Obyektyönlü proqramlaşdırma

C++

Dərs №11

C++ dili ilə obyektyönlü proqramlaşdırma

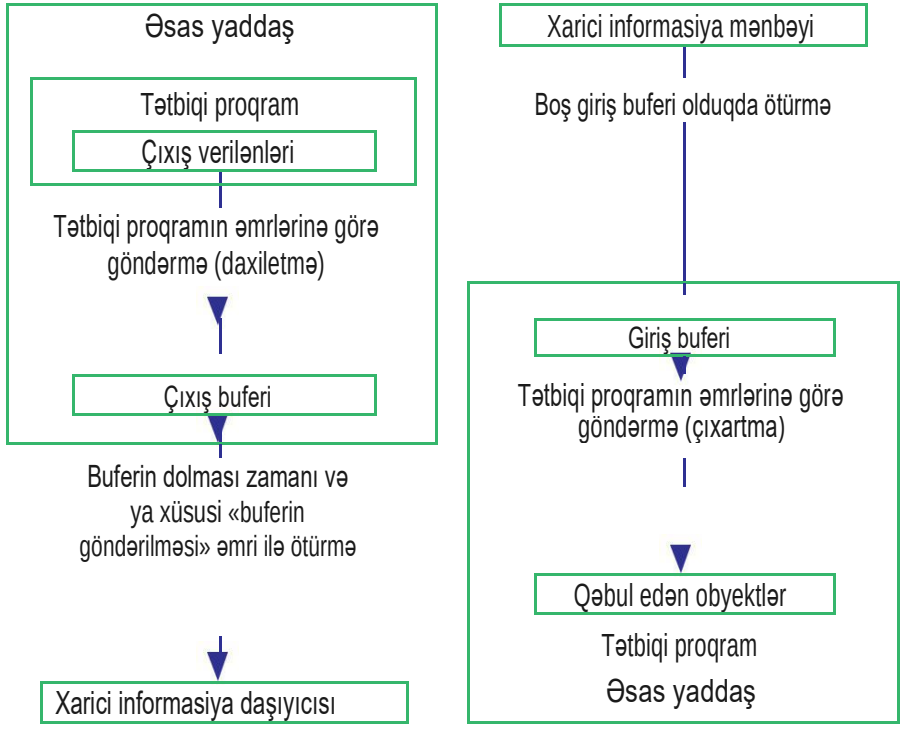
Mündəricat

Axın anlayışı

Bugün biz yenə də fayllar haqqında danışacağıq, lakin onlara başqa mövqedən baxaq. Başlanğıc üçün, bizə bir neçə yeni anlayış və sxemlərlə tanış olmaq lazımdır. Axın və onun iş prinsipi kimi.

Axın — verilənələrin mübadiləsinin aparıldığı qurğudan asılı olmayan baytlar ardıcıllığıdır.

Qeyd: Axınla verilənlərin mübadiləsi çox zaman əlavə yaddaş sahəsi – axın buferi ilə müşayiət olunur.



Şəkildə verilmış proseslər növbəti şəkildə baş verir:

- Proqram vasitəsilə daxil edilən verilənlər xarici qurğuya ötürülməzdən əvvəl onlar axın buferinə yerləşdirilirlər.
- Verilənlərin daxil edilməsi zamanı onlar əvvəlcə buferə yerləşdirilirlər və yalnız bundan sonra proqram tərəfindən icra olunan yaddaş oblastına ötürülürlər.
- Buferin istifadə edilməsi verilənləri ötürmə sürətini artırır, belə ki, əsl ötürmələr yalnız onda baş verir ki, bufer artıq dolmuş olsun (çıxış zamanı) və ya boş olsun (giriş zamanı).

Proqramlarda axınlar növbəti şəkildə təsnifləndirilir:

- *Giriş axınları* — informasiyanın oxunduğu axınlardır.
- *Çıxış axınları* — informasiyanın yazıldığı edildiyi axınlardır.
- *İkiistiqamətli axınlar* — həm oxuma, həm də yazmağa imkan verən axınlardır.

Qeyd edək ki, giriş-çıxış buferinin doldurulması və boşaladılması ilə əlaqədar olan işi əməliyyat sistemi praktik olaraq həmişə öz üzərinə götürür və proqramçının iştirakı olmadan yerinə yetirir. Nəzərə almaq lazımdır ki, axında hər hansı bir əlifbanın kodları ilə saxlanılan baytların qiymətləri ilə heç bir əlaqə nəzərdə tutulmamışdır.

Axınlar vasitəsilə giriş-çıxış zamanı proqramçının vəzifəsi obyektlər və axının baytlar ardıcıllığı (ötürülən informasiyanın tipi haqqında hər hansı bir məlumatın olmadığı) arasında əlaqə qurmaqdan ibarətdir.

Məhz bununla biz bugün məşğul olmağa çalışacağıq.

C++ dilinidə giriş-çıxış

Xatırladırıq ki, C dilinin və onun məntiqi davamı olan C++ dilinin ən başdan öyrənilməsi zamanı biz proqramlarımıza iostream kitabxanasını qoşmuşduq. Əslində, bu kitabxananın təyinatı sinif mexanizmi əsasında qurulmuş giriş-çıxış funksiyalarının bir hissəsini qoşmaqdır.

Bu faylda təsvir edilmiş giriş-çıxış vasitələri veriənlərin axınlardan çıxarılması və axınlara daxil edilməsi üçün bir vasitədir.

Qeyd: Adların mənasını verək: *iostream.h: stream* — axın, *"i"* — qısaldılmış input — giriş, *"o"* — qısaldılmış output — çıxış.

iostream kitabxanasında standart giriş-çıxış ilə əlaqələndirilmiş dört təyin edilmiş axın-obyekti var. Onlara baxaq:

1. **cin** — standart buferləşdirilmiş giriş axını (adətən klaviaturadan) ilə əlaqədar olan istream sinifinin obyekti.
2. **cout** — standart buferləşdirilmiş çıxış axını (adətən ekran) ilə əlaqədar olan ostream sinfinin obyekti.
3. **cerr** — səhvlər haqqında məlumatın ötürüldüyü standart buferləşdirilməmiş çıxış axını (adətən ekran) ilə əlaqədar olan ostream sinfinin obyekti.

4. **clog** — eyni ilə **cerr** kimi, lakin buferləşdirilmiş.

Proqrama faylın qoşulması zamanı iostream-də cin, cout, cerr və clog obyektlərini formalaşdırılması baş verir, yəni, uyğun standart axınlar yaradılır və proqramçıya üçün onlarla əlaqəli olan giriş-çıxış vasitələri əlyetərli olur.

iostream kitabxanası ilə əlaqədar əsas müddəalar

- C++ axın kitabxanası iki əsas giriş-çıxış sinfi: istream və ostream nəzərdə tutur.
- ostream sinfi çıxış üçün axına yazma əməliyyatı adlanan təyin edilmiş sol sürüşdürmə əməliyyatını (<<) istifadə edir.
- istream sinfi giriş üçün axından oxuma əməliyyatı adlanan təyin edilmiş sağ sürüşdürmə əməliyyatını (>>) istifadə edir.
- Yazma və oxuma əməliyyatları çağırışların birləşməsinə imkan verirlər, belə ki, axına istinad qiymətini qaytarırlar.
- istream və ostream sinifləri bütün verilənlər tipləri üçün yerləşdirmə və oxuma əməliyyatlarına yükləyir. Bu cür yükləmə simvolların, sətirlərin, tam və həqiqi ədədlərin giriş və çıxışı üçün vahid sintaksis istifadə etməyə imkan verir.

■■Yazma/oxuma əməliyyatlarını istifadəçinin verilənlər tipləri üçün də asanlıqla istifadə etmək olar. Bunun üçün növbəti başlıqlı iki funksiya təyin etmək lazımdır:

```
istream& operator >> (istream&, tipin_adı&);
ostream& operator << (ostream&, tipin_adı&);
```

Qeyd: Siniflər üçün yazma/oxuma əməliyyatlarını təyin edərək, verilmiş siniflərin bağlı elementlərinə müraciəti təmin etmək üçün onları adətən dost edirlər.

Elə indicə biz ekrana çıxış ilə bağlı axını gözdən keçirdik. Artıq fayllara keçmək zamanıdır.

Axınların tətbiqi ilə fayl giriş-çıxışı

Beləliklə, Siz yəqin ki, razılaşarsınız ki, fayl başlanğıcı və sonu olan adlandırılmış baytlar ardıcıllığıdır. C++ dilində fstream adlanan kitabxana da vardır.
Onun vasitəsilə növbəti funksiyaları yerinə yetirmək olar:

■■Faylın yaradılması
■■Axının yaradılması
■■Faylın açılması
■■Faylın axına «qoşulması»
■■Axın vasitəsilə faylla mübadilə
■■Axının fayldan «ayrılması»
■■Faylın bağlanması
■■Faylın silinməsi

Qeyd: fstream kitabxanasının düzgün işləməsi üçün **using namespace std**; istifadə etmək lazımdır;

fstream kitabxanası iostream kitabxanası kimi verilənlərin fayla giriş-çıxışı üçün nəzərdə tutulmuş üç sinif ehtiva edir:

■■**ofstream** — verilənlərin fayla çıxışı (yazılması) üçün.
■■**ifstream** — verilənlərin fayldan girişi (oxunması) üçün.
■■**fstream** — verilənlərin oxunması və yazılması üçün.

Bu üç sinfin hər biri üçün dörd konstruktor nəzərdə tutulmuşdur. Onları gözdən keçirək:

```
fstream() //faylı açmadan axın yaradır;
fstream(
        const char* name,       //faylın adı
        int omode,              //açılma rejimi
        int = filebuf::openprot;//faylın qorunması
  //axın yaradır, faylı açır və
 //onu axınla əlaqələndirir.

fstream(
        int f; // faylın deskriptoru - axın yaradır
        //və onu artıq açıq olan faylla əlaqələndirir


fstream(
        int f,                  //faylın deskriptoru
        char *buf,              //bufer
        int len                 //buferin ölçüsü
//əvvəlki konstruktor ilə eynidir, axına

//bufer təyin edilir.
```

Yuxarıda göstərilən parametrlər arasında omode parametrini qeyd etmək olar. Bu faylı açmaq üçün bayraqlar dəstidir:

```
enum _Openmode {
    in          = 0x01, //yalnız oxumaq üçün açamaq
    out         = 0x02, //yalnız yazmaq üçün açamaq
                        //göstəricini faylın sonuna
    ate         = 0x04, //gətirmək
                        //verilənləri faylın sonuna əlavə
                        //etmək
                        //faylı sıfır uzunluğuna qədər
    trunc       = 0x10, //kəsmək
                        //əgər fayl mövcud deyilsə,    =
    _Nocreate  0x40, //açılma səhvi
    _Noreplace = 0x80, //əgər fayl artıq mövcuddursa,
                        //açılma səhvi
                        //ikilik mübadilə üçün faylın
         binary = 0x20  //açlması
```


Qeyd: bayrağın təyin edilməsi üçün ona ios:: əlavə etmək lazımdır. Məsələn, ios::in.

Fayllar ilə işləmək üçün funksiyalar

```
void open(const char *fileName, int mode =
susmaya_görə_qiymət, int protection =
susmaya_görə_qiymət);
```

Fayl axınını konkret fayla birləşdirir.

fileName — artıq mövcud olan və ya yaradılacaq faylın adı. Bu əməliyyat sistemi tərəfindən təyin edilən faylın tam və ya qısaldılmış adını verir.

mode — açılma rejimi.

protection — faylın qotrunması.

Funksiya istənilən üç sinif axınının obyekti vasitəsi ilə çağırılır.

```
int close();
```

Funksiya axın buferini boşaldır, axını fayldan ayırır və faylı bağlayır.

Qeyd: Bu funksiya proqramın icrası başa çatdıqda avtomatik çağırılır.

```
istream&amp;amp; istream::read(unsigned char *buf, int len);
istream&amp;amp; istream::read(signed char *buf, int len);
```

Simvollar blokunun oxunmasını həyata keçirir.

len — axından **buf** buferinə çıxarılacaq simvolların maksimal sayı.

```
ostream&amp;
      osteram::write(const unsigned char *buf, int
n); ostream&amp;
      osteram::write(const signed char *buf, int n);
```

Simvollar blokunun yazılmasını həyata keçirir.

n — **buf** buferindən axına yerləşdiriləcək sıfır-simvol da nəzərə alınmaqla simvolların sayı.

```
int istream::get();
istream& istream::get(unsigned char&);

istream& istream::get(signed char&);
```

Axından bir simvol oxuyur.

```
ostream& ostream::put(char);
```

Axına bir simvol yazır.

```
istream& istream::get(unsigned char *buf, int
                      n, char c = '\n');
istream& istream::get(signed char *buf, int

                      n, char c = '\n');
```

Axından sətri oxumaq.

Yuxarıda şərh edilən bütün funksiyalarda simvollar məhdudlaşdırıcı simvol tapılana qədər və ya n simvol oxunana qədər, ya da faylın sonu rast gəlinənə qədər oxunur və buferdə yerləşdirilir. Məhdudlaşdırıcı axından oxunmur və buferdə yerləşdirilmir.


```
istream& istream::getline(unsigned char *buf, int
                          n, char c = '\n');
istream& istream::getline(signed char *buf, int

                          n, char c = '\n');
```

Verilmiş funksiya get kimi eyni işi görür, lakin məhdudlaşdırıcı axından oxunur (buverə yerləşdirilmir).

```
istream& istream::ignore(int n = 1, int d = EOF);
```

Bu funksiya d məhdudlaşdırıcısı rast gələnə qədər və ya n sayda simvol oxunana qədər axından simvolları oxuyur.

```
int istream::gcount();
```

Bu funksiya sonuncu formatsız giriş funksiyası ilə oxunmuş simvolların sayını qaytarır.

```
int istream::peek();
```

Bu funksiya giriş axınının növbəti simvoluna baxmağa imkan verir — növbəti simvolun kodunu qaytarır (və ya əgər axın boşdursa, EOF qaytarır), lakin bu simvolu axında saxlayır. Ehtiyac olduqda bu simvolu axından kitabxananın digər vasitələri ilə oxumaq olar.

```
istream& istream::putback(char cc);
```

Bu funksiya axından heç bir şey oxumur, ora cari olacaq və axından oxunacaq növbəti simvol olacaq cc simvolunu yerləşdirir.

```
istream& istream::seekg(long pos);
```

Bu funksiya parametrin qiyməti kimi axından oxuma mövqeyini qurur.

```
istream& istream::seekg(long off, ios::seek_dir dir);
```

Bu funksiya oxuma mövqeyini axın boyunca seek_dir {beg, cur, end}; sadalanan çoxluqdan qiymət alan dir parametri ilə təyin edilən istiqamətdə yerdəyişməni yerinə yetirir. Yerdəyişmənin nisbi kəmiyyəti (baytlarla) long off parametrinin qiyməti ilə təyin edilir. Əgər istiqamət beg kimi təyin edilibsə, onda axının əvvəlindən yerdəyişmə baş verir; cur — cari mıvqeyindən; end — axının sonundan.

```
ostream& ostream::seekp(long pos);
```

Bu funksiya axına yazmanın mütləq mövqeyini qurur.

```
ostream& ostream::seekp(long off, ios::seek_dir dir);
```

Bu funksiya seekg() funksiyası kimidir, lakin ostream sinfinə məxsusdur və axına yazmanın nisbi yerdəyişməsini yerinə yetirir.

```
long istream::tellg();
```

Bu funksiya axından oxumanın cari mövqeyini təyin edir.


```
long ostream::tellp();
```

Bu funksiya axına yazmanın cari mövqeyini təyin edir.

Nəhayət, nəzəriyyə kifayətdir — artıq təcrübəyə keçmək lazımdır. Dərsin növbəti bölməsində biz bir neçə nümunəyə baxacağıq.

C++ üslubunda faylların istifadəsinə aid praktik nümunə

Artıq təcrübəyə keçmək vaxtıdır. İndi biz sizinlə C++ dilinin vasitələrini istifadə edərək, fayla onaltılıq şəkildə baxmağı reallaşdıran nümunəyə baxacağıq.

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string.h>
#include <conio.h>
using namespace std;

//faylın yolununmaksimal uzunluğu
#define MAX_PATH 260
//Ekrandakı sütunların sayı
#define NUM_COLS 18
//Ekrandakı sətirlərin sayı
#define NUM_ROWS 24

void main()
{
    char path[MAX_PATH];
    //Faylın yolunu soruşuruq
    cout << "Input path to file: ";
    cin.getline(path, MAX_PATH);

    int counter = 0, i = 0, j = 0;
    char text[NUM_COLS];
```



```
//Faylın ikilik rejimdə açılması
ifstream input(path, ios::in |
ios::binary); if (!input)
{
    cout << "Cannot open file for display!" <<
    endl; return;
}

//Onaltılıq çıxış üçün üst registrdə əks
//etdirmə rejimi
cout.setf(ios::uppercase);

//Nə qədər ki, faylın sonu deyil, onda verilənləri
//oxuyuruq və ekranda formatlaşdırılmış
//şəkildə əks etdiririk
while (!input.eof())
{
    //Simvol-simvol oxuma
        for (i = 0; (i < NUM_COLS && !input.eof()); i++)
           input.get(text[i]);

    if (i < NUM_COLS)
        i--;
    for (j = 0; j < i; j++)
        if((unsigned)text[j] < 0x10)
         //Ədədi əks etdirmək üçün simvolların
         //sayı ikidən asılıdır?
         cout << setw(2) << 0 << hex << unsigned)
        text[j]; else
           cout << setw(3) << hex << (unsigned)
    text[j]; //Tamamlanmamış sətir üçün
    //bərabərləşdirmə
    for (; j < NUM_COLS; j++)
        cout << "      ";
    cout << "\t";
```

```
        for (j = 0; j < i; j++)
            //Simvol idarəedici deyil?
            if(text[j] > 31 && text[j] <= 255)
                cout << text[j];
            else
                cout << ".";
        cout << "\n";

        //Əgər ekran artıq dolmuşdursa, dayandırırıq

        if (++counter == NUM_ROWS)
        {

            counter = 0;
            cout << "Press any key to continue..."
                    << flush;

            //İstənilən klavişin
            //basılmasını gözləyirik
            getch();
            cout << "\n";
        }
    }
    //faylı bağlayırıq
    input.close();
}
```


Praktik nümunə. Fayldan massivin girişi/fayla çıxışı

Massivi fayla yazama və fayldan oxumanı yerinə yetirən proqram nümunəsinə baxaq. İnformasiya növbəti formada saxlanılır:

- Birinci sətir — masssivin ölçüsü (boşluqla ayrılmış sətir və sutunların sayı).
- Bütün növbəti sətirlər — massivin elementləri.

```
#include <windows.h>
#include <fstream>
#include <iostream>
#include <iomanip>
using namespace std;

void main()
{
    char Answer;
    const MessageCount = 8;
    int i, j;

    //Kömək
        enum {CHOICE = 3, INPUT_FILENAME,
              INPUT_DIMENSIONS,
              INPUT_ELEMENTS, FILE_ERROR};

    //İsmarış
    char Msg[MessageCount][50] =
    {
```

```
        "1. Verilənləri mətn faylında oxuma\n",
        "2. Verilənlərin mətn faylına yazma\n",
        "3. Proqramdan çıxış\n",
        "\nSizin seçiminiz: ",
        "Emal olunan faylın adını daxil edin: ",
        "Matrisin ölçüsünü verin:\n",
        "Matrisin elementlərini daxil edin:\n",
        "Faylı açmaq mümkün deyil\n"
    };

    //İsmarışın rusifikasiyası
    //və menyunun ekrana çıxarılması

    for(i = 0; i < MessageCount; i++)
        CharToOem(Msg[i], Msg[i]);

    do
    {
        for(int i = 0; i < 4; i++)
            cout << Msg[i];
        cin >> Answer;
    } while (Answer < '1' || Answer > '3');

    if(Answer == '3')
        return;

    //Faylın adı üçün dəyişən
    char FileName[80];

    //Matrisin ölçüsü
    int M, N;

    int num;
    cout << "\n" << Msg[INPUT_FILENAME];
    cin >> FileName;
    //Əgər menyunun birinci bölməsi seçilmişdirsə,
    // onda verilənləri mətn faylından ekran veririk
```



```
    if(Answer == '1')
    {
        //Əgər verilmiş adla fayl mövcud deyilsə,
        //səhv haqqında ismarış cıxarırırq
        ifstream inF(FileName, ios::in |
        ios::_Nocreate); if (!inF)
        {
            cout << endl << Msg[FILE_ERROR];
            return;
        }
        //Massivin ölçüsünü oxuyuruq
        inF >> M;
        inF >> N;
        //Massivin elementlərini fayldan
        //oxuyuruq və onları ekrana veririk
        for (i = 0; i < M; i++)
        {
            for (j = 0; j < N; j++)
            {
                inF >> num;
                cout << setw(6) << num;
            }
            cout << endl;
        }
        inF.close();
    }

    //Əgər menyunun ikinci bılməsi seçilərsə, onda
    //istifadəçidən verilənləri soruşuruq və onları
    //mətn faylına yazırıq
    else
    {
        //Faylı yazmaq üçün açırıq.
        //Əgər verilmiş adla fayl mövcud deyisə,
        //onda proqram onu yaradır
        ofstream outF(FileName, ios::out);
        if (!outF)
        {
```

```
            cout << "\n" << Msg[FILE_ERROR];
            return;
        }
        //Matrisin ölçüsünü soruşuruq və
        //verilənləri fayla yazırıq
        cout << Msg[INPUT_DIMENSIONS];
        cout << "M: ";
        cin >> M;
        cout << "N: ";
        cin >> N;

        outF << M << ' ' << N << "\n";

        cout << Msg[INPUT_ELEMENTS];
        //Massivin elementlərini soruşuruq və
        //onları fayla yazırıq
        for (i = 0; i < M; i++)
        {
            for(j = 0; j < N; j++)
            {
               cout << "A[" << i << "][" << j << "]
               = "; cin >> num;
               outF << num << " ";
            }
            outF << "\n";
        }
        outF.close();
    }
}
```


Sinif obyektinin fayla yazılmasına aid praktik nümunə

İndi isə, bugün öyrəndiyimiz sinfin obyektinin sahələrinin məzmununun fayla yazılması vasitələrini istifadə etməyə çalışıaq. Bizim proqram əvvəlki qiymətləri silmədən əlavə etməni yerinə yetirəcək. Qeyd edək ki, verilmiş nümunədə, fayl diskdə artıq mövcud olmalıdır, buna görə də onu əvvəlcədən yaratmağı unutmayın.

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <windows.h>
using namespace std;

void RussianMessage(char *message){
    char rmessage[256];
    CharToOem(message,rmessage);
    cout<<rmessage;
}
int RussianMenu(){
        RussianMessage("\nYeni strukturu fayla əlavə
        etmək üçün 1 daxil edin\n");
    RussianMessage("Bütün strukturu fayldan
                    oxuyub göstərmək üçün 2
                    daxil edin\n");
    RussianMessage("Çıxış üçün 3 daxil
   edin\n"); int choice;
   cin>>choice;
```

```
    return choice;
}

class Man{
    //yaş üçün dəyişən
    int age;
    //ad üçün dəyişən
    char *name;
    //soyad üçün dəyişən
    char *surname;
public:
    //parametrli konstruktor
    Man(char *n,char *s,int a);
    //konstruktor
    Man();
    //destruktor
    ~Man();
public:
    //verilənlərin daxil edilməsi funksiyası
    void Put();
    //verilənlərin göstərilməsi funksiyası
    void Show();
    //faylın saxlanılması funksiyası
    void SaveToFile();
    //Faylın məzmununun göstərilməsi funksiyası
    static void ShowFromFile();
};
//konstruktor
Man::Man(){
    name=0;
    surname=0;
    age=0;
}
//parametrli konstruktor
Man::Man(char *n,char *s,int a){
    name=new char[strlen(n)+1];
    if(!name){
```



```
        RussianMessage("Yaddaşın ayrılması zamanı
        səhv!!!");
        exit(1);
    }
    strcpy(name,n);
    surname=new char[strlen(s)+1];
    if(!surname){
     RussianMessage("Yaddaşın ayrılması zamanı səhv!!!");
        exit(1);
    }
    strcpy(surname,s);

    age=a;
}

    //destruktor
    Man::~Man(){
        delete[] name;
        delete[] surname;
    }

    //verilənlərin daxil edilməsi funksiyası
    void Man::Put(){
        char temp[1024];
        RussianMessage("\nAdı daxil edin:\n");
    cin>>temp;

    if(name)
        delete[] name;

    name=new char[strlen(temp)+1];
    if(!name){
    RussianMessage("Yaddaşın ayrılması zamanı səhv!!!");
        exit(1);
    }
    strcpy(name,temp);
    RussianMessage("\nSoyadı daxil edin:\n");
    cin>>temp;
```

```
    if(surname)
        delete[] surname;

    surname=new char[strlen(temp)+1];
    if(!surname){
   RussianMessage("Yaddaşın ayrılması zamanı səhv!!!");
   exit(1);
    }
    strcpy(surname,temp);

    RussianMessage("\nYaşı daxil edin\n");
    cin>>age;
}

//Verilənləri ngöstərilməsi funksiyası
void Man::Show(){
    RussianMessage("\nAd:\n");
    cout<<name;
    RussianMessage("\nSoyad:\n");
    cout<<surname;
    RussianMessage("\nYaş:\n");
    cout<<age<<"\n";
}
//Faylda saxlama funksiyası
void Man::SaveToFile(){
    int size;
    fstream f("men.txt",ios::out|ios::binary|ios::app);
    if(!f){
       RussianMessage("Fayl yazmaq üçün açılmadı!!!");
        exit(1);
    }
    //yaşı yazırıq
    f.write((char*)&age,sizeof(age));
    size=strlen(name);
    //Addakı simvolların sayını yazırıq

    f.write((char*)&size,sizeof(int));
```



```
    //Adı yazırıq
    f.write((char*)name,size*sizeof(char));
    size=strlen(surname);
    //Soyaddakı simvollarin sayını yazırıq
    f.write((char*)&size,sizeof(int));
    //Soyadı yazırıq
    f.write((char*)surname,size*sizeof(char));
    f.close();
}

//Faylın məzmununu göstərmə funksiyası
void Man::ShowFromFile(){
    fstream f("men.txt",ios::in|ios::binary);
    if(!f){
        RussianMessage("Fayl oxumaq üçün
                        açılmadı!!!");
        exit(1);
    }
    char *n,*s;
    int a;
    int temp;
    //Dövrdə faylın məzmununu oxuyuruq

    while (f.read((char*)&a,sizeof(int))){

        RussianMessage("\nAd:\n");

        f.read((char*)&temp,sizeof(int));
        n=new char[temp+1];
        if(!n){
            RussianMessage("Yaddaşın ayrılması
                            zamanı səhv!!!");
            exit(1);
        }
        f.read((char*)n,temp*sizeof(char));
        n[temp]='\0';
        cout<<n;
```

```
        RussianMessage("\nSoyad:\n");
        f.read((char*)&temp,sizeof(int));
        s=new
        char[temp+1];
        if(!s){
            RussianMessage("Yaddaşın ayrılması
                            zamanı səhv!!!");
            exit(1);
        }
        f.read((char*)s,temp*sizeof(char));
        s[temp]='\0';
        cout<<s;

        RussianMessage("\nYaş:\n");
        cout<<a<<"\n";
        delete []n;
        delete []s;
    }
}

void main(){
    Man *a;
    //Proqramın əsas dövrü
    do{
        switch(RussianMenu()){
        case 1:  //Yazının əlavə edilməsi
            a=new Man;
            a->Put();
            a->SaveToFile();
            delete a;
            break;
        case 2:  //Bütün yazının göstərilməsi
            Man::ShowFromFile();
            break;
        case  3: //İstifadəçi  ilə  vidalaşma
            RussianMessage("Görüşənədək\n")
            ; return;
```



```
        default: //Yanlış giriş
            RussianMessage("Yanlış Giriş\n");
        }

    }while(1);
}
```

Ev tapşırığı

1. **SORĞUKİTABÇASI sinfini növbəti sahələri ilə yaratmalı:**
   a) Firmanın adı;
   b) Sahibi;
   c) Telefon;
   d) Ünvan;
   e) Fəaliyyət formasi.
2. **Növbəti imkanları reallaşdırmalı:**
   ■■Ada görə axtarış;
   ■■Sahibə görə axtarış;
   ■■Telefon nömrəsinə görə axtarış;
   ■■Fəaliyyət sahəsinə görə axtarış;
   ■■Bütün yazıların göstərilməsi və əlavə edilməsi.

Bütün məlumatlar, təbii olaraq, əlavə etmə imkanı olmaqla faylda saxlanılır.