O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARNI RIVOJLANTIRISH VAZIRLIGI

TOSHKENT AXBOROT TEXNOLOGIYALAR UNIVERSITETI SAMARQAND FILIALI

KOMPYUTER INJINIRINGI FAKULTETI
DASTURIY INJINIRING KAFEDRASI

5330200 -“Informatika va axborot texnologiyalari” yo’nalishi bo’yicha bakalavr akademik darajasini olish uchun

BITIRUV MALAKAVIY ISHI

Mavzu: O’zbek tilining elektron lug’ati uchun dasturlar majmuasini yaratish.

Ish kafedraning 2020 yil ___ _____dagi____-sonli majlisda muhokama qilindi va himoyaga tavsiya etildi.

Kafedra mudiri _____

dots. A.B.Qarshiyev

“___”_____2020y.

Bajardi: 404-guruh talabasi
_____Maxmidov Shirinboy

Ilmiy rahbar: _____

dots. A.B.Qarshiyev

S A M A R Q A N D – 2020

M U N D A R I J A

KIRISH

Mavzuning dolzarbligi. O’zbekiston Respublikasi Prezidentining 2012 yil 21 martdagi «Zamonaviy axborot-kommunikasiya texnologiyalarini yanada joriy etish va rivojlantirish chora-tadbirlari to’g’risida» PQ-1730 sonli qarori Respublikamizda axborot-kommunikatsiya texnologiyalari taraqqiyotida yana bir muhim bosqich bo’lib qoladi. Prezidentimiz I.Karimov XXI asrni ma’naviy- ma’rifat asri, ilm-fan madaniyat va axborotlar asri deb ta’riflar ekan, bu bilan hozirgi davrning muhim belgisi va xususiyatini ham belgilab berganini payqab olish qiyin emas [1].

Hozirgi kunda mamlakatimizni kompyuterlashtirish jadal rivojlanmoqda. Ayni vaqtda tilshunos olimlarimiz o’z sohasida o’zbek tilining so’z boyligi, yozuvchilarimiz asarlarida qancha so’zdan foydalanganligini aniqlashi muammolardan biridir. Buning uchun kompyuter dasturiy vositalari ishlab chiqilmoqda. Ulardan misol qilib har xil lug’at(o’zbekcha-inglizcha, o’zbekcha-ruscha)larni yoki kirildan lotinchaga o’tkazish dasturlarnini olish mumkin. O’zbek tili statistik tahlili uchun dasturiy ta’minotlar yaratish oxirigacha yetkazilmagan. BMI dagi dasturiy ta’minot o’zbek tili matnlarning statistik tahlili uchun qism dasturiy vosita bo’lib hisoblanadi.

BMI da qaralayotgan dasturiy vosita dolzarbliligi va zamonaviyligi shundaki o’zbek tili so’z boyligini aniqlash uchun yordam beradi. Undan tashqari yozuvchilarning asarlarida qancha so’zlardan foydalangan, qaysi so’zni o’z asarida ko’p foydalanganligini aniqlash mumkin. Bu esa yozuvchimizning so’z boyligi aniqlash statistik tahlili uchun foydalanishga yordam beradi.

Ishning maqsadi. BMI ishining maqsadi o’zbek tilining elektron lug’atini hosil qilish dasturiy ta’minotini yaratish.

Ushbu maqsadlarga erishish uchun quyidagi **vazifalarni** bajarish zarur:

- kompyuter lingvistikasiga oid, xususan, lug’atlar va ularni tuzishgaoid adabiyotlarni o’rganish;

- o’zbek tilining kompyuter yordamida statistik tahlili, alfavit-chastotali lug’atlar tuzilishi, mavjud elektron lug’atlar yaratish kabi ishlar bajarilishining zamonaviy darajasini tahlil qilish;
- o’zbek tili elektron lug’atini yaratishning dasturiy majmuasini loyihalash;
- elektron lug’at yaratish algoritmlarini ishlab chiqish;
- dasturiy majmuani kodlashtirish, sozlash va sinovdan o’tkazish;
- dasturiy majmuani o’zbek tili elektron lug’atini yaratish uchun tadbiq etish.

Ishning amaliy ahamiyati. Matnlarning statistik tahlili va uning natijalari asosida xulosalar chiqarish tilshunos olimlar faoliyatida muhim ahamiyatga ega. Matnlar statistikasi uchun kompyuter dasturlari yaratish va ularni turli lug‘atlar tuzish jarayonida qo‘llash tilshunoslar mehnatini engillatuvchi bir vositadir. Ilgari S.Muxamedov, S.Rizaev, X.Arzikulov, S.Karimov va boshqa olimlar tomondan o‘zbek tilining statistik tadqiqotlariga oid ilmiy ishlar bajarilgan va o‘sha paytda mavjud katta EHMlar uchun dasturlar majmualari ham yaratilgan. Lekin, hozirgi kunda matnlar statistikasiga oid masalalarni zamonaviy kompyuter texnologiyalari yordamida hal qilishga mo‘ljallangan dasturiy mahsulotlar mavjud emas. Xorijda va MDH mamlakatlarida tabiiy til matnlarini statistik tahlil qilishga va turli lug‘atlar tuzishga mo‘ljallangan kompyuter dasturlari ko‘plab yaratilgan. Lekin ularni o‘zbek tili uchun bevosita ishlatib bo‘lmaydi. Shuning uchun, katta hajmdagi matnlarning kompyuter yordamida statistik tahliliga mo‘ljallangan amaliy dasturlar majmuasini (ADM) yaratish amaliy ahamiyatga ega ishdir [2, 3].

Dasturni o’zbek tili elektron lug’ati hosil qilshda foydalanishi mumkin. Undan tashqari tadqiqotchi filologlar ham dasturdan foydalanishlari mumkin.

Bitiruv malakaviy ish kirish, 3ta bob, xotima qismi, adabiyotlar ro’yxati va ilovalardan iborat. Kirishda mavzuning dolzarbligi asoslangan, ishning maqsadi, vazufalari va amaliy ahamiyati keltirilgan, ishning tarkibiy qismlari tavsiflangan.

Birinchi bobda dasturiy ta’minotni loyihalashga doir nazariy ma’lumotlar keltirilgan, elektorn lug’at yaratish dasturiy majmuasini loyihasi ishlab chiqilgan.

Ikkinchi bobda yaratilgan elektron lug’atni faylga yozish va fayldan ko’rish funksiyalari, hamda so’zlarni saralash algoritmlari keltirilgan.

Uchinchi bobda elektron lug’at yaratilish strukturasi, dasturning tavsifi va elektron lug’at dasturidan foydalanish ketma-ketlik jarayoni berilgan.

Xotima qismida birituv malakaviy ishni bajarish jarayonida amalga oshirilgan ishlar va natijalar sanab o’tilgan, olingan natijalar asosida xulosalar chiqarilgan. Adabiyotlar ro’yxati 2ta banddan iborat bo’lib, BMI ni bajarishda foydalanilgan huquqiy-me’yoriy, ilmiy-uslubiy va badiiy adabiyotlarni o’z ichiga olgan.

Ilova qismida yaratilgan dasturlarning C++ tilidagi kodi berilgan.

I. BOB. ELEKTRON LUG‘AT DASTURIY TA’MINOTINI LOYIHALASH

1.1. Dasturiy ta’minotni loyihalashtirishning nazariy asoslari

Loyihalash deganda, foydalanuvchi yoki kuzatuvchi nuqtai nazaridan ishlab chiqarilayotgan mahsulotning harakati tushuniladi. Bu jarayonning maqsadi sifatida hosil qilinayotgan mahsulotning qismi orasidagi o‘zaro bog‘liqlik tashqi muammolar amalga oshirilishini keltirish mumkin. Ular tashqi spetsifikatsiya ko‘rinishda rasmiylashtirilib bu spetsifikatsiyalar foydalanuvchilarga va kenga ommaga qaratilgan bo‘lishi kerak. Tashqi loyihalashning maxsus usullari va ko‘rinishlari bo‘lmasligidan qat’iy nazar mahsulotni tashkil qiluvchi tashqi funksiyalar konsiptual birligiga ega bo‘lishlari lozim. Konseptual birlik esa foydalanuvchi bilan muloqotni mezoni bo‘lib xisoblanadi [9].

Har qanday dasturiy ta’minotni loyihalash uning tuzilmasini, ya’ni uning tarkibiy qismlari va ular orasidagi bog’lanishlarni aniqlashdan boshlanadi.Tuzilmani aniqlasht natijasi tuzilmaviy yoki funksional sxema va komponentlar (tarkibiy qismlar) tavsifi sifatida tasvirlanishi mumkin.

Dasturning tuzilmaviy sxemasi deganda uning tarkibini va qismlarni boshqarishdagi o’zaro ta’sirlarni aks ettiruvchi sxemani tushunamiz. Biz loyihalayotgan dasturda tuzilmaviy kompanentlar faqat qismiy dasturlardan iborat. Umuman olganda dasturning komponentalari, qismiy dasturlar, ma’lumotlar bazalari, resurslar bibliotekasi va hokazolardan iborat bo’lishi mumkin.

Tashqi loyihalashning maxsus usullari va ko‘rinishlari bo‘lmasligidan qat’iy nazar mahsulotni tashkil qiluvchi tashqi funksiyalar konsiptual birligiga ega bo‘lishlari lozim. Konseptual birlik esa foydalanuvchi bilan muloqotni mezoni bo‘lib xisoblanadi.

Konseptual birlikka ega bo‘lmagan mahsulotlarda foydalanuvchi bilan muloqot kamiga qiyinchilik bilan amalga oshiriladi.

“Tashqi loyihalash dasturlash bilan umuman bog‘liq emas. Bunda ko‘proq atrof-muhitni ko‘zda tutishi, inson psixologiyasini, foydalanuvchi muammolarini e’tiborga olish kerak.

Tashqi loyihalarni hosil qilishda ishlab chiqaruvchilarning dasturiy mahsulot ishonchliligini oshiruvchi uch faktorni ko‘zda tutish lozim”.

1. Foydalanuvchi xatolarini minimumga keltirish.

2. Xatoliklar mavjud bo‘lsa, ularni aniqlash.

3. Dasturlash vositasi murakkabligini kamaytirish.

Tashqi loyiha murakkabligiga qarab va uning dasturiy mahsulotda tutgan o‘rniga ko‘ra quyidagi standart bo‘yicha tasdiqlangan qoidalarni hisobga olishshart.

1. Foydalanuvchi bilan bo‘lgan muloqotda uning tayyorgarligi bilan foydalanuvchi ishlayapgan muxitdagi cheklanishlar xisobga olinadi.

2. Natijaviy qiymatlar talab qilinayotgan ko‘rinishga ega bo‘lib, imkoniyat boricha izoxlanmog‘i lozim.

3. Foydalanuvchi iloji boricha klaviaturadan kamroq axborot kiritsin.

4. Kiritilayotgan va chiqarilayotgan qiymatlar uchun konseptual birlik ta’minlanmog‘i lozim.

5. Dastur yordam ko‘rinishidagi vositalariga ega bo‘lishi mumkin.

6. Ekrandan unumli foydalanish.

7. Tizim (dastur) shu ko‘rinishda loyihalashtirilgan bo‘lishi lozimki, foydalanuvchi xoxlagan vaqtida ishini tamomlash yoki dastlabki xolatga qaytishi mumkin bo‘lsin.

8.Foydalanuvchini har bir harakati tizim tomonidan tekshirilsin.

Modul – bu kamida bitta operatordan iborat, tugallangan dastur.

Kant qonunlari:

1) Ayrim modullarni ayrim funksiyalardan tashkil topgan ko‘rinishda rasmiylashtirish modullar mustahkamligini ta’minlaydi;

2) Modullararo bog’liqlikni rasmiy mexanizmga ko‘ra almashishini xisobga olib kamaytirish -bu modullararo bog’liqlikni kuchsizlantiradi.

3) Bog’liklikni amalga oshirishda standart qoidalardan foydalanish lozim. Bunda boshqarish va axborot almashish orqali bog’liqlik ko‘zda tutiladi.

4) Dasturlash kompleksi unchalik katta hajmga ega bo‘lmagan modullardan tashkil topgan bo‘lib, bog’liqlikni ierarxik tuzilishda akslantirmog’i lozim. Bu tuzilish orqali har bir dasturchi har bir modul va dasturni ish xakidagi ma’lumotga ega bo‘lishi lozim

5) Qoidaga ko‘ra har bir modul 10tadan 100tagacha tashkil topgan bo‘lishi kerak.

6) Modul mustahkamlik xususiyatiga ega bo‘lishi lozim. Modulning mustahkamligi uning ichki aloqalari orqali belgilanadi.

7) Modulning ishini oldindan ko‘ra bilish lozim ya’ni modul ko‘rinisha mustakil bo‘lishi lozimki u o‘zining dastlabki ishlatishlariga boklik bo‘lmasin.

8) Yechimlarni qabul qilish strukturasi aniqlangan bo‘lishi lozim, bu talablarga ko‘ra qabul qilinayotgan yechimlar ta’sir qilayotgan modullar chiqariladigan modullar sifatida rasmiylashtirilishi lozim.

Ma’lumotlarga murojaat qilinish minimumga keltirilsin ya’ni har bir modul talab qilayotgan ma’lumotlar hajmi iloji boricha kichikroq bo‘lishi lozim [9].

1.2. Elektron lug’at yaratish dasturlarining loyihasini ishlab chiqish

Axborot texnologiyalar sohasida kompyuterda biron-bir masalani yechilishini ta’minlaydigan ishdir. Qo’yilgan maqsadni amalga oshirish uchun kerakli ma’lumotlar tarkibi, tuzilishi, ifodalanishi aniqlangan bo’lib, ular orasidagi bog’lanishlar aniq ifodalangan bo’lsa, masala qo’yilgan deb hisoblanadi.

Yaratilgan dasturiy mahsulotni tilshunoslik sohasidagi ilmiy tadqiqotlar faoliyatida foydalanish nazarda tutiladi.

Ushbu bitiruv ishda kompyuter yordamida o’zbek tilining elektron lug’atini hosil qilish dasturiy majmua yaratilishi nazarda tutilgan. Shuning uchun qaralayotgan masala dolzarb ahamiyatga ega.

Shunday qilib, yaratiladigan dasturning vazifasi o’zbek tili matnini dasturga kiritish va so’zlarni hisoblab borish jarayoni nazarda tutilgan. Hosil bo’lgan elektron lug’atni faylga saqlab borish kerak. Bunda fayldagi so’zlar yangi lug’at bilan solishtirilib faylda mavjud bo’lmagan so’z olinadi. Shu tariqa jarayon davom

ettirilib, elektron lug’at ma’lum hajmi oshishi kamayib boradi. Ma’lum hajmga yetganda elektron lug’at o’zgarmay qoladi. Bu esa o’zbek tilining so’z boyligini aniqlashga imkon beradi.

Dasturiy mahsulotga qo’yiladigan talablar

I. Funksional talablar.

Dastur quyidagi funksiyalarni bajara olishi zarur:

- Boshlang’ich qiymatlarni kiritish;
- Kiritilgan matndagi so’zlani ajratib olish;
- So’zlar ro’yxatini tuzish

Natijalar:

- So’zlar ro’yxatini faylga kiritish;
- So’zlarni miqdorini aniqlash.

II. Ishonchlilikka qo’yiladigan talablar:

- Dasturda kiritilgan matnni so’zlarga ajratib olmasdan so’zlar ro’yxatini yaratib bo’lmaydi.
- Foydalanuvchining dastur bilan ishlayotganda foydalanuchiga ko’rinmaydigan har bir harakat haqida ma’lumot beradi.

III. Texnik vositalar tarkibi va parametrlariga qo’yiladigan talablar:

- Dastur IBM turidagi barcha shaxsiy kompyuterlarda ishlashi kerak;
- Minimal konfiguratsiya:

Prosessor turi: Pentium va undan yuqori.

Operativ xotira hajmi: 32 Mb va undan yuqori.

IV. Axborot va dasturiy moslashuvga talablar.

Dastur Windows oilasiga mansub operasion tizimlarda (Windows 95, Windows 98,Windows 2000, Windows XP va h.k.) ishlaydi.

Dasturning tuzilmasi

Dasturning tuzilmasini aniqlashni qadamma-qadam detallashtirib borish usuli asosida bajaramiz.

Qadamma-qadam detallashtirish dastur yaratishning “*quyiga borish*” yondashuvini amalga oshirib, strukturali dasturlashning asosiy kostruksiyalarini ishlatishga tayanadi. Bu usulda algoritmni ishlab chiqish qadamma-qadam ketma-ketlikda bajariladi. Har bir qadamda vazifalar qismiy vazifalarga bo’linadi. Birinchi qadamda qo’yilgan masala yechimi tavsiflanib, qismiy masalalar ajratilsa, navbatdagi qadamda qismiy masalalar yechimi tavsiflanib, bir darajaga quyi bo’lgan qismiy masalalar aniqlanadi. Har bir qadamda loyihalanayotgan dasturiy ta’minot funksiyalari aniqlashtirilib boriladi. Bu jarayon algoritmi eng sodda qismiy masalalar hosil bo’lgunga qadar davom ettiriladi.

Shunday qilib ushbu bitiruv malakaviy ish uchun qo’yilgan masala o’zbek tilining elektron lug’atini hosil qilish dasturiy mahsulot yaratishdan iborat. BMI da elektron lug’at yaratish dasturi masalasini qo’yilishi quyidagi shartlarga asoslangan:

- Matnni dasturga kiritish;
- Kiritilgan matnni so’zlarga ajratish;
- Matndagi so’zlarning ro’yxatini tuzish va so’zlarning chastotasini (so’z matnda necha marta ishtirok etganligini) hisoblash;
- So’zlar ro’yxatini alfavit bo’yicha tartiblash;
- Ro’yhatdagi har bir so’zni elektron lug’atga kiritish;
- Elektron lug’atni chiqarish;

Dastur ishining algoritmi

Axborot texnologiyalar sohasida kompyuterda biron-bir masalani yechilishini ta’minlaydigan ishdir. Qo’yilgan maqsadni amalga oshirish uchun kerakli ma’lumotlar tarkibi, tuzilishi, ifodalanishi aniqlangan bo’lib, ular orasidagi bog’lanishlar aniq ifodalangan bo’lsa, masala qo’yilgan deb hisoblanadi.

Algoritmni yozish uchun psevdokoddan foydalanamiz. Dastur foydalanuvchi bilan an’anaviy iyerarxik menyu orqali muloqotda bo’ladi deb hisoblaymiz. Menyu quyidagi bandlardan iborat:

Boshlash, kiritish, matnni so'zlarga ajratish, lug'atni hosil qilish, lug'atni faylga chiqarish, tamom.

Dasturni ishga tushirib ixtiyoriy matn kiritamiz va so'zlarni ajratamiz. Bu dastur menyu bilan ishlashni amalga oshiradi.

Dastur.

Global o'zgaruvchilar initsializatsiyasi

Sarlavha va menyuni chiqarish

Boshlash

Matn faylini tanlash

Toki *matn boshi* **dan** *Matn oxiri* **gacha**

Bajarish

Agar *so'z ajralsa*

U holda *jadvalga so'z qo'shsin*

Tugatish

Toki *1-soz* **dan** *oxirgi so'z* **gacha**

Bajarish

Agar *so'z1 = so'z2*

U holda *so'z1chastota = so'z1chastota+ so'z2chastota*

Aks holda *jadvalga so'z2ni qo'shsin*

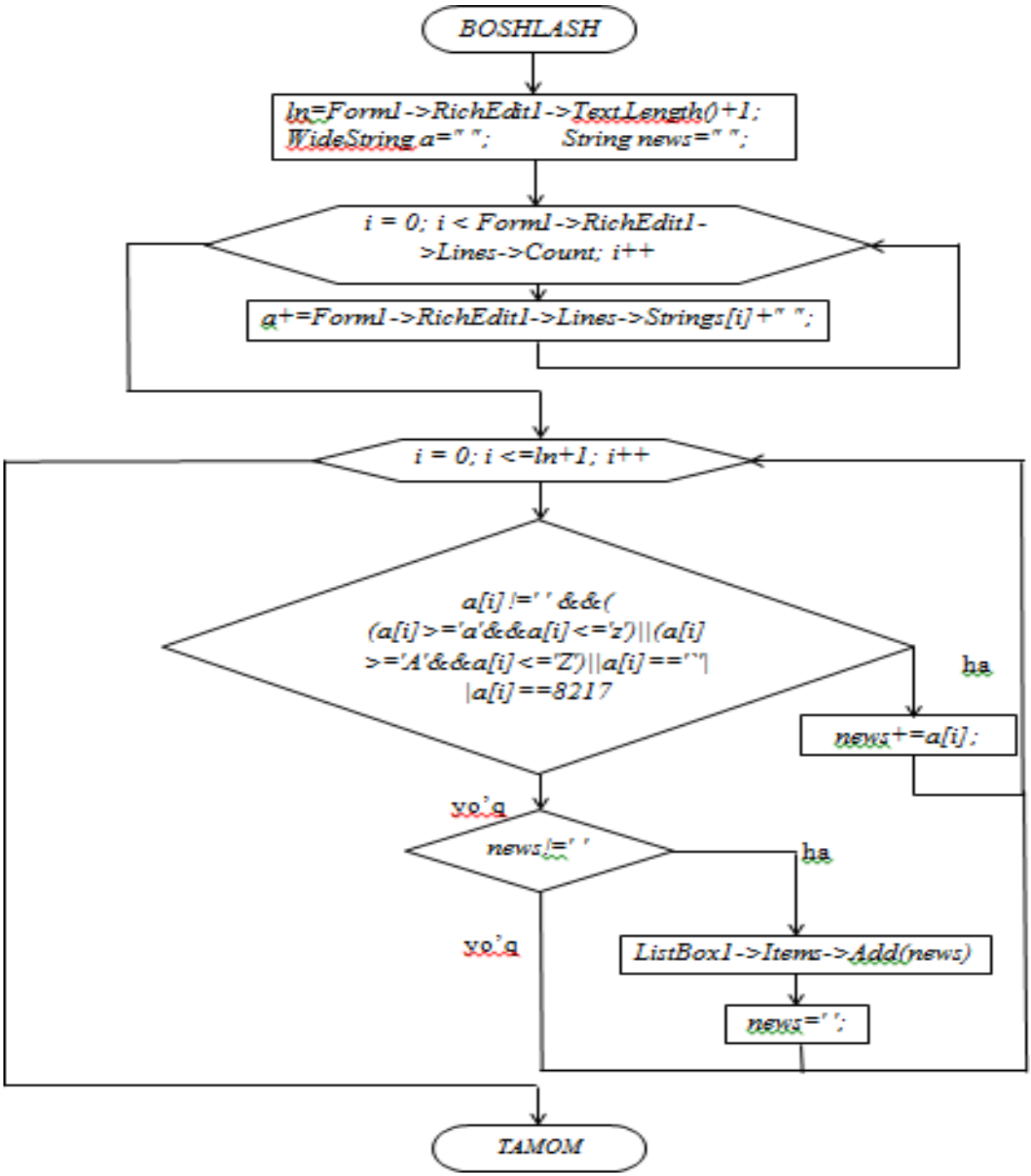
So'zlar ro'yxatini faylga yozish

Tugatish

Tamom.

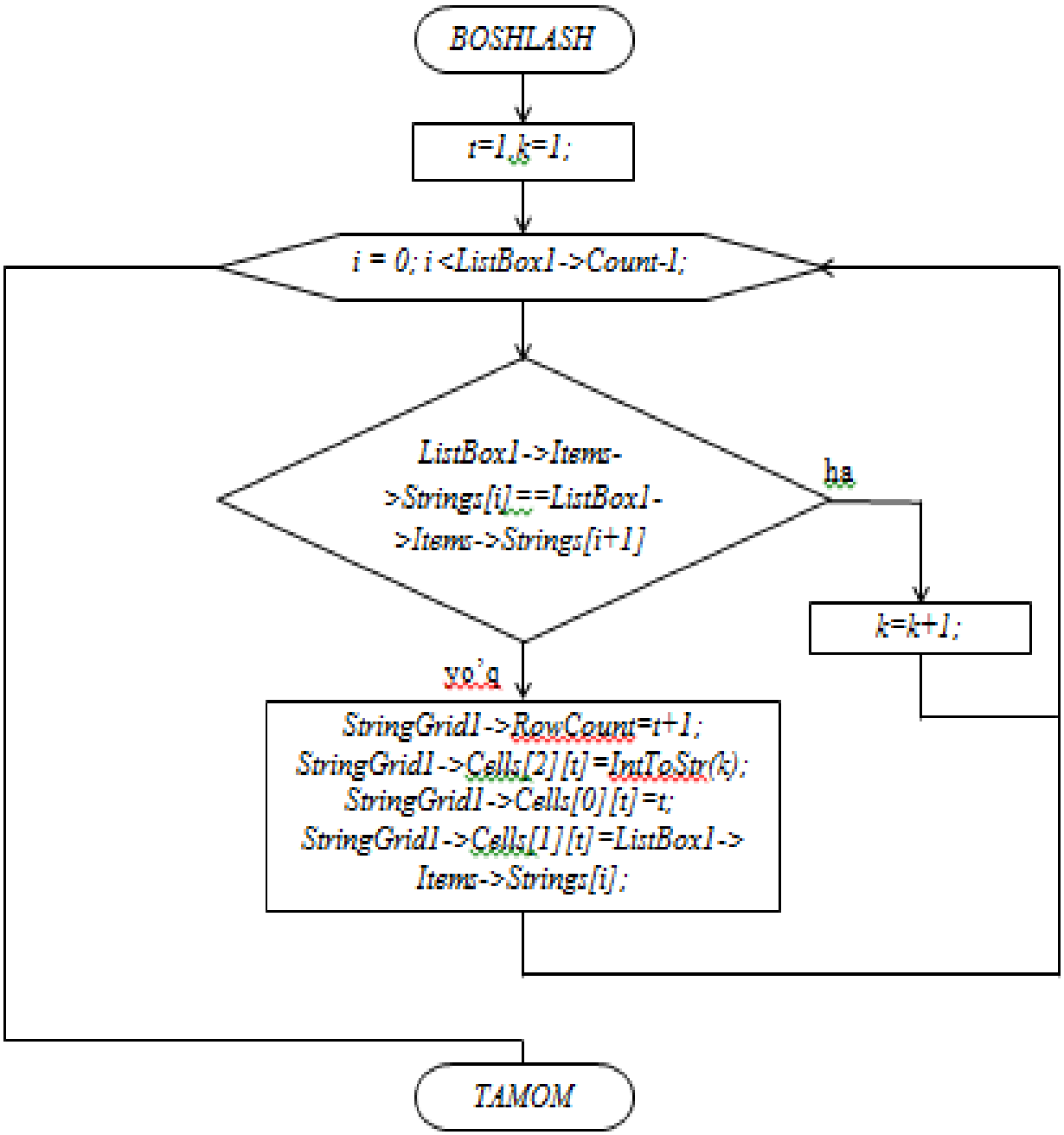
Endi qaysi fragmentlarni qismiy dastur sifatida yozish kerakligini aniqlaymiz. Birinchidan, "*Sarlavha va menyuni chiqarish*" fragmenti operatorlarning yetarlicha uzun chiziqli ketma-ketligidan iborat bo'lgani tufayli,uni alohida protsedura sifatida tashkillashtirish boshqaruvchi dasturni qisqartirishga olib keladi. "*Dasturga matn kiritish*", "*So'zlarga ajratish*", "*So'zlar ro'yxatini chastotali hosilqilish ", "Faylga chiqarish"* kabi amallar yetarlichamurakkab bo'lib, ularni alohida protseduralar sifatida yozish maqsadga muvofiq.

BMI dagi dasturning ish jarayonida dastlab matn fayldan o’qiladi va undagi so’zlar ajratilib olinib, alohida so’zlar massivi hosil qilinadi. Matndan tinish belgilari, raqamlar va so’z tarkibiga kirmaydigan boshqa xil belgilar tashlab yuboriladi. So’ngra so’zlar massivi alfavit bo’yicha tartiblanadi. Bunda takrorlanayotgan bir xil so’zlar massivda ketma-ket elementlarga joylashib qoladi va so’zlar chastotasini topish osongina bajatilishi mumkin. Uning algoritmi quyidagicha:



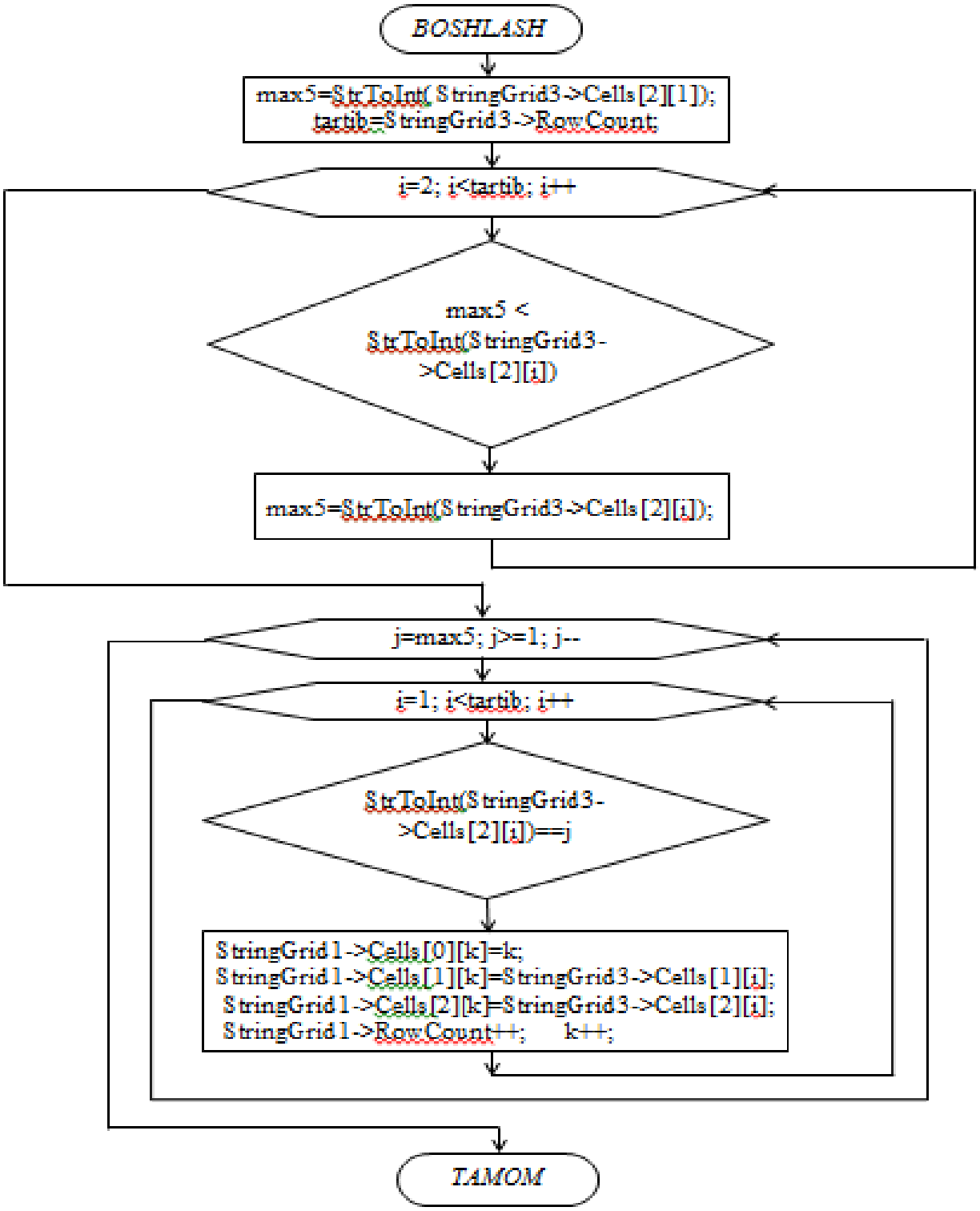
1.1-rasm. Kiritilgan matndan so’zlarni ajratib olish algoritmi.

Massivdagi har bir so’z o’zidan keyingisi bilan taqqoslanadi; agar ular bir xil bo’lsa, chastota miqdori 1 ga oshiriladi va takrorlangan so’z massivdan o’chiriladi, aks holda chastota o’zgartirilmasdan, navbatdagi so’zga o’tiladi. Bu jarayon oxirgi elementgacha davom ettirilib, massivda berilgan matnda uchragan so’zlar ro’yxati va ularning takrorlanish chastotalari hosil bo’ladi. Bu jarayonning algoritmi quyidagicha:



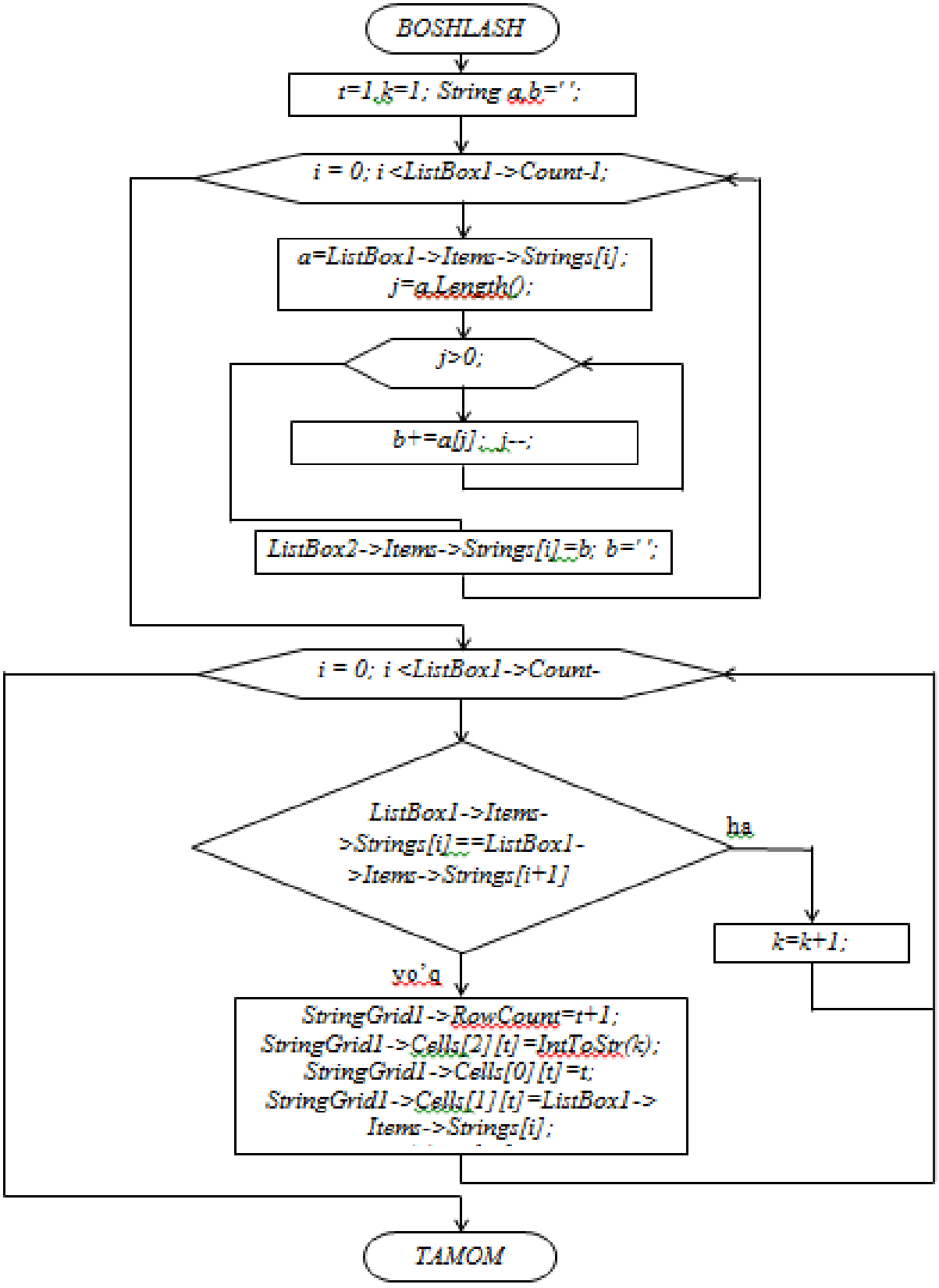
1.2-rasm. Alfavit-chastotali lug’at hosil qilish algoritmi.

Chastotali lug’at yaratish uchun mavjud lug’atdan eng katta chastotali so’z topiladi topiladi va shu chastotali so’zlar jadvalni birinchi satriga yoziladi. Undan skichik chastota topilib u ham jadvalga qo’shib boriladi. Toki chastota birga teng bo’lguncha tartiblash amalga oshiriladi. Shu tariqa chastotali lug’at yaratiladi. Uning algoritmi quyidagicha:



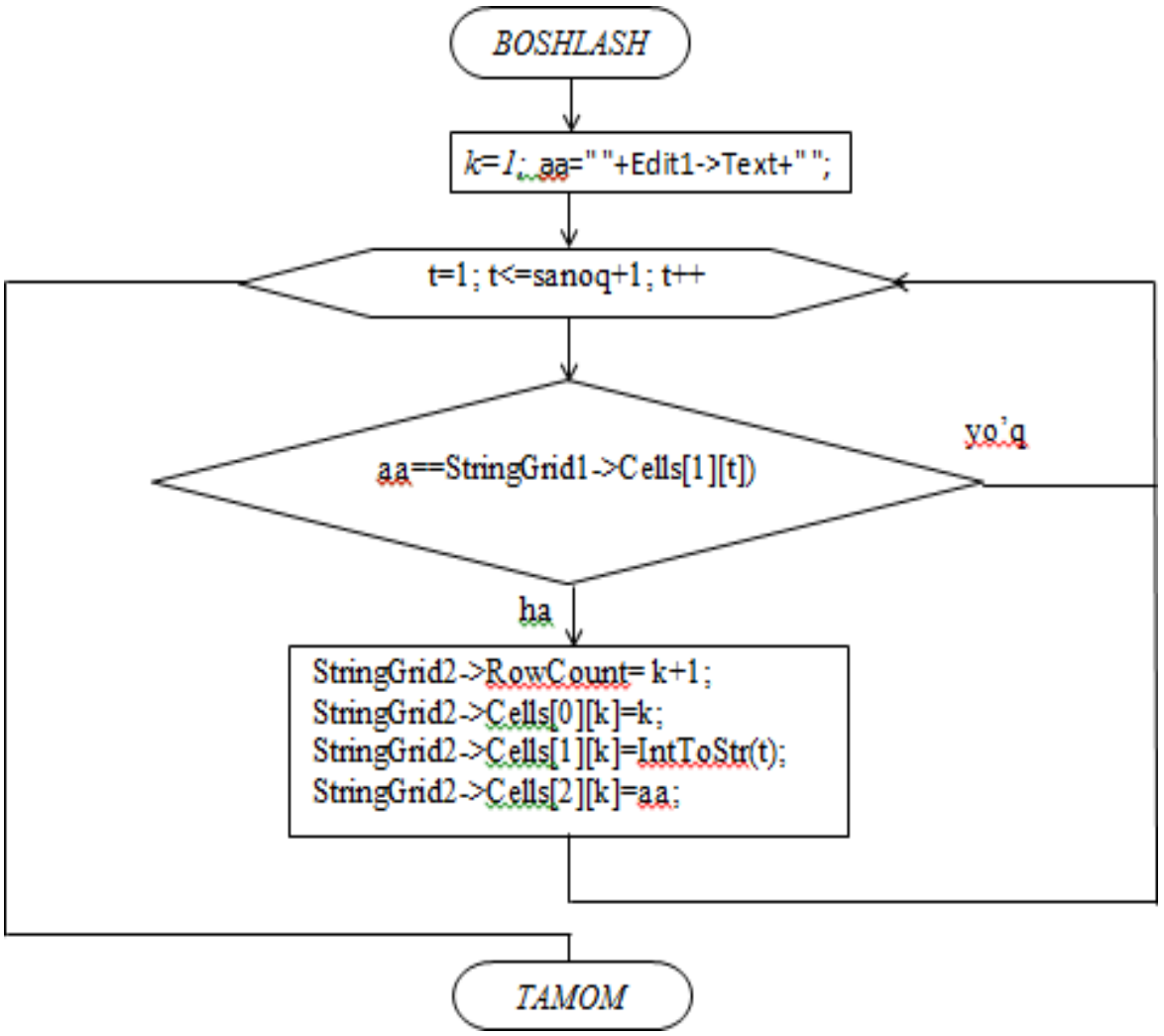
1.3-rasm. Lug’atdagi so’zlarni chastotasi kamayib borish tartibidagi algoritm.

Ters lug’at yaratish uchun har bir so’zni oxiridan boshlab o’qishga tushuniladi. Uning algoritmi quyidagicha bo’ladi:



1.4-rasm. So’zlarni teskari tartibda o’qish algoritmi.

Hosil bo’lgan elektron lug’atdan ma’lumotlarni qidirish algoritmi mavjud bo’lib, unda so’z bo’yicha qidirish mumkin yoki so’zning chastotasi bo’yicha qidirish mumkin. Topilgan ma’lumot o’zining xos parametrlari bo’yicha yangi jadvalda hosil bo’ladi. Ma’lumotlarni qidirish algoritmi quyidagicha:



1.5-rasm. Ma’lumotlarni qidirish algoritmi.

II. BO'LIM. ELEKTRON LUG'AT YARATISH VA UNING USTIDA TURLI AMALLAR BAJARISH ALGORITMLARI

2.1. Lug'at fayliga yangi elementlar kiritish

Lug'at faylini hosil qilish uchun fayldan foydalanildi. C++da fayllar bilan ishlash ftsream kutubxonasidagi biron-bir sinflar yordamida amalga oshiriladi. fstream kutubxonasi fayllarni o'qib olish uchun javob beradigan ifstream sinfiga, xamda faylga axobotning yozib olinishiga javob beradigan oftsream sinfiga ega.

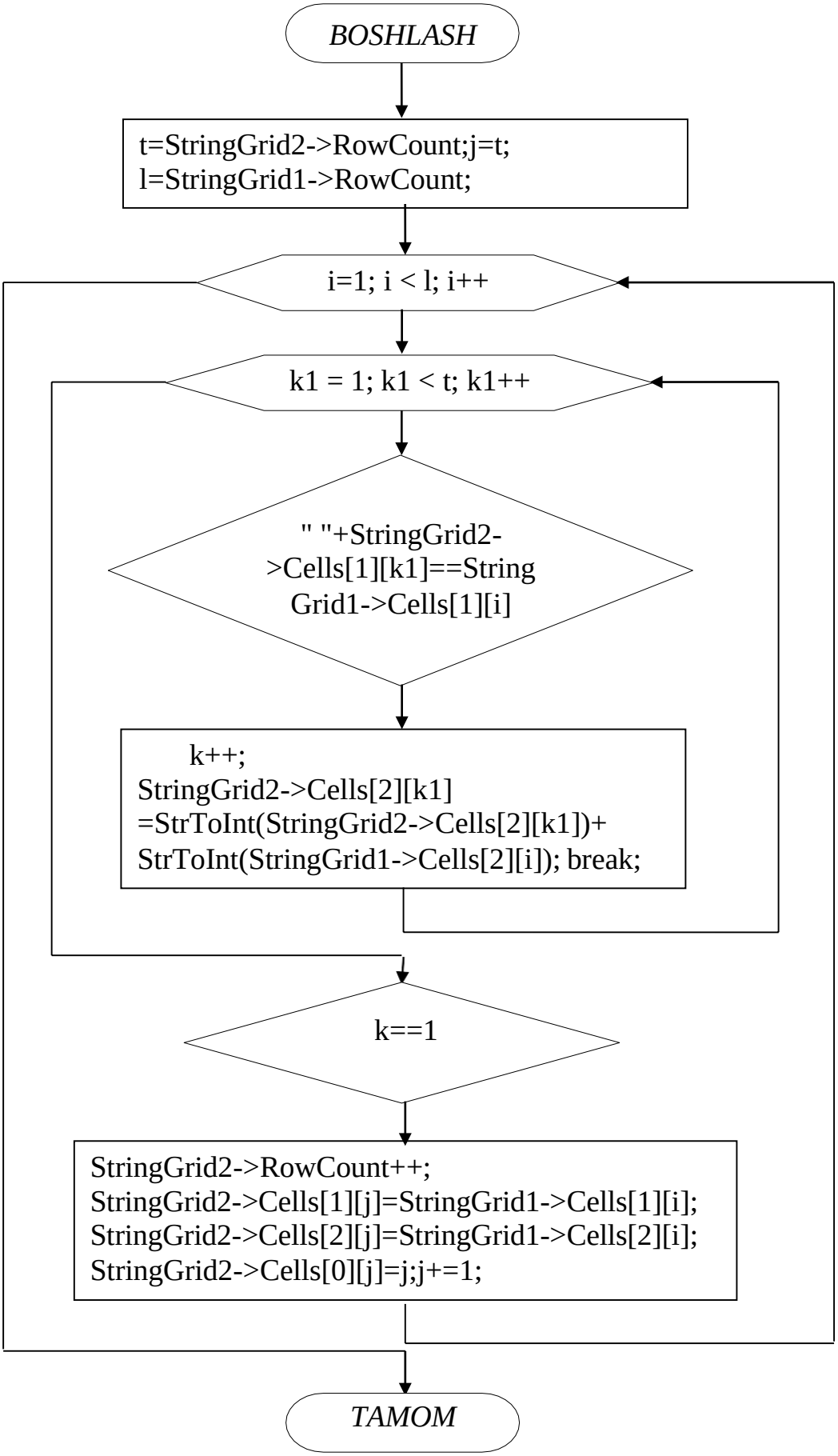
Dasturda faylni yozish uchun oftsream turdagi yoki o'qish uchun ochish iftsream turdagi o'zgaruvchini yaratish kerak. Bunday o'zgaruvchini initsiallashda fayl nomi o'zgaruvchi nomidan keyin qavs ichida berilgan belgilar massivi ko'rinishida uzatiladi [6].

Dasturda lug'at elementlarini faylga yozish uchun quyidagi algoritmidan foydalanildi. Elektron lug'at yaratish dasturiy vositada lug'atni faylga yozish uchun LL TStringList turidagi dinamik o'zgaruvchisi olindi. Jadvaldagi lug'at LL o'zgaruvchisiga ta'minlandi. Undan so'ng LL o'zgaruvchisi faylga saqlanib o'zgaruvchi o'chiriladi. Uning algoritmi quyidagicha:

```
TStringList *LL=new TStringList;
        long long int i=1;
        ProgressBar2->Max=StringGrid2->RowCount-1;
        ProgressBar2->Position=0;
         while(i<=StringGrid2->RowCount-1)
          {
          LL->Add(StringGrid2->Cells[1][i]+" "+StringGrid2->Cells[2][i]);
                i++;
                ProgressBar2->Position+=1;
          }
        LL->SaveToFile("Lugat.txt");
        delete LL;
```

Bazaga yangi element qo'shish uchun har bir so'z bazadagi so'zlar bilan solishtirib chiqiladi. Agar u so'z bazada bo'lsa chastotasini bazadagi so'z chastotasiga qo'shib qo'yadi, aks holda yangi so'z bazadan topilmasa fayl oxiridan shu so'z chastotasi bilan qo'shib yoziladi.

Uning algoritmi va dasturi quyidagicha:



2.1-rasm. Lug’at bazasiga yangi element kiritish algoritmi.

```
for (i=1; i < l; i++)  {
    for (k1 = 1; k1 < t; k1++)
       {
        if(" "+StringGrid2->Cells[1][k1]==StringGrid1->Cells[1][i]) {
             k++;
StringGrid2->Cells[2][k1]=StrToInt(StringGrid2-
>Cells[2][k1])+StrToInt(StringGrid1->Cells[2][i]) ;
              }
        if( k==1){
          StringGrid2->RowCount++;
               StringGrid2->Cells[1][j]=StringGrid1->Cells[1][i];
         StringGrid2->Cells[2][j]=StringGrid1->Cells[2][i];
         StringGrid2->Cells[0][j]=j;j+=1;
        }
        }
t=StringGrid2->RowCount;
j=t;

l=StringGrid1->RowCount;
```

2.2. Lug’at elementlarini chiqarish

Fayldan elementlarni chiqarish. Axborotni fayldan o‘qib olish uchun ‘’>>’’ operatoriga ekvivalent bo‘lgan get funksiyasi qo‘llanadi. Put funksiyasikabi, get funksiyasi xam har qanday o‘zgaruvchilarning standart turlari yoki / va belgilar massivlari bilan ishlay oladi. SHuningdek get ga har jixatdan ekvivalent bo‘lgan getline funksiyasi mavjud: farqi faqat shundaki, getline funksiyasi satr oxiridan oxirgi belgini qaytarmaydi [6].

```
ifstream ofI (’’C:\text.txt’’);
char s; char ss[9];
s=ofI.get ();
cout<<s;
ofI.get(s);
cout<<s;
ofI.getline(ss,9);
cout<<ss;
ofI>>ss;
cout<<ss;
```

Fayl oxirini aniqlash. Fayl ichidagisini, fayl oxiri uchramaguncha, o‘kish dasturdagi oddiy fayl operatsiyasi xisoblanadi. Fayl oxirini aniqlash uchun,

dasturlar oqim ob’ektining eof funksiyasidan foydalanishlari mumkin. Agar fayl oxiri xali uchramagan bo‘lsa, bu funksiya 0 qiymatini qaytarib beradi, agar fayl oxiri uchrasa, 1 qimatini qaytaradi. Whilessiklidan foydalanib, dasturlar, fayl oxirini topmagunlaricha, qo‘yida ko‘rsatilganidek, uning ichidagilarini uzluksiz o‘qishlari mumkin:

```
while (! Input_file.eof())
{
  //Operatorlar
}
```

Ushbu xolda dastur, eof funksiyasi yolg‘on (0) ni qaytarguncha,ssiklni bajarishda davom etadi [7].

Xuddi shunday, keyingi dastur - WORD_EOF.CPP fayl ichidagisini bitta so‘z bo‘yicha bir martada, fayl oxiri uchramaguncha, o‘qiydi:

```
#include <iostream.h>
#include <fstream.h>
void main(void)
{
  ifstream input_file("BOOKINFO.DAT");
  char word[64] ;
  while (! input_file.eof())
  {
    input_file >> word;
    cout << word << endl;
  }
}
```

Elektron lug’at yaratish dasturiy vositada lug’atni faylda yozish uchun struktura yaratilib olinadi. Fayldan struktura orqali jadval to’ldiriladi. Uning algoritmi quyidagicha:

```
struct zapis{
        char suz[30];
        int chast;
};
//-------------------------------------------------
zapis soz;   long long i=1;
 ifstream fayl;
 fayl.open("Lugat.txt");
 StringGrid2->RowCount=2;
 ProgressBar1->Max=StrToInt(Label1->Caption);
 ProgressBar1->Position=0;
```

```
while(!fayl.eof())
 {
            fayl>>soz.suz>>soz.chast;
            StringGrid2->Cells[0][i]=i;
            StringGrid2->Cells[1][i]=soz.suz;
            StringGrid2->Cells[2][i]=soz.chast;
            StringGrid2->RowCount++;
            ProgressBar1->Position+=2;
            i++;
}
fayl.close();
```

2.3. Elektron lug‘at faylini tartiblash algoritmlari

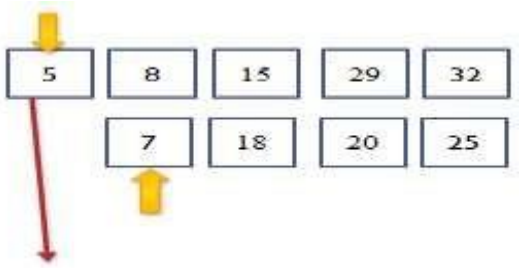
Saralash algoritmi turli xil bo’lib dasturda MergeSort saralash algoritmidan foydalanilgan. Bu algoritmdan foydalanilganidan maqsad MergeSort algoritmi katta hajmdagi massivlarni saralaganda samaradorligi qolganlariga nisbatan yuqoriroq.

Birlashtirish orqali (MergeSort) saralash. MergeSort O(NlogN) da ishlaydigan optimal saralash algoritmlaridan biri. Eng yaxshi holatda ham, eng yomon holatda ham O(NLogN) assimptotikada ishlaydi. Hamda, u barqaror saralaydi, ya'ni sonlar ketma-ketligini buzmagan holda. Masalan, sonlardan tashqari, ularning kalit sonlari bo'lsa, kalit soni 3 bo'lgan 2 soni kalit soni 5 bo'lgan 2 sonidan oldin kelsa, MergeSortda saralangandan so'ng ham kalit soni 3 bo'lgan 2 soni oldin keladi. QuickSortda doim ham bu shart bajarilavermaydi.

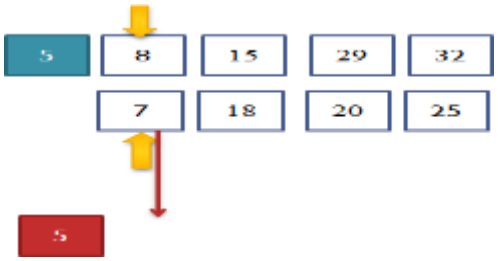
Algoritmning ishlash prinsipi quyidagicha: Massivni teng ikkiga bo'lamiz. O'ng tomonni alohida saralaymiz, chap tomonni alohida. So'ng ikki saralangan qismni O(n) ya'ni n ta operatsiyada qo'shib, to'liq saralangan massiv olamiz. Aynan shu operatsiya, ya'ni qo'shish - Merge ning hisobiga algoritm shunday nom olgan. Endi chap va o'ng tomonlarini saralash uchun ham, ularni ikkiga bo'libxuddi shu ishni bajaramiz va hk. Massivni bo'lishni to unda 2 ta element qolgunchadavom ettiramiz [14].

Faraz qilaylik, bizda Merge(a: Array of Integer, Left, Mid, Right: Integer) funksiyasi bo'lsin. Ya'ni, a massivning L -- Mid qismi saralangan, Mid + 1 -- R

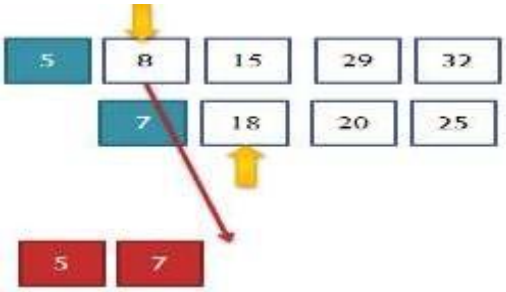
qismi saralangan. Shu qismlarni qo'shib, L -- R kesmada saralangan qilish. Bunda Mid L va R kesma o'rtasi.



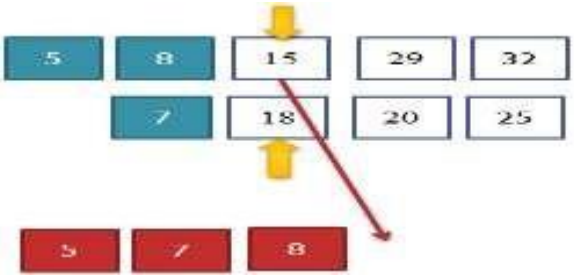
2.2-rasm.



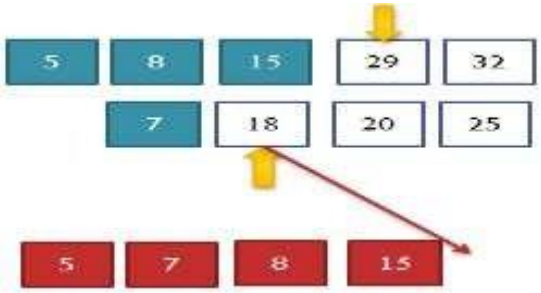
2.3-rasm.



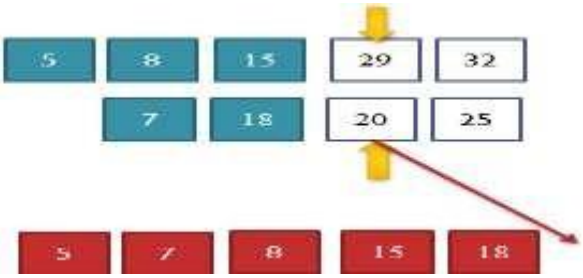
2.4-rasm.



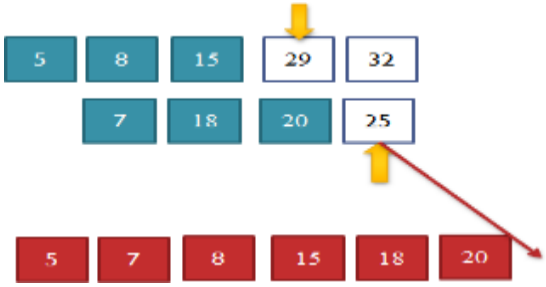
2.5-rasm.



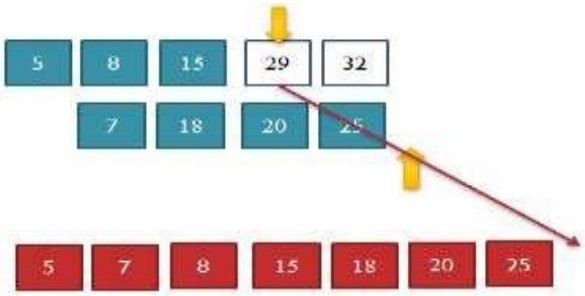
2.6-rasm.



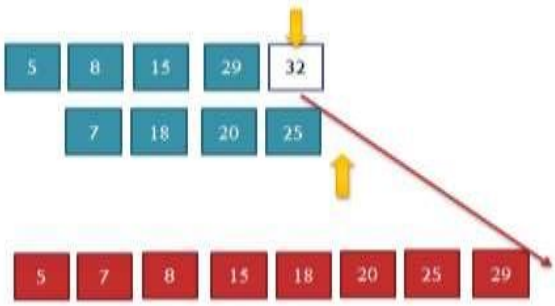
2.7-rasm.



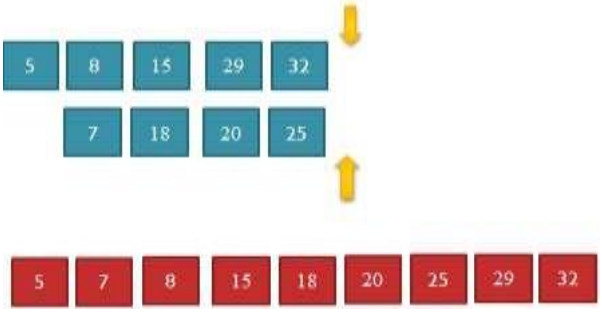
2.8-rasm.



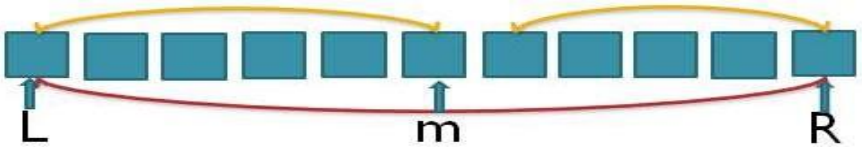
2.9-rasm.



2.10-rasm.

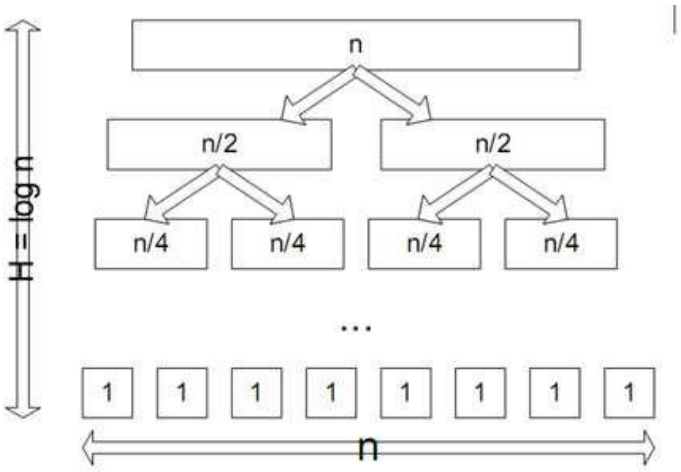


2.11-rasm.



2.12-rasm.

[L, R] oraliq m=(L+R) / 2 o’rtasi orqali ikkita [L, m] va [m+1, R] oraliqqa ajratiladi va ular alohida saralanadi.



2.13-rasm.

Shunday Merge funksiya asosida MergeSortni quyidagicha yozish mumkin.

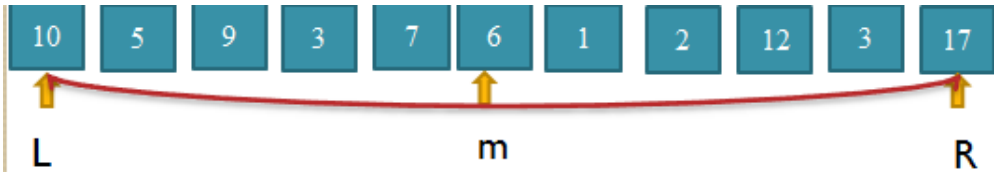
```
#include <iostream>
using namespace std;
int helper[1000001];
void Merge(int a[],int left,int middle, int right){
        for (int i=left,j=middle,k=left;k<right;k++){
           if (i==middle) { helper[k]=a[j++]; continue;}
           if (j==right ) { helper[k]=a[i++]; continue;}
           helper[k]= ( a[i] < a[j] ) ? a[i++] : a[j++];
        }
        for(int k=left;k<right;k++)a[k]=helper[k];
}
void MergeSort(int a[],int left,int right){
     if (right-left==1)  return ;
     int middle=(left+right)>>1;
     MergeSort(a,left,middle);
     MergeSort(a,middle,right);
     Merge(a,left,middle,right);
}
int main(){
    int a[10000] , n , i;
    cin>>n;
    for(i=0;i<n;i++)  cin>>a[i];
```

```
    MergeSort(a,0,n);
    for (i=0;i<n;i++)   cout<<a[i]<<" ";
    return 0;
}
```

Tezkor saralash (Quick Sort) algoritmi. Saralashning bir nechta effektiv algoritmlari topilgan. Ulardan biri bu QuickSort() bo'lib (tez saralash) u quyidagicha ishlaydi.

1964 yilda Charlz Hoar tamonidan taklif qilingan. Charlz Hoar ingliz olimi, informatika va hisoblash texnikasi sohasida yetuk mutaxassis. Uning “Tezkor saralash” algoritmi saralash bo’yicha eng ommobop algoritm [14].

Massivning o'rtasidagi element olamiz va uni x deymiz. Endi, barcha x dan katta sonlarni x dan o'ng tomonga, kichiklarini chap tomonga o'tkazamiz. Keyin shu ishni x dan chap tomondagi (x dan kichkina sonlar) uchun va x dan o'ng tomondagi (x dan katta sonlar) uchun qilamiz. Bu ishni rekursiv qilgan yaxshiroq (programma qisqa chiqadi).



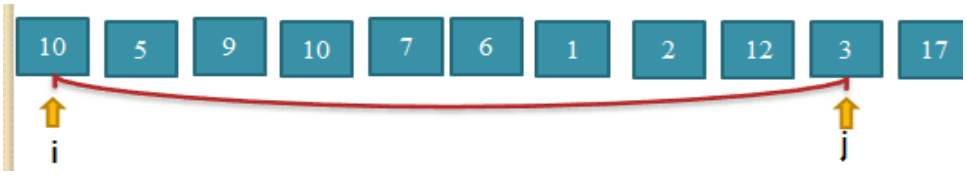
2.14-rasm.

[L, R] saralanishi kerak bo’lgan oraliq.

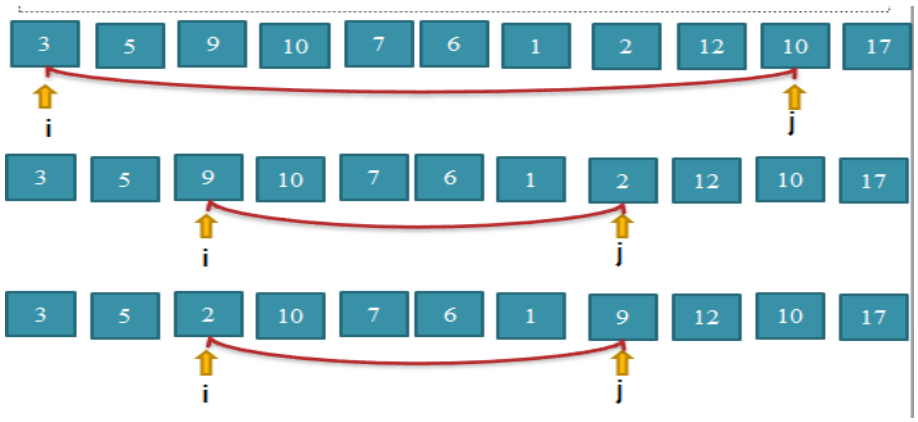
m = (L+R) / 2 – kesmanig o’rtasi.

X=a[m] - bo’luvchi element.

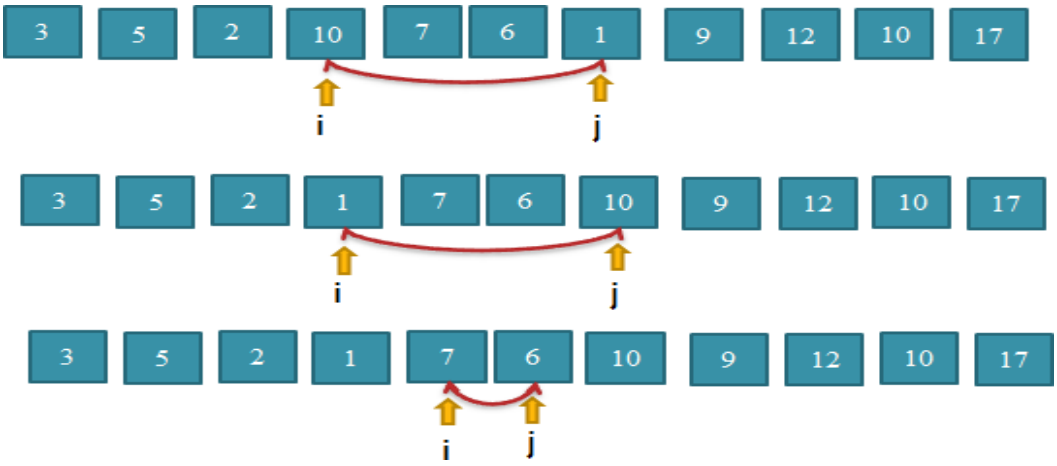
Chap tamondan X dan kichik katta yoki teng bo’lgan birinchi elementni topamiz. O’ng tamondan X dan katta bo’lmagan birinchi elementni topamiz. Ikkalasining qiymatini almashtirzmiz [2.15-2.18-rasmlar].



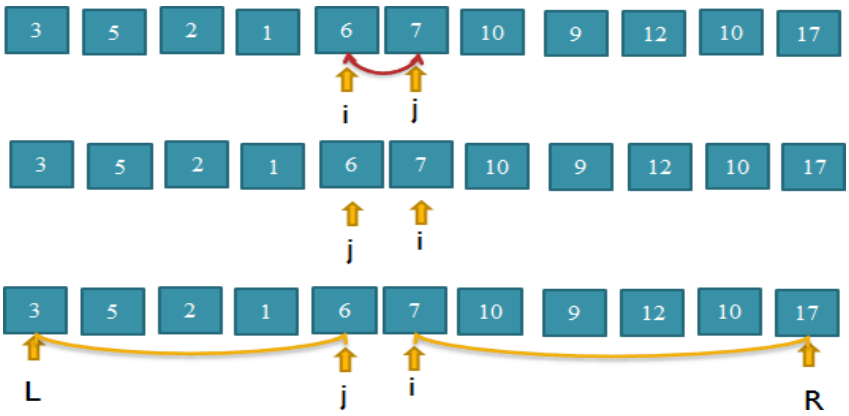
2.15-rasm.



2.16-rasm.



2.17-rasm.



2.18-rasm.

Algoritm assimptotikasi - o'rtacha O(NlogN). Eng yomon holatda O(N^2) Bunday holatda pand yemaslik uchun, ya'ni O(N^2) da ishlaydigan test bersa, Quicksortni random variantini ishlatgan ma'qul. Ya'ni, har safar kesmadagi ixtiyoriy elementni x deb olamiz, bunda algoritm tezligi 5-10 % pasayishi mumkin, lekin bu vaqtga uncha ta'sir qilmaydi:

```
#include <stdio.h>
int a[1000000];
void qsort(int left, int right)
{
    int l = left, r = right, m = a[(l+r) / 2], tmp;
    while(l <= r){
          while( a[l]<m)l ++;
          while(a[r]>m)r --;
          if(l<=r){
            tmp = a[l];
            a[l] = a[r];
            a[r] = tmp;
            l ++ ;
            r -- ;
          }     }
    if(l<right) qsort(l,right);
    if(left <r) qsort(left, r);
}
int main()
{
   int n,i;
   cin>>n;
   cin>>a[i];
   qsort(0,n-1);
   cout<<a[i]<<" ";
   return 0;   }
```

Shell usuli. 1959 yilda D.L.Shell tomonidan taklif etilgan va keng foydalaniladigan bu usul juda kam xotira talab etadi va saralashda yuqori tezlikni ta’minlaydi. Usul qo’yish usuli kabi elementlarni solishtirish va joyini almashtirishdan foydalanadi, lekin undan farqli o’laroq solishtirishda qo’shni elementlar emas, balki bir-biridan muayyan masofada bo’lgan elementlar solishtiriladi. Almashtirish zaruriyati tug’ilganida elementlar qo’yish usulidagi kabi bitta pozitsiyaga emas, shu masofaning o’ziga sakrab o’tadi []8.

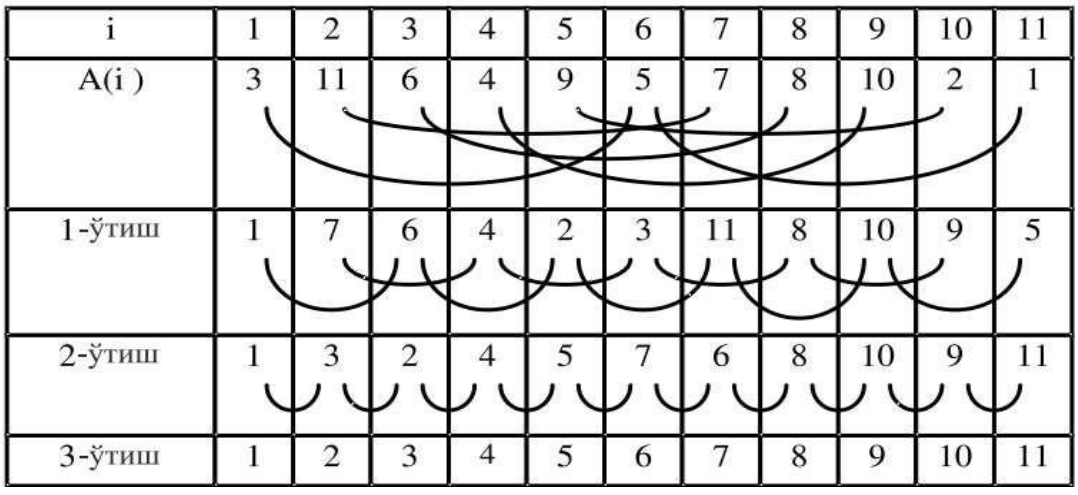
Yuqoridagi usul bilan saralash uchu N elementdan iborat ketma-ketlik N/2 yoki N toq son bo’lsa (N-1)/2 guruhga bo’linadi. Har bir guruh ikki elementdan iborat bo’ladi. Agar elementlar soni toq bo’lsa, bir qismi uch elementdan iborat bo’ladi. Bitta guruhga mansub elementlar bir-biridan N/2 pozitsiyada joylashadi. Bu masofa **qadam** deb ataladi. 2.19-rasmda dastlabki ketma-ketlik A ning o’n bitta

elementi beshga teng qadam bilan beshta guruhga bo'linadi. Bitta guruhga mansub elementlar qavslar bilan birlashtiriladi.

Birinchi o'tish davomida har bir guruh elementlari qo'yish usuli bilan tartibga solinadi. 18-rasmdagi misolga murojaat etamiz. Birinchi o'tish natijasida birinchi guruh elementlarining kelish tartibi o'zgartiriladi. Element birinchi pozitsiyani egallaydi, 3 va 5-elementlar o'ng tomonga suriladi va tegishlicha oltinchi hamda o'n birinchi pozitsiyalarni egallaydi. Shuningdek ikkinchi guruh elementlari (21 va 7) va beshinchi guruh elementlari (9 va 2) joy almashadi.

Keyingi har bir o'tishni amalga oshirish uchun Shell oldingi qadam (kasr sonlarda uning butun qismi olinadi)ning yarmiga teng bo'lgan qadam belgilashni taklif etdi. Bunday holda ko'rib chiqilayotgan misolimiz uchun guruh elementlari o'rtasidagi qadam ikkinchi o'tishda ikkiga teng.

Ikkinchi o'tishda ikki guruh elementlari tartibga solinadi: 1, 6, 2, 11, 10, 5 elementlaridan iborat bo'lgan birinchi guruh va 7, 4, 3, 8, 9 elementlaridan iborat bo'lgan ikkinchi guruh. Ikkinchi o'tish natijasida bu guruhning elementlari ularning qiymati oshib borishi bo'yicha tartibga solingan bo'ladi.



2.19-rasm. Shell usulida saralash misoli

Uchinchi o'tish uchun 1 ga teng bo'lgan qadam belgilanadi va yagona guruh tartibga solinadi. Juft solishtirishlar va almashtirishlar natijasida dastlabki ketma-ketlik uchinchi o'tishdan so'ng to'la tartibga solngan bo'ladi.

N elementdan iborat ketma-ketlikni saralash uchun log2N ga yaqin o'tishlar talab etiladi. Shell usulida saralash uchun zarur bo'lgan solishtirishlar soni

qadamga juda bog'liqdir. Shu vaqtgacha qadamlarning ketma-ketligini qanday tanlash zarur degan masala muhokama qilib kelinmoqda. Shellning o'zi tomonidan N/2, N/4, N/8 va hokazo ketma-ketlik taklif etilgan. Solishtirishlar sonini baholash Nlog2N formula bo'yicha amalga oshiriladi [8].

```
void shellSort(int numbers[], int array_size) {
 int  i,j, increment, temp;
increment = 3;
while (increment > 0)
{
for( i =0;  i< array_size; i++ )
 {
j=i;
temp = numbers[i];
whie ((j>=increment) && (numbers[j-increment]>temp))
   {
     numbers[j] = numbers[j-increment];
    j = j-increment;
    }
        numbers[j] = temp;
}
if ( increment/2 != 0)
increment = increment/2;
else if (increment == 1)
increment = 0;
else
increment = 1;
}  }
```

Saralash usullarini tanlashda hisobga olinadigan omillar. Ko'rib chiqilgan saralash usullari turli shaklda bo'lishi mumkin; ulardan har birini amalga oshirishda turli protseduralar zarur bo'ladi. Masalan almashtirish usulining faqat bitta modifikatsiyasi – pufakcha usuli bayon etilgan. Bu usulning boshqa modifikatsiyalari ham bo'lishi mumkin. Elementlarni qo'yish yo'li bilan saralash mohiyat jihatidan tartibga solingan ketma-ketlikka yangi elementlarni ketma-ket qo'yishga asoslangan saralash usullari guruhning umumiy nomlanishidir. Ko'rib chiqilgan usulning prinsipini to'la ko'rsatib beradigan chiziqiy qo'yishdan tashqari yana markazlashgan va ikkilangan qo'yishlar ham mavjuddir.

Bir qator monografiyalar hamda maxsus tadqiqotlar saralashning turli usullari hamda algoritmlarini ko'rib chiqish va baholashga bag'ishlangan. Odatda, kompyuterning asosiy xotirasida amalga oshiriladigan saralash ko'p vaqt talab etmaydi va aksariyat hollarda ko'rib chiqilgan usullarda istalganidan foydalaniladi.

Lekin ba'zida muayyan talablarga javob beradigan saralash usulini tanlash yoki ishlab chiqish zaruriyati yuzaga keladi. Bunday vaziyat OXning bo'sh hajmiga qat'iy cheklashlar qo'yilgan hollarda, shuningdek saralanadigan ma'lumotlarning tavsiflari qandaydir oddiy bo'lmagan tarzda qo'shilishi natijasida odatdagi yaxshi usullardan foydalanish uncha samara bermaydigan holatlarda yuzaga kelishi mumkin. Bunday vazifani hal qilish uchun saralash samaradorligiga ta;sir qilishi mumkin. Bo'lgan omillarni tahlil qilish va tanlangan usulni test dasturlarida sinab ko'rish zarur.

Tashqi saralash dasturini ishlab chiqish ancha murakkab mustaqil vazifa bo'lib, samarali saralash-qo'shish dasturlarini yaratish bilan tor soha mutaxassislari shug'ullanadilar. Odatda, EHM ning matematik ta'minoti tarkibida tayyor tashqi saralash paketi mavjud bo'ladi. Yoki uni sotib olish mumkin. Faqat dastlabki ma'lumotlar yoki muayyan kompyuterning konfiguratsiyasiga bog'liq bo'lgan u yoki bu sabablarga ko'ra tayyor paketdan foydalanish mumkin bo'lmagan hollarda mustaqil ravishda tashqi saralash dasturini ishlab chiqishga to'g'ri keladi.

Saralashning u yoki bu usulini tanlash va baholashda hisobga olinishi zarur bo'lgan asosiy omillarni ko'rib chiqamiz [8].

Saralanadigan massivning o'lchami. Saralanishi zarur bo'lgan massivdagi yozuvlar miqdorini baholash tashqi saralash kerakligi yoki kerakmasligini, ya'ni saralash uchun OXning bo'sh hajmi yetarli ekanligini aniqlash imkonini beradi. Bunda xotiraning eng kichik hajmidan foydalanuvchi usullarni qo'llash zaruriyati ham aniqlab olinadi [8].

Kalit uzunligi. Kalitning uzunligi va yozuv chegarasidagi joylashgan o'rin solishtirish operatsiyalarini bajarish uchun zarur vaqtni belgilab beradi. Bunda kalit erkin foydalanish mumkin bo'lgan yozuv maydoni ekanligini yoki kalitni yozuvdan chiqarib olish uchunqo'shimcha tadbirlar, masalan "maskirovkalash"

zaruriyatini aniqlab olish kerak. Keying holda kalitni chiqarib olish uchun sarflanadigan vaqtni baholash zarur

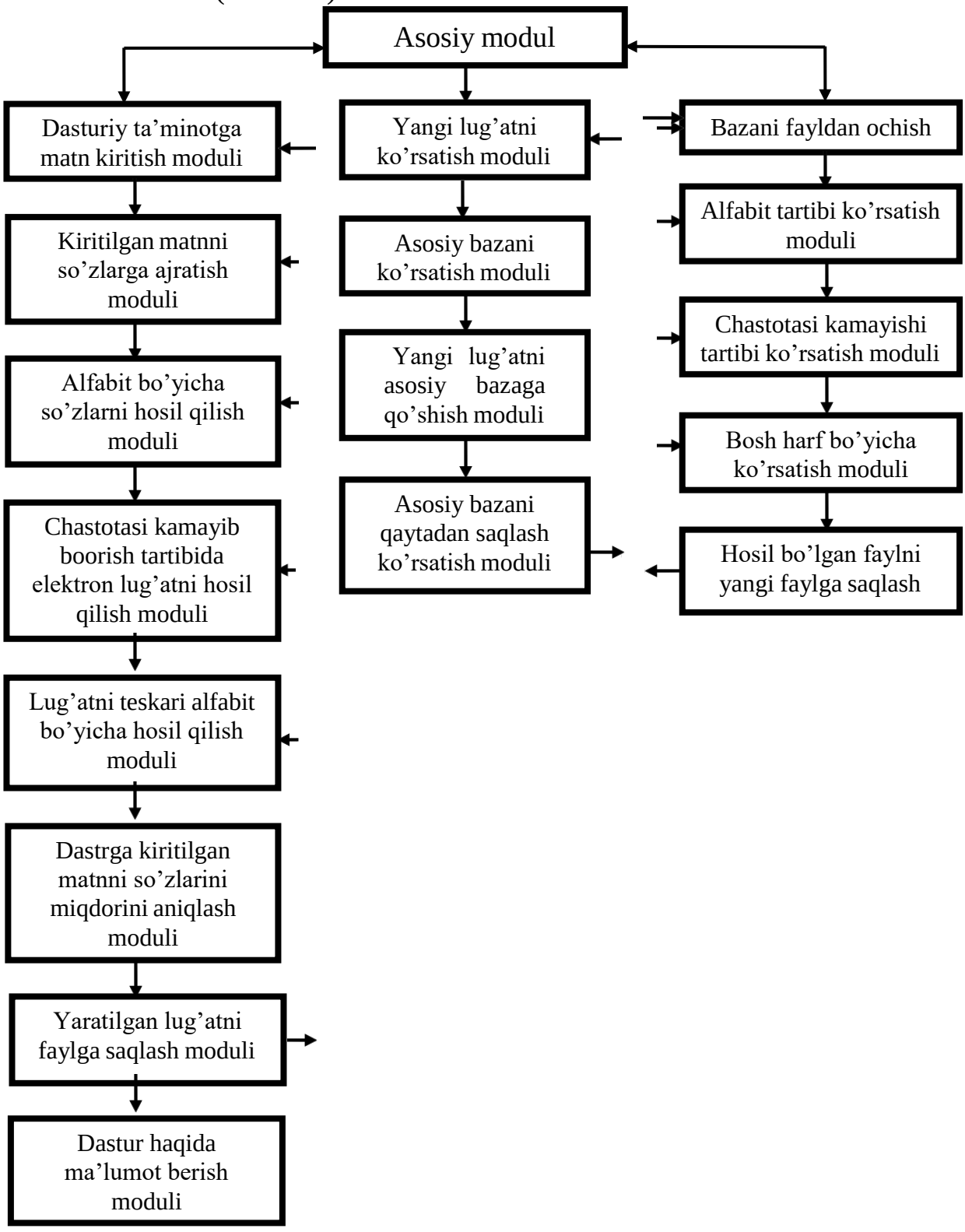
Kalit mashina bilan bevosita ishlov beriladigan tuzilmaga to’g’ri keladimi yo’qmi, buni bilish ham juda muhimdir. Balki kalitga oldindan ishlov berish va uni ancha qulayroq shaklda taqdim etish zaruriyati yuzaga kelishi mumkin [8].

Kalitlar turi. Solishtirish vaqti ma’lumotlarning ichki taqdim etilishiga va ularni turli tiplarini solishtirish buyruqlarining mavjudligiga bog’liq. Masalan, agar kompyuterda “o’nliklar bo’yicha solishtirish” buyrug’i bo’lmasa, kalitni ikkitali kodga aylantirishga to’g’ri kelishi mumkin. Kalitlarni o’zgartirish vaqti saralash vaqtiga qo’shiladi [8].

III.BOB. DASTURIY MAJMUA TAVSIFI VA UNDAN FOYDALANISH TARTIBI

3.1. Dasturiy majmuaning strukturasi

Dasturiy majmuaning funksional strukturasi 3 ta asosiy va 18 ta qo'shimcha modullardan iborat(3.1-rasm):



3.1-rasm. Elektron lug'at hosil qilish tuzilmaviy sxemasi.

1. **Asosiy modul** – barcha modullarni boshqaradi.

2. **Dasturiy ta’minotga matn kiritish moduli -** Dasturga matn kiritish uchun foydalaniladi va asosiy oynada kiritilgan matn ko’rinadi.

2.1. **Kiritilgan matnni so’zlarga ajratish moduli -** Bu modul orqali kiritilgan matn so’zlarga ajratib olinadi va so’zlar alfavit tartibi bo’yicha aks ettiriladi.

2.2. **Alfavit bo’yicha so’zlarni hosil qilish moduli** - Ajratilgan so’zlar chastotalarini hisoblab alfavit bo’yicha tartiblab beradi.

2.3. **Chastotasi kamayib borish tartibida elektron lug’atni hosil qilish moduli** – so’zlarni matnda eng ko’p ishtirok etgani bo’yicha kamayib boorish tartibida tashkil etiladi.

2.4. **Lug’atni teskari alfavit bo’yicha hosil qilish moduli** –Lug’atdagi har bir so’zni teskarisiga o’qiydi va alfavit tartibi bo’yicha tartiblash vazifasini bajaradi.

2.5. **Dasturga kiritilgan matnni so’zlarini miqdorini aniqlash moduli –** Matndagi nechta so’z borligini aniqlab beradi va ishtirok etgan so’zlar hajmini hisoblab berish moduli hisoblanadi.

2.6. **Yaratilgan lug’atni faylga saqlash moduli -** Yangi yaratilgan lug’atni bazadan tashqari boshqa faylga saqlaydi.

2.6. Dastur haqida ma’lumot berish moduli **-** Yangi yaratilgan lug’atni bazadan tashqari boshqa faylga saqlaydi.

3. **Yangi lug’atni ko’rsatish moduli –** Bu modulda yangi lug’at o’zining harakteristikasi (Alfavit, chastotali, ters) bo’yicha ko’rinadi.

3.1. **Asosiy bazani ko’rsatish moduli –** Lug’at bazasini ochadi.

3.2. **Yangi lug’atni asosiy bazaga qo’shish moduli –** Bunda yangi lug’atni asosiy lug’atni davomidan qo’shib boradi.

3.3. **Asosiy bazani qaytadan saqlash ko’rsatish moduli** – Hosil bo’lgan lug’atni bazaga qaytadan saqlaydi.

4. **Bazani fayldan ochish –** Chop etishga bergan buyruq holatiga ko’ra bazadan lug’atni hosil qilib beradi.

4.1. **Alfavit tartibini ko’rsatish moduli -** Lug’atni alfavit bo’yicha ekranga chiqarib beradi.

4.2. **Chastotasi kamayishi tartibi ko’rsatish moduli -** Lug’atni chastotasi kamayishi tatribida lug’atni hosil qiladi.

4.3. **Bosh harf bo’yicha ko’rsatish moduli -** Bu modulda bitta harfni tanlash kerak va tanlangan harf bo’yicha bazadagi lug’atni bosh harfi bo’yicha chiqarib beradi.

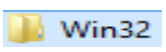
4.4. **Hosil bo’lgan faylni yangi faylga saqlash –** Hosil bo’lgan lug’atni yangi faylga saqlaydi.

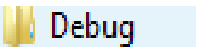
3.2. Dasturiy majmuaning tavsifi.

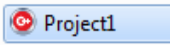
Ish stolning 

papkasida dasturiy majmaning modullari va ularning Embarcadero XE3 C++ Builder dasturlash muhitida tuzilgan kodlari joylangan(3.2 – rasm):

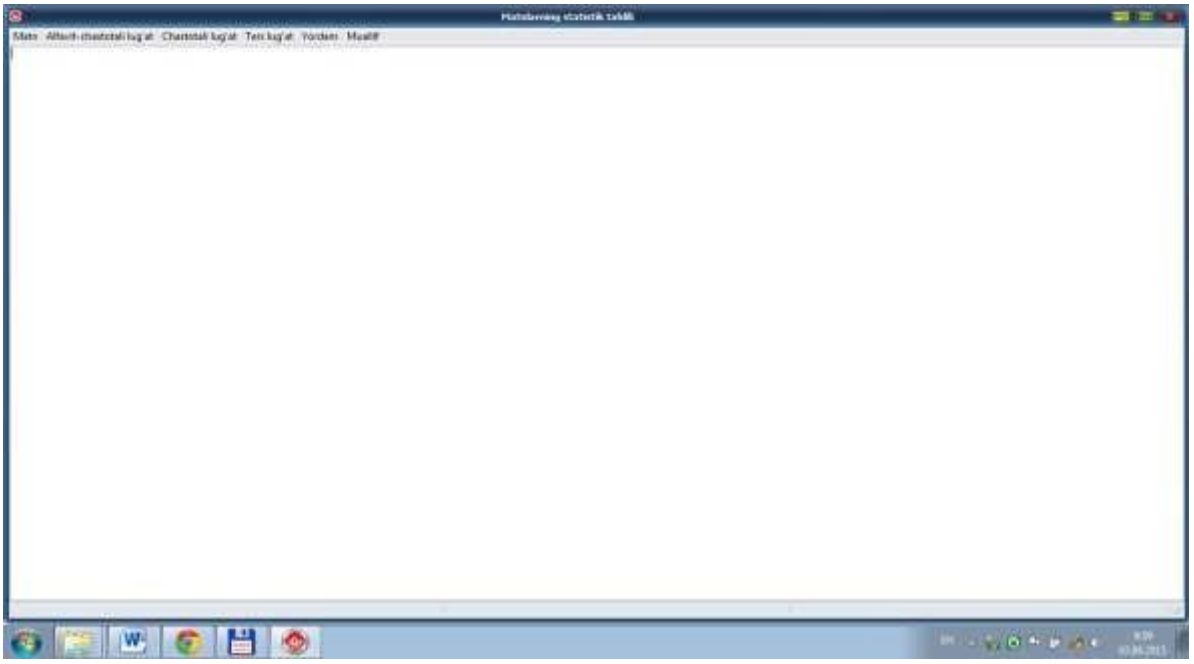
3.2 – rasm. Dasturni ishga tushirish.

3.2 – rasmdagi papkaning ichida 

va 

papkalariga ketma-ket kiriladi va dasturiy modulning 

fayli ishga tushiriladi. Natijada ekranda quyidagi bosh oyna hosil bo’ladi (3.3 – rasm):

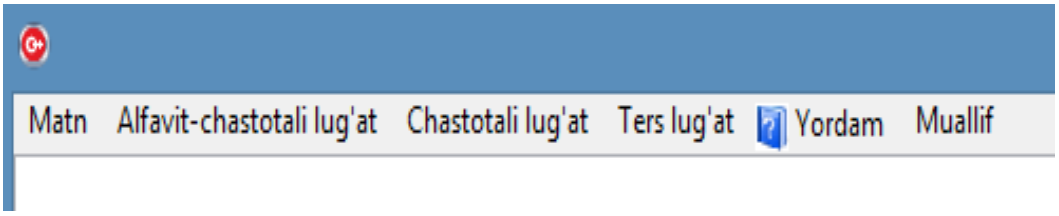


3.3 – rasm. Dasturning asosiy oynasi.

3.3 – rasmdagi oyna menyu va maydon bo’limidan iborat. Kiritilgan matn maydonda hosil bo’ladi. Kiritilgan matn bo’yicha amallar bajarilish menyu bo’limida amalga oshiriladi.

3.4 – rasmdagi bosh oynaning menyu paneli quyidagilardan tashkil topgan.

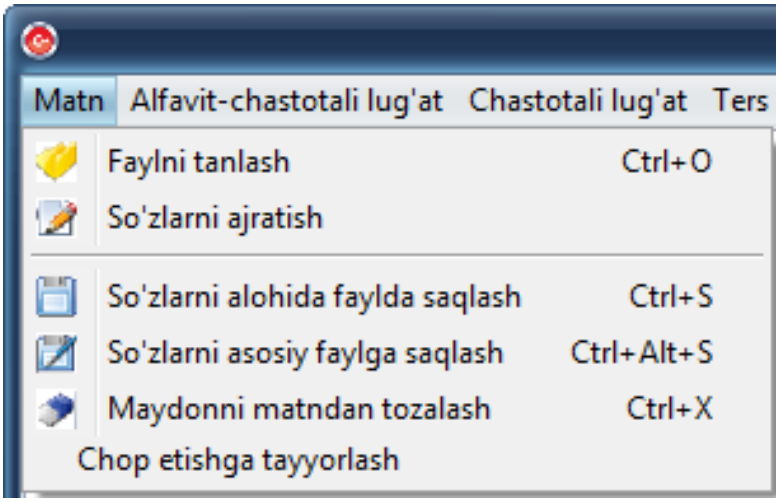
- Matn;
- Alfavit –chastotali lug’at;
- Chastotali lug’at;
- Ters lug’at;
- Yordam;
- Muallif;



3.4-rasm. Dastur menyu qatori.

Har bir bo’lim o’z vazifasiga ko’ra tavsiflanadi.

“Matn” bo’limi ichida quyidagi bo’limlar mavjud (3.5-rasm).

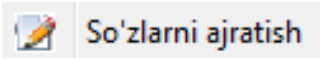


3.5-rasm. Matn bo’limining vazifalari.

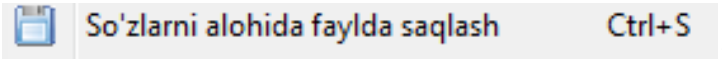
- Faylni ochish;
- Alfavit –chastotali lug’at hosil qilish;
- Chastotali lug’at hosil qilish;
- Ters lug’at hosil qilish;
- So’zlarni saqlash;
    - So’zlarni alohida faylda saqlash;
    - So’zlarni asosiy faylga saqlash;
- Maydonni matndan tozalash;

Faylni ochish 

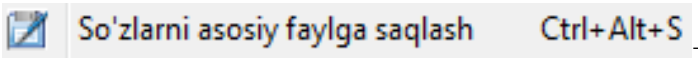
– bu bo’lim orqali maydon matn bilan to’ldiriladi.

So’zlarni ajratish 

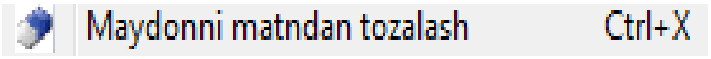
- bu band orqali kiritilgan matn so’zlarga ajratiladi.

So’zlarni alohida faylda saqlash 

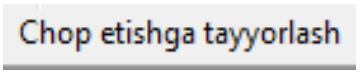
– Bunda ajratib olingan so’zlarni hosil qilingan lug’at bo’yicha yangi faylga saqlaydi.

So’zlarni asosiy faylga saqlash 

– Bu bo’limda lug’atni asosiybazaga qo’shish uchun ikkinchi interfeysni ochadi va yangi lug’atni u interfeysga olib o’tadi.

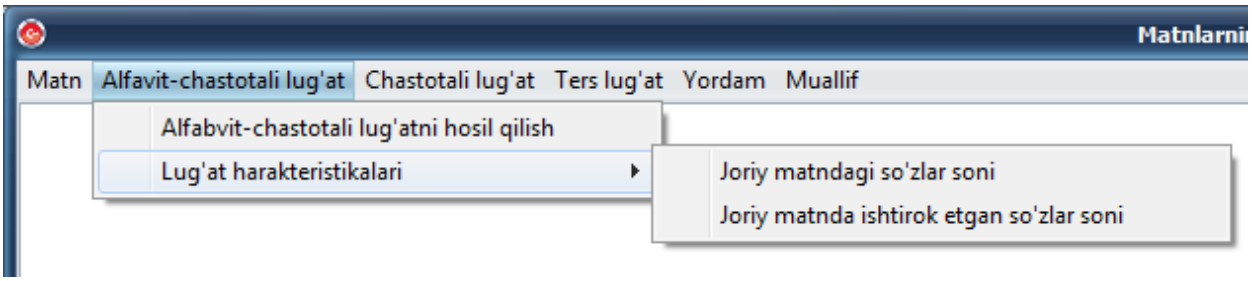
Maydonni matndan tozalash 

– Bu tugma yordamida joriy oynada avval ochilgan matnni o’chiriladi.

Chop etishga tayyorlash 

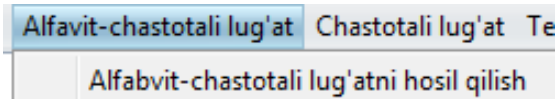
- bu band orqali uchinchi interfeys ochiladi. Uchinchi interfeysda lug’atni chop etishga tayyorlash mumkin.

Alfavit-chastotali lug’at yaratish bo’limi quyidagi bandlardan iborat (3.6- rasm).

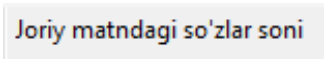


3.6-rasm. Alfavit-chastotali lug’at bo’limining vazifalari.

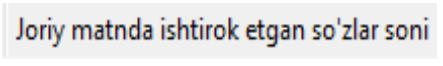
- Alfavit-chastotali lug’atni hosil qilish;
- Lug’at harakteristikalari;
    - Joriy matndagi so’zlar soni;
    - Joriy matnda ishtirok etgan so’zlar soni;

Alfavit –chastotali lug’at hosil qilish 

– bu tugma yordamida Alfavit bo’yicha saralangan takrorlanmas so’zlardan iborat jadval hosil qiladi.

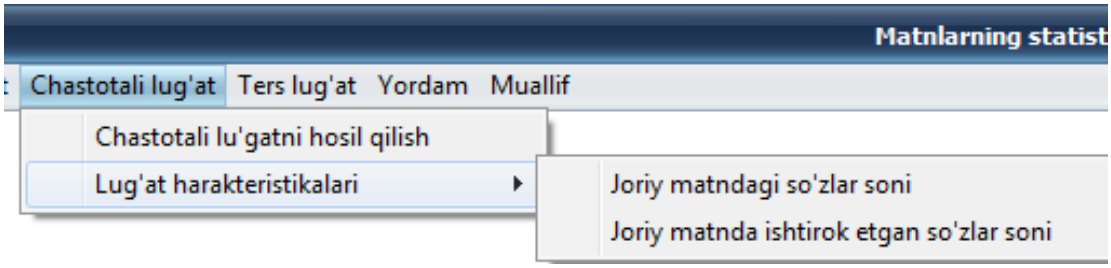
Joriy matndagi so’zlar soni 

– bu band orqali kiritgan matndagi so’zlar miqdorini aniqlash mumkin.

Joriy matnda ishtirok etgan so’zlar soni 

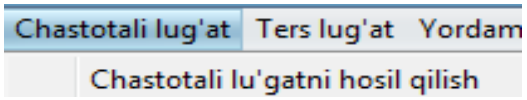
– bu bo’limda kiritilgan matndagi bir-biriga o’xshamas so’zlar sonini miqdorini aniqlab beradi.

Chastotali lug’at yaratish bo’limi quyidagi bandlardan iborat (3.7- rasm).

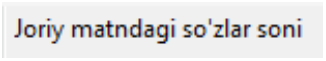


3.7-rasm. Chastotali lug’at bo’limining vazifalari.

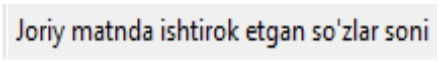
- Chastotali lug’atni hosil qilish;
- Lug’at harakteristikalari;
    - Joriy matndagi so’zlar soni;
    - Joriy matnda ishtirok etgan so’zlar soni;

Chastotali lug’at hosil qilish 

– bunda matn ishtirok etgan so’zlarni chastotasi bo’yicha kamayish tartibida saralaydi.

Joriy matndagi so’zlar soni 

– bu band orqali kiritgan matndagi so’zlar miqdorini aniqlash mumkin.

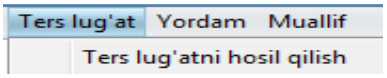
Joriy matnda ishtirok etgan so’zlar soni 

– bu bo’limda kiritilgan matndagi bir-biriga o’xshamas so’zlar sonini miqdorini aniqlab beradi.

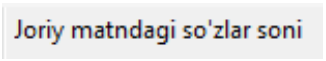
Ters lug’at yaratish bo’limi quyidagi bandlardan iborat (3.8- rasm).

3.8-rasm. Ters lug’at bo’limining vazifalari.

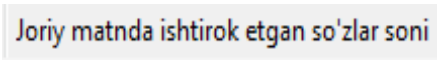
- Ters lug’atni hosil qilish;
- Lug’at harakteristikalari;
    - Joriy matndagi so’zlar soni;
    - Joriy matnda ishtirok etgan so’zlar soni;

Ters lug’at hosil qilish 

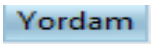
– bu tugma yordamida matndan ajratib olingan har bir so'zni teskari holatda ko'rsatadi. Unda so'ng Alfavit bo'yicha saralaydi.

Joriy matndagi so’zlar soni 

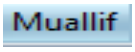
– bu band orqali kiritgan matndagi so’zlar miqdorini aniqlash mumkin.

Joriy matnda ishtirok etgan so’zlar soni 

– bu bo’limda kiritilgan matndagi bir-biriga o’xshamas so’zlar sonini miqdorini aniqlab beradi.

Yordam 

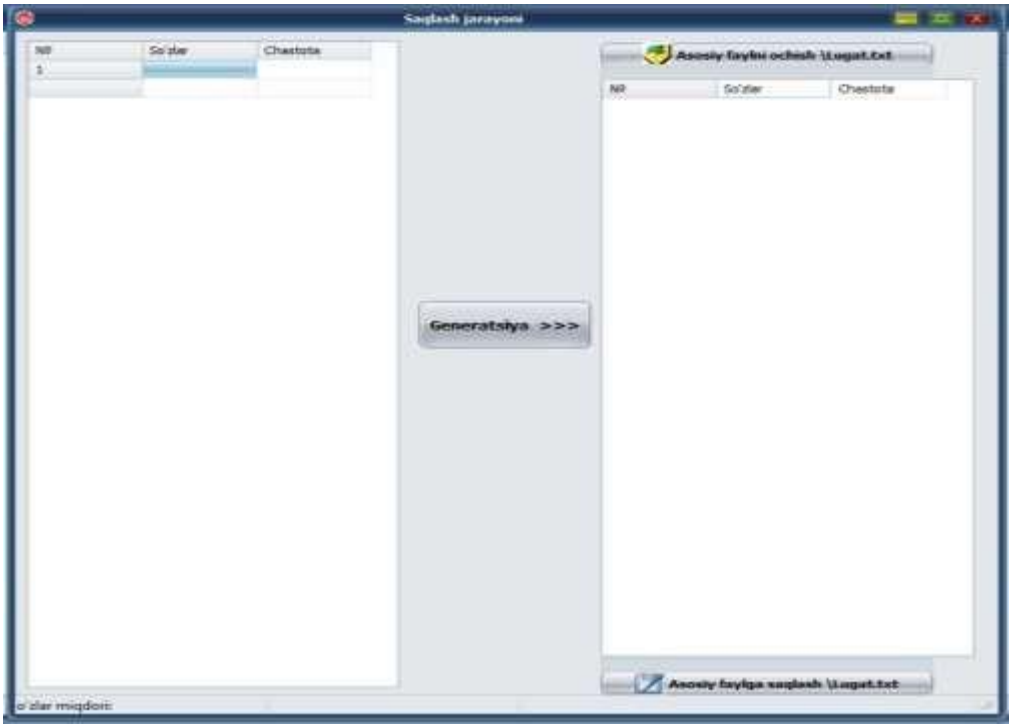
– yordam tugmasi orqali dastur haqida ma’lumotga ega bo’lish mumkin.

Muallif 

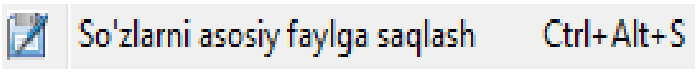
– bu bo’limda dastur muallifi va muallif haqida ma’lumotga ega bo’lish mumkin.

Yangi lug’atni bazaga qo’shish oynasi quyidagi rasmda keltirilgan (3.8-rasm). U quyidagi bandlardan iborat.

- Bazani ochish;
- Generatsiya;
- Asosiy faylga saqlash;



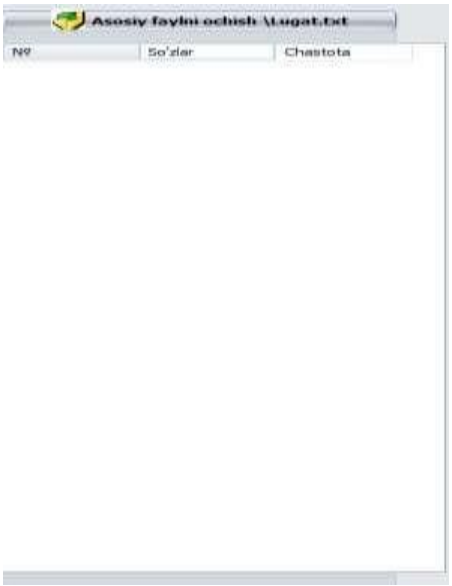
3.9-rasm. Yangi elementlarni bazaga kiritish.



tugmani bosganimizda yangi lug’atni quyidagi oynada hosil qiladi (3.10- rasm).

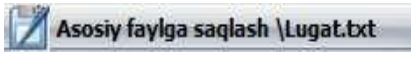
3.10- rasm. Yangi lug’at ko’rinadi.

- bu tugmani bosganimizda bazadagi lug’atni quyidagi oynada ko’rsatadi (3.11-rasm).



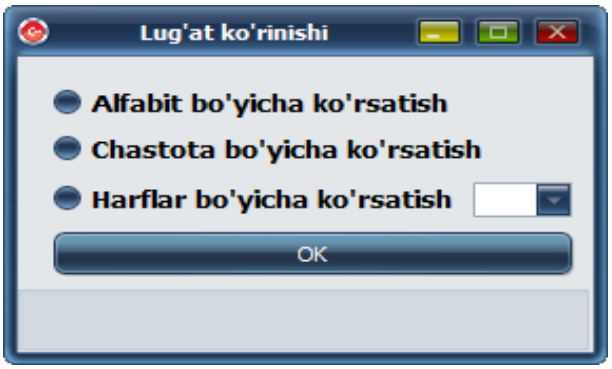
3.11- rasm. Asosiy fayl ko’rinadi.

- bu tugma yordamida asosiy bazaga yangi lug’atni qo’shish imkoniyati mavjud.

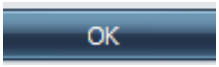


- bu tugma hosil bo’lgan yangi bazani qaytadan faylga saqlaydi.

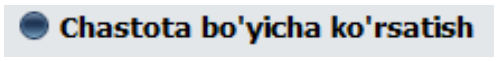
Chop etish oynasi ikkita oynaga bo’linadi. Birinchi oynada lu’gatni qanaqa harakteristikada chop etish kerakligini tanlaymiz (3.12-rasm). Ikkinchi oynakerakli lug’atni ko’rsatadi (3.13-rasm).



3.12-rasm. Lug’atni chop etishga tayyorlash.

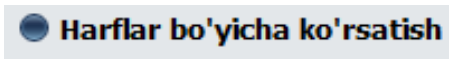
- tugmani belgilab 

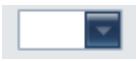
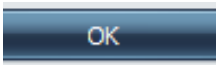
tugmasini bosganda baza alfavit tartibi bo’yicha hosil bo’ladi.



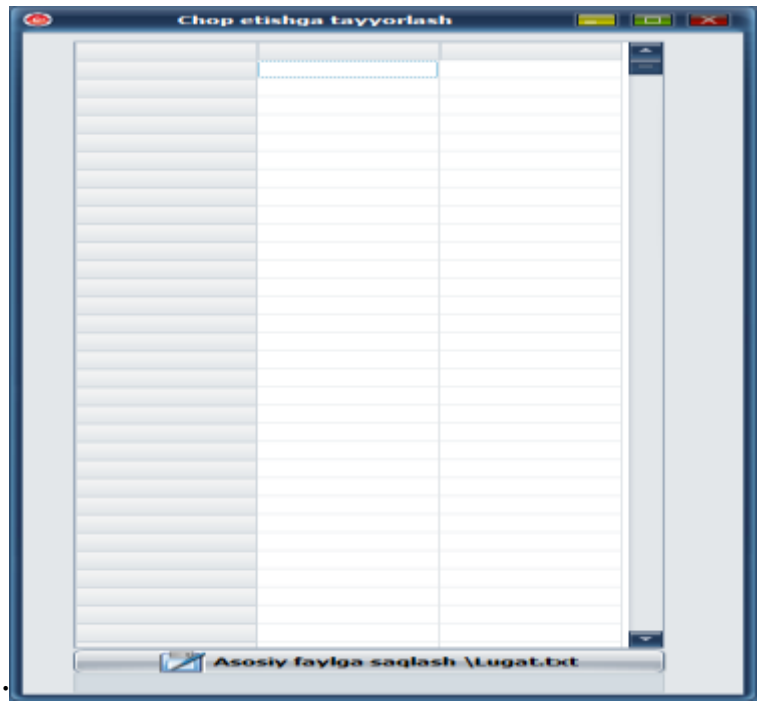
- bu tugmani belgilab 

tugmasini bosganda lug’at bazasini chastotasi kamayishi tartibida hosil qiladi.



- tugmasini belgilab 
bu tugmadan kerakli harfni belgilash kerak. Kerakli harfni belgilagandan so’ng 

tugmasi bosilganda, kerakli harf bo’yicha jadval chiqarishi kerak.



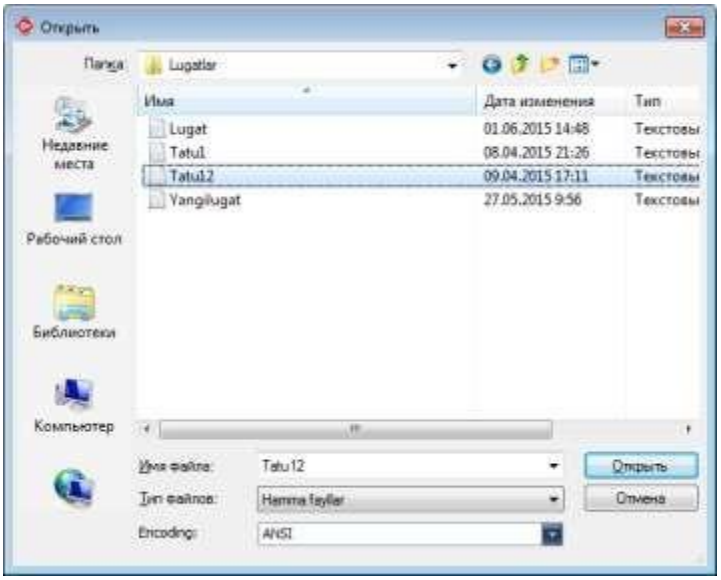
3.13-rasm. 3.12-rasm bo’yicha lug’at hosil bo’ladi.

- tugma orqali hosil qilingan lug’at faylga saqlanadi.

3.3. Dasturiy majmuadan foydalanish tartibi.

3.3-rasmdagi bosh oynadan 

- tugma yordamida fayl ochiladi.



3.14-rasm. Ixtiyoriy fayl tanlanadi.

3.14-rasmdan kerakli matn tanlanadi va 

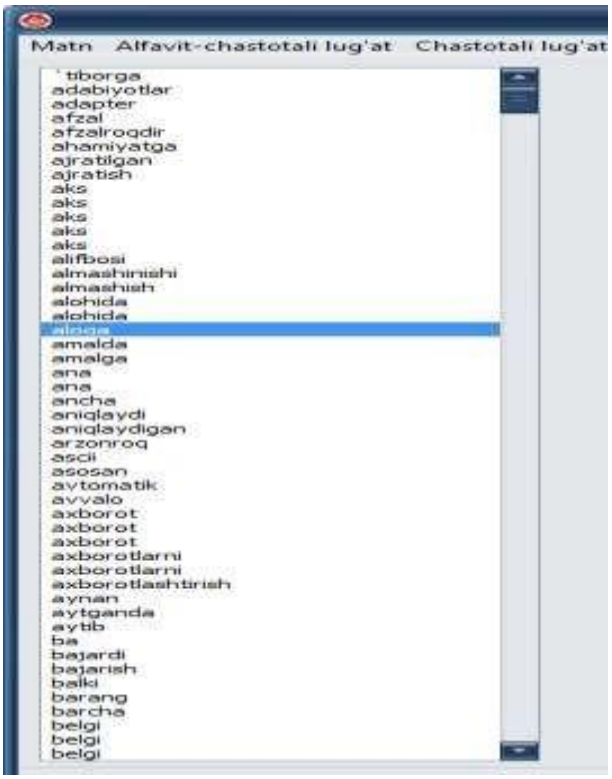
tugmasi bosiladi. Bunda matn quyidagi oynada hosil bo’ladi (3.15-rasm).

3.15-rasm. Tanlangan fayl ochilgan holati.

Agar matnni so’zlarga ajratmay alfavitli, chastotali va ters lug’atni hosil qilmoqchi bo’lsak, dastur kerakli xabarni beradi (3.16-rasm) . Shuning uchun matn ochilgandan so’ng matndagi so’zlarni 

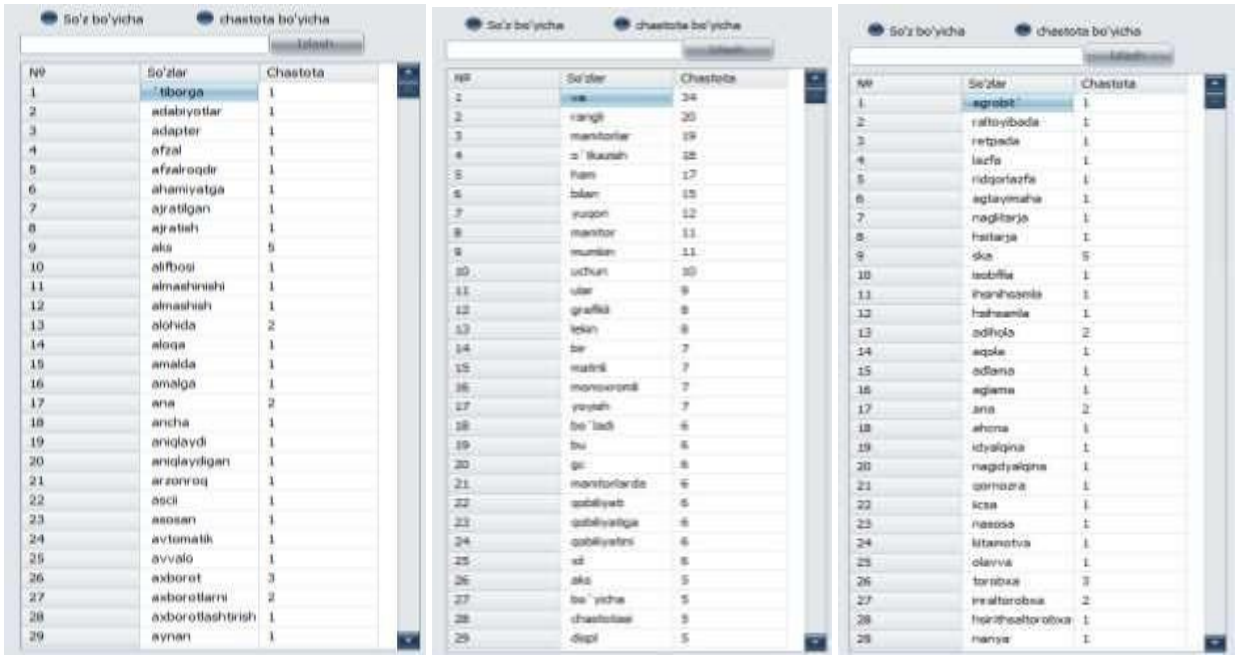
tugmasi yordamida ajratib alfavit tartibida ko’rsatadi (3.17-rasm).

3.16-rasm. Matn so’zlarga ajratilmasa beriladigan xatolik.



3.17-rasm. Matn so’zlarga ajratib beriladi.

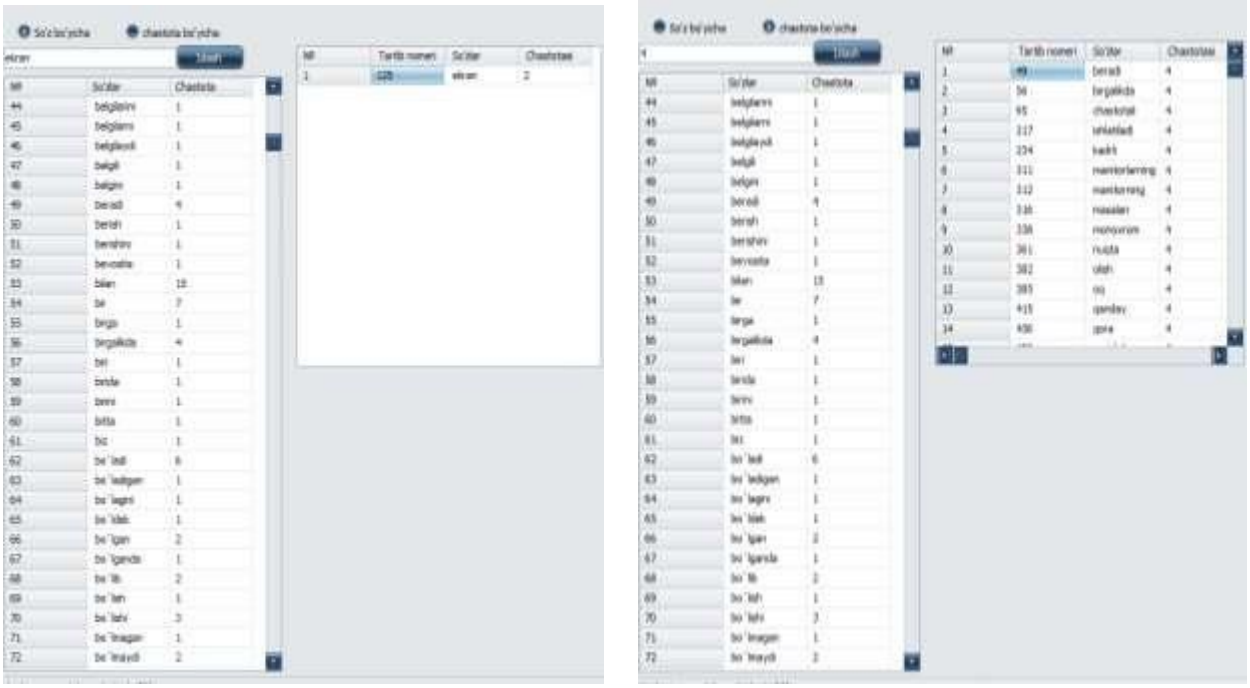
Undan so’ng alfavit-chastotali lug’at yoki chastotali lug’at yoki ters lug’at hosil qilinishi mumkin(3.18 a, b, c -rasmlar).



a) Alfavitli b) Chastotali c) Ters lug’at

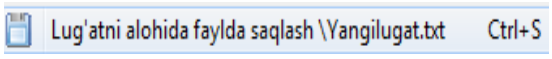
3.18-rasm. Lug’at turlari bo’yicha hosil bo’lishi

Uchta holatda ham ikki qismdan (so’z bo’yicha va chastota bo’yicha) iborat izlash qismi bor (3.19 a, b -rasmlar). Buning uchun kerakli tugmani tanlanadi. Undan so’ng tanlangan tugma bo’yicha maydonga ma’lumot kiritiladi va “Izlash” tugmasi bosiladi. So’ngra natija ekranda so’zni turgan o’rni, so’z va chastotasini hosil qiladi.



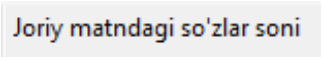
a) So’z bo’yicha qidiruv; b) Chastota bo’yicha qidiruv;

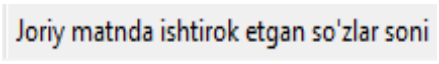
3.19-rasm. Qidiruv oynasi.



- tugmasi orqali “Yangilugat.txt” fayliga saqlanadi (3.20-rasm).

3.20-rasm. Hosil bo’lgan elektron lug’at faylda ko’rinishi.

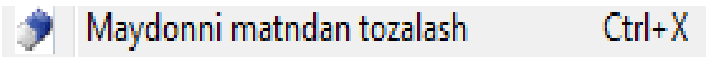
Lug’atni matndagi so’zlarini miqdorini (3.21 a-rasm) 

tugma bilan yoki matnda ishtirok etgan so’zlarni miqdorini (3.21 b-rasm) 

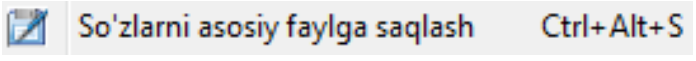
tugma bilan aniqlash mumkin. Kiritilgan matnda quyidagi natijalar olingan.

a) matndagi so’zlarini miqdori. b) matnda ishtirok etgan so’zlarni miqdori.

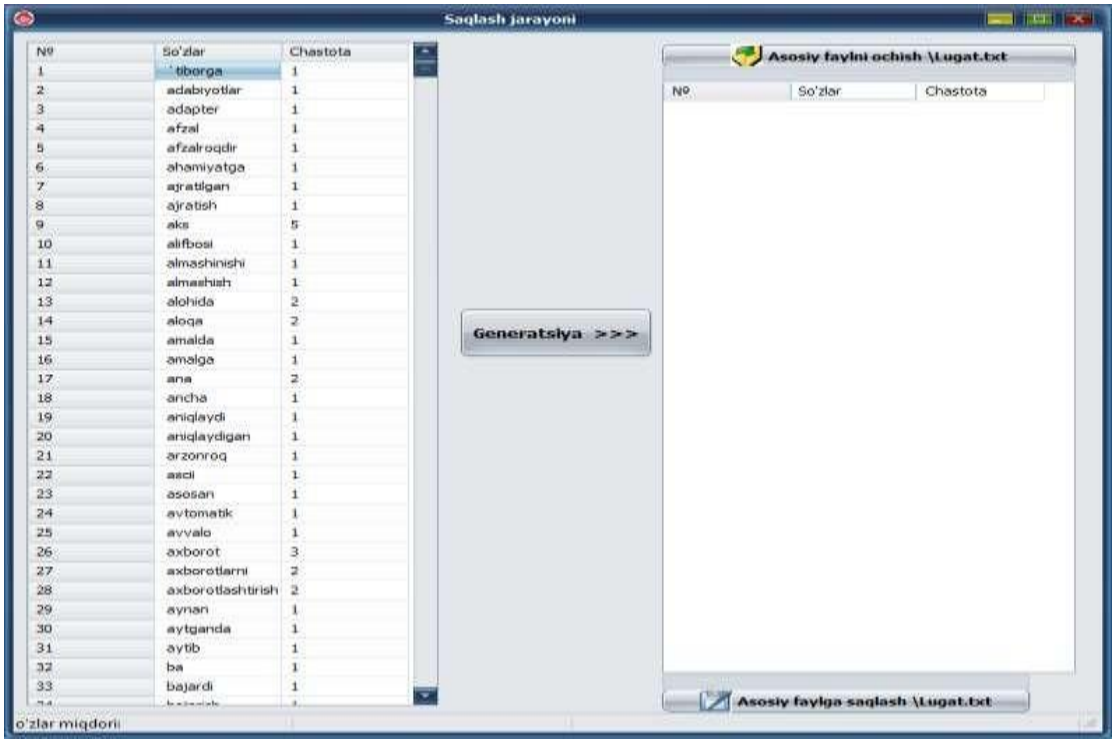
3.21-rasm. Matndagi so’zlar miqdorini aniqlash.



- tugmasi orqali dasturni dastlabki holatiga o’tiladi.

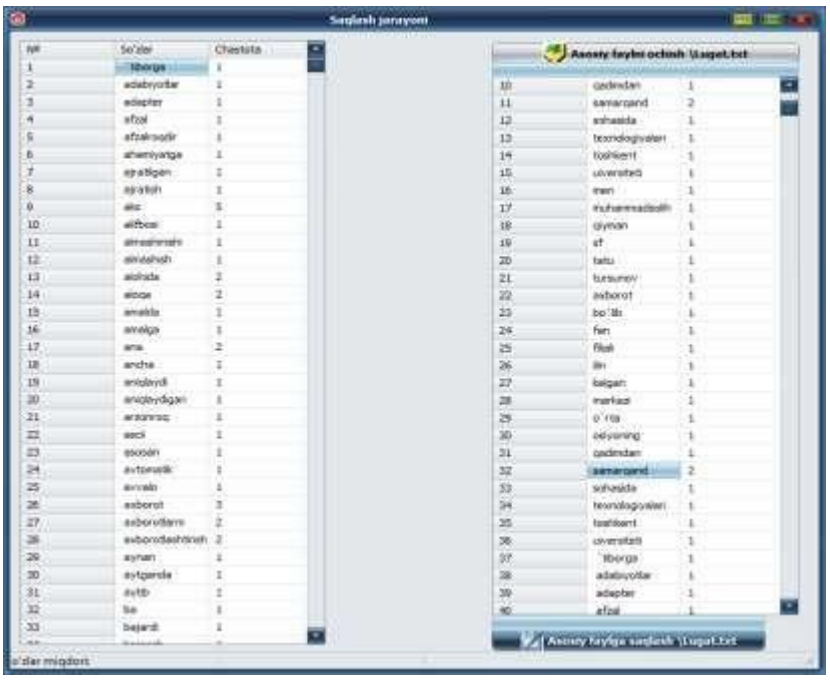


- tugmasi orqali yangi lug’atni lug’at bazasiga qo’shish interfeysi ochiladi shu bilan birga ochilgan interfeysining jadvaliga yangi lug’atni olib o’tadi (3.22-rasm).



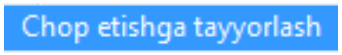
3.22-rasm. Yangi lug’atni asosiy lug’atga qo’shish oynasi.

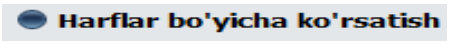
3.22-rasmdagi “Asosiy faylni ochish” tugmasi orqali lug’at bazasi ochiladi (Lugat.txt). “Generatsiya” tugmasi yordamida yangi lug’atni bazaga qo’shadi. Har bir so’zni bazaga qo’shayotganda so’zlarni solishtirib chiqadi. Agar yangi so’z asosiy faylda mavjud bo’lmasa, u holda yangi so’zni asosiy faylga qo’shib qo’yadi, aks holda ya’ni agar yangi so’z faylda mavjud bo’lsa so’zni qo’shmay uning chastotasini mavjud so’z chastotasiga qo’shadi (3.23-rasm).



3.23-rasm. Yangi lug’atni asosiy lug’atga qo’shgandagi ko’rinishi.

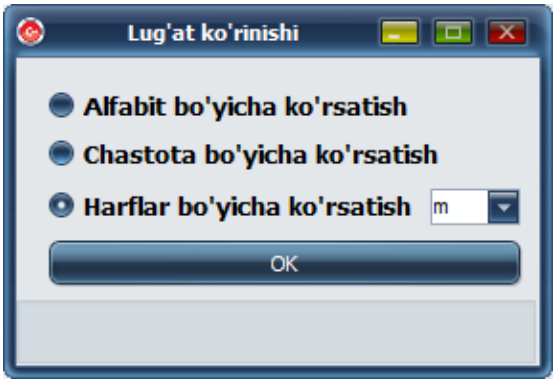
- bu tugma orqali yangi baza faylga saqlanadi.



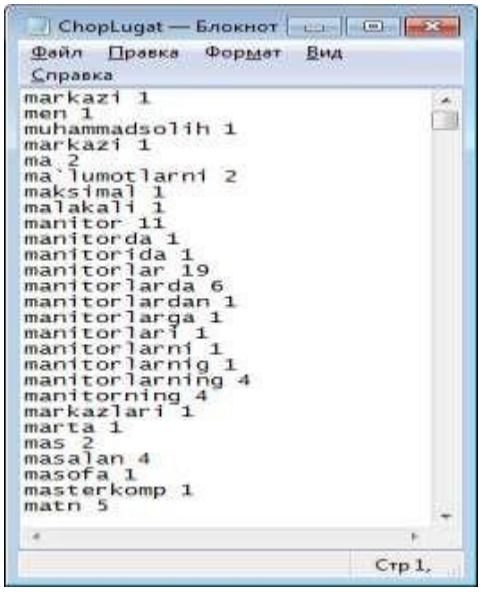
- bu tugma orqali uchinchi interfeysga o'tiladi ya'ni lug'atni harateristika bo'yicha taxlash mumkin (3.12-rasm). Uchta banddan bittasini tanlab "OK" tugmasi bosiladi. Chastotasi kamayishi tartibida lug'at hosil qilish kerak bo'lsa kerakli band tanlanib "OK" tugmasi bosiladi (3.24-rasm). Agar bosh harf bo'yicha lug'at hosil qilish kerak bo'lsa 

tugmasini belgilab  bu tugmadan kerakli harfni belgilash kerak. Undan so'ng "OK" tugmasini bosish orqali kerakli harf bo'yicha lug'at hosil qilinadi. Belgilangan harf bo'yicha bazadan yangi lug'at hosil qilinadi (3.25-rasm). Hosil qilingan lug'at kerakli faylga saqlanadi (3.26-rasm).

3.24-rasm. Asosiy faylni chastota bo'yicha ko'rsatish.



3.25-rasm. Asosiy faylni bosh harf bo'yicha ko'rsatish



3.26-rasm. Elektron lug’atni faylda saqlanishi.

Yordam oynasi quyidagicha bo’ladi (3.27-rasm). Unda har bir tugmani bosilsa kerakli ma’lumot beradi.

3.27-rasm. Dastur haqida ma’lumotnoma.

3.4. Dasturning ish natijalari

Yaratilgan dastur asosida o’zbek yozuvchisi Pirimqul Qodirovning “Yulduzli tunlar” asarining elektron lug’ati olindi. Asar MS Word dasturida tayyorlangan 1040 Kb hajmli *.txt formatdagi fayldan iborat. Ushbu asar dastur yordamida qayta ishlangan so’ng unda jami 25012ta so’z ishlatilgan va ularning qo’llanilishi 117796 so’zni tashkil qilgan; Elektron lug’atda 25012ta birlik hosil bo’ldi.

Xuddi shunday, o’zbek yozuvchisi Abdulla Qodiriyning “O’tgan kunlar” asari MS Word dasturida tayyorlangan 700 Kb hajmli *.txt formatdagi fayl bo’lib, unda jami 91255ta so’z ishlatilgan va ularning qo’llanilishi 21338 so’zni tashkil qilgan; elektron lug’ati 21338ta birlikdan iborat;

Ikki asarning lug’ati biriktirilganda 41325ta birlikdan iborat elektron lug’at hosil bo’ldi.

Asarlar shu tarzda ketma-ket kiritilib borib, o’zbek tilining badiiy asarlar bo’yicha elektron lug’atni hosil qilish mumkin.

3.5. Kompyuter bilan ishlashda texnika xavfsizligi qoidalari va talablari

Kompyuter bilan ishlash davomida bir qator faoliyat bilan bog’liq fiziologik jarayonlar sodir bo’ladi va inson organizmiga turli darajada ta’sir ko’rsatadi.

Bu kabi surunkali ta’sirlar va xavfli holatlar muxandis-dasturlarni ish faoliyatiga ta’sir ko’rsatib kasb kasaliklarini keltirib chiqarishi tasdiqlangan.

Yong’in xavfsizligi qoidalariga binoan EHM joylashtirilgan xonalar, ko’paytirish qiyofa ko’chirish uskunalari atrofida tutunli xabar berish moslamalari o’rnatiladi. Yonqin boshlangan holatda plastmassalarni tutashi natijasida zaharli is gazi ajralib chiqadi va ishchilarni bug’ishi mumkin. Shu sababli, belgilangan talablarga mos keladigan o’chirish usullari va nazariyasi- dan foydalanish maqsadidga muvofiqdir. Olib borilayotgan tadqiqotchilar va tahlillar asosida aynan shu holatga mos keluvchi usullar tanlanadi. Ish o’rnini samarali tashkil yetishda asosiy ye’tibor elektromagnit va elektrostatik maydonlar ta’sirini kamaytirishga qaratilishi darkor. Bugungi kunda ish o’rnini yergonomik talablar asosida tashkil etish, komfort ish sharoitini tashkil etish dolzarb sosial masaladir. Olingan tadqiqotlar asosida ish stolining optimal o’lchamlari, qo’lay konfigurasiya, ish o’rnini to’g’ri tashkil etishda asosiy manbadir [11].

Kompyuterlar joylashtirilgan xonalar va dasturlovchi o’rni yong’in xavfsizligi talablari bo’yicha «D» toifaga mansub bo’lib, tez yonuvchi materiallar kam hisoblanadi.

Elektr hisoblash mashinalari va aloqa vositalari joylashtirilgan xonalarda yong'in xavfi mavjud bo'lib, bu holat katta miqdorda tez eriydigan yengil materiallar. Bu xonalarda yong'in asosan yonuvchan moddalarning oksidlanishi, issiqlik ta'sirida erishi natijasida sodir bo'ladi. Mavjud xonalarda o'rnatilgan akustik va estetik talablarga mos keluvchi eshik, pol, kabellar izolyasiyasi vaboshqalar yonuvchi komponentlar toifasiga kiradi.

Yong'inga qarshi kurash bu kompleks tashkiliy va texnik tadbirlar bo'lib, xodimlarni xavfsizligini ta'minlash, yong'inni samarali o'chirish uni tez tarqalishini oldini olishdan iborat. Uchqun chiqaruvchi manbalar esa, elektron sxemalar, elektr ta'minot tarmoqlari, turli elektron uskunalarning nosozligi, qisqa tutashuv, yer bilan notug'ri biriktirish, ulashdagi nosozliklar hisoblanadi [12].

Zamonaviy kompyuterlar va EHMlarida elektron sxemalar va detallar zich joylashganligi va kabellar hamda ulash simlari yaqinligi sababli bu xavf ancha yuqoridir. Bulardan elektr tokini oqishi himoya qavatini erishiga olib keladi. Kompyuterlarda bu kabi ortiqcha issiqlik maxsus shamolatish bloki orqali chiqarib yuboriladi. Bu blokni ishdan chiqishi yesa juda xavflidir. Bunday hollarda xavfsizlik choralarini ko'rish va qo'shimcha shamollatish tizimini tashkil yetish maqsadga muvofiqdir [12].

Ma'lumki, yong'in holatini komp'yuter choralarida qo'llanilgan qurilish materiallarining yong'inga chidamligini, umumiy mustahkamlikni ta'minlaydi. Elektron uskunalarni yuqori narxini e'tiborga olgan holda bu turdagi xonalarni 1 yoki 2 toifali yong'inga chidamlilik deb darajalash mumkin.

Kichik hajmdagi yong'in o'choqlarini bartaraf etish uchun binolarda yong'in krani, o't o'chirish moslamasi, quruq qo'm, asbestli ko'rpa va boshqalarda foydalaniladi [12].

Belgilangan talablarga javob beradigan sharoit yaratish, ish sifatini va unumdorligini oshirish, dastur ishini yaxshilash, nuqsonsiz ishlashni ta'minlash imkonini beradi.

Kompyuter bilan ishlash davomida insonlarning ko'rish organlarida yuqori darajada kuchlanish holati sodir bo'ladi.

Turli o’lchamdagi va hajmdagi grafik materiallar, kichik o’lchamlar, hajmini ortishi natijasida ko’zda surunkali jiddiy kuchlanish yoki zo’riqish sodir bo’ladi. Bu kabi salbiy holatlar ko’zni ko’rish qobiliyatini pasayishiga olib keladi.

Belgilangan me’yorlarga binoan foydalanuvchi operator ko’zi bilan yekran oralig’idagi masofa 600-700 mm bo’lishi lozim, Ayrim hollarda harf va raqamlar kattaligini e’tiborga olgan holda bu masafani 500 mm bo’lishiga ruxsat beriladi.

Ko’zni zo’riqishini kamaytirish maqsadida har 1 soatda 15 minutlik tanaffus qilish tavsiya etiladi [11].

Bugungi kunda bir qator olimlar tomonidan komp’yuter monitorlaridan tushayotgan turli nurlarni inson organizmiga ta’siri va ma’lum ionlanish sodir bo’lishi tasdiqdangan. Lekin, bu masalada to’laqonli belgilar mavjud emas.

Elektormagnit ionlashgan nurlarning monitordan tushadigan miqdori qo’yidagi jadvalda keltirilgan.

Ma’lumki, tegishli me’yorlarga asosan ish joyida komp’yuter monitoridan ajraladigan maksimal rengen nurlarning maksimal miqdori 10 mk ber/ch va ultrabinafsha va infraqizil nurlarning bu kabi miqdori yesa 10-100 mVt/m2 bo’lishi aniqlangan [12].

Xulosa

Ushbu bitiruv malakaviy ishni bajarishda quyidagilar amalga oshirildi:

- kompyuter lingvistikasiga oid, xususan, lug’atlar va ularni tuzishgaoid adabiyotlar o’rganildi;
- o’zbek tilining kompyuter yordamida statistik tahlili, alfavit-chastotali lug’atlar tuzilishi, mavjud elektron lug’atlar yaratish kabi ishlar bajarilishining zamonaviy darajasi tahlil qilindi;
- o’zbek tili elektron lug’atini yaratishning dasturiy majmuasi loyihalandi;
- elektron lug’at yaratish algoritmlari ishlab chiqildi;
- dasturlar majmuasi kodlashtirildi, sozlandi va sinovdan o’tkazildi;
- dasturiy majmua yordamida o’zbek yozuvchisi Abdulla Qodiriyning “O’tgan kunlar” asari va Pirimqul Qodirovning “Yulduzli tunlar” asarlarining elektron lug’ati yaratildi.

Dastur ishining natijalarida o’zbek tilshunosligi uchun muhim ma’lumotlar olindi:

o’zbek yozuvchisi Pirimqul Qodirovning “Yulduzli tunlar” asari MS Word dasturida tayyorlangan 1040 Kb hajmli *.txt formatdagi fayl bo’lib, unda jami 25012ta so’z ishlatilgan va ularning qo’llanilishi 117796 so’zni tashkil qilgan; elektron lug’at 25012ta birlikdan iborat;

o’zbek yozuvchisi Abdulla Qodiriyning “O’tgan kunlar” asari MS Word dasturida tayyorlangan 700 Kb hajmli *.txt formatdagi fayl bo’lib, unda jami91255ta so’z ishlatilgan va ularning qo’llanilishi 21338 so’zni tashkil qilgan; elektron lug’atida 21338ta birlikdan iborat;

ikki asarning lug’ati biriktirilganda 41325ta birlikdan iborat elektron lug’at hosil bo’ldi.

Foydalanilgan adabiyotlar ro’yxati

1. O‘zbekiston Respublikasi Prezidentining 2012 yil 21 martdagi «Zamonaviy axborot-kommunikatsiya texnologiyalarini yanada joriy etish va rivojlantirish chora-tadbirlari to‘g‘risida» PQ-1730 sonli qarori.

2. Каршиев А.Б., Пиатровский Р.Г. Восходящий и нисходящий подходы при построении систем автоматизированной переработки текста (АПТ) и машинного перевода (МП). В кн. “Вопросы моделирлвания языка и машинного перевода” Сб. науч. трудов СамГУ. - Самарканд, 1984, стр 21-23.

3. Каршиев А.Б. Ўзбекча атама сўз шаклларини автоматик синтез қилиш дастури. ЭҲМлар учун яратилган дастур. УзР давлат патент идорасида расмий руйхатдан ўтказилган. Гувоҳнома № DGU 00605. 29.01.2003 й.

4. Karshiev A.B., Saidvaliev U. Computer System of Uzbek Spellchecking. International Conference on “IT Promotion in Asia 2009”, September 21-25, 2009, TUIT, Uzbekistan. –P. 173-175.

5. Арзикулов Х., Сафаров Ш., Қаршиев А.Б. Автоматизированная система переработки текстов. //Строй языка и методы обучения иноязычному общению. Сборник научных. статьей факультета иностранных языков СамГУ. - Самарканд, 1997, стр. 5-19.

6. Каримов С., Қаршиев А.Б., Исроилова Г. Абдулла Каҳҳор асарлари тилининг лугати. Алфавитли лугат. Тошкент: «Узбекистон миллий энциклопедияси» Давлат илмий нашриёти, 2007.

7. Qarshiyev A.B., Tursunov M.S. “Matnlarni statistik tahlili uchun amaliy dasturlar majmuasi” //“Axborot va telekommunikatsiya texnologiyalari muammolari” ilmiy-texnik konferensiyaning ma’ruzalar to’plami, II qism. –Toshkent, 2015. Sahifalar 35-37.

8. Qarshiyev A.B., Tursunov M.S., Xamiyev A. “O’zbek tilining statistik lug’atlarini tuzish uchun amaliy dasturlar yaratish” //“Fan-ta’lim va ishlab chiqarish integratsiyada axborot kommunikatsiya texnologiyalarni qo’llashning hozirgi zamon masalalari” respublika ilmiy-texnika anjumanining ma’ruzalar to’plami, III qism. –Nukus, 2015. Sahifalar 50-51.

9. J.Jumayev. “Informatika va dasturlash” o’quv metodik qo’llanma. Buxoro, “Ziyo-Rizograf”, 2005.

10. Пиатровский Р.Г. Текст, машина, человек. –Ленинград, «Судостроение», 1975.

11. Fuqaro muhofazasi me’yorlari va qoidalari SNIP ITM GZ-93,1993.

12. Qudratov A., G’aniev T. va boshqalar. Hayot faoliyati xavfsizligi. Мухаммедов С.А., Пиотровский Р.Г. Инженерная лингвистика и опыт системно-статистического исследования узбекских текстов. Ташкент, «Фан», 1986.

13. Karimov S.A. Zulfiya asarlari lingvostilistikasi. –Samarqand, SamDU, 2006.

14. Qosimov S.S. “Axborot texnologiyalari”: Toshkent. Aloqachi. 2006 yil.

15. Po‘latov A. Kompyuter lingvistikasi. Toshkent, «Akademnashr», 2011.

16. Страуструп Б. Язык программирования C++. Третье издание, М.: Бином, 1999.

17. Шмидский Я.К. Программирование на языке C++: Самоучитель. Учебное пособие. Диалектика. 361 стр, 2004 г.

18. Э. Дейкстра. Заметки по структурному программированию// У. Дал, Э. Дейкстра, К. Хоор. Структурное программирование. - М.: Мир, 1975.

19. Toshkent, Aloqachi, 2005

20. http://ziyonet.uz

21. http://acm.tuit.uz

22. http://intuit.uz

Ilovalar.

Asosiy oynaning ilovasi:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <string.h>
#include <fstream.h>
#include <stdlib.h>
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#include "Unit6.h"
#include "Unit7.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
 int sanoq,umumsanoq;
//_____________________________________________
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//_____________________________________________
void__fastcall TForm1::Ochish1Click(TObject *Sender)
{
Edit1->Visible=False;
Button1->Visible=False;
RichEdit1->Visible=True;
Amalbajarish1->Enabled=True;
ListBox1->Visible=False;
StringGrid1->Visible=False;
StringGrid2->Visible=False;
Ozgartirish1->Enabled=True;
erslugat1->Enabled=True;
RadioButton1->Visible=False;
RadioButton2->Visible=False;
if(OpenTextFileDialog1->Execute())
RichEdit1->Lines->LoadFromFile(OpenTextFileDialog1->FileName);
}
//_____________________________________________
void__fastcall TForm1::FormCreate(TObject *Sender)
{
```

```
 StringGrid1->Cells[0][0]="№";
          StringGrid1->Cells[1][0]="So'zlar";
          StringGrid1->Cells[2][0]="Chastota";
}
//_____________________________________________________
void__fastcall TForm1::Saralash1Click(TObject *Sender)
{     RichEdit1->Visible=False;
ListBox1->Sorted=true;
}
//_____________________________________________________
void__fastcall TForm1::Saqlash1Click(TObject *Sender)
{
}
//_____________________________________________________
void__fastcall TForm1::Joriyoynadagimatnniochirish1Click(TObject *Sender)
{
StringGrid2->Visible=False;
StringGrid1->Visible=False;
RichEdit1->Visible=True;
RichEdit1->Clear();
Amalbajarish1->Enabled=False;
Ozgartirish1->Enabled=False;
erslugat1->Enabled=False;
RadioButton1->Visible=False;
RadioButton2->Visible=False;
ListBox2->Visible=False;
ListBox1->Visible=False;
Button1->Visible=False;
Edit1->Visible=False;
}
//_____________________________________________________
void__fastcall TForm1::Sozlarniajratish2Click(TObject *Sender)
{
 ListBox1->Clear();
RadioButton1->Visible=False;
RadioButton2->Visible=False;
StringGrid2->Visible=False;
ListBox2->Visible=False;
RichEdit1->Visible=False;
        long long int k=0,t;    long long int i,ln;
  ln=Form1->RichEdit1->Text.Length()+1;
        WideString a=" ";          String news=" ";
                        for (i = 0; i < Form1->RichEdit1->Lines->Count; i++)
                a+=Form1->RichEdit1->Lines->Strings[i]+" ";
   for (i = 0; i <=ln+1; i++) {
```

```
        if (a[i]!=' ' &&(
(a[i]>='a'&&a[i]<='z')||(a[i]>='A'&&a[i]<='Z')||a[i]=='`'||a[i]==8217))
                        {       if (a[i]=='A') {
                                        a[i]='a';
                                        }
                                         else if (a[i]=='B') {
                                        a[i]='b';
                                        }
                                         else if (a[i]=='C') {
                                        a[i]='c';
                                        }
                                         else if (a[i]=='D') {
                                        a[i]='d';
                                        }
                                         else if (a[i]=='E') {
                                        a[i]='e';
                                        }
                                         else if (a[i]=='F') {
                                        a[i]='f';
                                        }
                                         else if (a[i]=='G') {
                                        a[i]='g';
                                        }
                                         else if (a[i]=='H') {
                                        a[i]='h';
                                        }
                                         else if (a[i]=='I') {
                                        a[i]='i';
                                        }
                                         else if (a[i]=='J') {
                                        a[i]='j';
                                        }
                                         else if (a[i]=='K') {
                                        a[i]='k';
                                        }
                                         else if (a[i]=='L') {
                                        a[i]='l';
                                        }
                                         else if (a[i]=='M') {
                                        a[i]='m';
                                        }
                                         else if (a[i]=='N') {
                                        a[i]='n';
                                        }
                                         else if (a[i]=='O') {
```

```
                                        a[i]='o';
                                        }
                                         else if (a[i]=='P') {
                                        a[i]='p';
                                        }
                                         else if (a[i]=='Q') {
                                        a[i]='q';
                                        }
                                         else if (a[i]=='R') {
                                        a[i]='r';
                                        }
                                         else if (a[i]=='S') {
                                        a[i]='s';
                                        }
                                         else if (a[i]=='T') {
                                        a[i]='t';
                                        }
                                         else if (a[i]=='U') {
                                        a[i]='u';
                                        }
                                         else if (a[i]=='V') {
                                        a[i]='v';
                                        }
                                         else if (a[i]=='W') {
                                        a[i]='w';
                                        }
                                         else if (a[i]=='X') {
                                        a[i]='x';
                                        } else if (a[i]=='Y') {
                                        a[i]='y';
                                        } else if (a[i]=='Z') {
                                        a[i]='z';
                                        }
                    news+=a[i];
          }
                                            else{
                    if(news!=' ' && news!=" ta" && news!=" gi" && news!=" dan" && news!=" da" && news!=" ga" && news!=" tasi" && news!=" ni"  && news!=" ning"
                    && news!=" a"&& news!=" b"&& news!=" e"&& news!=" d"&& news!=" f"&& news!=" g"&& news!=" h"&& news!=" i"&& news!="j"&& news!=" k"&& news!=" l"
                    && news!=" m"&& news!=" n"&& news!=" o"&& news!=" p"&& news!=" q"&& news!=" r"&& news!=" s"&& news!=" t"&& news!=" u"&& news!=" v"&& news!=" x"
```

```
                    && news!=" y"&& news!=" z"&& news!=" ch"&& news!="
sh") {
                     ListBox1->Items->Add(news) ;
                      }
                                 news=' ';
                                 k++;
                                }    }
                                ListBox1->Visible=True;
                           Label2->Caption=k;
                           StatusBar1->Panels->Items[0]->Text     =     "Matndagi
so'zlar miqdori:  "+Form1->ListBox1->Count;;
                            RichEdit1->Visible=False;
ListBox1->Sorted=true;
}
//_____________________________________________________
void __fastcall TForm1::Alfabvitchastotalilugatnihosilqilish1Click(TObject
*Sender)
{
 ListBox2->Visible=False;
StringGrid2->Visible=False;
if (ListBox1->Sorted==true) {
RadioButton1->Visible=True;
RadioButton2->Visible=True;
Edit1->Visible=True;
Button1->Visible=True;
RichEdit1->Visible=False;
StringGrid1->Visible=True;
int i,t=1,k=1;

                                for (i = 0; i <ListBox1->Count-1; i++)
if  (ListBox1->Items->Strings[i]==ListBox1->Items->Strings[i+1]  )  {
k=k+1;            }
               else  {
                     StringGrid1->RowCount=t+1;
                     StringGrid1->Cells[2][t]=IntToStr(k);
                          StringGrid1->Cells[0][t]=t;
                          StringGrid1->Cells[1][t]=ListBox1->Items->Strings[i];
                                    t++;    k=1;
               }
               // oxirgi so'z olish
if(ListBox1->Items->Strings[ListBox1->Count-1]!=ListBox1->Items-
>Strings[ListBox1->Count-2]){
               StringGrid1->RowCount=t+1;
               StringGrid1->Cells[2][t]=IntToStr(1);
                          StringGrid1->Cells[0][t]=t;
```

```
                    StringGrid1->Cells[1][t]=ListBox1->Items->Strings[ListBox1-
>Count-1];  }
                    else {
                    StringGrid1->Cells[2][t]=IntToStr(k)+1;
                    }
                    // oxirgi so'z olish tugadi

                   sanoq=t;
                   Label1->Caption=sanoq;
                    StatusBar1->Panels->Items[1]->Text = "Ishtirok etgan so'zlar
miqdori:  "+Label1->Caption;
                                        }
                     else{
                            ShowMessage("Matndan so'zlarni ajrating !");
                     }
}

//_____________________________________________________
void__fastcall TForm1::Joriymatndagisozlarsoni1Click(TObject *Sender)
{ StringGrid2->Visible=False;
Form3->ShowModal();
Form3->Panel1->Visible=True;
Form3->Label1->Visible=False;
Form3->Label2->Visible=False;
}
//_____________________________________________________
void __fastcall TForm1::Joriymatndaishtiriketgansozlarsoni1Click(TObject
*Sender)
{ StringGrid2->Visible=False;
Form2->ShowModal();
Form2->Panel1->Visible=True;
Form2->Label1->Visible=False;
Form2->Label2->Visible=False;
}
//_____________________________________________________
void__fastcall TForm1::Alfavitchastotalilugat1Click(TObject *Sender)
{
RichEdit1->Visible=False;
StringGrid1->Visible=True;
StringGrid2->Visible=False;
RadioButton1->Visible=True;
RadioButton2->Visible=True;
ListBox2->Visible=False;
int i,t=1,k=1;
                                    for (i = 0; i <ListBox1->Count-1; i++)
if (ListBox1->Items->Strings[i]==ListBox1->Items->Strings[i+1] )   {
```

```
k=k+1;            }
                  else  {
                  StringGrid1->RowCount=t+1;
                        StringGrid1->Cells[2][t]=IntToStr(k);
                              StringGrid1->Cells[0][t]=t;
                              StringGrid1->Cells[1][t]=ListBox1->Items->Strings[i];
                                    t++;    k=1;
                  }
                        sanoq=t-1;
                        Label1->Caption=sanoq;
               StatusBar1->Panels->Items[1]->Text = "Ishtirok etgan so'zlar
miqdori:  "+Label1->Caption;
}
//_____________________________________________________
void__fastcall TForm1::Button1Click(TObject *Sender)
{
StringGrid2->Visible=True;
          StringGrid2->Cells[0][0]="№";
          StringGrid2->Cells[1][0]=" Tartib nomeri";
          StringGrid2->Cells[2][0]=" So'zlar";
          StringGrid2->Cells[3][0]=" Chastotasi ";
int t,k=1;
String aa;
if(RadioButton1->Checked==True){
if(ListBox2->Visible==True){
        aa=" "+Edit1->Text+" ";
 for(t=1; t<=sanoq+1; t++)
        if (aa==StringGrid1->Cells[1][t]) {
        StringGrid2->RowCount= k+1;
                  StringGrid2->Cells[0][k]=k;
                  StringGrid2->Cells[1][k]=IntToStr(t);
                  StringGrid2->Cells[2][k]=aa;
                  StringGrid2->Cells[3][k]=StringGrid1->Cells[2][t];
                  k++;        }  }
                  else{
aa=" "+Edit1->Text;
         for(t=1; t<=sanoq+1; t++)
        if (aa==StringGrid1->Cells[1][t]) {
        StringGrid2->RowCount= k+1;
                  StringGrid2->Cells[0][k]=k;
                  StringGrid2->Cells[1][k]=IntToStr(t);
                  StringGrid2->Cells[2][k]=aa;
                  StringGrid2->Cells[3][k]=StringGrid1->Cells[2][t];
                  k++;        }
}
```

```
        }
else
 if (RadioButton2->Checked==True) {
         for(t=1; t<=sanoq+1; t++)
               if (Edit1->Text==StringGrid1->Cells[2][t]) {
                 StringGrid2->RowCount= k+1;
                 StringGrid2->Cells[0][k]=k;
                 StringGrid2->Cells[1][k]=IntToStr(t);
                 StringGrid2->Cells[2][k]=StringGrid1->Cells[1][t];
                 StringGrid2->Cells[3][k]=StringGrid1->Cells[2][t];
                 k++;
 }
       }   }
//---------------------------------------------------------------------------
void__fastcall TForm1::Chastotalilugat1Click(TObject *Sender)
{
StringGrid2->Visible=False;
RadioButton1->Visible=True;
RadioButton2->Visible=True;
ListBox2->Visible=False;
}
//---------------------------------------------------------------------------
void__fastcall TForm1::erslugat2Click(TObject *Sender)
{
StringGrid2->Visible=False;
RadioButton1->Visible=True;
RadioButton2->Visible=True;
ListBox2->Visible=True;
}
//---------------------------------------------------------------------------
void__fastcall TForm1::Jamisozlarsoni1Click(TObject *Sender)
{
StringGrid2->Visible=False;
}
//---------------------------------------------------------------------------

void__fastcall TForm1::Chastotalilugatnihosilqilish1Click(TObject *Sender)
{   ListBox2->Visible=False;
StringGrid2->Visible=False;
   StringGrid1->RowCount=2;
if (ListBox1->Sorted==true) {
RadioButton1->Visible=True;
RadioButton2->Visible=True;
Edit1->Visible=True;
Button1->Visible=True;
```

```
RichEdit1->Visible=False;
StringGrid1->Visible=True;
int i,t=1,k=1;
                                        for (i = 0; i <ListBox1->Count-1; i++)
if (ListBox1->Items->Strings[i]==ListBox1->Items->Strings[i+1] )    {
 k=k+1;              }
                  else  {
                          StringGrid3->RowCount=t+1;
                          StringGrid3->Cells[2][t]=IntToStr(k);
                                  StringGrid3->Cells[0][t]=t;
                                  StringGrid3->Cells[1][t]=ListBox1->Items->Strings[i];
                                          t++;    k=1;
                  }
                  // oxirgi so'z olish
if(ListBox1->Items->Strings[ListBox1->Count-1]!=ListBox1->Items-
>Strings[ListBox1->Count-2]){
                  StringGrid3->RowCount=t+1;
                  StringGrid3->Cells[2][t]=IntToStr(1);
                                  StringGrid3->Cells[0][t]=t;
 StringGrid3->Cells[1][t]=ListBox1->Items->Strings[ListBox1->Count-1]; }
                        else {
                        StringGrid3->Cells[2][t]=IntToStr(k)+1;
                        }
                        // oxirgi so'z olish tugadi
                       sanoq=t;
                       Label1->Caption=sanoq;
                       StatusBar1->Panels->Items[1]->Text = "Ishtirok etgan so'zlar
miqdori:  "+Label1->Caption;
                                                          }
                         else{
                                 ShowMessage("Matndan so'zlarni ajrating !");
                         }

          // tartib bilan yozilishi chastota bo'yicha
            __int64 tartib,tartib1,i,k; int max5,j;
                        max5=StrToInt( StringGrid3->Cells[2][1]);
            tartib=StringGrid3->RowCount;
          for(i=2; i<tartib; i++){
                  if(max5<StrToInt(StringGrid3->Cells[2][i])){
                  max5=StrToInt(StringGrid3->Cells[2][i]); }
            }
                                        k=1;
         for(j=max5; j>=1; j--){
           for(i=1; i<tartib; i++){

                        if(StrToInt(StringGrid3->Cells[2][i])==j){
```

```
StringGrid1->Cells[0][k]=k;
```

```
                                StringGrid1->Cells[1][k]=StringGrid3->Cells[1][i];
                                StringGrid1->Cells[2][k]=StringGrid3->Cells[2][i];
                                StringGrid1->RowCount++;
                                k++;
                        }}
                }
}
//---------------------------------------------------------------------------
void__fastcall TForm1::Jamisozlarsoni2Click(TObject *Sender)
{
StringGrid2->Visible=False;
}
//---------------------------------------------------------------------------
void__fastcall TForm1::Joriymatndagisozlarsoni2Click(TObject *Sender)
{
 StringGrid2->Visible=False;
Form3->ShowModal();
Form3->Panel1->Visible=True;
Form3->Label1->Visible=False;
Form3->Label2->Visible=False;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Joriymatndaishtiroketgansozlarsoni1Click(TObject
*Sender)
{
StringGrid2->Visible=False;
Form2->ShowModal();
Form2->Panel1->Visible=True;
Form2->Label1->Visible=False;
Form2->Label2->Visible=False;
}
//---------------------------------------------------------------------------
void__fastcall TForm1::erslugatnihosilqilish1Click(TObject *Sender)
{
ListBox2->Visible=True;
StringGrid2->Visible=False;
RadioButton1->Visible=True;
RadioButton2->Visible=True;
Edit1->Visible=True;
Button1->Visible=True;
RichEdit1->Visible=False;
StringGrid1->Visible=True;
int i,t=1,k=1,j;     String a,b=' ';
                                int max=0;
                                // ters lugat
```

```
                                    for (i = 0; i <ListBox1->Count; i++) {
                                    a=ListBox1->Items->Strings[i];
                                    j=a.Length();
                                     while(j>0){
                                     b+=a[j];
                                     j--;
                                     }
                                ListBox2->Items->Strings[i]=b;
                                b=' ';
}
ListBox2->Sorted;

for (i = 0; i <ListBox2->Count-1; i++)
if (ListBox2->Items->Strings[i]==ListBox2->Items->Strings[i+1] )   {
 k=k+1;             }
                  else  {
                         StringGrid1->RowCount=t+1;
                         StringGrid1->Cells[2][t]=IntToStr(k);
                                StringGrid1->Cells[0][t]=t;
                                StringGrid1->Cells[1][t]=ListBox2->Items->Strings[i];
                                       t++;    k=1;
                  }
                  // oxirgi so'z olish
if(ListBox1->Items->Strings[ListBox2->Count-1]!=ListBox2->Items-
>Strings[ListBox2->Count-2]){
                  StringGrid1->RowCount=t+1;
                  StringGrid1->Cells[2][t]=IntToStr(1);
                                StringGrid1->Cells[0][t]=t;
        StringGrid1->Cells[1][t]=ListBox2->Items->Strings[ListBox2->Count-1]; }
                          else {
                          StringGrid1->Cells[2][t]=IntToStr(k)+1;
                          }
                          // oxirgi so'z olish tugadi
                         sanoq=t-1;
                         Label1->Caption=sanoq;
                         StatusBar1->Panels->Items[1]->Text = "Ishtirok etgan so'zlar
miqdori:  "+Label1->Caption;
                         }
//_____________________________________________________
void__fastcall TForm1::Qogozgachiqarish3Click(TObject *Sender)
{
StringGrid2->Visible=False;
RadioButton1->Visible=False;
RadioButton2->Visible=False;
}
```

//

```
void__fastcall TForm1::Jamisozlarsoni3Click(TObject *Sender)
{
StringGrid2->Visible=False;
}
//_____________________________________________________
void__fastcall TForm1::Joriymatndagisozlarsoni3Click(TObject *Sender)
{
 StringGrid2->Visible=False;
Form3->ShowModal();
Form3->Panel1->Visible=True;
Form3->Label1->Visible=False;
Form3->Label2->Visible=False;
}
//_____________________________________________________
void __fastcall TForm1::Joriymatndaishtiroketgansozlarsoni2Click(TObject
*Sender)

{
StringGrid2->Visible=False;
Form2->ShowModal();
Form2->Panel1->Visible=True;
Form2->Label1->Visible=False;
Form2->Label2->Visible=False;
}
//_____________________________________________________
void__fastcall TForm1::Yordam1Click(TObject *Sender)
{
StringGrid2->Visible=False;
RadioButton1->Visible=False;
RadioButton2->Visible=False;
Form4->ShowModal();
}
//_____________________________________________________
void__fastcall TForm1::RadioButton1Click(TObject *Sender)
{
if(RadioButton1->Checked==True){
 Edit1->Enabled=True;
 Button1->Enabled=True;}
}
//_____________________________________________________
void__fastcall TForm1::RadioButton2Click(TObject *Sender)
{if(RadioButton2->Checked==True){
 Edit1->Enabled=True;
 Button1->Enabled=True;}
}
```

```
//_____________________________________________________
void__fastcall TForm1::Sozlarnialohidafayldasaqlash1Click(TObject *Sender)
{             long long int i=1;
          TStringList *LL=new TStringList;

         while(i<=StringGrid1->RowCount-1)
          {
                LL->Add(StringGrid1->Cells[1][i]+" "+StringGrid1->Cells[2][i]);
                i++;
          }
        LL->SaveToFile("Yangilugat.txt");
        delete LL;
}
//_____________________________________________________
void__fastcall TForm1::Sozlarniasosiyfaylgasaqlash1Click(TObject *Sender)
{
 int i;
 Form5->StringGrid2->RowCount=Form5->StringGrid2->RowCount-1;
 for(i=1; i<StringGrid1->RowCount; i++){
        Form5->StringGrid1->RowCount++;
        Form5->StringGrid1->Cells[1][i]=StringGrid1->Cells[1][i];
        Form5->StringGrid1->Cells[2][i]=StringGrid1->Cells[2][i];
        Form5->StringGrid1->Cells[0][i]=i;
}
Form5->ShowModal();
 Form5->StringGrid1->RowCount=Form5->StringGrid1->RowCount-1;
}
//_____________________________________________________
void__fastcall TForm1::Muallif1Click(TObject *Sender)
{
Form6->ShowModal() ;
}
//_____________________________________________________
void__fastcall TForm1::SSSs1Click(TObject *Sender)
{
Form7->ShowModal();
}
//_____________________________________________________
```

Dasturda elektron lug’atni asosiy bazaga qo’shish oynasi ilovasi:

```
//_____________________________________________________
#include <vcl.h>
#pragma hdrstop
#include <fstream.h>
```

```
#include "Unit1.h"
#include "Unit5.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
struct zapis{
        char suz[30];
        int chast;
};
//---------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void__fastcall TForm5::BitBtn1Click(TObject *Sender)
{
zapis soz;   long long i=1;
  ifstream fayl;
  fayl.open("Lugat.txt");
  StringGrid2->RowCount=2;
  ProgressBar1->Max=StrToInt(Label1->Caption);
  ProgressBar1->Position=0;
  while(!fayl.eof())
   {
                fayl>>soz.suz>>soz.chast;
                StringGrid2->Cells[0][i]=i;
                StringGrid2->Cells[1][i]=soz.suz;
                StringGrid2->Cells[2][i]=soz.chast;
                StringGrid2->RowCount++;
                ProgressBar1->Position+=2;
                i++;
  }
  fayl.close();
 StringGrid2->RowCount=StringGrid2->RowCount-1;

 if (StringGrid2->RowCount==1) {
 ShowMessage("Baza mavjud emas, yangi lug'at o'tkazildi");
        long long int j,i,l,t,k,k1;
 l=StringGrid1->RowCount;
        for (i=1; i < l; i++)  {
          StringGrid2->RowCount++;
                StringGrid2->Cells[1][i]=StringGrid1->Cells[1][i];
         StringGrid2->Cells[2][i]=StringGrid1->Cells[2][i];
```

```
          StringGrid2->Cells[0][i]=i;
         }
         StringGrid2->RowCount--;
 }
   if (StringGrid2->Cells[1][StringGrid2->RowCount]=="") {
                StringGrid2->RowCount=StringGrid2->RowCount-1;
 }
}
//---------------------------------------------------------------------------
void__fastcall TForm5::FormCreate(TObject *Sender)
{
 StringGrid1->Cells[0][0]="№";
          StringGrid1->Cells[1][0]="So'zlar";
          StringGrid1->Cells[2][0]="Chastota";
  StringGrid2->Cells[0][0]="№";
          StringGrid2->Cells[1][0]="So'zlar";
          StringGrid2->Cells[2][0]="Chastota";
          StatusBar1->Panels->Items[0]->Text = "So'zlar miqdori:   "+StringGrid1-
>RowCount-1;
}
//---------------------------------------------------------------------------
void__fastcall TForm5::BitBtn3Click(TObject *Sender)
{     Label1->Caption=StringGrid2->RowCount-1;
   TStringList *LL=new TStringList;
        long long int i=1;
        ProgressBar2->Max=StringGrid2->RowCount-1;
        ProgressBar2->Position=0;
        while(i<=StringGrid2->RowCount-1)
          {
                LL->Add(StringGrid2->Cells[1][i]+" "+StringGrid2->Cells[2][i]);
                i++;
                ProgressBar2->Position+=1;
          }
        LL->SaveToFile("Lugat.txt");
        delete LL;
}
//---------------------------------------------------------------------------
void__fastcall TForm5::BitBtn2Click(TObject *Sender)
{
long long int j,i,l,t,k,k1;
 if (StringGrid2->RowCount==1) {
       ShowMessage(" Avval bazani oching !!!");
 }
 else{
t=StringGrid2->RowCount;
```

```
j=t;

l=StringGrid1->RowCount;
  ProgressBar4->Max=l;
  ProgressBar4->Position=0;

for (i=1; i < l; i++) {
        ProgressBar3->Max=t;
  ProgressBar3->Position=0; k=1;
for (k1 = 1; k1 < t; k1++)
  {
        if(" "+StringGrid2->Cells[1][k1]==StringGrid1->Cells[1][i]) {
          k++;
StringGrid2->Cells[2][k1]=StrToInt(StringGrid2-
>Cells[2][k1])+StrToInt(StringGrid1->Cells[2][i]) ;
         break; } ProgressBar3->Position+=2;
        }
        if( k==1){
          StringGrid2->RowCount++;
                StringGrid2->Cells[1][j]=StringGrid1->Cells[1][i];
         StringGrid2->Cells[2][j]=StringGrid1->Cells[2][i];
         StringGrid2->Cells[0][j]=j;j+=1;
        }
        ProgressBar4->Position+=2; }

   StringGrid2->RowCount--;
}
  }
//_____________________________________________________
```

Dasturda elektron lug’atni chop etishga tayyorlash oynasi ilovasi:

```
//_____________________________________________________
#include <vcl.h>
#pragma hdrstop
#include <fstream.h>
#include "Unit1.h"
#include "Unit7.h"
#include "Unit8.h"
//_____________________________________________________
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
struct zapis{
        char suz[30];
```

```
        int chast;
};
//_____________________________________________________
__fastcall TForm7::TForm7(TComponent* Owner)
        : TForm(Owner)
{
}
String a[1000000];    long int aa[1000000];
void qsort(int left, int right)
{
        int l = left, r = right, tmpp;   String tmp, m = a[(l+r) / 2];
        while(l <= r){
                        while(a[l]<m)l ++;
                        while(a[r]>m)r --;
                        if(l<=r){
                                tmp = a[l];
                                tmpp=aa[l];
                                a[l] = a[r];
                                aa[l]=aa[r];
                                a[r]  =  tmp;
                                aa[r]=tmpp;
                                l ++ ;
                                r -- ;
                        }
        }
        if(l<right) qsort(l,right);
        if(left <r) qsort(left, r);
}
void__fastcall TForm7::Button1Click(TObject *Sender)
{
if          (RadioButton1->Checked==True          ||RadioButton2->Checked==True
||RadioButton3->Checked==True) {
zapis soz;   long long i=1;
  ifstream fayl;
  fayl.open("Lugat.txt");
  Form8->StringGrid1->RowCount=1;
  Form8->StringGrid2->RowCount=2;
  while(!fayl.eof())
   {  fayl>>soz.suz>>soz.chast;
                Form8->StringGrid2->Cells[0][i]=i;
                Form8->StringGrid2->Cells[1][i]=soz.suz;
                Form8->StringGrid2->Cells[2][i]=soz.chast;
                Form8->StringGrid2->RowCount++;
                        i++;
  }
```

```
 fayl.close();
 Form8->StringGrid2->RowCount=Form8->StringGrid2->RowCount-1;
 if (Form8->StringGrid2->Cells[1][Form8->StringGrid2->RowCount]=="") {
                Form8->StringGrid2->RowCount=Form8->StringGrid2->RowCount-
1;
 }
  Form8->StringGrid1->Cells[0][0]="Nomer";
  Form8->StringGrid1->Cells[1][0]="So'zlar";
  Form8->StringGrid1->Cells[2][0]="Chastota";
// chastota
 if (RadioButton1->Checked==True) {
        long int n,i;
        n=Form8->StringGrid2->RowCount;
        for(i=1;i<n;i++){
         a[i]= Form8->StringGrid2->Cells[1][i];
         aa[i]=StrToInt(Form8->StringGrid2->Cells[2][i]);
        }
        qsort(0,n-1);
        ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                        ProgressBar1->Position=0;
                for(i=1;i<n;i++){
         Form8->StringGrid1->Cells[1][i]=a[i] ;
         Form8->StringGrid1->Cells[2][i]=aa[i];
         Form8->StringGrid1->Cells[0][i]=i;
         Form8->StringGrid1->RowCount++;
    ProgressBar1->Position+=2;   }
                }
 else
 if (RadioButton2->Checked==True) {
                    __int64 tartib,tartib1,i,k; int max5,j;
                        max5=StrToInt( Form8->StringGrid2->Cells[2][1]);
          tartib=Form8->StringGrid2->RowCount;
         for(i=2; i<tartib; i++){
                 if(max5<StrToInt(Form8->StringGrid2->Cells[2][i])){
                 max5=StrToInt(Form8->StringGrid2->Cells[2][i]); }
          }
                                        k=1;
                        ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                        ProgressBar1->Position=0;
        for(j=max5; j>=1; j--){
          for(i=1; i<tartib; i++){
                        if(StrToInt(Form8->StringGrid2->Cells[2][i])==j){
                                Form8->StringGrid1->RowCount++;
                                Form8->StringGrid1->Cells[0][k]=k;
                Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
```

```
                Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                        k++;
                    }}      ProgressBar1->Position+=2;    }    }
// HArflar buyicha
else{
        if (RadioButton3->Checked==True) {
         String s;       long int k=1;
         // a harfi
                switch(ComboBox1->ItemIndex){
                case 1:
                  ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                  ProgressBar1->Position=0;
                      for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                            s=Form8->StringGrid2->Cells[1][i];
                            if(s[1]=='a'){
                            Form8->StringGrid1->RowCount++;
               Form8->StringGrid1->Cells[0][k]=k;
               Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
               Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                            k++;
                            }   ProgressBar1->Position+=2;}    break;
                case 2:
                ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                  ProgressBar1->Position=0;
                      for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                            s=Form8->StringGrid2->Cells[1][i];
                            if(s[1]=='b'){
                              Form8->StringGrid1->RowCount++;
                              Form8->StringGrid1->Cells[0][k]=k;
                Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
               Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                              k++;
                            }   ProgressBar1->Position+=2;}  break;
                case 3:
                ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                  ProgressBar1->Position=0;
                      for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                            s=Form8->StringGrid2->Cells[1][i];
                            if(s[1]=='c'){
                              Form8->StringGrid1->RowCount++;
                              Form8->StringGrid1->Cells[0][k]=k;
               Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
               Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                              k++;
                            }   ProgressBar1->Position+=2;}  break;
```

```
        case 4:
        ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
              for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                    s=Form8->StringGrid2->Cells[1][i];
                    if(s[1]=='d'){
                      Form8->StringGrid1->RowCount++;
                      Form8->StringGrid1->Cells[0][k]=k;
        Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
        Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                      k++;
                    }   ProgressBar1->Position+=2;}  break;
        case 5:
        ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
              for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                    s=Form8->StringGrid2->Cells[1][i];
                    if(s[1]=='e'){
                      Form8->StringGrid1->RowCount++;
                      Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                      k++;
                    }   ProgressBar1->Position+=2;}  break;
        case 6:
        ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
              for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                    s=Form8->StringGrid2->Cells[1][i];
                    if(s[1]=='f'){
                      Form8->StringGrid1->RowCount++;
                      Form8->StringGrid1->Cells[0][k]=k;
 Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
 Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                      k++;
                    }   ProgressBar1->Position+=2;}  break;
        case 7:
        ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
              for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                    s=Form8->StringGrid2->Cells[1][i];
                    if(s[1]=='g'){
                      Form8->StringGrid1->RowCount++;
                      Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
```

```
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                    k++;
                  }   ProgressBar1->Position+=2;}  break;
         case 8:
         ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
               for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                     s=Form8->StringGrid2->Cells[1][i];
                     if(s[1]=='h'){
                       Form8->StringGrid1->RowCount++;
                       Form8->StringGrid1->Cells[0][k]=k;
        Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
        Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                       k++;
                     }   ProgressBar1->Position+=2;}  break;
         case 9:
         ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
               for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                     s=Form8->StringGrid2->Cells[1][i];
                     if(s[1]=='i'){
                     Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
        Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
        Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                       k++;
                     }   ProgressBar1->Position+=2;}  break;
         case 10:
         ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
               for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                     s=Form8->StringGrid2->Cells[1][i];
                     if(s[1]=='j'){
                       Form8->StringGrid1->RowCount++;
                       Form8->StringGrid1->Cells[0][k]=k;
        Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
        Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                       k++;
                     }   ProgressBar1->Position+=2;}  break;
         case 11:
         ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
          ProgressBar1->Position=0;
               for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                     s=Form8->StringGrid2->Cells[1][i];
                     if(s[1]=='k'){
```

```
                Form8->StringGrid1->RowCount++;
                Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                k++;
              }   ProgressBar1->Position+=2;}  break;
 case 12:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
       for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
              s=Form8->StringGrid2->Cells[1][i];
              if(s[1]=='l'){
              Form8->StringGrid1->RowCount++;
                Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                k++;
              }   ProgressBar1->Position+=2;}  break;
 case 13:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
       for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
              s=Form8->StringGrid2->Cells[1][i];
              if(s[1]=='m'){
              Form8->StringGrid1->RowCount++;
                Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                k++;
              }   ProgressBar1->Position+=2;}  break;
 case 14:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
       for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
              s=Form8->StringGrid2->Cells[1][i];
              if(s[1]=='n'){
              Form8->StringGrid1->RowCount++;
                Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                k++;
              }   ProgressBar1->Position+=2;}  break;
 case 15:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
```

```
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='o'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                  k++;
                }   ProgressBar1->Position+=2;}  break;
 case 16:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='p'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                  k++;
                }   ProgressBar1->Position+=2;}  break;
 case 17:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='q'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                  k++;
                }   ProgressBar1->Position+=2;}  break;
 case 18:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='r'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                  k++;
                }   ProgressBar1->Position+=2;}  break;
```

```
 case 19:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='s'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                  k++;
                }   ProgressBar1->Position+=2;}  break;
 case 20:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='t'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                  k++;
                }  ProgressBar1->Position+=2; }  break;
 case 21:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='u'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                  k++;
                }  ProgressBar1->Position+=2; }  break;
 case 22:
 ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
  ProgressBar1->Position=0;
        for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                s=Form8->StringGrid2->Cells[1][i];
                if(s[1]=='v'){
                Form8->StringGrid1->RowCount++;
                  Form8->StringGrid1->Cells[0][k]=k;
Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
```

```
                Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                        k++;
                      }  ProgressBar1->Position+=2; }  break;
                case 23:
                ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                  ProgressBar1->Position=0;
                      for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                            s=Form8->StringGrid2->Cells[1][i];
                            if(s[1]=='x'){
                            Form8->StringGrid1->RowCount++;
                              Form8->StringGrid1->Cells[0][k]=k;
               Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
               Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                              k++;
                            }  ProgressBar1->Position+=2; }  break;
                case 24:
                ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                  ProgressBar1->Position=0;
                      for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                            s=Form8->StringGrid2->Cells[1][i];
                            if(s[1]=='y'){
                            Form8->StringGrid1->RowCount++;
                              Form8->StringGrid1->Cells[0][k]=k;
               Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
               Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                              k++;
                            }  ProgressBar1->Position+=2; }  break;
                case 25:
                ProgressBar1->Max=Form8->StringGrid2->RowCount-1;
                  ProgressBar1->Position=0;
                      for (i = 1; i < Form8->StringGrid2->RowCount; i++) {
                            s=Form8->StringGrid2->Cells[1][i];
                            if(s[1]=='z'){
                            Form8->StringGrid1->RowCount++;
                              Form8->StringGrid1->Cells[0][k]=k;
               Form8->StringGrid1->Cells[1][k]=Form8->StringGrid2->Cells[1][i];
                Form8->StringGrid1->Cells[2][k]=Form8->StringGrid2->Cells[2][i];
                              k++;
                            } ProgressBar1->Position+=2; } break;
                      default: ShowMessage("Xato");
                      exit; }    }    }
Form8->ShowModal();
}
```