}
```

2. Bir bog‘lamli ro‘yhat boshidan elementni o‘chirish

Ro‘yhatda birinchi element info informatsion maydonidagi ma’lumotni esda saqlab qolib uni ro‘yhatdan o‘chiramiz (3.7-rasm).



3.7-rasm. Ro‘yhat boshidagi elementni o‘chirish

Yuqorida aytilganlarni amalga oshirish uchun quyidagi ishlarni bajarish lozim:

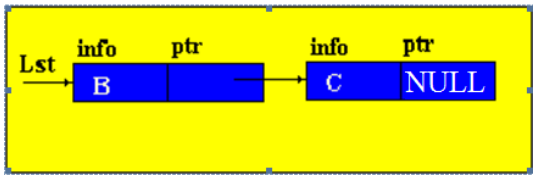
a) o‘chirilayotgan elementni ko‘rsatuvchi p ko‘rsatkich kiritish: p=lst;

b) p ko‘rsatkich ko‘rsatayotgan element info maydonini qandaydir x o‘zgaruvchida saqlash: x=p->info;

c) lst ko‘rsatkichni yangi ro‘yhat boshiga ko‘chirish: lst=p->ptr;

d) p ko‘rsatkich ko‘rsatayotgan elementni o‘chirish: delete(p);

Natijada 3.8-rasmdagi ko‘rinishga ega bo‘lamiz.



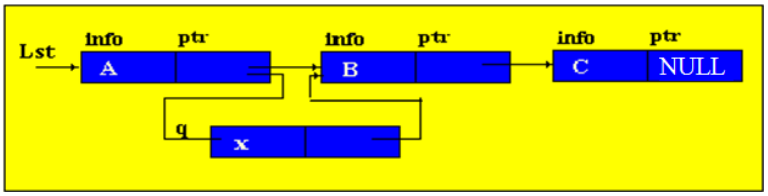
3.8-rasm. Ro‘yhatning natijaviy ko‘rinishi

Endi shu algoritmni C++ tilidagi realizatsiyasini ko‘rib chiqsak.

```
Node* p = new Node;
        if (lst == NULL){
            cout<<"ro'yhat bo'sh";
             system("pause");
             system("CLS");
                }
       else { p = lst;
             lst = p->next ;
            delete(p);
            }
```

3. Elementni ro‘yhatga qo‘shish

Berilgan ro‘yhatda p ko‘rsatkich ko‘rsatayotgan elementdan keyin informatsion maydoni x bo‘lgan elementni qo‘yamiz (3.9-rasm).



3.9-rasm. Ro‘yhatga yangi element qo‘shish

Aytilganlarni amalga oshirish uchun quyidagi amallarni bajarish lozim:

a) q ko‘rsatkich ko‘rsatuvchi bo‘sh elementni yaratish: Node *q=new Node;

b) Yaratilgan element informatsion maydoniga x ni kiritish: q->info=x;

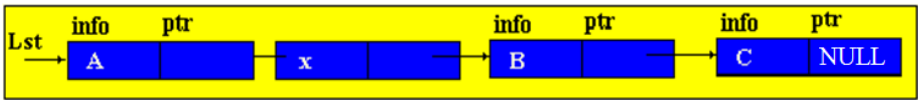
c) q elementni p elementdan keyingi element bilan bog‘lash.

q->ptr=p->ptr – yaratilgan element ko‘rsatkichiga p element ko‘rsatkichini o‘zlashtirish.

d) p element bilan q elementni bog‘lash.

p->ptr=q – bu amal p elementdan keyingi element q ko‘rsatkich murojaat qilgan element bo‘lishini anglatadi.

Natijada quyidagi rasmdagidek ko‘rinishga ega bo‘lamiz.



3.10-rasm. Natijaviy ro‘yhat ko‘rinishi

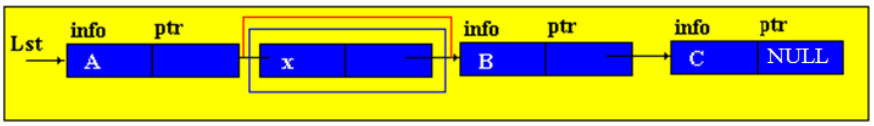
Endi shu algoritmni C++ tilidagi realizatsiyasini ko‘rib chiqsak.

```
Node * p = lst;
    Node * q = new Node;
    int numb = -1;
    cout<<"son kiriting: ";
    cin>>numb;
```

```
q->number = numb;
 int k;
cout<<"nechta elementdan keyin kiritasiz k=";cin>>k;
for(int i=0;i<k-1;i++) p=p->next;
q->next = p->next;
p->next = q;
```

4. Bir bog‘lamli ro‘yhatdan elementni o‘chirish

Ro‘yhatda p ko‘rsatkich ko‘rsatayotgan elementdan keyingi elementni o‘chiramiz (3.11-rasm).



3.11-rasm. Ro‘yhat o‘rtasidan element o‘chirish

Buni ro‘yobga chiqarish uchun quyidagi ishlarni amalga oshirish lozim:

a) O‘chirilayotgan elementni ko‘rsatuvchi q ko‘rsatkichni kiritish.

q=p->ptr;

b) p elementni q elementdan keyingi element bilan bog‘lash.

p->ptr=q->ptr;

c) O‘chirilayotgan element info maydonidagi informatsiyani yodda saqlash (agar zarur bo‘lsa) k=q->info;

d) q ko‘rsatkich ko‘rsatayotgan elementni o‘chirish.

delete(q)

Natijada ro‘yhat quyidagi ko‘rinishga ega bo‘ladi:

3.12-rasm. Natijaviy ro‘yhat ko‘rinishi

Shu algoritm dasturi:

```
Node* p = lst;
Node* q = new Node;
int k;
cout<<"k=";cin>>k;
for(int i=0;i<k-1;i++) p=p->next;
q = p->next;
p->next = q->next;
delete(q);
```

Ishni bajarishga namuna

Topshiriq variantlariga o'xshash bitta misolni yechish dasturini ko'rib chiqamiz. Quyidagicha masala qo'yilgan bo'lsin. Ro'yhatning maksimal elementi topilsin. Ushbu masalaning algoritmi, dasturiy kodi va natijasi quyida keltirilgan.

Algoritm

1. Ekranga menyu chiqaramiz: 1 - *element qo'shish; 2 - ro'yhatni ko'rish; 3 - ro'yhat maksimalini topish; 0 - chiqish;* tanlash uchun *tanla* o'zgaruvchisiga qiymat so'raymiz. 2-qadamga o'tish.

2. Agar *tanla*=1 bo'lsa, 3-qadamga, 2 ga teng bo'lsa, 4-qadamga, 3 tanlansa, 6-qadamga o'tish, 0 tanlansa dasturni yakunlash.

3. Navbatdagi elementni yaratish p; (p ning info maydoniga qiymat so'rab olib yozish va ptr maydoniga NULL yozish) Agar ro'yhat boshi ko'rsatkichi lst=NULL bo'lsa, lst=p va last=p; aks holda last – ro'yhat oxirgi elementi ptr maydoniga p ni yozib, p elementni last qilib belgilaymiz. 1-qadamga o'tamiz.

4. Agar lst NULL ga teng bo'lsa, ro'yhat bo'shligini ekranga chiqarib, 1-qadamga o'tish. Aks holda, p=lst va 5-qadamga o'tish.

5. Agar p ning ptr maydoni NULL bo'lmasa, p ning info maydonini ekranga chiqaramiz va keyingi elementga o'tamiz, ya'ni p=p->ptr, 5-qadamga o'tamiz, aks holda, 1-qadamga o'tamiz.

6. max=lst->info, ya'ni max o'zgaruvchisiga ro'yhat 1-elementi info maydoni qiymatini o'zlashtiramiz. p=lst va 7-qadamga o'tish.

7. Agar p NULL ga teng bo'lmasa, 8-qadamga o'tamiz, aks holda max ni ekranga chiqaramiz va 1-qadamga o'tamiz.

8. Agar max< p->info bo'lsa, max=p->info. Keyingi elementga o'tamiz, ya'ni p=p->ptr. 7-qadamga o'tamiz.

Dastur kodi

```
#include <iostream>
using namespace std;
class Node{
public:  int number;
         Node* next;
};
int main()
{      Node* head = NULL;
       Node* lastPtr = NULL;
       short action = -1;
       while (1)
       {      cout<<"1. element qo'shish\n";
              cout<<"2. ro'yhatni ko'rish\n";
              cout<<"3. ro'yhat maksimalini topish\n";
              cout<<"0. chiqish\n\n";
              cout<<"tanlang: ";
              cin>>action;
              if (action == 0) {
                     system("CLS");
                     break;}
               if (action == 1)
              {      system("CLS");
                     Node* ptr = new Node;
                     int numb = -1;
```

```
            cout<<"son kiriting: ";
            cin>>numb;
            ptr->number = numb;
            ptr->next = NULL;
            if (head == 0)
            {     head = ptr;
                  lastPtr = ptr;
                  system("CLS");
                  continue;
            }
            lastPtr->next = ptr;
            lastPtr = ptr;
            system("CLS");
            continue;
      }
if (action == 2){
            Node* ptr = NULL;
            system("CLS");
            if (head == 0)
            {     cout<<"\t!!! ro'yhat bo'sh !!!\n\n";
                  system("PAUSE");
                  system("CLS");
                  continue;
            }
            cout<<"* * * * * ro'yhat * * * * *\n\n";
            ptr = head;
            while (1) {
                  cout<<ptr->number<<" ";
                  if (ptr->next == 0)  break;
```

```
ptr = ptr->next;
                }
                cout<<"\n\n";
                system("PAUSE");
                system("CLS");
                continue;
            }
          if (action == 3)
            {
                system("CLS");
                Node* p = head;
                Node* q = new Node;
                Node* last = new Node;
                int max=p->number; q=head;
                while(p){
                    if(max<p->number){ max=p->number;}
                    p=p->next;
                    }
                system("CLS");
                cout<<"max="<<max;
                system("pause");
                continue;
            }
}}
```

Dastur bajarilishi natijasi

1. element qo'shish

2. ro'yhatni ko'rish

3. ro'yhat maksimalini topish

0. chiqish

tanlang:1

{1,2,3,55,4,6} sonlari kiritildi. 2-holat tanlanganda natija:

******ro'yhat******

1 2 3 55 4 6

3-holat tanlanganda natija:

max=55

Nazorat savollari

1. Dinamik ma'lumotlar tuzilmasi nima va uning statik tuzilmalardan afzalligini tushuntiring?
2. Ro'yhat tuzilmasi nima va ro'yhatning qanday turlarini bilasiz?
3. Ro'hat tuzilmasini dasturda ifodalash qanday amalga oshiriladi?
4. Ro'hat tuzilmasi ustida amal bajarish algoritmlarini tushuntiring
5. Ikki bog'lamli ro'yhat nima va uni bir bog'lamli ro'hatdan afzalligi va kamchiligini tushuntiring.

Topshiriq

Variantlar:

1. Elementni n pozitsiyaga siljitish dasturini tuzing.
2. Ro'yhat nusxasini yarating.
3. Ro'yhat boshiga element qo'yish.
4. Ikkita ro'yhat birlashtirilsin.
5. Ro'yhatning n-inchi elementi o'chirilsin.
6. Ro'yhat n-inchi elementidan keyin yangi element qo'yilsin.
7. Ikkita ro'yhat umumiy elementlaridan tashkil topgan ro'yhat yaratilsin.
8. Ro'yhat elementlari o'sish tartibida joylashtirilsin.
9. Ro'yhat har ikkinchi elementi o'chirilsin.
10. Ro'yhat har uchinchi elementi o'chirilsin.
11. Ro'yhat elementlari kamayish tartibida joylashtirilsin.
12. Ro'yhat tozalansin.

13. Futbol jamosining 20 ta o’yinchilari familiyalaridan tashkil topgan halqasimon ro’yhat berilgan. O’yinchilar 2 ta guruhga 10 tadan ajratilsin. Ikkinchi guruhga umumiy o’yinchilarni har 12-inchisi kirsin.

14. Sportchi familiyalaridan tashkil topgan ikkita halqasimon ro’yhat berilgan. Qura tashlash amalga oshirilsin. Birinchi guruhdagi har n-inchi sportchi, ikkinchi guruhdagi har m-inchi sportchi bilan raqib bo’lsin.

15. Lotoreya ishtirokchilari familiyalari va mukofotlar nomlaridan tashkil topgan 2 ta halqasimon ro’yhat berilgan. N ta ishtirokchi g’olib bo’lsin (har K-inchi). Mukofotlarni qayta hisoblash soni - t.

16. O’quvchilar familiyalari va imtihon biletlari raqamlaridan tashkil topgan 2 ta halqasimon ro’yhat berilgan. O’quvchilar tomonidan olingan bilet raqamlari aniqlansin. Imtihon biletlari uchun qayta hisoblash soni - E, o’quvchilar uchun esa - K.

17. Mahsulot nomlaridan tashkil topgan ro’yhat berilgan. Ro’yhat elementlaridagi SONY firmasida ishlab chiqilgan mahsulotlardan tashkil topgan yangi ro’yhat yarating.

18. 2 ta guruh talabalari familiyalaridan tashkil topgan 2 ta ro’yhat berilgan. Birinchi guruhdan L ta talaba ikkinchi guruhga o’tkazilsin. Qayta hisoblashlar soni - K.

19. BOSCH va PHILIPS konsernlari tomonidan ishlab chiqilgan mahsulot nomlaridan tashkil topgan ikkita ro’yhat berilgan. Har ikkala firma tomonidan ishlab chiqilgan bir xil mahsulotlar ro’yhati tuzilsin.

20. Futbol jamoasining asosiy va zahira tarkibi o’yinchilari familiyalaridan tashkil topgan ikkita ro’yhat berilgan. K ta o’yinchi almashtirilsin.

21. 1- va 2-vzvod askarlari familiyalaridan tashkil topgan ikkita ro’yhat berilgan. Hujum natijasida 1-chi vzvoddan M ta askar halok bo’ldi. Ikkinchi vzvod askarlaridan birinchi vzvod to’ldirilsin.

22. Mahsulot nomlari va xaridorlar familiyalaridan tashkil topgan ikkita ro’yhat berilgan. Har bir N-chi xaridor M-chi mahsulotni sotib oladi. Xarid

qilingan mahsulotlar ro’yhatini chiqaring.

23. SONY va SHARP firmalari tomonidan ishlab chiqilgan mahsulot nomlaridan tashkil topgan ikkita ro’yhat berilgan. O’zaro raqobat qiluvchi mahsulotlar ro’yhatini tuzing.

24. Talabalar ismlaridan iborat ro’yhat berilgan. Ismining uzunligi eng katta bo’lgan talabani ro’yhat boshiga joylang.

25. Talabalar familiyalaridan iborat halqasimon ro’yhat berilgan. Har k-inchi talabadan 3 tasi ro’yhatdan ajratib olinsin.

26. Talabalar ismlaridan iborat massiv elementlarini berilgan halqasimon ro’yhatning har k-elementidan keyin joylashtiring.

27. 2 ta halqasimon ro’yhatni galma-galdan har 3-elementidan umumiy bitta yangi ro’yhat hosil qiling.

28. 2 ta ro’yhatning bir xil qiymatli elementlaridan yangi halqasimon ro’yhat yarating.

29. 2 ta ro’yhatning bir xil qiymatli elementlarini ro’yhat boshiga o’tkazing.

30. 2 ta ro’yhatning bir xil qiymatli elementlarini ro’yhat oxiriga joylashtiring.

4-tajriba ishi. DARAXTSIMON TUZILMALAR

Ishdan maqsad: Talabalar daraxtsimon tuzilmalar, binar daraxtlarni e’lon qilish, uning ustida amallar bajarish algoritmlarini tadqiq qilishlari va o’rganishlari kerak, bu algoritmlarning dasturiy realizatsiyasini amalga oshirish ko’nikmasiga ega bo’lishlari kerak.

Qo’yilgan masala: Har bir talaba topshiriq varianti olib, undagi masalaning qo’yilishiga mos binar daraxtlarni tadqiq qilishga oid dasturni ishlab chiqishlari kerak.

Ish tartibi:

- Tajriba ishi nazariy ma’lumotlarini o’rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- C++ dasturlash muhitida dasturni yaratish;
- Natijalarni tekshirish;
- Hisobotni tayyorlash va topshirish.

4.1. Daraxt ko’rinishidagi ma’lumotlar tuzilmasi haqida umumiy tushunchalar.

Uzellar (elementlar) va ularning munosabatlaridan iborat elementlar to’plamining ierarxik tuzilmasiga daraxtsimon ma’lumotlar tuzilmasi deyiladi.

Daraxt – bu shunday chiziqsiz bog’langan ma’lumotlar tuzilmasiki, u quyidagi belgilari bilan tavsiflanadi:

- daraxtda shunday bitta element borki, unga boshqa elementlardan murojaat yo’q. Bu element daraxt ildizi deyiladi;

- daraxtda ixtiyoriy element chekli sondagi ko’rsatkichlar yordamida boshqa tugunlarga murojaat qilishi mumkin;

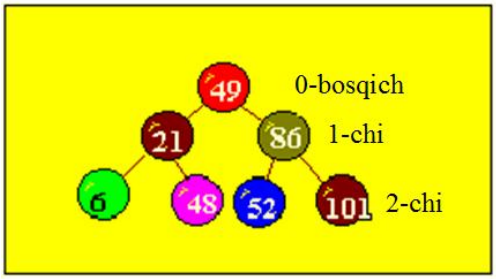
- daraxtning har bir elementi faqatgina o’zidan oldingi kelgan bitta element bilan bog’langan.

4.2. Binar daraxtlarni qurish

Binar daraxtda har bir tugun-elementdan ko’pi bilan 2 ta shox chiqadi. Daraxtlarni xotirada tasvirlashda uning ildizini ko’rsatuvchi ko’rsatkich berilishi kerak. Daraxtlarni kompyuter xotirasida tasvirlanishiga ko’ra har bir element (binar daraxt tuguni) to’rtta maydonga ega yozuv shaklida bo’ladi, ya’ni kalit maydon, informatsion maydon, ushbu elementni o’ngida va chapida joylashgan elementlarning xotiradagi adreslari saqlanadigan maydonlar.

Shuni esda tutish lozimki, daraxt hosil qilinayotganda, otaga nisbatan chap tomondagi o’g’il qiymati kichik kalitga, o’ng tomondagi o’g’il esa katta qiymatli kalitga ega bo’ladi. Har safar daraxtga yangi element kelib qo’shilayotganda u avvalambor daraxt ildizi bilan solishtiriladi. Agar element ildiz kalit qiymatidan kichik bo’lsa, uning chap shoxiga, aks holda o’ng shoxiga o’tiladi. Agar o’tib ketilgan shoxda tugun mavjud bo’lsa, ushbu tugun bilan ham solishtirish amalga oshiriladi, aks holda, ya’ni u shoxda tugun mavjud bo’lmasa, bu element shu tugunga joylashtiriladi.

Masalan, daraxt tugunlari quyidagi qiymatlarga ega 6, 21, 48, 49, 52, 86, 101. U holda binar daraxt ko’rinishi quyidagi 4.1-rasmdagidek bo’ladi:



4.1-rasm. Binar daraxt ko’rinishi

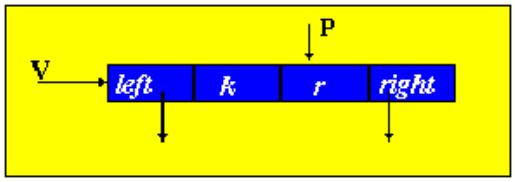
Natijada, o’ng va chap qism daraxtlari bir xil bosqichli tartiblangan binar daraxt hosil qildik. Agar daraxtning o’ng va chap qism daraxtlari bosqichlarining farqi birdan kichik bo’lsa, bunday daraxt ideal muvozanatlangan daraxt deyiladi. Yuqorida hosil qilgan binar daraxtimiz ideal muvozanatlangan daraxtga misol

bo’ladi. Daraxtni muvozanatlash algoritmini sal keyinroq ko’rib chiqamiz. Undan oldin binar daraxtni yaratish algoritmini o’rganamiz.

4.3. Algoritm

Binar daraxt yaratish funksiyasi

Binar daraxtni hosil qilish uchun kompyuter xotirasida elementlar quyidagi 4.2-rasmdagidek toifada bo’lishi lozim.



4.2-rasm. Binar daraxt elementining tuzilishi

p – yangi element ko’rsatkichi

next, last – ishchi ko’rsatkichlar, ya’ni joriy elementdan keyingi va oldingi elementlar ko’rsatkichlari

r=rec – element haqidagi birorta ma’lumot yoziladigan maydon

k=key – elementning unikal kalit maydoni

left=NULL – joriy elementning chap tomonida joylashgan element adresi

right=NULL – joriy elementning o’ng tomonida joylashgan element adresi.

Dastlab yangi element hosil qilinayotganda bu ikkala maydonning qiymati 0 ga teng bo’ladi.

tree – daraxt ildizi ko’rsatkichi

n – daraxtdagi elementlar soni

Boshida birinchi kalit qiymat va yozuv maydoni ma’lumotlari kiritiladi, element hosil qilinadi va u daraxt ildiziga joylashadi, ya’ni tree ga o’zlashtiriladi. Har bir hosil qilingan yangi elementning left va right maydonlari qiymati 0 ga tenglashtiriladi. Chunki bu element daraxtga terminal tugun sifatida joylashtiriladi, hali uning farzand tugunlari mavjud emas. Qolgan elementlar ham shu kabi hosil

qilinib, kerakli joyga joylashtiriladi. Ya’ni kalit qiymati ildiz kalit qiymatidan kichik bo’lgan elementlar chap shoxga, katta elementlar o’ng tomonga joylashtiriladi. Bunda agar yangi element birorta elementning u yoki bu tomoniga joylashishi kerak bo’lsa, mos ravishda left yoki right maydonlarga yangi element adresi yozib qo’yiladi.

Binar daraxtni hosil qilishda har bir element yuqorida ko’rsatilgan toifada bo’lishi kerak. Lekin hozir biz o’zlashtirish osonroq va tushunarli bo’lishi uchun key va rec maydonlarni bitta qilib info maydon deb ishlatamiz.

4.3-rasm. Binar daraxt elementining tuzilishi

Ushbu toifada element hosil qilish uchun oldin bu toifani yaratib olishimiz kerak. Uni turli usullar bilan amalga oshirish mumkin. Masalan, ***node*** nomli yangi toifa yaratamiz:

```
class node{
    public:
        int info;
        node *left;
        node *right;
    };
```

Endi yuqoridagi belgilashlarda keltirilgan ko’rsatkichlarni shu toifada yaratib olamiz.

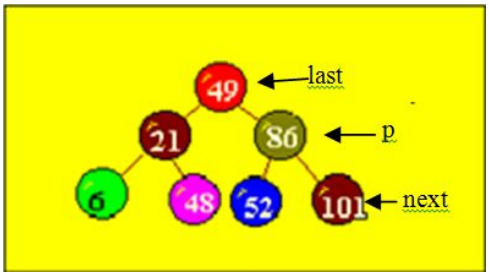
```
node *tree=NULL;
node *next=NULL;
int n,key; cout<<"n=";cin>>n;
```

Nechta element (n) kiritilishini aniqlab oldik va endi har bir element qiymatini kiritib, binar daraxt tuzishni boshlaymiz.

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

```
node *p=new node;
    node *last=new node;
    cin>>key;
     p->info=key;
     p->left=NULL;
    p->right=NULL;
        if(i==0){ tree=p; next=tree;sontinue;}
        next=tree;
              while(1){    last=next;
           if(p->info<next->info) next=next->left; else next=next->right;
           if(next==NULL) break;
           }
if(p->info<last->info) last->left=p; else last->right=p;
}
```

Bu yerda *p* hali aytganimizdek, kiritilgan kalitga mos hosil qilingan yangi element ko'rsatkichi, *next* yangi element joylashishi kerak bo'lgan joyga olib boradigan shox adresi ko'rsatkichi, ya'ni u har doim *p* dan bitta qadam oldinda yuradi, *last* esa ko'rilayotgan element kimning avlodi ekanligini bildiradi, ya'ni u har doim *p* dan bir qadam orqada yuradi (4.4-rasm).

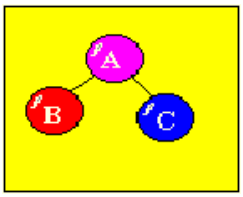


4.4-rasm. Binar daraxt elementlarini belgilash

Shunday qilib binar daraxtini ham yaratib oldik. Endigi masala uni ekranda tasvirlash kerak, ya'ni u ko'rikdan o'tkaziladi yoki vizuallashtirsa ham bo'ladi.

4.4. Daraxt “koʼrigi” funksiyalari

4.5-rasmdagidek binar daraxt berilgan boʼlsin:



4.5-rasm. 3 ta elemetdan iborat binar daraxt

Binar daraxtlari koʼrigini uchta tamoyili mavjud. Ularni berilgan daraxt misolida koʼrib chiqaylik:

1) Yuqoridan pastga koʼrik (daraxt ildizini qism daraxtlarga nisbatan oldinroq koʼrikdan oʼtkaziladi): A, B, C ;

2) Chapdan oʼngga: B, A, C ;

3) Quyidan yuqoriga (ildiz qism daraxtlardan keyin koʼriladi): B, C, A .

Daraxt koʼrigi koʼpincha ikkinchi usul bilan, yaʼni tugunlarga kirish ularning kalit qiymatlarini oʼsish tartibida amalga oshiriladi.

4.5. Daraxt koʼrigining rekursiv funksiyalari

1. *int pretrave(node *tree){*

if(tree!=NULL) {int a=0,b=0;

if(tree->left!=NULL) a=tree->left->info;

if(tree->right!=NULL) b=tree->right->info;

cout<<tree->info<<" - chapida "<<a<<" - oʼngida "<<b<<" \n";

pretrave(tree->left);

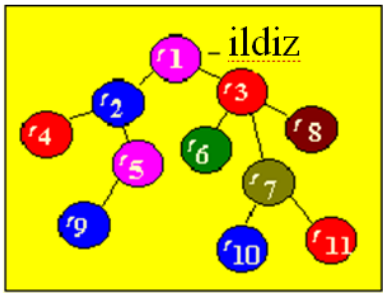
pretrave(tree->right);

}

return 0;

```
};
2. int intrave(node *tree){
    if(tree!=NULL) {
            intrave(tree->left);
            cout<<tree->info;
            intrave(tree->right);
            }
    return 0;
    };
3. int postrave(node *tree){
    if(tree!=NULL) {
            postrave(tree->left);
            postrave(tree->right);
            cout<<tree->info;
            }
    return 0;
    };
```

Daraxtning har bir tuguni 4.6-rasmdagidek oraliq (2, 3, 5, 7 elementlar) yoki terminal (daraxt “barg”i) (4, 9, 10, 11, 8, 6 elementlar) bo’lishi mumkin.



4.6-rasm. Daraxtsimon tuzilma

1. Agar tugunning otasi yo’q bo’lsa, bu tugun ildiz hisoblanadi. Buni aniqlash uchun dastur kodini keltiramiz. Dasturda p izlanayotgan tugun.

if(p==tree) cout<<"bu tugun ildiz ekan";

else cout<<"bu tugun ildiz emas";

2. Biz izlayotgan element daraxtda oraliq tugun ekanligini tekshirish uchun uning yoki o'ng shoxi, yoki chap shoxi, yoki ikkalasiyam mavjudligini tekshirish kerak. Agar ikkala shoxi NULL dan farqli bo'lsa, bu 2 ta farzandga ega oraliq tugun hisoblanadi, yoki ikkalasidan bittasi NULL ga teng bo'lsa, bu tugun 1 ta farzandga ega oraliq tugun hisoblanadi. Berilgan p element daraxtning oraliq tugun ekanligini aniqlash dastur kodini keltiramiz.

if(p!=tree){

if((p->left!=NULL)&&(p->right!=NULL)) cout<<"bu tugun 2 ta farzandga ega oraliq tugun";

else if((p->left!=NULL)||(p->right!=NULL) cout<<"bu 1 ta farzandga ega oraliq tugun";

} else cout<<"bu tugun oraliq tugun emas";

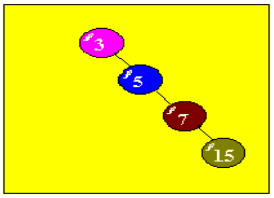
3. Biz izlayotgan tugun terminal tugunligini tekshirishni ko'rib chiqamiz. Agar tugunning har ikkala shoxi NULL ga teng bo'lsa, bu ***terminal tugun*** hisoblanadi. Dastur kodini keltiramiz.

if((p->left==NULL)&&(p->right==NULL)) cout<<"bu tugun terminal tugun";

else cout<<"bu terminal tugun emas";

4.6. Binar daraxt bo'yicha qidiruv funksiyasi

Mazkur funksiyaning vazifasi shundan iboratki, u berilgan kalit bo'yicha daraxt tuguni qidiruvini amalga oshiradi. Qidiruv operatsiyasining davomiyligi daraxt tuzilishiga bog'liq bo'ladi. Haqiqatdan, agar elementlar daraxtga kalit qiymatlari o'sish (kamayish) tartibida kelib tushgan bo'lsa, u holda daraxt 4.7-rasmdagidek bir tomonga yo'nalgan ro'yhat hosil qiladi (chiqish darajasi bir bo'ladi, ya'ni yagona shoxga ega), masalan:



4.7-rasm. Bir tomonlama yo’naltirilgan binar daraxt tuzilishi

Bu holda daraxtda qidiruv vaqti, bir tomonlama yo’naltirilgan ro’yhatdagi kabi bo’lib, o’rtacha qarab chiqishlar soni N/2 bo’ladi. Agar daraxt muvozanatlangan bo’lsa, u holda qidiruv eng samarali natija beradi. Bu holda qidiruv $\log_2 N$ dan ko’p bo’lmagan elementlarni ko’rib chiqadi.

Qidiruv funksiyasini ko’rib chiqamiz. *search* fuksiyasi daraxtdan *key* kalitga mos elementning adresini aniqlaydi.

```
int search(node *tree, int key){
node *next; next=tree;
while(next!=NULL)
  {     if (next->info==key){cout<<"Binar daraxtda "<<key<<" mavjud";
 return next; }
       if (next->info>key) next=next->left;
          else  next=next->right;
   }
cout<<"tuzilmada izlangan element yo’q!!!"<<endl;
return 0;
}
```

4.7. Daraxtga yangi element qo‘shish funksiyasi

Daraxtga biror bir elementni qo’shishdan oldin daraxtda berilgan kalit bo’yicha qidiruvni amalga oshirish lozim bo’ladi. Agar berilgan kalitga teng kalit mavjud bo’lsa, u holda dastur o’z ishini yakunlaydi, aks holda daraxtga element qo’shish amalga oshiriladi.

Daraxtga yangi yozuvni kiritish uchun, avvalo daraxtning shunday tugunini topish lozimki, natijada mazkur tugunga yangi element qo’shish mumkin bo’lsin. Kerakli tugunni qidirish algoritmi ham xuddi berilgan kalit bo’yicha tugunni topish algoritmi kabi bo’ladi.

Daraxtda qo’shilayotgan element kalitiga teng kalitli element yo’q bo’lgan holda elementni tuzilmaga qo’shish funksiyasini keltirib o’tamiz.

```
Node *q=NULL;
Node *p=tree;
while(p!=NULL){
    q=p;
    if(key==p->key){
       search=p;
       return 0;
       }
    If(key<p->key)  p=p->left;
      else p=p->right;
  }
```

Berilgan kalitga teng tugun topilmadi, element qo’shish talab qilinadi. Ota bo’lishi mumkin tugunga q ko’rsatkich beriladi, elementning o’zi esa *yangi* nomli ko’rsatkichi bilan beriladi.

```
node *q=new node;
```

Qo’yilayotgan yangi element chap yoki o’ng o’g’il bo’lishini aniqlash lozim.

```
 If(key<q->key)   q->left=yangi;
    else q->right=yangi;
search=yangi;
return 0;
```

4.8. Binar daraxtdan elementni oʼchirish funksiyasi

Tugunni o’chirib tashlash natijasida daraxtning tartiblanganligi buzilmasligi lozim.

Tugun daraxtda o’chirilayotganda 3 xil variant bo’lishi mumkin:

1) Topilgan tugun terminal (barg). Bu holatda tugun otasining qaysi tomonida turgan bo’lsa, otasining o’sha tomonidagi shoxi o’chiriladi va tugunning xotirada joylashgan sohasi tozalanadi.

2) Topilgan tugun faqatgina bitta o’g’ilga ega. U holda o’g’il ota o’rniga joylashtiriladi.

3) O’chirilayotgan tugun ikkita o’g’ilga ega. Bunday holatda shunday qism daraxtlar zvenosini topish lozimki, uni o’chirilayotgan tugun o’rniga qo’yish mumkin bo’lsin. Bunday zveno har doim mavjud bo’ladi:

- bu yoki chap qism daraxtning eng o’ng tomondagi elementi (ushbu zvenoga erishish uchun keyingi uchiga chap shox orqali o’tib, navbatdagi uchlariga esa, murojaat ***NULL*** bo’lmaguncha, faqatgina o’ng shoxlari orqali o’tish zarur);

- yoki o’ng qism daraxtning eng chap elementi (ushbu zvenoga erishish uchun keyingi uchiga o’ng shox orqali o’tib, navbatdagi uchlariga esa, murojaat ***NULL*** bo’lmaguncha, faqatgina chap shoxlari orqali o’tish zarur).

O’chirlayotgan element chap qism daraxtining eng o’ngidagi element o’chirilayotgan element uchun merosxo’r bo’ladi (12 uchun – 11 bo’ladi). Merosxo’r esa o’ng qism daraxtning eng chapidagi tuguni (12 uchun - 13).

Merosxo’rni topish algoritmini ishlab chiqaylik (4.8-rasmga qarang).

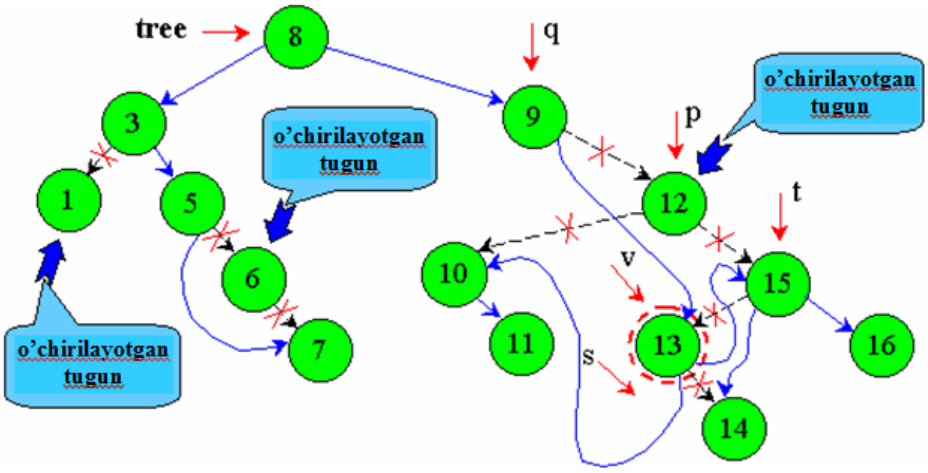
p – ishchi ko’rsatkich;

q - p dan bir qadam orqadagi ko’rsatkich;

v – o’chirilayotgan tugun merosxo’rini ko’rsatadi;

t – v dan bir qadam orqada yuradi;

s - v dan bir qadam oldinda yuradi (chap o’g’ilni yoki bo’sh joyni ko’rsatib boradi).



4.8-rasm. Binar daraxtdan oraliq tugunni o'chirich tartibi

Yuqoridagi daraxt bo'yicha qaraydigan bo'lsak, oxir oqibatda, v ko'rsatkich 13 tugunni, s esa bo'sh joyni ko'rsatishi lozim.

1) Elementni qidirish funksiyasi orqali o'chirilayotgan elementni topamiz. p ko'rsatkich o'chirilayotgan elementni ko'rsatadi.

2) O'chiriladigan elementning o'rniga qo'yiluvchi tugunga v ko'rsatkich qo'yamiz.

```
node *del(node *tree,int key){
node *p=new node;
node *next=tree;
node *q=NULL;
while(next!=NULL)
{ if (next->info==key){cout<<"Binar daraxtda "<<key<<" Mavjud"<<endl; p=next;break; }
  if (next->info>key){ q=next; next=next->left; }
    else {q=next;next=next->right;}
}
  if(next==NULL) cout<<"tuzilmada izlangan element yo'q!!!"<<endl;
  node *v=NULL,*t=NULL,*s=NULL;
  if(p->left==NULL)v=p->right;
  else
```

```
if(p->right==NULL)   v=p->left;
   if((p->left!=NULL)&&(p->right!=NULL)){t=p; v=p->right; s=v->left;}
   while(s!=NULL){
         t=v;
         v=s;
         s=v->left;
         }
   if((t!=NULL)&&(t!=p)){
     t->left=v->right;
     v->right=p->right;
     v->left=p->left;
     }
    if(t==p)    v->left=p->left;
    if(q==NULL){
     cout<<v->info<<" ildiz\n";
     tree=v;
     delete(p);
     return tree;
     }
   if(p==q->left)
       q->left=v;
   else q->right=v;
   delete(p); // o'chirilgan element joylashgan xotira yacheykasini tozalash
  return tree;
}
```

4.9. Daraxtni muvozanatlash algoritmi

Binar daraxt muvozanatlangan yoki AVL-muvozanatlangan bo'lishi mumkin. Daraxt AVL-muvozanatlangan (1962 yil sovet olimlari Adelson, Velsk

Georgiy Maksimovich va Landis Yevgeniya Mihaylovichlar tomonidan taklif qilingan) deyiladi, agar daraxtdagi har bir tugunning chap va o'ng qismdaraxtlari balandliklari farqi 1 tadan ko'p bo'lmasa.

Berilgan butun sonlar – kalitlar ketma-ketligidan binar daraxt yaratib olamiz va uni muvozanatlaymiz. Daraxtni muvozanatlashdan maqsad, bunday daraxtga yangi element kiritish va daraxtdan element izlash algoritmi samaradorligini oshirishdan iborat, ya'ni bu amallarni bajarishdagi solishtirishlar soni kamayadi. Binar daraxtni muvozanatlash algoritmi quyidagicha bo'ladi.

Algoritm

1. Binar daraxtni yaratib olamiz.

2. Binar daraxtni chapdan o'ngga ko'rikdan o'tkazamiz va tugunlarning info maydonlaridan a[..] massiv hosil qilamiz. Tabiiyki, massiv o'sish bo'yicha tartiblangan bo'ladi.

Muvozanatlangan daraxtning tugunlarini belgilash uchun massivni ko'riladigan oralig'ini belgilab olamiz, ya'ni start=0 va end=n-1.

3. Massivning ko'rilayotgan oralig'i o'rtasida joylashgan elementni, ya'ni mid=(start+end)/2 va a[mid] ni muvozanatlangan daraxtning tuguni qilib olinadi. Agar ko'rilayotgan oraliqda bitta ham element qolmagan bo'lsa, ya'ni start>end bo'lsa, bajarilish joriy seansdan keyingisiga uzatiladi.

4. Ko'rilayotgan tugunning chap qismdaraxtini hosil qilish uchun massivning ko'rilayotgan oralig'ining 1-yarmini olamiz, ya'ni start=0 va end=mid-1. 3-5 qadamlarni takrorlaymiz.

5. Ko'rilayotgan tugunning o'ng qismdaraxtini hosil qilish uchun massivning ko'rilayotgan oralig'ining 2-yarmini olamiz, ya'ni start=mid+1 va end=end (oldingi qadamdagi end). 3-5 qadamlarni takrorlaymiz.

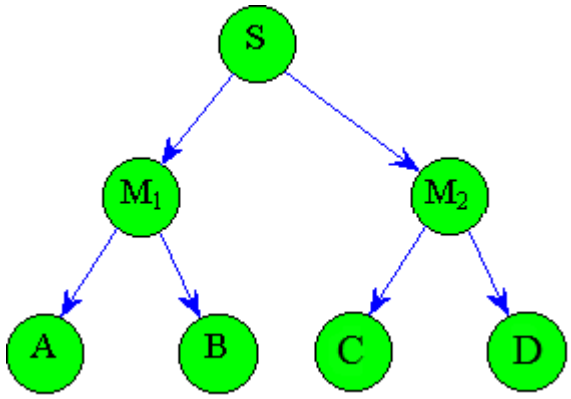
Datur kodi

```
node *new_tree(int *arr, int start, int end)
{
   if(start>end) return NULL;
   else {
```

```
        int mid=(start+end)/2;
        node *tree=new node;
        tree->info=arr[mid];
        tree->left=new_tree(arr,start,mid-1);
        tree->right=new_tree(arr,mid+1,end);
        return tree;
    }
}
```

4.10. Binar daraxt balandligi

Binar daraxtning balandligi deb daraxt bosqichlari soniga aytiladi. Binar daraxt balandligini aniqlash uchun uning har bir tuguni chap va o'ng qismdaraxtlari balandliklari solishtiriladi va maksimal qiymat balandlik deb olinadi. Misol uchun quyidagi 4.9-rasmdagi daraxtning balandligi 2 ga teng.



4.9-rasm. Binar daraxt balandligi

Daraxt balandligini aniqlash dastur kodini keltiramiz.

```
int height(node *tree){
  int h1,h2;
  if  (tree==NULL) return (-1);
  else  {
   h1 = height(tree->left);
```

```
  h2 = height(tree->right);
  if  (h1>h2) return (1 + h1);
  else  return (1 + h2);
 }
}
```

4.11. Binar daraxtni muvozanatlanganmi yoki yo'qligini tekshirish

Daraxtning balandligini aniqlashni o'rganganimizdan keyin uning muvozanatlanganligini tekshirish mumkin. Binar daraxtning muvozanatlanganligini tekshirish uchun uning har bir tugunini har ikkala qismdaraxti balandliklarini hisoblab, farqlarini tekshirib ko'rish kerak. Agar farq ***0*** yoki ***1*** ga teng bo'lsa, bu muvozanatlangan daraxt hisoblanadi. Quyida binar daraxtni muvozanatlanganlikka tekshirishning rekursiv funksiyasini qo'llovchi dastur keltirilgan.

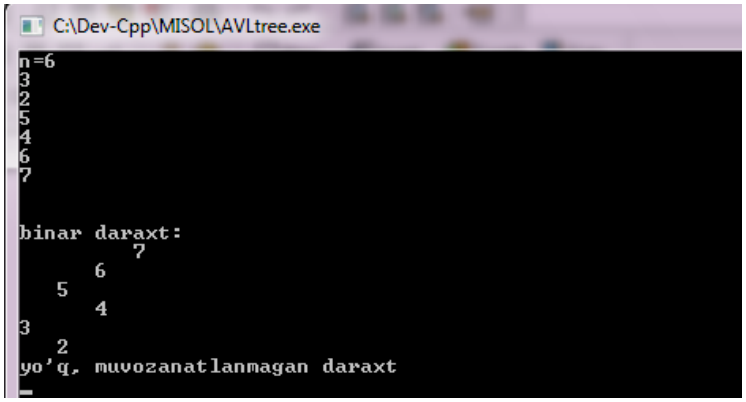
Dastur kodi

```
#include <conio.h>
#include <iostream>
using namespace std;
class node{
     public: int info;
     node *left;
     node *right;
     };
    int k=0,Flag=1;
int height(node *tree){
 int h1,h2;
 if  (tree==NULL) return (-1);
 else  {
  h1 = height(tree->left);
  h2 = height(tree->right);
  if  (h1>h2) return (1 + h1);
```

```
   else  return (1 + h2);
  }
}
void vizual(node *tree,int l)
{  int i;
  if(tree!=NULL)  {
   vizual(tree->right,l+1);
   for  (i=1; i<=l; i++) cout<<"   ";
   cout<<tree->info<<endl;
   vizual(tree->left,l+1);
  }
}
int AVLtree (node *tree){
  int t;
  if (tree!=NULL){
   t = height (tree->left) - height (tree->right);
   if ((t<-1) || (t>1)) { Flag = 0; return Flag; }
   AVLtree (tree->left); AVLtree (tree->right);
  }
 }
 int GetFlag(){return Flag;}
int main()
{ int n,key,s; node *tree=NULL,*next=NULL;
cout<<"n="; cin>>n; int arr[n];
for(int i=0; i<n; i++){
        node *p=new node;
        node *last=new node;
        cin>>s;
        p->info=s;
        p->left=NULL;
```

```
        p->right=NULL;
        if(i==0){tree=p; next=tree; continue; }
        next=tree;
    while(1)
        { last=next;
        if(p->info<next->info)next=next->left;
        else next=next->right;
        if(next==NULL)break; }
    if(p->info<last->info)last->left=p;
      else last->right=p;}
        cout<<endl;
        cout<<"\nbinar daraxt:\n";
        vizual(tree,0);
        AVLtree(tree);
     if(GetFlag()) cout<<"ha, muvozanatlangan daraxt"; else cout<<"yo'q, muvozanatlanmagan daraxt";cout<<endl;
        getch();
    }
```

Dastur natijasi



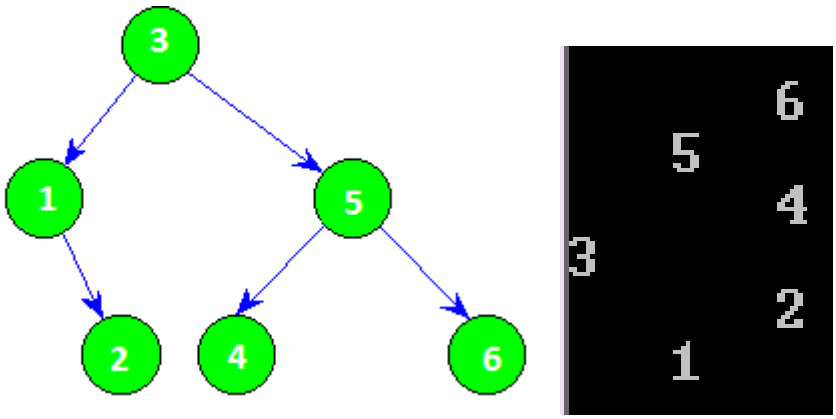
4.12. Binar daraxtni vizuallashtirish

Binar daraxtni ko'rikdan o'tkazayotganda biz yuqorida har bir

tugunni o’ngida va chapida turgan tugunlarni so’z bilan ifodaladik. Lekin bu usul bir muncha noqulay. Daraxtni vizual ko’rinishda ifodalash uni anglashning juda qulay usuli hisoblanadi. Daraxtni vizuallashtirishning grafik ko’rinishi va konsol oynasida ifodalash kabi turlari mavjud. Shundan konsol oynasida daraxtni vizuallashtirishni ko’rib chiqamiz. Bunda sonlar daraxt shaklida joylashtiriladi. Quyida bunday usulning dastur kodi keltirilgan.

```
void vizual(node *tree,int l)
{  int i;
  if(tree!=NULL)  {
   vizual(tree->right,l+1);
   for  (i=1; i<=l; i++) cout<<"   ";
   cout<<tree->info<<endl;
   vizual(tree->left,l+1);
  }
}
```

Dastur kodi quyidagi 4.10 a-rasmdagi daraxtni konsol ekranida 4.10 b-rasm ko’rinishda ifodalaydi.



a. b.

4.10-rasm. a - binar daraxt; b - binar daraxtning ekranda namoyon bo’lishi

Yuqorida keltirilgan bir nechta algoritmlarni qo’llab bitta misol ko’rib chiqamiz.

Misol: berilgan binar daraxtning balandligini aniqlang va muvozanatlang.

Dastur kodi

```
#include <conio.h>
#include <iostream>
using namespace std;
class node{
      public: int info;
      node *left;
      node *right;
      };
     int k=0;
int intrave(node *tree){
    if (tree!=NULL){int a=NULL, b=NULL;
    if (tree->left!=NULL){ a=tree->left->info; }
    if (tree->right!=NULL){ b=tree->right->info;  }
    cout<<tree->info<<"--chapida=>"<<a<<"  o'ngida=>"<<b<<"\n";
    intrave(tree->left);
    intrave(tree->right);   }
    return 0;
    }
int height(node *tree){
  int h1,h2;
  if  (tree==NULL) return (-1);
  else  {
   h1 = height(tree->left);
   h2 = height(tree->right);
   if  (h1>h2) return (1 + h1);
   else  return (1 + h2);
  }
}
```

```
int create_arr(node *tree,int *arr){
   if(!tree) return 0;
   else{
        create_arr(tree->left,arr);
        arr[k++]=tree->info;
        create_arr(tree->right,arr);
        }
   }
node *new_tree(int *arr, int start, int end)
{
   if(start>end) return NULL;
   else  {
      int mid=(start+end)/2;
      node *tree=new node;
      tree->info=arr[mid];
      tree->left=new_tree(arr,start,mid-1);
      tree->right=new_tree(arr,mid+1,end);
      return tree;
   }
}
void vizual(node *tree,int l)
{  int i;
  if(tree!=NULL)  {
   vizual(tree->right,l+1);
   for  (i=1; i<=l; i++) cout<<"   ";
   cout<<tree->info<<endl;
   vizual(tree->left,l+1);
  }
}
int main()
```

```
{ int n,key,s; node *tree=NULL,*next=NULL;
cout<<"n="; cin>>n; int arr[n];
for(int i=0; i<n; i++){
        node *p=new node;
        node *last=new node;
        cin>>s;
        p->info=s;
        p->left=NULL;
        p->right=NULL;
        if(i==0){tree=p; next=tree; continue; }
        next=tree;
while(1)
        { last=next;
        if(p->info<next->info)next=next->left;
        else next=next->right;
        if(next==NULL)break; }
if(p->info<last->info)last->left=p;
   else last->right=p;}
       cout<<endl;
        intrave(tree);
        cout<<"\nya'ni\n";
        vizual(tree,0);
        int h=height(tree);
        cout<<"balandligi="<<h<<endl;
        create_arr(tree,arr);
       for(int i=0;i<k;i++)cout<<arr[i]<<"   ";cout<<endl<<endl;
       tree=new_tree(arr,0,k-1);
       vizual(tree,0);
       getch();
}
```

Dastur natijasi

Ishni bajarishga namuna

Topshiriq variantlariga o'xshash bitta misolning algoritmi va to'liq dasturini ko'rib chiqaylik.

Misol: berilgan binar daraxtdan ko'rsatilgan *key* kalitga mos tugunni o'chirish dasturini tuzing.

Algoritm

Asosiy dastur tanasi - *int main()*

1. ***i=0; n*** – daraxtga kiritiladigan elementlar sonini aniqlash. Daraxt ildizi ko'rsatkichi ***tree=NULL***. ***Next*** yangi elementni joylashtiradigan shoxga o'tishda ishlatiladi va **last next** dan 1 qadam orqada yuradi.

2. Agar ***i<n*** bo'lsa, daraxtga kiritiladigan navbatdagi elementga qiymat kiritish va uni yangi ***p*** element ***info*** maydoniga yozish, ***left*** va ***right*** maydonlarga ***NULL*** yozish. Aks holda 8-qadamga o'tish.

3. Agar ***tree=NULL*** bo'lsa, ***p*** ni daraxt ildizi qilish, ya'ni ***tree=p*** va

next=last=p.

4. Agar ***p->info next->info*** dan kichik bo'lsa, chap shoxga o'tish kerak, ya'ni ***last=next*** va ***next=next->left***, aks holda o'ng shoxga o'tamiz, ya'ni ***last=next*** va ***next=next->right***.

5. Agar ***next=NULL*** bo'lsa, 6-qadamga o'tish, aks holda 4-qadamga o'tish.

6. Agar ***p->info<last->info*** bo'lsa, ***last->left=p***, aks holda ***last->right=p***.

7. ***i***++, 2-qadamga o'tish.

8. ***intrave(tree)*** funksiyasini ishlatish.

9. ***Key*** kalitga mos elementni daraxtdan o'chiradigan ***del(tree,key)*** funksiyasini ishlatish.

10. Natijaviy daraxtni ko'rikdan o'tkazish uchun ***intrave(tree)*** funksiyasini ishlatish va algoritmni yakunlash.

intrave(tree) **funksiyasining ishlash algoritmi**

1. Agar funksiyaning kirishiga berilgan tugun NULL bo'lmasa, 2-qadamga o'tish, aks holda funksiya chaqirilgan joyga qaytib borish.

2. Agar tugunning chap shoxi tuguni NULL bo'lmasa, uning info maydonini yangi butun toifali a ga o'zlashtirish, aks holda a=0.

3. Agar tugunning o'ng shoxi tuguni NULL bo'lmasa, uning info maydonini yangi butun toifali b ga o'zlashtirish, aks holda b=0.

4. Ekranga tugunning info maydoni qiymatini, tugunning chapidagi a va o'ngidagi b ni chiqaramiz.

5. Endi shu intrave() funksiyasining kirishiga joriy tugunning chap shoxi tugunini berib chaqiramiz, ya'ni yuqoridagi 4 ta amalni joriy tugunning chap shoxidagi tugun ustida bajaramiz.

6. Endi shu intrave() funksiyasining kirishiga joriy tugunning o'ng shoxi tugunini berib chaqiramiz, ya'ni yuqoridagi 4 ta amalni joriy tugunning o'ng shoxidagi tugun ustida bajaramiz.

del() funksiyasining ishlash algoritmi

Funksiyaning kirishiga daraxt ildizi ko'rsatkichi ***tree*** va o'chirilishi kerak

bo’lgan tugunning ***info*** maydoni qiymati ***key*** beriladi. Daraxtning ***key*** kalitli tugunini terminal tugungacha izlaymiz. Dastlab ***next=tree***.

1. Toki ***next NULL*** bo’lguncha, agar ***next*** tugunning ***info*** maydoni ***key*** ga teng bo’lsa, izlayotgan tugunni topdik va uning adresini ***p*** ga joylaymiz va 4-qadamga o’tamiz. Agar ***next NULL*** bo’lsa, 3-qadamga o’tamiz.

2. Agar ***key next*** ning ***info***sidan kichik bo’lsa, joriy tugunning chap tomonidagi tugunga o’tamiz, ya’ni ***next=next->left***, aks holda o’ng shoxdagi tugunga o’tamiz. 1-qadamga qaytamiz.

3. Agar ***next NULL*** ga teng bo’lsa, biz izlagan tugun tuzilmada yo’q. Tugunni o’chirish algoritmi tugaydi va dastur bajarilishi o’chirish funksiyasi chaqirilgan joyga qaytib boradi.

4. Agar ***p*** o’chirilayotgan tugunning chap tomonida tugun yo’q bo’lsa (ya’ni p->left=NULL bo’lsa), uning o’ng tomonidagi tugun adresini ***v*** ga o’zlashtiramiz.

5. Agar ***p*** o’chirilayotgan tugunning o’ng tomonida tugun yo’q bo’lsa, uning chap tomonidagi tugun adresini ***v*** ga o’zlashtiramiz.

6. Agar ***p*** o’chirilayotgan tugunning chapi va o’ngida element mavjud bo’lsa, bu tugunning o’rniga da’vo qiladigan tugunni topish uchun shu tugundan 1 marta o’ngga va oxirigacha chap shox tuguniga o’tamiz. Ya’ni ***v=p->right***, ***v p*** ning o’ng tomonida turibdi, ***t=p*** va ***s=v->left***, ya’ni ***s v*** ning chapida turibdi. Endi to ***s NULL*** bo’lguncha chapga ketamiz, undan 1 ta orqada ***v*** va ***v*** dan 1 ta orqada ***t*** keladi. Mana endi biz ***p*** ning o’rniga ***v*** olib borib qo’yishimiz mumkin.

7. Agar ***t NULL*** bo’lmasa va ***t p*** ga teng bo’lmasa (agar ***p*** ning bitta farzandi mavjud bo’lsa, uning o’rniga keladigan tugunni izlashga xojat yo’q, chunki uning o’sha farzandi aynan ***p*** ning o’rniga joylashadi. Agar o’chirilayotgan ***p*** tugunning 2 ta farzandi mavjud bo’lsa, shu shart bajariladi), u holda, ***p*** ning o’rniga ketayotgan ***v*** tugunning farzandi (agar u mavjud bo’lsa) ***v*** ning otasi bo’lmish ***t*** ga meros qoldiriladi, ya’ni ***v->right v*** ning o’rniga keladi. ***t->left=v->right***. Endigi ish ***p*** ning har ikkala tomonidagi tugunlarni ***v*** ga o’zlashtiramiz.

8. Agar ***t p*** ga teng bo’lsa (ya’ni p o’chayotgan tugunning o’rniga o’zining

farzandi kelayotgan bo’lsa), ***p*** ning chapidagi tugunni ***v*** ning chapiga o’zlashtiramiz.

9. Mana ***p*** tugunning o’rniga ***v*** tugun keldi. Endigi vazifa ***v*** ni ***p*** ning otasi bilan ulash kerak. Buning uchun aniqlash kerak – ***p*** tugunning otasi ***q NULL*** ga teng emasmi? Agar ***q NULL*** bo’lsa, biz daraxt ildizini o’chirgan bo’lamiz. Bu holda daraxt ildizi ko’rsatkichi ***tree*** ni ***v*** ga tenglab qo’yamiz. Aks holda, 10-qadamga o’tamiz.

10. ***p*** tugun otasi ***q*** tugunning qaysi tomonida turgan edi? Agar ***p q*** ning chapida turgan bo’lsa, ***p*** ning o’rniga, ya’ni ***q->left*** ga ***v*** ni joylaymiz, aks holda ***q->right*** ga ***v*** ni joylaymiz.

11. ***p*** tugun joylashgan xotira yacheykasini tozalab qo’yamiz va algoritm yakunlanadi.

Dastur kodi

```
#include <conio.h>
#include <iostream>
using namespace std;
class node{
   public: int info;
   node *left;
   node *right;
   };
int intrave(node *tree){
  if (tree!=NULL){int a=0, b=0;
  if (tree->left!=NULL){ a=tree->left->info; }
  if (tree->right!=NULL){ b=tree->right->info;  }
  cout<<tree->info<<"--chapida=>"<<a<<"  o’ngida=>"<<b<<"\n";
  intrave(tree->left);
  intrave(tree->right);   }
  return 0;
  }
```

```
node *del(node *tree,int key){
node *p=new node;
node *next=tree;
node *q=NULL;
while(next!=NULL)
{ if (next->info==key){cout<<"Binar daraxtda "<<key<<" Mavjud"<<endl;
p=next;break; }
if (next->info>key){ q=next; next=next->left; }
else {q=next;next=next->right;}
}
if(next==NULL) cout<<"tuzilmada izlangan element yo'q!!!"<<endl;
node *v=NULL,*t=NULL,*s=NULL;
if(p->left==NULL) v=p->right;
else if(p->right==NULL) v=p->left;
if((p->left!=NULL)&&(p->right!=NULL)){t=p; v=p->right; s=v->left;}
while(s!=NULL){
t=v;
v=s;
s=v->left;
}
if((t!=NULL)&&(t!=p)){
t->left=v->right;
v->right=p->right;
v->left=p->left;
}
if(t==p) v->left=p->left;
if(q==NULL){
cout<<v->info<<" ildiz\n";
tree=v;
```

```
        delete(p);
        return tree;
        }
    if(p==q->left)
        q->left=v;
    else q->right=v;
    delete(p); // o'chirilgan element joylashgan xotira yacheykasini tozalash
   return tree;
}
int main()
{ int n,key,s; node *tree=NULL,*next=NULL;
cout<<"n="; cin>>n;
for(int i=0; i<n; i++){
        node *p=new node;
        node *last=new node;
        cin>>s;
        p->info=s;
        p->left=NULL;
        p->right=NULL;
        if(i==0){tree=p; next=tree; continue; }
        next=tree;
while(1)
        { last=next;
        if(p->info<next->info)next=next->left;
        else next=next->right;
        if(next==NULL)break; }
if(p->info<last->info)last->left=p;
   else last->right=p;}
       cout<<endl;
        intrave(tree);
```

```
        cout<<"delete qilinadigan elementni kiriting \n";
        cout<<"key="; cin>>key;
        tree=del(tree,key);
        intrave(tree);
       getch();
}
```

Dasturning ishlashi natijasi

n=10

8 3 9 12 10 15 13 11 16 14

8—chapida=>3 o'ngida=>9

3—chapida=>0 o'ngida=>0

9—chapida=>0 o'ngida=>12

12—chapida=>10 o'ngida=>15

10—chapida=>0 o'ngida=>11

11—chapida=>0 o'ngida=>0

15—chapida=>13 o'ngida=>16

13—chapida=>0 o'ngida=>14

14—chapida=>0 o'ngida=>0

16—chapida=>0 o'ngida=>0

delete qilinadigan elementni kiriting

key=12

Binar daraxtda 12 Mavjud

8—chapida=>3 o'ngida=>9

3—chapida=>0 o'ngida=>0

9—chapida=>0 o'ngida=>13

13—chapida=>10 o'ngida=>15

10—chapida=>0 o'ngida=>11

11—chapida=>0 o'ngida=>0

15—chapida=>14 o'ngida=>16

14—chapida=>0 o'ngida=>0

16—chapida=>0 o’ngida=>0

Nazorat savollari

1. Daraxtsimon ma’lumotlar tuzilmasi nima?
2. Binar daraxt tuzilmasi nima va uni tuzishga misol keltiring?
3. Binar daraxti tuzilmasi ustida qanday amallar bajatirilishi mumkin?
4. Binar daraxtini ko’rikdan o’tkazish algoritmi qanday?
5. Binar daraxtiga yangi element qo’shish algoritmini tushuntiring.
6. Binar daraxti elementini o’chirish algoritmini tushuntiring.

Topshiriq

Variantlar:

1. Talabalar ismlari ketma-ketligidan binar daraxt hosil qilish algoritmi va dasturini tuzing.
2. Berilgan binar daraxtning terminal tugunlaridan tashkil topgan yangi muvozanatlangan binar daraxt hosil qilish algoritmi va dasturini tuzing.
3. Berilgan binar daraxtning har bir tuguni chap tomoni tugunlaridan tashkil topgan muvozanatlangan binar daraxt hosil qilish algoritmi va dasturini tuzing.
4. Daraxt tugunlari haqiqiy sonlar bo’lsin. Daraxt barcha tugunlarini o’rta arifmetigini hisoblash algoritmi va dasturini keltiring.
5. Daraxt tugunlari haqiqiy sonlar bo’lsin. Yozuvi manfiy bo’lgan daraxt tugunlarini o’chiruvchi dastur tuzing.
6. Daraxt tugunlari haqiqiy sonlar bo’lsin. Yozuvi berilgan kalit qiymatidan katta bo’lgan daraxt tugunlarini o’chiruvchi dastur tuzing.
7. Berilgan binar daraxtning balandligini aniqlash algoritmi va dasturini keltiring.
8. Berilgan binar daraxtning har bir juft elementi balandligini aniqlash algoritmi va dasturini keltiring.

9. Berilgan binar daraxtning terminal tugunlari balandliklarini aniqlash algoritmi va dasturini keltiring.

10. Daraxt tugunlari haqiqiy sonlar bo’lsin. Daraxt barcha tugunlarini o’rta arifmetigiga teng qiymatli tugunni berilgan binar daraxtga kiritish algoritmi va dasturini keltiring.

11. Berilgan binar daraxtning oraliq tugunlaridan tashkil topgan yangi binar daraxt tuzish algoritmi va dasturini keltiring.

12. Berilgan binar daraxtdan kalitlari o’sish tartibida joylashgan bir bog’lamli ro’yhat hosil qilish algoritmi va dasturini keltiring.

13. Binar daraxtning barcha barglari yozuvini chop etuvchi dastur ishlab chiqing.

14. Binar daraxtning barcha oraliq tugunlari yozuvini chop etuvchi dastur ishlab chiqing.

15. Binar daraxtning juft qiymatli kalitga ega elementlaridan yangi daraxt qurish algoritmi va dasturini keltiring.

16. Binar daraxtning tugunlari sonini aniqlashning algoritmi va dasturini keltiring.

17. Binar daraxtda berilgan tugungacha bo’lgan masofani aniqlashning algoritmi va dasturini keltiring.

18. Bo’sh bo’lmagan binar daraxtning eng katta va eng kichik kalitli tugunlarini aniqlashning algoritmi va dasturini keltiring.

19. T1 va T2 binar daraxtlar tengligini tekshiruvchi dastur tuzing. (Daraxtlar teng deyiladi, agar ikkala daraxt mos uchlarining yozuv va kalitlari o’zaro teng bo’lsa).

20. Binar daraxtni o’ngdan chapga va chapdan o’ngga ko’rik o’tkazish dasturi va algoritmini keltiring.

21. Daraxt tugunlari haqiqiy sonlar bo’lsin. Yozuvi (a,b) oraliqqa tegishli bo’lmagan daraxt tugunlarini o’chiruvchi dastur tuzing.

22. Daraxt tugunlari haqiqiy sonlar bo’lsin. Yozuvi (a,b) oraliqqa tegishli bo’lgan daraxt tugunlarini o’chiruvchi dastur tuzing.

23. Berilgan binar daraxtdan kalit qiymatlari kamayish tartibida joylashgan bir bog’lamli ro’yhat hosil qilish algoritmi va dasturini keltiring.

24. Bo’sh bo’lmagan binar daraxtning eng katta va eng kichik kalitli tugunlarini o’rta arifmetigiga teng kalitli tugunni berilgan daraxtga qo’yish algoritmi va dasturini keltiring.

25. Berilgan binar daraxtda kalit qiymati ildizning kalit qiymatiga eng yaqin bo’lgan tugun kaliti va yozuvini chop etish algoritmi va dasturini keltiring.

26. Berilgan binar daraxtda kalit qiymati ildizning kalit qiymatiga eng uzoq bo’lgan tugun kaliti va yozuvini chop etish algoritmi va dasturini keltiring.

27. Butun sonlardan iborat binar daraxtning toq qiymatli tugunlaridan yangi muvozanatlangan daraxt hosil qiling.

28. Berilgan binar daraxt muvozanatlanganmi yoki yo’qligini tekshiring.

29. Berilgan muvozanatlangan binar daraxtdan qaysi tugunlar o’chirilsa, uning muvozanatlanganligi buzilmasligini ko’rsatish dasturini tuzing.

30. Berilgan ro’yhat binar daraxt bo’la oladimi, yo’qmi, shuni aniqlash dasturini keltiring.

5-tajriba ishi. QIDIRUV USULLARINI TADQIQ QILISH

Ishdan maqsad: talabalar berilgan tuzilmaning shakliga qarab biror kalitga mos elementni qidirishning optimal usulini qo'llashni o'rganishlari va qidiruv usullarining samaradorligini taqqoslashlari kerak.

Qo'yilgan masala: topshiriq variantidagi masalani so'ralayotgan qidiruv usuli yordamida yechishning C++ tilidagi dasturini yaratish ko'nikmasiga ega bo'lish.

Ish tartibi:

- Laboratoriya ishi nazariy ma'lumotlarini o'rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- C++ dasturlash muhitida dasturni yaratish;
- Natijalarni tekshirish;
- Hisobotni tayyorlash va topshirish.

5.1. Ma'lumotlarni tuzilmadan qidirish

Kompyuterda ma'lumotlarni qayta ishlashda qidiruv asosiy amallardan biri hisoblanadi. Uning vazifasi berilgan argument bo'yicha massiv ma'lumotlari ichidan mazkur argumentga mos ma'lumotlarni topish yoki bunday ma'lumot yo'qligini aniqlashdan iborat.

Ixtiyoriy ma'lumotlar majmuasi **jadval** yoki **fayl** deb ataladi. Ixtiyoriy ma'lumot (yoki tuzilma elementi) boshqa ma'lumotdan biror bir belgisi orqali farq qiladi. Mazkur belgi **kalit** deb ataladi. Kalit noyob bo'lishi, ya'ni mazkur kalitga ega ma'lumot jadvalda yagona bo'lishi mumkin. Bunday noyob kalitga **boshlang'ich (birinchi) kalit** deyiladi. **Ikkinchi kalit** bir jadvalda takrorlansada u orqali ham qidiruvni amalga oshirish mumkin. Ma'lumotlar kalitini bir joyga yig'ish (boshqa jadvalga) yoki yozuv sifatida ifodalab bitta maydonga kalitlarni yozish mumkin. Agar kalitlar ma'lumotlar jadvalidan ajratib olinib alohida fayl sifatida saqlansa, u holda bunday kalitlar **tashqi kalitlar** deyiladi. Aks

holda, ya’ni yozuvning bir maydoni sifatida jadvalda saqlansa **ichki kalit** deyiladi.

Kalitni berilgan argument bilan mosligini aniqlovchi algoritmga berilgan argument bo’yicha **qidiruv** deb ataladi. Qidiruv algoritmi vazifasi kerakli ma’lumotni jadvaldan topish yoki yo’qligini aniqlashdan iboratdir. Agar kerakli ma’lumot yo’q bo’lsa, u holda ikkita ishni amalga oshirish mumkin:

1. Ma’lumot yo’qligini indikatsiya qilish (belgilash)
2. Jadvalga ma’lumotni qo’yish.

Faraz qilaylik, **k** – kalitlar massivi. Har bir **k(i)** uchun **r(i)** – ma’lumot mavjud. **Key** – qidiruv argumenti. Unga **rec** - informatsion yozuv mos qo’yiladi. Jadvaldagi ma’lumotlarning tuzilmasiga qarab qidiruvning bir necha turlari mavjud.

5.2. Ketma-ket qidiruv algoritmi

Mazkur ko’rinishdagi qidiruv agar ma’lumotlar tartibsiz yoki ular tuzilishi noaniq bo’lganda qo’llaniladi. Bunda ma’lumotlar butun jadval bo’yicha operativ xotirada kichik adresdan boshlab, to katta adresgacha ketma-ket qarab chiqiladi.

Massivda ketma-ket qidiruv (search o’zgaruvchi topilgan element tartib raqamini saqlaydi).

Ketma-ket qidiruv algoritmi C++ tilida quyidagicha bo’ladi:

```
int qidiruv(int key){
for (int i=0;i<n;i++)
  if (k[i]==key) { search = i;return search;}
  search = -1;
  return search;
}}
```

Massivda ketma-ket qidiruv algoritmi samaradorligini bajarilgan taqqoslashlar soni **M** bilan aniqlash mumkin. $\mathbf{M_{min} = 1,\ M_{max} = n}$. Agar ma’lumotlar massiv yacheykasida bir xil ehtimollik bilan taqsimlangan bo’lsa, u holda $\mathbf{M_{o'rt} \approx (n + 1)/2}$ bo’ladi.

Agar kerakli element jadvalda yo’q bo’lib, uni jadvalga qo’shish lozim bo’lsa, u holda yuqorida keltirilgan algoritmdagi oxirgi ikkita operator quyidagicha almashtiriladi.

n=n+1;

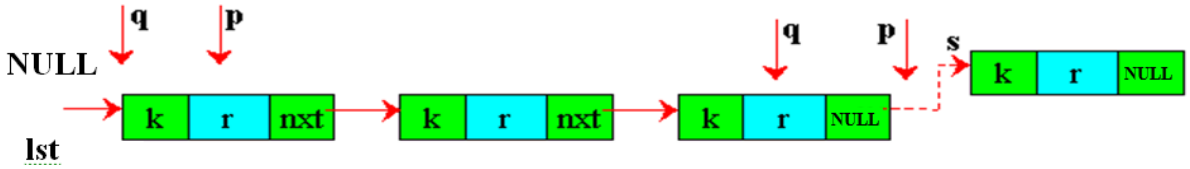
k[n-1]:=key;

r[n-1]:=rec;

search:=n-1;

return search;

Agar ma’lumotlar jadvali bir bog’lamli ro’yhat ko’rinishida berilgan bo’lsa (5.1-rasm), u holda ketma-ket qidiruv ro’yhatda amalga oshiriladi.



5.1-rasm. Bir bog’lamli ro’yhatning ko’rinishi

Chiziqli bir bog’lamli ro’yhatdan key kalitga mos elementni ketma-ket qidiruv usuli yordamida izlab topish dasturi.

```
Node *q=NULL;
Node *p=lst;
while (p !=NULL){
    if (p->k == key){
       search = p;
       return search;
       }
  q = p;
  p = p->nxt;
}
Node *s=new Node;;
s->k=key;
```

```
s->r=rec;
s->nxt= NULL;
if (q == NULL){ s->nxt=lst; lst = s; }
        else q->nxt = s;
search= s;
return search;
```

Ro’yhatli tuzilmaning afzalligi shundan iboratki, ro’yhatga elementni qo’shish yoki o’chirish tez amalga oshadi, bunda qo’shish yoki o’chirish element soniga bog’liq bo’lmaydi, massivda esa elementni qo’shish yoki o’chirish o’rta hisobda barcha elementlarning yarmini siljitishni talab qiladi. Ro’yhatda qidiruvning samaradorligi taxminan massivniki bilan bir xil bo’ladi.

5.3. Teng bo‘lish orqali qidiruv (ikkilik qidiruv) algoritmi

Faraz qilaylik, o’sish tartibida tartiblangan sonlar massivi berilgan bo’lsin. Ushbu usulning asosiy g’oyasi shundan iboratki, tasodifiy qandaydir $\mathbf{A_M}$ element olinadi va u **X** qidiruv argumenti bilan taqqoslanadi. Agar $\mathbf{A_M=X}$ bo’lsa, u holda qidiruv yakunlanadi; agar $\mathbf{A_M < X}$ bo’lsa, u holda indekslari **M** dan kichik yoki teng bo’lgan barcha elementlar kelgusi qidiruvdan chiqarib yuboriladi. Xuddi shuningdek, agar $\mathbf{A_M > X}$ bo’lsa, u holda indekslari **M** dan katta bo’lgan barcha elementlar kelgusi qidiruvdan chiqarib yuboriladi.

M ixtiyoriy tanlanganda ham taklif qilinayotgan algoritm korrekt ishlaydi. Shu sababali **M** ni shunday tanlash lozimki, tadqiq qilinayotgan algoritm samaraliroq natija bersin, ya’ni uni shunday tanlaylikki, iloji boricha kelgusi jarayonlarda ishtirok etuvchi elementlar soni kam bo’lsin. Agar biz o’rtacha elementni, ya’ni massiv o’rtasini tanlasak yechim mukammal bo’ladi. Misol uchun butun sonlardan iborat, o’sish bo’yicha tartiblangan massivdan ikkilik qidiruv usuli yordamida key kalitga mos elementni izlash dasturini ko’rib chiqamiz.

Dastur kodi

```
#include<iostream>
using namespace std;
int main(){
   int n;cout<<"n=";cin>>n;
   int k[n];
   for(int i=0;i<n;i++) cin>>k[i];
   int key, search;
   cout<<"qidirilayotgan elementni kiriting=";cin>>key;
int low = 0;
int hi = n-1; int j=0;
while (low <= hi){
   int mid = (low + hi) / 2;j++;
   if (key == k[mid]){
      search = mid;
      cout<<"qidirilayotgan element "<<search+1<<" o'rinda turibdi va u "<<j<<" ta solishtirishda toplidi\n";
      system("pause");
      exit(0);
   }
   if (key < k[mid])
         hi = mid - 1;
    else low = mid + 1;
   }
   search=-1;
   cout<<j<<" ta solishtirish amalga oshirildi va qidirilayotgan element topilmadi\n";
system("pause");
}
```

Dastur natijasi

n=6

1 2 3 4 5 6

qidirilayotgan elementni kiriting=6

qidirilayotgan element 6 o'rinda turibdi va u 3 ta solishtirishda toplidi

5.4. Qidiruv jadvalini qayta tartibga keltirish

Umuman olganda, jadvalda har bir elementni qidirish ehtimolligini qandaydir bir qiymat bilan izohlash mumkin. Faraz qilaylik jadvalda qidirilayotgan element mavjud. U holda qidiruv amalga oshirilayotgan jadvalni diskret holatga ega tizim sifatida qarash mumkin hamda unda qidirilayotgan elementni topish ehtimolligi – bu tizim i-chi holati ehtimolligi ***p(i)*** deb olish mumkin.

$$\sum_{i=1}^{n} p(i) = 1$$

Jadvalni diskret tizim sifatida qaraganimizda, undagi taqqoslashlar soni diskret tasodifiy miqdorlar qiymatlarini matematik kutilmasini ifodalaydi.

*Z=Q=1*p(1)+2*p(2)+3*p(3)+…+n*p(n)*

Ma’lumotlar jadvalda quyidagi ko‘rinishda tartiblangan bo‘lishi lozim:

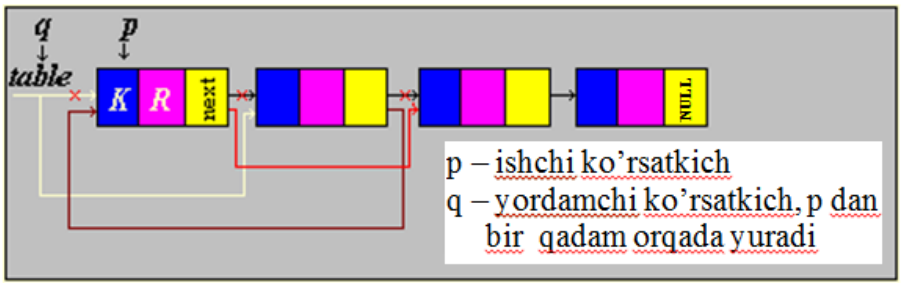
p(1)≥p(2) ≥p(3) ≥…≥p(n).

Bu shart taqqoslashlar sonini kamaytirib, samaradorlikni oshiradi. Sababi, ketma-ket qidiruv birinchi elementdan boshlanganligi uchun eng ko‘p murojaat qilinadigan elementni birinchiga qo‘yish lozim.

Qidiruv jadvalini qayta tartibga keltirishning eng ko‘p ishlatiladigan ikkita usuli mavjud. Ularni bir bog‘lamli ro‘yhatlar misolida ko‘rib chiqamiz.

1. Topilgan elementni ro‘yhat boshiga qo‘yish orqali qayta tartibga keltirish.
2. Transpozitsiya usuli.

5.5. Topilgan elementni ro‘yhat boshiga qo‘yish orqali qayta tartibga keltirish



5.2-rasm. Ro‘yhatni qayta tartibga keltirish

Topilgan element 5.2-rasmdagidek birdaniga ro‘yhat boshiga joylashtiriladi. Tuzilmadan har safar birorta element izlab topilsa va u ro‘yhat boshiga olib borib qo‘yilaversa, natijada oxirgi izlangan elementlar ro‘yhat boshiga joylashib qoladi va biz oxirgi vaqtlarda izlangan elementlarni tez izlab topish imkoniga ega bo‘lamiz.

Boshida **q** ko‘rsatkich bo‘sh, **p** esa ro‘yhat boshini ko‘rsatadi; **p** ikkinchi elementni ko‘rsatganda, **q** birinchini ko‘rsatadi. Ro‘yhat boshi ko‘rsatkichi (**table**) birinchi elementni ko‘rsatadi. Ro‘yhatda **key** kalitli element topilsa, u **p** ko‘rsatkich bilan, undan oldingi element esa **q** ko‘rsatkich bilan belgilanadi. Shu topilgan **p** elementni ro‘yhat boshiga joylashtiriladi.

Dastur kodi

```
node *q=NULL;
node *p=table;
while (p !=NULL){
    if (key == p->k){
      if (q == NULL) { //o‘rinlashtirish shart emas
         search = p;
       exit(0);
       }
     q->nxt = p->nxt;
```

```
        p->nxt = table;
        table = p;
        exit(0);
      }
    q = p;
    p = p->nxt;
}
search = NULL;
  exit(0);
```

5.6. Transpozitsiya usuli

Ushbu usulda topilgan element ro‘yhatda bitta oldingi element bilan o‘rin almashtiriladi. Agarda mazkur elementga ko‘p murojaat qilinsa, bittadan oldinga surilib borib natijada ro‘yhat boshiga kelib qoladi. Ushbu usulning afzalligi shundaki, tuzilmada ko‘p murojaat qilinadigan elementlar ro‘yhat boshiga bitta qadam bilan intiladi.

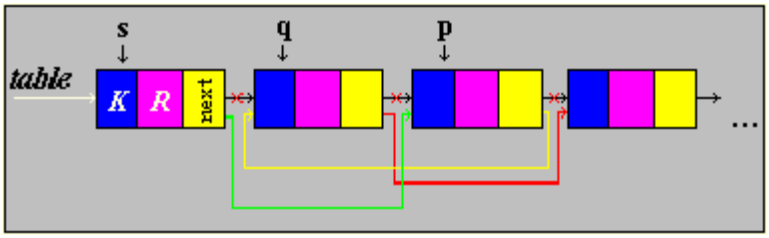
Ushbu usulning qulayligi u nafaqat ro‘yhatda, balki tartiblanmagan massivda ham samarali ishlaydi (sababi faqatgina ikkita yonma-yon turgan element o‘rin almashtiriladi).

Bu usulda uchta ko‘rsatkichdan foydalanamiz (5.3-rasm):

p – ishchi ko‘rsatkich

q – yordamchi ko‘rsatkich, **p** dan bitta qadam orqada bo‘ladi

s – yordamchi ko‘rsatkich, **p** dan ikkita qadam orqada bo‘ladi



5.3-rasm. Transpozitsiya usuli bilan ro‘yhatni qayta tartibga keltirish

Biz tomonimizdan topilgan uchinchi element ro‘yhat boshiga bir qadam suriladi (ya’ni ikkinchi bo‘lib qoladi). Birinchi element ko‘rsatkichi uchinchi elementga joylashtiriladi, ikkinchi element ko‘rsatkichi to‘rtinchi, shunday qilib uchinchi element ikkinchi joyga joylashib qoladi. Agar mazkur elementga yana bir bor murojaat qilinsa, u holda u ro‘yhat boshida bo‘lib qoladi.

```
node *s=NULL;
node *q=NULL;
node *p=table;
while (p != NULL){
   if (key == p->k){ //transponerlaymiz
        if( q ==NULL){//o‘rinlashtirish shart emas
          search=p;
          exit(0);
         }
        q->nxt=p->nxt;
        p->nxt=q;
        if (s == NULL) table = p;
        else s->nxt = p;
   search=p;
   exit(0);
   }
   s=q;
   q=p;
   p=p->nxt;
}
   search=NULL;
   exit(0);
```

Ishni bajarishga oid namuna

Talabalar ma’lumotlaridan – FIO va adresdan iborat jadval berilgan. Binar qidiruvdan foydalanib TTJ da yashaydigan talabalar ro‘yhatini hosil qiling.

Algoritm

1. Jadvalga **n** ta talaba **FIO** va **adres**larini kiritamiz.
2. Binar qidiruvni jadvalning birorta maydonida amalga oshirish uchun jadvalni shu maydoni bo‘yicha tartiblab olish kerak. Shuning uchun masalaning qo‘yilishida adresi TTJ bo‘lgan talabalarni topish kerakligi sababli jadval ma’lumotlarini adres maydoni bo‘yicha saralab olamiz. Masalani yechishda to‘g‘ridan-to‘g‘ri tanlash orqali saralashdan foydalanilgan.
3. **key** kalitga mos elementni izlash chegaralarini aniqlab olamiz. Dastlab u **[0,n]** oralig‘ida, ya’ni **low=0,hi=n**.
4. Agar **low<=hi** bo‘lsa, oraliq o‘rtasini hisoblaymiz. **mid=(low+hi)/2**
5. Agar **mid** o‘rnida turgan talaba adresi TTJ bo‘lsa, element topildi, **search=mid** va 7-qadamga o‘tiladi, aks holda keyingi qadamga o‘tiladi.
6. Agar “TTJ” so‘zi alifbo bo‘yicha **mid** o‘rnida turgan talaba adresi qiymatidan kichik bo‘lsa, izlash quyi chegarasi o‘zgaradi, ya’ni **mid** o‘rnida turgan elementdan bitta oldingi elementgacha olinadi, ya’ni **hi=mid-1.** Aks holda, yuqori chegara o‘zgaradi – **mid** dan keyingi elementdan to oxirgi elementlar oralig‘i olinadi, ya’ni **low=mid+1**. 4-qadamga o‘tiladi.
7. Agar topilgan elementdan oldin turgan elementning (**mid-1**) ham adres maydoni TTJ bo‘lsa, **search--,** ya’ni bitta oldingi elementga o‘tamiz va shu qadamni boshidan bajaramiz. Aks holda keyingi qadamga o‘tiladi.
8. Joriy (**search** ko‘rsatayotgan) elementdan boshlab adresi “TTJ” ga teng bo‘lgan talaba ma’lumotlarini ekranga chiqaramiz. Agar adresi “TTJ” dan farq qiladigan talaba chiqib qolsa, algoritm tugallanadi.

Dastur kodi

```
#include <iostream>
using namespace std;
int main(){
int n;cout<<"n=";cin>>n;
```

```
struct Guruh{
    string fio,adres;
    }talaba[n];
    for(int i=0;i<n;i++){
        cout<<i+1<<"-talabaning fio=";cin>>talaba[i].fio;
        cout<<"adres=";cin>>talaba[i].adres;
        }
    //jadval binar qidiruv olib boriladigan maydoni bo‘yicha tartiblangan
//bo‘lishi kerak
    for(int i=0;i<n-1;i++)
    for(int j=i+1;j<n;j++)
        if(talaba[i].adres>talaba[j].adres){
        Guruh h=talaba[i];
        talaba[i]=talaba[j];
        talaba[j]=h;
        }
     for(int i=0;i<n;i++)
          cout<<talaba[i].fio<<" "<<talaba[i].adres<<endl;
          cout<<endl;
int low = 0,hi = n-1,search=-1,q=0;
string key="TTJ";
while(low<=hi){
   int mid = (low + hi) / 2;
   q++;
   if (key == talaba[mid].adres){
      search = mid;
      break;
   }
   if (key < talaba[mid].adres)
        hi = mid - 1;
```

```
        else low = mid + 1;
    }
      if(search!=-1)  cout<<"qidirilayotgan  el  "<<search+1<<"  –  o‘rinda
turibdi va "<<q<<" ta solishtirishda topildi"<<endl;
      else {cout<<q<<" ta solishtirish amalga oshirildi va topilmadi"<<endl;
          system("PAUSE");
          return EXIT_SUCCESS;
          }
          while(talaba[search-1].adres==key) search--;
          while(talaba[search].adres==key) {
                cout<<talaba[search].fio<<"
"<<talaba[search].adres<<endl;
                search++; }
          system("pause");
    }
```

Dastur natijasi:

```
n=5
1-talabaning fio=fam1
adres=Toshkent
2-talabaning fio=fam2
adres=TTJ
3-talabaning fio=fam3
adres=ijarada
4-talabaning fio=fam4
adres=uchastkada
5-talabaning fio=fam5
adres=TTJ
fam2 TTJ
fam5 TTJ
fam1 Toshkent
```

fam3 ijarada

fam4 uchastkada

qidirilayotgan el 1-orinda turubdi va 2 ta solishtirishda topildi

fam2 TTJ

fam5 TTJ

Nazorat savollari

1. Qanday qidiruv algoritmlarini bilasiz?
2. Qidiruv jarayonining tezligi nimalarga bog'liq?
3. Statik tuzilmadan birorta elementni izlashning qanday usullari mavjud?
4. Ro'yhat tuzilmasidan elementlarni izlab topish tezligini oshirish uchun qanday algoritmlar mavjud?
5. Binar qidiruvni ro'yhat tuzilmasiga qo'llab bo'ladimi? Sababini asoslang.

Topshiriq

Variantlar:

1. Ketma-ket qidiruv usulidan foydalanib, ro'yhat eng kichik elementini toping.

2. Ketma-ket qidiruv usulidan foydalanib, ro'yhatda berilgan kalitdan katta elementlarni toping.

3. Ketma-ket qidiruv usulidan foydalanib, ro'yhat eng kichik elementini toping.

4. Ketma-ket va binar qidiruv usulidan foydalanib, A massivdan elementni va taqqoslashlar sonini toping.

5. Binar qidiruvdan foydalanib elementlarni tasodifiy ravishda toping.

6. Mashina raqamlari ro'yhati berilgan: 345, 368, 876, 945, 564, 387, 230. Binar qidiruvdan foydalanib berilgan raqamli mashina qaysi joyda turganini toping.

7. Ketma-ket qidiruv usulidan foydalanib ro'yhatda har ikkinchi elementni qidiring va taqqoslashlar sonini aniqlang.

8. Binar qidiruvdan foydalanib massivdan berilgan kalitga karrali kalitli elementni va solishtirishlar sonini toping.

9. Boshiga qo‘yish va transpozitsiya usulidan foydalanib massiv eng katta elementi topilsin.

10. Boshiga qo‘yish usulidan foydalanib ro‘yhatda 11 ga butun bo‘linuvchi eng katta sonni toping (agar bunday sonlar ko‘p bo‘lsa, u holda ularning eng kattasini toping; agar bunday son mavjud bo‘lmasa – shunga mos ma’lumot chiqaring).

11. Transpozitsiya usulidan foydalanib ro‘yhatda 11 ga butun bo‘linuvchi eng katta sonni toping (agar bunday sonlar ko‘p bo‘lsa, u holda ularning eng kichigini toping; agar bunday son mavjud bo‘lmasa – shunga mos ma’lumot chiqaring).

12. Boshiga qo‘yish usulidan foydalanib ro‘yhatda qo‘shni elementlari ayrimasi 72 dan kichik bo‘lgan elementni toping. Agar bunday elementlar ko‘p bo‘lsa, u holda ularning eng kattasini toping; agar bunday element mavjud bo‘lmasa – shunga mos ma’lumot chiqaring.

13. Transpozitsiya usulidan foydalanib ro‘yhatda qo‘shni elementlari bo‘linmasi juft son bo‘lgan elementni toping. Agar bunday elementlar ko‘p bo‘lsa, u holda ularning eng kattasi yoki eng kichigini toping; agar bunday element mavjud bo‘lmasa – shunga mos ma’lumot chiqaring.

14. Boshiga qo‘yish usulidan foydalanib ro‘yhatda qo‘shni elementlar ayrimasi juft bo‘lgan elementni toping. Agar bunday elementlar ko‘p bo‘lsa, u holda ularning eng kattasi yoki eng kichigini toping; agar bunday element mavjud bo‘lmasa – shunga mos ma’lumot chiqaring.

15. Transpozitsiya usulidan foydalanib ro‘yhatda kerakli elementgacha bo‘lgan elementlarning o‘rta arifmetigi 12 ga teng bo‘lgan element topilsin. Agar bunday element mavjud bo‘lmasa – shunga mos ma’lumot chiqaring.

16. Boshiga qo‘yish usulidan foydalanib ro‘yhatda 10 ga bo‘linuvchi maksimal elementni toping. Agar bunday element mavjud bo‘lmasa – shunga mos ma’lumot chiqaring.

17. Boshiga qo‘yish va transpozitsiya usulidan foydalanib massiv eng kichik elementi topilsin.

18. Transpozitsiya usulidan foydalanib ro‘yhatda qo‘shni elementlari ayirmasi juft va 3 ga bo‘linadigan elementni toping. Agar bunday element mavjud bo‘lmasa – shunga mos ma’lumot chiqaring.

19. Boshiga qo‘yish usulidan foydalanib ro‘yhatda kerakli elementdan keyingi elementlarning o‘rtacha kvadratik qiymati 10 dan kichik bo‘lgan elementni toping. Agar bunday elementlar ko‘p bo‘lsa, u holda ularning eng kattasini toping; agar bunday element mavjud bo‘lmasa – shunga mos ma’lumot chiqaring.

20. Transpozitsiya usulidan foydalanib har bir x element uchun tg(x) qiymatini aniqlang va eng katta qiymatga ega bo‘lgan elementni 1-o‘ringa qo‘ying.

21. Berilgan ro‘yhatda qidirilayotgan element transpozitsiya usuli bilan qancha murojaatda ro‘yhat boshiga kelishini aniqlash dasturini tuzing.

22. Massivdan boshiga qo‘yish usuli yordamida key kalitli elementni izlash dasturini tuzing.

23. Binar qidiruv usuli yordamida massivga yangi elementni kiriting.

24. Binar qidiruv usuli yordamida massivning key kalitli elementini o‘chiring.

25. Ro‘yhatda transpozitsiya usuli yordamida toq elementlarni topish dasturini tuzing.

26. Berilgan massivda key kalitli elementni ketma-ket va binar qidiruv usullari yordamida izlang va qaysi usul ushbu qidiruv holatida samara berganligini aniqlash dasturini keltiring.

27. Talabalar ismi va umumiy ballaridan iborat jadvaldan ketma-ket qidiruv usuli bilan balli maksimal bo‘lgan talabani toping.

28. Talabalar ismi va umumiy ballaridan iborat jadvaldan binar qidiruv usuli yordamida so‘ralgan talabaning umumiy balini chiqarish dasturini tuzing.

29. Boshiga qo‘yish usuli yordamida talabalar ismlaridan iborat massiv elementlariga ko‘p marta murojaat qilib massivni qayta tartiblang.

30. Transpozisiya usuli yordamida talabalar ismlaridan iborat ro‘yhat elementlariga ko‘p marta murojaat qilib massivni qayta tartiblang.

6-laboratoriya ishi. MA’LUMOTLARNI SARALASH USULLARI

Ishdan maqsad: Ushbu laboratoriya ishining maqsadi talabalar qanday saralash usullari va algoritmlari mavjudligini va ularning samaradorliklarini baholashni o‘rganishlari kerak. Shu asosda saralash usullarini qiyosiy tahlil qilishlari va ularga oid dasturlar tuzishni o‘zlashtirishlari kerak.

Qo‘yilgan masala: Talabalar topshiriq variantiga mos saralash usuli yordamida masalani yechish dasturini yaratish ko‘nikmasiga ega bo‘lishlari kerak.

Ish tartibi:

- Tajriba ishi nazariy ma’lumotlarini o‘rganish;
- Berilgan topshiriqnitп algoritmini ishlab chiqish;
- C++ dasturlash muhitida dasturni yaratish;
- Natijalarni tekshirish;
- Hisobotni tayyorlash va topshirish.

6.1. Tuzilma elementlarini saralash

Ma’lumotlarni kompyuterda qayta ishlashda elementning informatsion maydoni va uning mashina xotirasida joylashishini bilish zarur. Shu maqsadda ma’lumotlarni saralash amalga oshiriladi. Demak, saralash – bu ma’lumotlarni kalitlari bo‘yicha doimiy ko‘rinishda mashina xotirasida joylashtirishdan iborat. Bu yerda doimiylik ma’lumotlarni massivda kalitlari bo‘yicha o‘sishi tartibida berilishi tushuniladi.

Ma’lumotlarga qayta ishlov berilayotganda ma’lumotning informatsion maydonini hamda uning mashinada joylashishini (adresini) bilish zarur.

Saralashning ikkita turi mavjud: **ichki** va **tashqi**:

– ichki saralash bu operativ xotiradagi saralash;

– tashqi saralash – tashqi xotirada saralash.

Agar saralanayotgan yozuvlar xotirada katta hajmni egallasa, u holda ularni almashtirishlar katta sarf (vaqt va xotira ma’nosida) talab qiladi. Ushbu sarfni

kamaytirish maqsadida, saralash kalitlar adresi jadvalida amalga oshiriladi. Bunda faqatgina ma’lumot ko‘rsatkichlari almashtirilib, massiv o‘z joyida qoladi. Bu usul adreslar jadvalini saralash usuli deyiladi.

Saralanayotganda bir xil kalitlar uchrashi mumkin, bu holda saralangandan keyin bir xil kalitlilar boshlang‘ich tartibda qanday joylashgan bo‘lsa, shu tartibda qoldirilishi maqsadga muvofiq bo‘ladi (Bir xil kalitlilar o‘zlariga nisbatan). Bunday usulga turg‘un saralash deyiladi.

Saralash samaradorligini bir necha mezonlar bo‘yicha baholash mumkin:

– saralashga ketgan vaqt;

– saralash uchun talab qilingan operativ xotira;

– dasturni ishlab chiqishga ketgan vaqt.

Birinchi mezonni qarab chiqaylik. Saralash bajarilganda taqqoslashlar yoki almashtirishlar sonini hisoblash mumkin.

Faraz qilaylik, $\mathbf{N = 0{,}01n^2 + 10n}$ – taqqoslashlar soni. Agar $\mathbf{n < 1000}$ bo‘lsa, u holda ikkinchi qo‘shiluvchi katta, aks holda ya’ni, $\mathbf{n > 1000}$ bo‘lsa, birinchi qo‘shiluvchi katta bo‘ladi.

Demak, kichkina **n** larda taqqoslashlar soni **n** ga teng bo‘ladi, katta **n** larda esa $\mathbf{n^2}$ ga teng bo‘ladi.

Saralashda taqqoslashlar soni quyidagi oraliqlarda bo‘ladi:

$O(n\log n)$ dan $O(n^2)$gacha; $O(n)$ – ideal holatda.

Saralashning quyidagicha usullari bor:

– qat’iy (to‘g‘ridan-to‘g‘ri) usullar;

– yaxshilangan usullar.

Qat’iy usullarning afzalliklarini ko‘rib chiqaylik:

1. Bilamizki, dasturlarning o‘zlari ham xotirada joy egallaydi. To‘g‘ridan-to‘g‘ri saralash usullarining dasturlari qisqa bo‘lib, ular tushunishga oson.

2. To‘g‘ridan-to‘g‘ri saralash usullari orqali saralash tamoyillarining asosiy xususiyatlarini tushuntirish qulay.

3. Murakkablashtirilgan usullarda uncha ko‘p amallarni bajarish talab

qilinmasada, ushbu amallarning o‘zlari ham ancha murakkabdir. Garchi yetarlicha katta n larda ulardan foydalanish tavsiya etilmasada, kichik n larda mazkur usullar tezroq ishlaydi.

Shu joyni o‘zida qat’iy usullarni ishlash tamoyillariga ko‘ra 3 ta toifaga bo‘lish mumkin:

1. To‘g‘ridan-to‘g‘ri qo‘shish usuli (by insertion);
2. To‘g‘ridan-to‘g‘ri tanlash usuli (by selection);
3. To‘g‘ridan-to‘g‘ri almashtirish usuli (by exchange).

6.2. To‘g‘ridan-to‘g‘ri qo‘shish usuli bilan saralash algoritmi

Bunday usul karta o‘yinida keng qo‘llaniladi. Elementlar (kartalar) hayolan “tayyor” **a(1),...,a(i-1)** va boshlang‘ich ketma-ketliklarga bo‘linadi. Har bir qadamda (**i=2** dan boshlanib, har bir qadamda bir birlikka oshirib boriladi) boshlang‘ich ketma-ketlikdan i-chi element ajratib olinib tayyor ketma-ketlikning kerakli joyiga qo‘yiladi.

To‘g‘ridan-to‘g‘ri qo‘shish orqali saralash algoritmi quyidagicha bo‘ladi:

```
for (int i=1;i<n;i++){
 x=a[i];
 x ni a[0]...a[i] oraliqning mos joyiga qo‘shish
}
```

Kerakli joyni qidirish jarayonini quyidagi tartibda olib borish qulay bo‘ladi. 2-elementdan boshlab har bir elementni qarab chiqamiz, ya’ni har bir element o‘zidan oldin turgan element bilan solishtiriladi. Agar qaralayotgan element kichik bo‘lsa, oldinda turgan element bilan o‘rin almashadi va yana o‘zidan oldinda turgan element bilan solishtiriladi, jarayon shu kabi davom etadi. Bu jarayon quyidagi shartlarning birortasi bajarilganda to‘xtatiladi:

1. **x** elementi oldida uning kalitidan kichik kalitli **a(j)** elementi chiqqanda.
2. **x** elementi oldida element qolmaganda.

```
for (int i=1;i<n;i++){
```

```
    int j=i;
    while(a[j]<a[j-1]){
        int t=a[j-1];
        a[j-1]=a[j];
        a[j]=t;
        j=j-1;
    }
}
```

Algoritm samaradorligi

Faraz qilaylik, taqqoslashlar soni **C**, o‘rinlashtirishlar soni **M** bo‘lsin. Agar massiv elementlari kamayish tartibida bo‘lsa, u holda taqqoslashlar soni eng katta bo‘lib, u $C_{\max} = \frac{n(n-1)}{2}$ ga teng bo‘ladi, ya’ni $O(n^2)$. O‘rinlashtirishlar soni esa $M_{\max} = C_{\max} + 3(n-1)$ ga teng bo‘ladi, ya’ni $O(n^2)$. Agar berilgan massiv o‘sish tartibida saralangan bo‘lsa, u holda taqqoslashlar va o‘rinlashtirishlar soni eng kichik bo‘ladi, ya’ni $C_{\min} = n-1$, $M_{\min} = 3(n-1)$.

6.3. Tanlash orqali saralash algoritmi

Mazkur usul quyidagi tamoyillarga asoslangan:

1. Eng kichik kalitga ega element tanlanadi.

2. Ushbu element a_0 birinchi element bilan o‘rin almashinadi.

3. Keyin mazkur jarayon qolgan **n-1, n-2** elementlar bilan takrorlanib, to bitta eng “katta” element qolguncha davom ettiriladi.

```
for(int i=0;i<n-1;i++)
for(int j=i+1;j<n;j++)
    if (a[i] > a[j]){
      int k = a[j];
       a[j]= a[i];
```

```
        a[i] = k;
    }
```

Algoritm samaradorligi:

- Taqqoslashlar soni

$$C_{min} = \frac{N(N-1)}{2} = \frac{N^2-N}{2}$$

- Massiv tartiblanganda o‘rinlashtirishlar soni

$$M_{min} = 3(N-1)$$

- Massiv teskari tartiblanganda o‘rinlashtirishlar soni

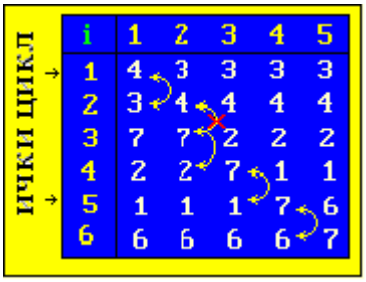
$$M_{max} = \frac{N(N-1)}{2} + 3(N-1) = \frac{N^2+5N-6}{2}$$

Ushbu usul bo‘yicha saralash bajarilsa, eng yomon holda taqqoslashlar va o‘rinlashtirishlar soni tartibi n^2 bo‘ladi.

6.4. Pufaksimon saralash algoritmi

Ushbu usulning g‘oyasi quyidagicha: **n - 1** marta massivda quyidan yuqoriga qarab yurib kalitlar jufti-jufti bilan taqqoslanadi. Agar pastki kalit qiymati yuqoridagi jufti kalitidan kichik bo‘lsa, u holda ularning o‘rni almashtiriladi (6.1-rasm).

Misol : massiv - 4, 3, 7, 2, 1, 6.



6.1-rasm. Pufaksimon saralash usulida massiv elementlarining o‘rnini almashtirish

Pufaksimon usulni massiv elementlarida pastdan yuqoriga va

yuqoridan pastga o‘tishni bir vaqtda amalga oshirish natijasida yaxshilash mumkin.

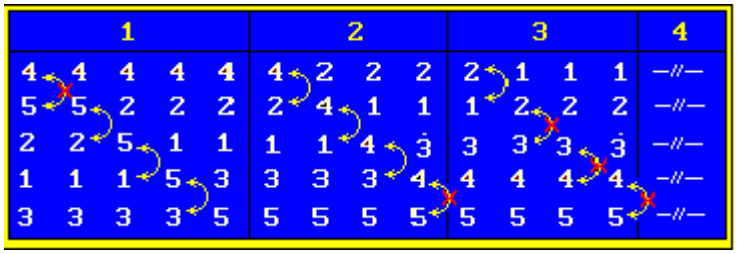
Taqqoslashlar soni:

$$M = \frac{n}{2} \cdot \frac{n}{2} = \frac{n^2}{4}$$

Almashtirishlar soni:

$$C_{mzx} = 3 \cdot \frac{n^2}{4}$$

“Pufaksimon” saralash usulini hisoblashga misol



6.2-rasm. Massivni pufaksimon saralashga misol

6.2-rasmda berilgan misolda 5 ta elementdan iborat massiv berilgan. Demak, massivda pastdan yuqoriga (yuqoridan pastga) o‘tishlar soni 5-1=4 marta bo‘ladi. Misoldan ko‘rinib turibdiki, algoritm ichki siklda 3-qadamdan boshlab massivni “bekor” qayta ishlaydi, 4-qadamni bajarmasa ham bo‘ladi.

Berilgan usullarning afzalligi:

1) Eng sodda algoritm;

2) Amalga oshirish sodda;

3) Qo‘shimcha o‘zgaruvchilar shart emas.

Kamchiliklari:

1) Katta massivlarni uzoq qayta ishlaydi;

2) Har qanday holatda ham o‘tishlar soni kamaymaydi.

6.5. “Pufaksimon” usulni yaxshilash

1) Agar massivda o‘tishlar nafaqat yuqoridan pastga, balki bir vaqtning

o‘zida pastdan yuqoriga ham bo‘lsa, u holda “yengil” elementlar “yuqoriga suzib” chiqadi va “og‘ir” elementlar esa “cho‘kadi”.

2) Massivda “bekor” o‘tishni yo‘q qilish uchun, tashqi siklda massiv saralanganligini tekshiruvchi belgi qo‘yish lozim.

```
for (int i=0;i<n;i++)
for (int j=n-1;j>i;j--)
   if (a[j] < a[j - 1]){
      int x= a[j - 1];
      a[j - 1] = a[j];
      a[j] = x;
     }
```

O‘rinlashtirish va taqqoslashlar soni: **(n* log(n)).**

6.6. Quiksort – tez saralash algoritmi

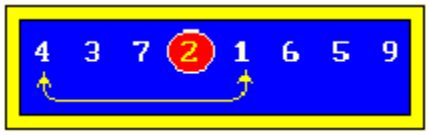
Bu algoritm “bo‘lib ol va egalik qil” tamoyilining yaqqol misolidir. Bu algotirm rekursiv bo‘lib, o‘rtacha $\mathbf{N*log_2N}$ ta solishtirish natijasida saralaydi. Algoritm berilgan massivni saralash uchun uni 2 taga bo‘lib oladi. Bo‘lib olish uchun ixtiyoriy elementni tanlab undan 2 ta qismga ajratiladi. Lekin o‘rtadagi elementni tanlab, massivning teng yarmidan 2 ga ajratgan ma’qul. Tanlangan kalit elementga nisbatan chapdagi va o‘ngdagi har bir element solishtiriladi. Kalit elementdan kichiklar chapga, kattalar o‘ng tomonga o‘tkaziladi (6.3-rasm). Endi massivning har ikkala tomonida xuddi yuqoridagi amallar takrorlanadi. Ya’ni bu oraliqlarning o‘rtasidagi elementlar kalit sifatida olinadi va h.k.

Misol uchun rasmdagi massivni saralash algoritmini ko‘rib chiqamiz.

1. Oraliq sifatida **0** dan **n-1** gacha bo‘lgan massivning barcha elementlarini olamiz.

2. Oraliq o‘rtasidagi kalit elementni tanlaymiz, ya’ni

key=(<oraliq_boshi>+<oraliq_oxiri>)/2, i=<oraliq_boshi>, j=<oraliq_oxiri>.



6.3-rasm. Quicksort algoritmida o‘rinlashtirish

3. Chapdagi i-elementni **key** bilan solishtiramiz. Agar **key** kichik bo‘lsa, keyingi qadamga o‘tamiz. Aks holda **i++** va shu qadamni takrorlaymiz.

4. O‘ngdagi **j**-element bilan **key** solishtiriladi. Agar **key** katta bo‘lsa, keyingi qadamga o‘tamiz, aks holda **j--** va shu qadamni takrorlaymiz.

5. **i-** va **j**-elementlarning o‘rni almashtiriladi. Agar **i<=j** bo‘lsa, 3-qadamga o‘tiladi.

Birinchi o‘tishdan keyin tanlangan element o‘zining joyiga kelib joylashadi.

6. Endi shu ko‘rilayotgan oraliqda **key** kalitning chap tomonida elementlar mavjud bo‘lsa, ular ustida yuqoridagi amallarni bajarish lozim, ya’ni ko‘riladigan oraliq **0** dan **key-1** gacha deb belgilanadi va 2-qadamga o‘tiladi. Aks holda keyingi qadamga o‘tiladi.

7. Endi shu ko‘rilayotgan oraliqda **key** kalitning o‘ng tomonida elementlar mavjud bo‘lsa, ular ustida yuqoridagi amallarni bajarish lozim, ya’ni ko‘riladigan oraliq **key+1** dan **n-1** gacha deb belgilanadi va 2-qadamga o‘tiladi. Aks holda algoritm tugaydi.

Shu algoritmga misol ko‘rib chiqamiz.

Misol: Talabalar ism-sharifi va tartib raqamidan iborat jadvalni quicksort algoritmi bilan saralang va nechta o‘rinlashtirish amalga oshirilganini aniqlang.

Dastur kodi

```
#include <cstring>
#include <iostream>
using namespace std;
struct table{
    int t;
    string FIO;
```

```
        };
        int q=0;
    void qs(table *a,int first,int last){
      int i = first, j = last;table x =a[(first + last) / 2];
      do {
         while (a[i].FIO < x.FIO) i++;
         while (a[j].FIO > x.FIO) j--;
         if(i <= j) {
            if (i < j){ swap(a[i], a[j]);q++;}
            i++;
            j--;
         }
      } while (i <= j);
      if (i < last)
         qs(a,i,last);
      if (first < j)
         qs(a,first,j);
    }
    int main(int args, char *argv[])
    {    int n;cout<<"n=";cin>>n;
        table talaba[n];
        for(int i=0;i<n;i++){
              talaba[i].t=i+1;
              cin>>talaba[i].FIO;
              }
        qs(talaba,0,n-1);
             for(int i=0;i<n;i++)
                cout<<talaba[i].t<<"  "<<talaba[i].FIO<<endl;
                cout<<"quicksort  algoritmi  "<<q<<"  ta  o‘rinlashtirishda
saraladi\n";
```

```
        system("PAUSE");
}
```

Dastur natijasi:

talabalar sonini kiriting=5

5 ta talabalar FIO sini kiriting

Farhod

Asror

Sobir

Bobur

Vali

| 2 | Asror |

| 4 | Bobur |

| 1 | Farhod |

| 3 | Sobir |

| 5 | Vali |

bu algoritm jadvalni 3 ta o‘rinlashtirishda saraladi

Ishni bajarishga namuna

Masalaning qo‘yilishi – tabalarning ism, familiyalarini optimallashtirilgan pufaksimon usuli bilan tartibga keltirish dasturini tuzamiz va saralash nechta o‘rin almashtirish bilan amalga oshirilganini aniqlaymiz.

Algoritm

1. Jadvalga talabalar ism-sharifini kiritamiz.

2. Jadvaldagi **1**-elementni olamiz, **i=0.**

3. Jadvaldagi **n-1** oxirgi elementdan to **i**-elementgacha barcha elementni FIO maydonini o‘zidan oldin turgan element FIO maydoni bilan solishtiramiz. Agar zarur bo‘lsa, o‘rin almashtiramiz va o‘rin almashtirishlar hisoblagichi **l** ning qiymatini bittaga oshiramiz, ya’ni **l++.**

4. Agar **i<n** bo‘lsa, **i++** va 3-qadamga o‘tamiz.

5. Natijaviy saralangan massivni ekranga chiqaramiz.

Dastur kodi

```
#include <cstring>
#include <iostream>
using namespace std;
int main(int args, char *argv[])
{
 int n; cout<<"talabalar sonini kiriting=";cin>>n;
 struct table{
        int t;
        char FIO[20];
        } talaba[n];
cout<<n<<" ta talabalar FIO sini kiriting"<<endl;
        for(int i=0;i<n;i++){
              talaba[i].t=i+1;
              cin>>talaba[i].FIO;
              }
         int l=0;
        for(int i=0;i<n;i++){
        for(int j=n-1;j>i;j--){
         if (strcmp(talaba[j-1].FIO,talaba[j].FIO)==1){
                                        l++;
                                    table k=talaba[j];
                                    talaba[j]=talaba[j-1];
                                    talaba[j-1]=k;
                                    }
                              }
                            }
          for(int i=0;i<n;i++)
              cout<<"| "<<talaba[i].t<<" | "<<talaba[i].FIO<<" |"<<endl;
          cout<<"bu algoritm jadvalni "<<l<<" ta o‘rinlashtirishda
        saraladi\n";
```

```
        system("PAUSE");
}
```

Dastur natijasi:

talabalar sonini kiriting=5

5 ta talabalar FIO sini kiriting

Farhod

Asror

Sobir

Bobur

Vali

| 2 | Asror |

| 4 | Bobur |

| 1 | Farhod |

| 3 | Sobir |

| 5 | Vali |

bu algoritm jadvalni 10 ta solishtirishda saraladi

Nazorat savollari

1. Qanday saralash algoritmlarini bilasiz?
2. Saralash algoritmlari samaradorligini qanday baholash mumkin?
3. Pufaksimon saralash algoritmi va uni yahshilangan usulini tushuntiring.
4. To’g’ridan-to’g’ri qo’shish, tanlash algoritmlarini farqini tushuntiring.
5. Shella saralash algoritmini tushuntiring.
6. Quicksort algoritmini tushuntiring.

Topshiriq

Quyida har 10 ta variant uchun umumiy bo‘lgan masalaning berilishi va talab qilinayotgan saralash usuli keltirilgan. Talabalar topshiriq olib so‘ralayotgan usul bilan o‘zlari tomonidan tanlangan ixtiyoriy saralash

usulining samaradorligini solishtirish dasturini tuzishlari kerak. Usullarni solishtirishda o‘rin almashtirishlar soni nazarda tutiladi.

Ta’mirlash ustaxonasida bir nechta (N ta) mashina bor. Ular to‘g‘risida quyidagi ma’lumotlarga egamiz: raqami, markasi, egasining ismi, oxirgi marta ta’mirlanganligi sanasi (kuni, oyi, yili), ta’mirdan chiqishi lozim bo‘lgan sana (kun, oy, yil).

To‘g‘ridan-to‘g‘ri qo‘shish usulidan foydalanib, saralashni amalga oshirish dasturini ishlab chiqish (variantga mos ravishda):

1. Mashina egalarining ismlari bo‘yicha alifbo tartibida joylashtirilsin va mos ravishda ularning mashinalari haqidagi ma’lumotlar chiqarilsin.
2. Avtomobillarni ta’mirlash tartibi ishlab chiqilsin. Bu yerda ta’mir tugashi sanasi qaysi avtomobil uchun ertaroq bo‘lsa, shunga birinchi navbatda xizmat ko‘rsatiladi.
3. Oldingi ta’mir qilinganlar soni 2 ga teng bo‘lgan mashinalar raqamlari bo‘yicha kamayish tartibida joylashtirilsin.
4. Oldin ta’mir qilinmagan mashinalarni ta’mirdan chiqish sanasi bo‘yicha o‘sish tartibida joylashtiring.
5. "Mersedes" markali mashina egalarini alifbo bo‘yicha teskari tartibda joylashtiring.
6. Boshqalaridan oldinroq ta’mirlanadigan mashinalarni ularning markasi bo‘yicha alifbo tartibida joylashtiring (ta’mir tugatilishi sanasi 31.12.2012 dan erta).
7. "Nexia" markasidagi mashinalarni raqamlari bo‘yicha o‘sish tartibida joylashtiring.
8. O‘tgan yildan beri ta’mirlanmagan mashinalarni ularning egalari ismlari bo‘yicha alifbo tartibida joylashtiring.
9. Keyingi oyda ta’mirlanishi lozim bo‘lgan mashinalarni oxirgi marta ta’mirlanganlik sanasi bo‘yicha o‘sish tartibida keltiring.
10. "Mersedes" markasidagi mashinalarni raqamlari bo‘yicha kamayish tartibida joylashtiring.

N ta talabadan iborat guruh tuzilsin. Quyidagi ma’lumotlar berilgan: familiya, ism, tug‘ilgan yili, fanlar bo‘yicha bahosi: MTvaA, oliy matematika, fizika, dasturlash, topshirgan sessiya umumiy bali.

To‘g‘ri tanlov usulidan foydalanib, saralashni amalga oshirish dasturini ishlab chiqing (variantga mos ravishda):

11. Talabalar familiyalarini alifbo tartibida.
12. Talabalarni yoshi bo‘yicha o‘sish tartibida.
13. Talabalarni umumiy bali bo‘yicha o‘sish tartibida.
14. Talabalarni birinchi imtihoni natijasi bo‘yicha o‘sish tartibida.
15. Talabalarni ikkinchi imtihoni natijasi bo‘yicha kamayish tartibida.
16. Talabalarni uchinchi imtihoni natijasi bo‘yicha o‘sish tartibida.
17. Talabalarni to‘rtinchi imtihoni natijasi bo‘yicha kamayish tartibida.
18. Talabalarni birinchi va ikkinchi imtihoni natijalari bo‘yicha o‘sish tartibida.
19. Talabalarni birinchi va ikkinchi imtihoni natijalari bo‘yicha kamayish tartibida.
20. Talabalarni umumiy bali bo‘yicha kamayish tartibida.

Pufaksimon saralash usulidan foydalanib, saralashni amalga oshirish dasturini ishlab chiqish (variantga mos ravishda):

21. A massivning eng katta (eng kichik) elementini ekranga chiqarish dasturini tuzing.
22. A massiv elementlari qiymatlarini kamayish tartibida saralash dasturini tuzing.
23. A massivda elementlar berilgan. Mazkur massiv elementlaridan shunday

V massiv shakllantiruvchi shunday dastur tuzingki, V massiv elementlari kamayish tartibida saralangan bo‘lsin.

24. Elementlari o‘sish tartibida joylashgan A sonli massiv va a soni berilgan. a ni A massivga shunday qo‘shingki, tartiblanganlik buzilmasin.

25. Elementlari o‘sish tartibida joylashgan A massivni, elementlari kamayish tartibida joylashgan massiv ko‘rinishida tez quruvchi dastur tuzing.

26. Manfiy va musbat sonlardan tashkil topgan A massiv berilgan. Barcha manfiy sonlarni chiqarib, musbatlarini o‘sish tartibda joylashtiruvchi dastur tuzing.

27. Berilgan A massivdan ketma-ket sonlar olib, ulardan o‘sish tartibida shakllantirilgan V massiv hosil qiluvchi dastur tuzing.

28. Mualliflar ro‘yhati A massiv shaklida berilgan. Mualliflarni alifbo tartibida shakllantirish va shakllangan ro‘yhatni ekranga chiqarish dasturini tuzing.

29. Telefon stansiyasida n ta mijoz bor. Quyidagi shaklda ro‘yhat hosil qiluvchi dastur tuzing: telefon raqami, mijoz familiyasi (telefon raqamlari o‘sish tartibida joylashadi).

30. A massivni uzunliklari har xil bo‘lgan n ta so‘z tashkil qiladi. So‘zlarni uzunliklari bo‘yicha o‘sish tartibida joylashtiruvchi dastur tuzing.

FOYDALANILGAN ADABIYOTLAR

1. ***Алфред В. Ахо., Джон Э. Хопкрофт, Джефри Д. Ульман.*** Структура данных и алгоритмы//Учеб.пос., М. : Изд.дом: "Вильямс", 2000, - 384 с.

2. ***Бакнелл Джулиан М.*** Фундаментальные алгоритмы и структуры данных в Delphi//СПб: ООО «ДиаСофтЮП», 2003. 560с.

3. ***Роберт Седжвик.*** Фундаментальные алгоритмы на С++. Анализ, Структуры данных, Сортировка, Поиск//К.: Изд. «ДиаСофт», 2001.- 688 с.

4. ***Динман М.И.*** С++. Освой на примерах//СПБ.:БХВ-Петербург, 2006, 384.

5. ***Шилдт, Герберт.*** Полный справочник по C#//M. : Изд. дом "Вильямс", 2004, 752 с.

6. ***Вирт Н.*** Алгоритмы и структуры программы//М., Мир, 1985.

7. ***Лойко В.И.*** Структуры и алгоритмы обработки данных. Учебное пособие для вузов.- Краснодар: КубГАУ. 2000. - 261 с., ил.

8. ***Knuth, D. E.*** (1968). The Art of Computer Programming Vol. I: Fundamental Algorithms, Addison – Wesley, Reading, Mass. (Русский перевод: Кнут Д. Искусство программирования для ЭВМ. Том 1: Основные алгоритмы. – М., «Мир», 1976. Русский перевод переработанного издания: Кнут Д. Искусство программирования. Том 1: Основные алгоритмы. – М., Издательский дом «Вильямс», 2000.)

9. ***Джон Бентли***. Жемчужины программирования. СПб.: Питер, 2002.-272 с.

10. ***Акбаралиев Б.Б.*** Конспект лекций по курсу “Маълумотлар тузилмаси ва алгоритмлар” для студентов по специальности 5521900 “Информатика и информационные технологии”, Ташкент, 2008 г.

11. ***Акбаралиев Б.Б.*** Методические указания к лабораторным работам по курсу “Маълумотлар тузилмаси ва алгоритмлар” для студентов по

специальности 5521900 “Информатика и информационные технологии”, Ташкент, 2008 г.

12. ***Xudoyberdiyev M.X., Akbaraliyev B.B.*** “Ma’lumotlat tuzilmasi va algoritmlar” fanidan amaliy mashg‘ulotlar uchun topshiriqlar (uslubiy ko‘rsatmalari bilan). Toshklent, 2013 y.