Obyektyönlü proqramlaşdırma

C++

ШАГ
компьютерная
АКАДЕМИЯ

Dərs №9

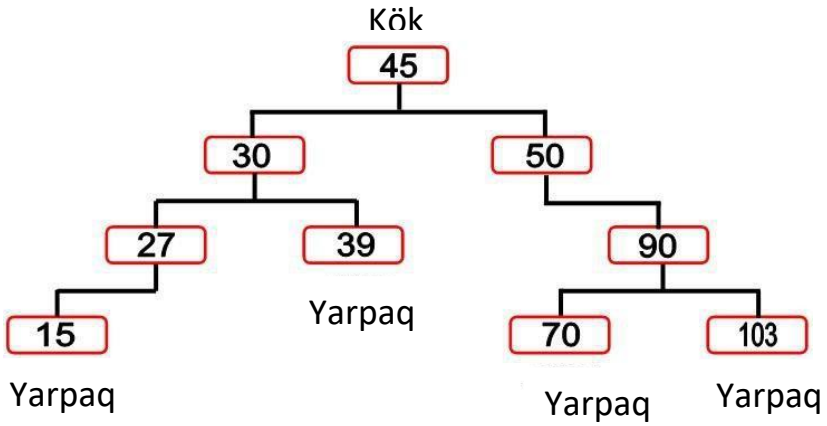
C++ dili ilə obyektyönlü c proqramlaşdırma

Mündəricat

İkilik ağac

Bugün biz sizinlə yeni, xətti olmayan verilənlər strukturu ilə tanış olacağıq. Bu struktur ikilik (və ya binar) struktur adlanır. Başlanğıc üçün strukturun təyinini verək, sonra isə onunla işləmək üçün istifadə edilən bir neçə terminə baxaq.

İkilik ağac (*binary tree*) **— *bu sıralanmış ağacvari strukturdur.*** Ağacın hər bir elementi (düyün) ondan sonra gələn ən çox iki element (nəsil) ehtiva edir və ozundən əvvəl 1-dən çox olmayan element (valideyn) ehtiva edir. İkilik ağacın sxematik təsvirinə baxaq:



Ağacın təsvirinin şərhi. Terminologiya.

1. İkilik ağacın əsas qurulma prinsipi ondan ibarətdir ki, hər bir düyün üçün növbəti qayda yerinə yetirilir: sol budaqda yalnız o açarlar yerləşir ki, qiyməti verilmiş düyünün qiymətindən kiçik olsun. Sağ budaqda yalnız o açarlar yerləşir ki, qiyməti verilmiş düyünün qiymətindən böyük olsun.

2. Hər bir düyün iki, bir və ya heç bir nəsil ehtiva edə bilər.
3. ***Yarpaq*** — nəsli olmayan düyündür.
4. Düyün öz nəsilləri üçün valideyn, əjdadları üçün isə oğullar hesab edilirlər.
5. ***Sol nəsil*** — cari düyündən solda olan oğul düyünü.
6. ***Sağ nəsil*** — cari düyündən sağda olan oğul düyünü.
7. ***Kök*** — valideynləri olmayan əsas (ilk) düyün.
8. Hər bir düyün dörd hissədən ibarətdir:
   - Qiymət.
   - Valideyn göstəricisi.
   - Sol nəsil göstəricisi.
   - Sağ nəsil göstəricisi.

Qeyd: İkilik ağac rekursiv strukturdur, belə ki, onun hər bir altağaçı ikilik ağacdır və nəticə etibarilə onun hər bir düyünü öz növbəsində sərbəst ağacın köküdür.

Valideyn göstəricisi	
Qiymət	
Sol nəsil göstəricisi	Sağ nəsil göstəricisi

Ağacla işin təşkili

Ağaclarla iş zamanı adətən rekursiv alqoritmlərdən istifadə edilir. Rekursiv funksiyaların istifadə edilməsi az səmərəlidir, belə ki, funksiyanın çox saylı çağırılması system resurslarını çox istifadə edir. Buna baxmayaraq, bu halda rekursiv funksiyaların istifadəsinə haqq qazandırmaq olar, belə ki, ağaclarla işləmək üçün rekursiv olmayan funksiyaların həm yazılması, həm də proqram kodunda əks etdirilməsi çox mürəkkəbdir. Burda biz ağacla işləmək üçün sxematik alqoritimlər verəcəyik. Lakin başa düşülən praktik nümunəyə dərsin növbəti bölməsində baxa bilərsiniz. Sxemdə istifadə edəcəyimiz növbəti bir neçə qiymətlər verilmişdir:

- **x** — ikilik ağacın təpəsi
- **left[x]** — sol altağac
- **right[x]** — sağ altağac
- **key[x]** — açar
- **p[x]** — təpənin valideyni

Bütün ağacda dolaşmaq

Çap(x)
Başlanğıc

1. Əgər x NULL-a bərabər deyilsə
2. Onda çap(left[x])
3. Çap key[x]
4. Çap (right[x])

Son

Qiymətin axtarılması

Axtarış(x, k)
Başlanğıc

1. Nə qədər ki, x bərabər deyil NULL və k bərabər deyil key[x]
2. Başlanğıc
3. Əgər k kiçikdir key[x] olarsa
4. Onda x bərabərdir left[x]
5. Əks halda x bərabərdir right[x]
6. Son
7. x qaytar

Son

Minimum və maksimumun tapılması

Minimum(x)
Başlanğıc

1. Nə qədər ki, left[x] bərabər deyil NULL
2. Başlanğıc
3. x bərabərdir left[x]
4. Son
5. x qaytar

Son
Maksimum(x)
Başlanğıc

1. Nə qədər ki, right[x] bərabər deyil NULL
2. Başlanğıc
3. x bərabərdir right[x]
4. Son
5. x qaytar

Son

Sonuncu və əvvəlki elementin əldə edilməsi

NövbətiElementiƏldəEt(x)

Başlanğıc

1. əgər right[x] bərabər deyil NULL, onda Minimum(right[x]) qaytar
2. y bərabərdir p[x]
3. nə qədər ki, y bərabər deyil NULL və x bərabərdir right[x]
4. Başlanğıc
5. x bərabərdir y
6. y bərabərdir p[y]
7. y qaytar
8. Son

Son

ƏvvəlkiElementiƏldəEt(x)

Başlanğıc

1. əgər left[x] bərabər deyil NULL, onda Maksimum(left [x]) qaytar
2. y bərabərdir p[x]
3. nə qədər ki, y bərabər deyil NULL və x bərabərdir left[x]
4. Başlanğıc
5. x bərabərdir y
6. y bərabərdir p[y]
7. y qaytar
8. Son

Son

Qiymətin əlavə edilməsi

ƏlavəEt(T, z)

Başlanğıc

1. y bərabərdir NULL
2. x bərabərdir root[T]
3. nə qədər ki, x bərabər deyil NULL
4. Başlanğıc

5. y bərabərdir x
6. əgər key[z] kiçikdir key[x], onda x bərabərdir left[x]
7. əks halda x bərabərdir right[x]
8. Son
9. p[z] bərabərdir y
10. əgər y bərabərdir NULL, onda root[T] bərabərdir z
11. əks halda, əgər key[z] kiçikdir key[y], onda left[y] bərabərdir z
12. əks halda right[y] bərabərdir z

Son

Qiymətin silinməsi

Silmə(T, z)

Başlanğıc

1. əgər left[z] bərabərdir NULL və ya right[z] bərabərdir NULL, onda y bərabərdir z
2. əks halda y bərabərdir NövbətiElementiƏldəEt(z)
3. əgər left[y] bərabər deyil NULL, onda x bərabərdir left[y]
4. əks halda x bərabərdir right[y]
5. əgər x bərabər deyil NULL, onda p[x] bərabərdir p[y]
6. əgər p[y] bərabərdir NULL, onda root[T] bərabərdir x
7. əks halda, əgər y bərabərdir left[p[y]], onda left[p[y]] bərabərdir x
8. əks halda right[p[y]] bərabərdir x
9. əgər y bərabər deyil z, onda key[z] bərabərdir key[y]
10. y ilə əlaqəli əlavə verilənlərin nüsxəsini alırıq
11. y-i sil

Son

Tətbiq

Son olaraq qeyd edək ki, verilənlərin ikilik ağac vasitəsi ilə təşkili, çox zaman lazım olan elementin axtarış müddətini əhəmiyyətli dərəcədə azaltmağa imkan verir. Xətti verilənlər strukturunda elementin axtarılması adətən verilmiş strukturda olan bütün elementlərin müqayisəsi yolu ilə həyata keçirilir. İkilik ağacda axtarış bütün elementlərin müqayisə edilməsini tələb etmir, buna görə də əhəmiyyətli dərəcədə az zaman tələb olunur. İkilik ağacda axtarış zamanı addımların maksimal sayı ağacın yüksəkliyinə, yəni, ağacın ierarxik strukturundakı səviyyələrin sayına bərabərdir.

İkillik ağacın reallaşdırılması

Futbol üzrə İngiltərə çempionatında komanda oyunlarının nəticələrinin saxlanması üçün ikilik ağacdan istifadə edilməsi.

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>

using namespace std;
struct Elem
{
    int OwnerPoints;    //Ev sahibinin xalları
    int OppPoints;      //Rəqibin xalları
    char Match[10];     //Hesab
    char Name[20];      //Komanda
    char Opponent[20];  //Rəqib

    Elem * left, * right, * parent;
};
```



```
class Tree
{   //kök
    Elem * root;
public:
    Tree();
    ~Tree();
    //göstərilmiş düyündən çap
    void Print(Elem * Node);
    //göstərilmiş düyündən axtarış
    Elem * Search(Elem * Node, char * key);
    //göstərilmiş düyündən min
    Elem * Min(Elem * Node);
    //göstərilmiş düyündən max
    Elem * Max(Elem * Node);
    //göstərilmiş düyün üçün növbəti
    Elem * Next(Elem * Node);
    //göstərilmiş düyün üçün əvvəlki
    Elem * Previous(Elem * Node);
    //düyünün yerləşdirilməsi
    void Insert(Elem * z);
    //göstərilmiş düyün üçün budağın silinməsi,
    //0 – bütün ağacın silinməsi
    void Del(Elem * z = 0);
    //kökün əldə edilməsi
    Elem * GetRoot();
};

Tree::Tree()
{
    root = NULL;
}

Tree::~Tree()
{
    Del();
}
```

```
//Ağacda rekursiv dolaşma
void Tree::Print(Elem * Node)
{
    if(Node != 0)
    {
        Print(Node->left);
        cout << Node->Name
            << Node->Match
            << Node->Opponent
            << endl;
        Print(Node->right);
    }
}
Elem * Tree::Search(Elem * Node, char * k)
{
    //Nə qədər ki, düyünlər var və açarlar üst-
    //üstə düşmürlər
    while(Node != 0 && strcmp(k, Node->Name) !=
    0) {
        if(strcmp(k, Node->Name) < 0)
            Node = Node->left;
        else
            Node = Node->right;
    }
    return Node;
}

Elem * Tree::Min(Elem * Node)
{
    //Ən sol düyünün axtarılması
    if(Node != 0)
        while(Node->left != 0)
            Node = Node->left;
    return Node;
}
Elem * Tree::Max(Elem * Node)
{
```



```
    // Ən sağ düyünün axtarılması
    if(Node != 0)
        while(Node->right != 0)
            Node = Node->right;

    return Node;
}

Elem * Tree::Next(Elem * Node)
{
    Elem * y = 0;
    if(Node != 0)
    {
        //əgər sağ nəsil varsa
        if(Node->right != 0)
            return Min(Node->right);

        //düyünün valideyni
        y = Node->parent;
        //əgər Node kök deyilsə və Node sağdakıdırsa
        while(y != 0 && Node == y->right)
        {
            //Yuxarı hərəkət edirik
            Node = y;
            y = y->parent;
        }
    }
    return y;
}

Elem * Tree::Previous(Elem * Node)
{
    Elem * y = 0;
    if(Node != 0)
    {
        //əgər sol nəsil varsa
        if(Node->left != 0)
            return Max(Node->left);
```

```
        //düyünün valideyni
        y = Node->parent;
        //əgər Node kök deyilsə və Node soldakıdırsa
        while(y != 0 && Node == y->left)
        {
            //Yuxarı hərəkət edirik
            Node = y;
            y = y->parent;
        }
    }
    return y;
}

Elem * Tree::GetRoot()
{
    return root;
}

void Tree::Insert(Elem * z)
{
    //nəsil yoxdur
    z->left = NULL;
    z->right = NULL;

    Elem * y = NULL;
    Elem * Node = root;

    //yerin axtarılması
    while(Node != 0)
    {
        //gələcək valideyn
        y = Node;
        if(strcmp(z->Name, Node->Name) < 0)
            Node = Node->left;
        else
            Node = Node->right;
    }
```



```
    //valideyni doldururuq
    z->parent = y;

    if(y == 0) //sağ element(yeganədir)
        root = z;
    //hansinin acarı böyükdür?
    else if(strcmp(z->Name, y->Name) < 0)
        y->left = z;
    else
        y->right = z;
}

void Tree::Del(Elem * z)
{
    //qrupun silinməsi
    if(z != 0)
    {
        Elem * Node, * y;

        //2 oğlu deyil
        if(z->left == 0 || z->right == 0)
            y = z;
        else
            y = Next(z);

        if(y->left != 0)
            Node = y->left;
        else
            Node = y->right;

        if(Node != 0)
            Node->parent = y->parent;
        //Kök düyün silinirsə?
        if(y->parent == 0)
            root = Node;
        else if(y == y->parent->left)
```

```
            //Valideyndən soldakı?
            y->parent->left = Node;
        else
            //Valideyndən sağdakı?
            y->parent->right = Node;
        if(y != z)
        {
            //Düyünün verilənlərinin köçürülmüsi
            strcpy(z->Name, y->Name);
            strcpy(z->Opponent, y->Opponent);
            strcpy(z->Match, y->Match);
            z->OppPoints = y->OppPoints;
            z->OwnerPoints = y->OwnerPoints;
        }

        delete y;
    }
    else //bütün ağacın silinməsi
        while(root != 0)
            Del(root);
}

//Turnir cədvəli
Tree tournament;

void Game(char Commands[][20], int N)
{
    int i, j;
    int p1, p2; //Счет

    //hər bir komanda digəri ilə 2 dəfə oynayır
    //evdə və qonaq kimi
    int k;

    Elem * temp;
    for(k = 0; k < 2; k++)
        for(i = 0; i < N - 1; i++)
        {
```



```
            for(j = i + 1; j < N; j++)
            {
                temp = new Elem;
                if(k == 0)
                {
                    //1 oyun
                    strcpy(temp->Name, Commands[i]);
                  strcpy(temp->Opponent, Commands[j]);
                }
                else
                {
                   //2 oyun
                   strcpy(temp->Name, Commands[j]);
                   strcpy(temp->Opponent, Commands[i]);
                }

                p1 = rand() % 6;
                p2 = rand() % 6;

                if(p1 > p2)
                {
                   temp->OwnerPoints = 3;
                   temp->OppPoints = 0;
                }
                else if(p1 == p2)
                {
                   temp->OwnerPoints = 1;
                   temp->OppPoints = 1;
                }
                else
                {
                   temp->OwnerPoints = 0;
                   temp->OppPoints = 3;
                }

                //Hesabın yazılması
                sprintf(temp->Match, " %d : %d ", p1, p2);
```

```
                //Yazının əlavə edilməsi
                tournament.Insert(temp);
            }
        }
}

void main()
{
    srand(time(0));

    const N = 4;
    char Commands[4][20] =
    {
        "Arsenal",
        "Liverpool",
        "Lids United",
        "Manchester United"
    };

    //Oyun
    Game(Commands, N);
    //Nəticənin çap edilməsi
    tournament.Print(tournament.GetRoot());

}
```


Fayllar

Giriş

İndi isə, biz fayllar haqqında danışaq. Proqramçı üçün fayllarla işləmək böyük məna kəsb edir. Siz gözəl anlayırsınız ki, informasiyanın əməli yaddaşda uzun müddət saxlanılması mümkün deyildir. Fayl isə informasiyanı diskdə saxlamağa imkan verir ki, buda ona istənilən vaxt müraciət etməyə imkan verir. Buna görə də bu mövzunun öyrənilməsi hava-su kimi lazımdır. Başlayaq. Əvvəlcə özünüzə sual verin: "Bəs fayl nədir" Yəqin ki, sizin variantların sayı çoxdur. Lakin, gəlin bu anlayışın dəqiq tərifini verək. Beləliklə,

Fayl – informasiya daşıyıcısında saxlanılan adlandırılmış informasiya blokudur.

İstənilən fayl növbəti bir sıra xüsusiyyətlərə malikdir:

- Fayl diskdə kəsilməz saxlanıla bilməz, lakin istifadəçiyə fayl tam ardıcıl bayt informasiyası bloku kimi təqdim edilir.
- Faylın adında növbəti simvollar ola bilməz: < > : " / \ |.
- Əksər faylların genişlənməsi (əməliyyat sisteminin faylın tipini təyin etməsi üçün simvollar toplusu) olur. Genişlənmə vacib deyil.
- Hər bir faylın, məsələn, ona müraciət saviyyəsini təyin edən atributları olur.

Atributları istifadə edərək əməliyyat sistemi verilmiş faylla necə işləmək lazım gəldiyini bilir.

İndi isə bir neçə yeni termin daxil edək:

1. ***Faylın deskriptoru (təsviri)*** — istənilən açıq fayla əməliyyat sisteminin ona digər fayllardan fərqləndirmək üçün verdiyi unikal nömrədir. Fayl bağlanan zaman sistem onda deskriptoru alır. Məhz bu unikal ədədi bizim proqramlarımızda konkret faylla işləmək üçün istifadə edəcəyik.
2. ***Fayl göstəricisi*** — açıq fayla avtomatik mənimsədilən xüsusi dəyişəndir və faylın cari mövqeyini saxlayır. O fayla yazma və oxuma anında sürüşdürülüür. Böyük mənada, siz bu dəyişəni istənilən mətn redaktorundakı kursor kimi təsəvvür edə bilərsiniz.

Müxtəlif formalı fayllar mövcuddur.

Windows əməliyyat sistemində bu müxtəliflik iki cürdür: ***mətn faylları və binar (ikilik) fayllar.*** Bu təsnifləndirmə kriteriya kimi informasiyanın saxlanılmasını istifadə edir. Beləliklə:

İki kateqoriyanın müqayisəli cədvəli

Mətn faylı	İkilik fayl
Fayla yazılan ədəd mətin kimi saxlanılır, yəni məşğul edilən sahənin ölçüsü bu ədəddəki rəqəmlərin sayına bərabərdir	Fayla yazılan ədəd ikilik formatda saxlanılır və məşğul edilən sahənin ölçüsü bu ədədin verilənlər tipinin ölçüsünə bərabərdir

Mətn faylı	İkilik fayl
Verilmiş faylı istənilən mətn redaktorunda oxumaq olar	Faylı istənilən mətn redaktorunda onun xüsusi proqramı vasitəsilə oxumaq olar
Bir neçə baytın itməsi zamanı informasiyanı mənasına görə bərpa etmək olar	Bir neçə baytın itməsi ikilik fayla müraciəti mümkünsüz edə bilər
İnformasiyanın proqram vasitəsilə oxunması çətindir, belə ki, fayl özü-özlüyündə vahid test bloku kimidir	İnformasiyanın proqram vasitəsilə oxunması sadələşdirilmişdir, belə ki, fayl özü-özlüyündə konkret verilənlər tipi olan informasiya bloku dəstidir
Öxumaq, redaktə etmək, çap etmək faylı kimi istifadə edilir	Proqram vasitəsilə informasiyanı rahat oxumaq, saxlamaq, redaktə etmək, yazmaq faylı kimi istifadə edilir

Qeyd: Faylı mətn və ya ikilik rejimdə acmaq olar. Diqqət, bunu mətn və ikilik formatla eyniləşdirmək olmaz!!! Açma rejimləri arasındakı fərq ondan ibarətdir ki, faylı ikilik rejimdə açarkən informasiya diskdə və ya yaddaşda olduğu kimi olacaq. Mətn rejimində isə, sətrin sonu işarəsi (\n) iki (\r\n) işarələri ilə əvəz ediləcəkdir. MS-DOS və Windows əməliyyat sistemlərinin birləşməsindən asılı deyil.

Fayllar ilə işləmək üçün C dillinin kitabxana funksiyaları

Fayllar nəzəriyyəsinə qısa səyahətdən sonra artıq praktikaya keçmək vaxtıdır. Dərsin bu bölməsi fayllarla işləmək üçün funksiyaların təsvirinə həsr edilir. Bu funksiyalar bizə fayllardan istifadə etməklə məsələlərin həlli üçün lazım olacaq.

stdio.h kitabxanasının funksiyaları

```
FILE *fopen(const char *filename, const char *mode)
```

Funksiya faylı açır.

filename — faylın yolu.
mode — müraciət tipi.

- **r** — oxuma, əgər fayl mövcud deyilsə, onda bu funksiya səhv emal edir və sıfır qiymətini qaytarır.
- **w** — yazma, əgər fayl mövcud deyilsə, onda o yaradılır, mövcuddursa, faylın cari məzmunu silinir.
- **a** — sona əlavə etmə, əgər fayl mövcud deyilsə, onda o yaradılır.
- **r+** — yazma və oxuma (fayl mövcud olmalıdır).
- **w+** — oxuma və yazma (iş prinsipi w-də olduğu kimidir).
- **a+** — əlavə etmə və oxuma (iş prinsipi a-da olduğu kimidir)..

Qeyd: Bütün yuxarıda şərh edilən rejimlər faylın mətin kimi açılışı üçündürlər. Faylın ikilik açılışı üçün rejimin əvvəlinə b əlavə etmək kifayətdir. Məsələn, br.

Əgər funksiya müvəffəqiyyətlə icra edilərsə, ondan faylın göstəricisi qaytarılır, əks halda sıfır qaytarılır.

Qeyd: Açıq fayl göstəricisini FILE* verilənlər tipində saxlamaq lazımdır.

```
int fclose( FILE *stream )
```

Funksiya faylı bağlayır.

stream — bağlanan faylın göstəricisi.

Əgər hər şey müvəffəqiyyətlə baş verirsə, onda bu funksiya 0 qaytarır və ya səhv baş verdiyi halda EOF qaytarır.

Qeyd: EOF (End Of File) — faylın sonuna işarədir.

```
char *fgets( char *string, int n, FILE *stream )
```

Sətri cari mövqedən başlayaraq oxuyur.
Oxuma dayandırılır:

- .... .Əgər yeni sətrə keçid simvolu tapılarsa (o sətrə yerləşdirilir);
- .... .əgər faylın sonuna gəlinərsə;
- .... .əgər n-1 simvol oxunarsa.

string — oxunan verilənlərin düşdüyü sətir, n — string-dəki elementlərin sayıdır.
stream — açıq faylın göstəricisi.

Əgər hər şey müvəffəqiyyətlə baş verərsə, funksiya oxunan sətri qaytarır, əks halda səhv baş verərsə və ya faylın sonuna gəlinərsə, 0 qiyməti qaytarılır.

```
int fputs( const char *string, FILE *stream )
```

Sətri cari mövqedən başlayaraq fayla yazır.
string — yazmaq üçün sətir.
stream — yazı əməliyyatının aparılacağı açıq faylın göstəricisi.

Əgər funksiya müvəffəqiyyətlə icra edilərsə, ondan mənfi olmayan qiymət qaytarılır. Səhv zamanı EOF. qaytarılır.

```
size_t fread( void *buffer, size_t size,
              size_t count, FILE *stream )
```

Funksiya verilənləri fayldan buferə oxuyur.

buffer — verilənlərin yazılacağı massivin ünvanı.

size — massivin elementinin baytlarla ölçüsü.

count — oxumaq üçün elementlərin maksimal sayı.

stream — Açıq faylın göstəricisi.

Funksiya oxunan baytların sayını qaytarır.

Qeyd: size_t verilənlər tipi stdio.h kitabxanasında növbəti şəkildə təyin edilmişdir: **typedef unsigned int size_t**;. Başqa sözlə, bu adi işarəsiz int tipidir.

```
size_t fwrite( const void *buffer, size_t size,
               size_t count, FILE *stream )
```

Funksiya verilənələr massivini fayla yazır.

buffer — verilənlərin saxlandığı massivin ünvanı.

size — massivin elementinin baytlarla ölçüsü.

count — fayla yazılmaq üçün elementlərin maksimal sayı.

stream — açıq faylın göstəricisi.

Funksiya yazılan baytların sayını qaytarır.


```
int feof( FILE *stream )
```

Faylın sonuna gəlindiyini yoxlayan funksiya.

stream — açıq faylın göstəricisi.

Faylın sonuna gəlindisə, funksiya sıfır olmayan qiymət qaytarır, əks halda sıfır qaytarır.

```
int _fileno( FILE *stream )
```

Bu funksiya faylın deskriptorunu qaytarır.

stream — açıq faylın göstəricisi.

```
int fseek ( FILE *stream, int offset [, int whence] )
```

Faylda yerdəyişməni təyin edir.

stream — açıq faylın göstəricisi.

offset — yerdəyişmə faylın əvvəlindən baytlarla ölçülür.

whence — yerdəyişmənin başladığı nöqtə.

- SEEK_SET (0) — faylın başlanğıcı,
- SEEK_CUR (1) —fayl göstəricisinin cari mövqeyi,
- SEEK_END (2) — faylın sonu (EOF).

Əgər faylın göstəricisi müvəffqiyyətlə yerdəyişmə etmiş olarsa, funksiya sıfır qiymətini qaytarır, əks halda sıfır olmayan qiymət qaytarır.

io.h kitabxanasının funksiyaları

```
int _access( const char *path, int mode )
```

Funksiya faylın genişlənməsini və ya kataloqu qaytarır.

path — faylın və ya kataloqun yolu.

mode — yoxlamaq üçün bayraqlar.

- 00 — mövcudluğun yoxlanılması,
- 02 — yazmağa icazənin yoxlanılması,
- 04 — oxumağa icazənin yoxlanılması,
- 06 — oxuma və yazmağın yoxlanılması.

Əgər icazə varsa, funksiya sıfır qaytarır, əks halda 1 qaytarır.

Qeyd: Kataloqların yalnız mövcudluğunu yoxlamaq olar.

```
long _filelength( int handle )
```

Faylın ölçüsünü baytlarla qaytarır.

handle — faylın deskriptoru.

Səhv baş verdiyi zaman funksiya 1 qaytarır.

```
int _locking( int handle, int mode, long nbytes )
```

Faylın cari mövqeyindən başlayaraq baytları təcrid edir və ya təcridi aradan qaldırır.

handle — faylın deskriptoru mode — təcrid etmə tipi.

- _LK_LOCK — baytları təcrid edir, əgər baytları təcrid etmək mümkün olmursa, bir saniyədən sonra yenidən cəhd edilir. Əgər 10 cəhtdən sonra baytların təcridi aradan qaldırılmazsa, funksiya səhv hasil edir və 1 qaytarır,
- _LK_NBLCK — baytları təcrid edir, əgər baytları təcrid etmək mümkün olmursa, funksiya səhv hasil edir və 1 qaytarır,
- _LK_NBRLCK — eynilə _LK_NBLCK kimi,
- _LK_RLCK — eynilə _LK_LOCK kimi,
- _LK_UNLCK — Öncədən təcrid edilmiş baytların təcridinin aradan qaldırılması.

nbytes — təcrid etmək üçün baytların sayı.

Əgər səhv baş verərsə, locking funksiyası 1 qaytarır, əks halda 0 qaytarır.

Qeyd: Bu funksiyanın işləməsi üçün io.h kitabxanasından başqa sys/locking.h kitabxanasını da daxil etmək lazımdır.

Proqram nümunəsi. Faylların köçürülməsi.

Məsələ. Bir faylın digərinə köçürülməsini təmin edən proqramı reallaşdırmalı. İstifadəci klaviaturadan köçürüləcək faylın yolunu, həmçinin, nüsxə üçün yol və adı daxil edir.

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

#include <io.h>
```

```
#include <stdio.h>
using namespace std;

//Ekrana sətir çıxaran funksiya
void RussianMessage(char *str){
    char message[100];
    //sətrin Windows kodlaşdırmadan MS-DOS
    //kodlaşdırmaya çevrilməsi
    CharToOem(str,message); cout<<message;
}
//Faylın köçürülməsi funksiyası
bool CopyFile(char *source,char
    *destination){ const int size=65536;
    FILE *src,*dest;
    //Faylın açılması
    if(!(src=fopen(source,"rb"))){
        return false;
    }
    //Faylın deskriptorunun əldə edilməsi
    int handle=_fileno(src);

    //Bufer üçün yaddaşın ayrılması
    char *data=new char[size];
    if(!data){
        return false;
    }

    //köçürmənın baş verəcəyi zaman faylın açılması
    if(!(dest=fopen(destination,"wb"))){
        delete []data;
        return false;
    }
    int realsize;

    while (!feof(src)){
        //Verilənlərin fayldan oxunması
        realsize=fread(data,sizeof(char),size,src)
        ;
        //Verilənlərin fayla yazılması
        fwrite(data,sizeof(char),realsize,dest);
    }

    //Faylların bağlanması
    fclose(src);
    fclose(dest);
    return true;
}
void main(){
    //MAX_PATH - yolun maksimal uzunluğunu təyin edən
    //sabit (konstant).
    //Verilmiş sabit stdlib.h kitabxanasında yerləşir
    char source[_MAX_PATH]; char
    destination[_MAX_PATH];
    char answer[20];
    RussianMessage("\nKöçürülən faylın yolunu və adını
    daxil edin:\n");
    //Birinci faylın yolunun əldə edilməsi
    cin.getline(source,_MAX_PATH);
    //Faylın mövcudluğunun yoxlanılması
    if(_access(source,00)==-1){
            RussianMessage("\nFaylın yolu və ya adı
                            düzgün verilməyib\n");
        return;
    }

    RussianMessage("\nYeni faylın yolunu və adını
                    daxil edin:\n");
    //İkinci faylın yolunun əldə edilməsi
    cin.getline(destination,_MAX_PATH);
    //Faylın mövcudluğunun yoxlanılması

    if(_access(destination,00)==0){
            RussianMessage("\nBu fayl mövcud deyil,
                            yaradılsınmı(1 - Hə/2 -
                            Yox)?\n");
```

```
        cin.getline(answer,20);
        if(!strcmp(answer,"2")){
         RussianMessage("\nƏməliyyat ləğv edildi.\n");
            return;
        }
        else if(strcmp(answer,"1")){
            RussianMessage("\nYanlış giriş.\n");
            return;
        }
        if(_access(destination,02)==-1){
            RussianMessage("\nYazıya müraciət haqqı
            yoxdur.\n"); return;
        }
    }
    //Faylın köçürülməsi
    if(!CopyFile(source,destination)){
        RussianMessage("\nFaylla iş zamanı
                       səhv.\n");
    }
}
```


"Darağacı" oyununa aid nümunə

Oyunun məğzi ondan ibarətdir ki, istifadəçi bir neçə cəhd ilə sözü tapmalıdır, bizim halda ingilis dilində olan sözü. Bu kodu kompilyasiya edəcəyiniz layihənin qovluğuna words.txt faylını yerləşdirmək lazımdır. Bu faylda tək sutunda alt-alta yerləşdirilmiş ingilis dilində bir neçə söz olmalıdır.

```
#include <windows.h>
#include <iostream>
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <time.h<
#include <sys\locking.h>
#include <string.h>
#include <ctype.h>

using namespace std;

//Sözün maksimal uzunluğu
#define MAX_WORD_LENGTH 21

//Cəhdlərin sayı
int Tries = 10;

//Tapılmış sözlərin sayı
int CountWords = 0;
```

```
//Sözün yüklənməsi
bool LoadWord(FILE * file, char * word)
{
    int i = 0;
    char s[MAX_WORD_LENGTH] = {0};
    //Fayldakı sözlərin sayı
    static int count = -1;

    if(count == -1)
    {
        //Sözlərin sayılması
        while(!feof(file))
        {
            fgets(s, MAX_WORD_LENGTH, file);
            count++;
        }
        //Söz yoxdur?
        if(count == 0)
            return false;
        //Fayl göstəricisinin faylın əvvəlinə
        //qaytarılması
        fseek(file, 0, 0);
    }
    //Təsadüfi söz
    int n = rand() % count;
    //Sözün axtarılması
    while(i <= n)
    {
        fgets(s, MAX_WORD_LENGTH, file);
        i++;
    }
    //Sözün uzunluğunu təyin edirik
    int wordlen = strlen(s);
    //Sözün minimal uzunluğu iki hərfdir
    if(wordlen <= 1)
        return false;
```



```
    //Enter-i silirik (DOS-da 2 bayt 13 10)
    if(s[wordlen - 1] == 10)
        s[wordlen - 2] = 0;
    else if(s[wordlen - 1] == 13)
        s[wordlen - 1] = 0;
    //Sözü köçürürük
    strcpy(word, s);
    //Faylın deskriptorunu əldə edirik
    int hFile = _fileno(file);
    //Faylın ölçüsünü hesablayırıq
    int size = _filelength(hFile);

    //Faylı təcrid edirik
    fseek(file, 0, 0);
    _locking(hFile, _LK_NBLCK, size);
    return true;
}

//Oyun
void Game(char * word)
{
    //Böyük hərflərə çevirmə
    strupr(word);

    int len = strlen(word);
    //Sətir-nüsxə
    char * copy = new char[len + 1];
    memset(copy, '*', len);
    copy[len] = 0;

    //Hərflər + boşluqlar
    char letters[52];

    int i, j = 0;
    for(i = 0; i < 26; i++)
    {
        letters[j++] = i + 'A';
```

```
        letters[j++] = ' ';
    }
    //Tamamlayan sıfır
    letters[51] = 0;

    //Hərf
    char letter;

    char * pos;
    bool replace = false;

    do {
        //Ekranın təmizlənməsi
        system("cls");
        cout << copy << endl << endl;
        cout << letters << endl << endl;
        cout << "Count of tries: " << Tries << endl
             << endl;
        cout << "Input any letter:\t";

        cin >> letter;
        //Səs siqnalı
        Beep(500, 200);

        if(letter >= 'A' && letter <= 'Z'
        || letter >= 'a' && letter <= 'z')

        //Hərfdir?
        if(!isalpha(letter))
        {
            cout << "It's not a letter" << endl;
            //Bir saniyə ləngitmə
            Sleep(1000);
            continue;
        }

        //Hərfləri böyük hərflərə çevirmə
        letter = toupper(letter);
```



```
        //Əlifbada sözün axtarılması
        pos = strchr(letters, letter);

        //Bü hərf artıq vardır
        if(pos == 0)
        {
            cout << "This letter have been
                      already pressed" << endl;
            Sleep(1000);
            continue;
        }
        else
        {
            //Əlifbadan hərfi silirik
            pos[0] = ' ';
        }

        //Sözdə hərfin axtarılması
        for(i = 0; i < len; i++)
        {
            if(word[i] == letter)
            {
                copy[i] = letter;
                replace = true;
            }
        }

        if(replace == false)
            Tries--;
        else
            replace = false;
        //Qələbə şərti
        if(strcmp(word, copy) == 0)
        {
            system("cls");
            cout << copy << endl << endl;
            cout << letters << endl << endl;
```

```
            cout << "Count of tries: " << Tries
                << endl << endl;
            cout << "Congratulation !!!" <<
            endl; CountWords++;
            break;
        }

    } while(Tries !=
    0); delete [] copy;
}

void main()
{
    //Faylı oxumaq üçün ikilik rejimdə açırıq
    FILE * f = fopen("words.txt", "rb");
    //Əgər fayl açılmadısa
    if(f == 0)
    {
        //Səhv
        perror("Open");
        return;
    }

    srand(time(0));

    char Word[20];
    //Sözü yükləməyə çalışırıq
    if(!LoadWord(f, Word))
    {
        //Əgər müvəffəq deyilsə
        cout << "Error !!!" << endl;
        fclose(f);
        return;
    }

char answer;
//Yorulana qədər oynayırıq
do
{
```



```
    Game(Word);
    //Əgər cəhd qalmayıbsa, çıxırıq
    if(Tries == 0)
    {
        cout << "Count of words: " << CountWords <<
        endl; cout << "Bye-bye" << endl;
        break;
    }
    //Cəhd qalıbsa
    cout << "Continue ??? (Y/N)\t";

    cin >> answer;
    //Yenə oynayırıq?
    if(answer == 'Y' || answer == 'y')
        if(!LoadWord(f, Word))
        {
            cout << "Error !!!" << endl;
            fclose(f);
            return;
        }

    }while(answer == 'Y' || answer == 'y');


    //Deskriptoru əldə edirik
    int hFile = _fileno(f);

    //Faylı təcriddən azad edilməsi
    int size = _filelength(hFile);
    fseek(f, 0, 0);
    _locking(hFile, _LK_UNLCK, size);
    fclose(f);
}
```

Ev tapşırığı

1. Növbəti əməliyyatları yerinə yetirmək üçün telefon sorğu kitabçası yaradın:
   - ■■ Bazaya abonentlərin əlavə edilməsi.
   - ■■ Bazadan abonenetlərin silinməsi.
   - ■■ Abonenetin verilənlərinin dəyişdirilməsi.
   - ■■ Telefon nömrəsinə və ya soyadına görə abonentin axtarılması.
   - ■■ Verilmiş diapazonda abonentlərin nömrələrini və ya soyadlarını əlifba sırasına görə çap etməli; məsələn, nömrələr üçün diapazon: 222222–333333 ola bilər, Soyadlar üçün isə: Babanlı-Behbudov (Budaqov diapazona daxil deyil).
   - ■■ Tapılmış informasiyanın faylda saxlanılma imkanı.
   - ■■ Bazanın faylda saxlanılması.
   - ■■ Bazanın fayldan yüklənilməsi.
2. DAM (Dövlət Avtomobil Müfəttişlıyi) verilənlər bazasını cərimə qəbzlərinə görə ikilik ağac vasitəsilə reallaşdırmalı. Açar kimi maşının nömrəsi, düyünün qiyməti kimi isə qaydapozanların siyahısı olacaq. Əgər qəbz ilk dəfə əlavə edilirsə, onda ağacda yeni düyün yaradılır, siyahıda isə qaydapozamaların verilənləri yerləşdirilir; əks halda verilənlər mövcud siyahıya əlavə edilirlər. Növbəti əməliyyatları reallaşdırmaq lazımdır:

   - ■■ Verilənlər bazasının (maşının nömrəsinə, qayda pozmaları siyahısına görə) tam çap edilməsi
   - ■■ Verilmiş nömrəyə görə verilənlərin çapı.
   - ■■ Nömrələr diapazonuna görə verilənlərin çapı.