

1. Feladat

Válogassuk ki a páros számokat az „inp.txt” szöveges állományban elhelyezett, és szóközzel elválasztott egész számok közül, és tegyük őket soronként egyesével az „out.txt” szöveges állományba!

Specifikáció:

$A = \text{infile}(N) \times \text{outfile}(N)$

$x \quad y$

$B = \text{infile}(N)$

x'

$Q = (x = x')$

$R = (y = \text{conc}_{i=1}^{dom(x')} f(x'_i))$

$f(dx) = \begin{cases} < dx > & \text{ha } dx \text{ páros} \\ \diamond & \text{kül} \end{cases}$

Absztrakt program:

$y := < >$	
$sx, dx, x : \text{read}$	
$sx = \text{norm}$	
$dx \text{ páros}$	
$y : \text{hiext}(dx)$	SKIP
$sx, dx, x : \text{read}$	

C++ program:

```
// Fordítási direktívák

int main()
{
    // Szöveges állományok előkészítése

    // Elemenkénti feldolgozás

    return 0;
}
```

Egész számokat tartalmazó szekvenciális inputfájl

[illegible]

Egész számokat tartalmazó szekvenciális outputfájl

[illegible]

Absztrakt program kódja

```
int dx;

x>>dx;
while (!x.eof()) {
    if (dx%2==0) {
        y<<dx<<endl;
    }
    x>>dx;
}
```

:

Szöveges állományok előkészítése:

```
ifstream x("inp.txt");
if ( x.fail() ){
    cerr<<"Nincs input fájl"<<endl;
    char barmi;
    cin>>barmi;
    return 2;
}
ofstream y("out.txt");
if ( y.fail() ){
    cerr<<"Nem lehet létrehozni az output fájlt"<<endl;
    char barmi;
    cin>>barmi;
    return 2;
}
```

:

A teljes megoldóprogram:

```
/* **** */
/* Feladat: Páros számok kiválogatása */
/*          szöveges állományban elhelyezett számok közül */
/*          egy másik szöveges állományba */
/* **** */

#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    // Szöveges állományok előkészítése
    ifstream x("inp.txt");
    if ( x.fail() ){
        cerr<<"Nincs input fájl"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }
    ofstream y("out.txt");
    if ( y.fail() ){
        cerr<<"Nem lehet létrehozni az output fájlt"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }

    //Elemenkénti feldolgozás
    int dx;

    x>>dx;
    while (!x.eof()) {
        if (dx%2==0) {
            y<<dx<<endl;
        }
        x>>dx;
    }

    return 0;
}
```

:

2. Feladat

Egy ékezetes magánhangzókat is tartalmazó szöveget alakítsuk át távirati stílusúra!

Specifikáció:

$$A = \text{infile}(K) \times \text{outfile}(K)$$

$$B = \text{infile}(K)$$

$$Q = (x = x')$$

$$R = (y = \text{conc}_{i=1}^{\text{dom}(x')} \text{távirat}(x_i'))$$

$$\text{távirat}(dx) = \begin{cases} "aa" & ha & dx = "á" \\ "ee" & ha & dx = "é" \\ "i" & ha & dx = "í" \\ "oe" & ha & dx = "ö" \text{ vagy } dx = "ő" \\ "ue" & ha & dx = "ü" \text{ vagy } dx = "ű" \\ "Aa" & ha & dx = "Á" \\ "Ee" & ha & dx = "É" \\ "I" & ha & dx = "Í" \\ "Oe" & ha & dx = "Ö" \text{ vagy } dx = "Ő" \\ "Ue" & ha & dx = "Ü" \text{ vagy } dx = "Ű" \\ dx & \text{különben} \end{cases}$$


Absztrakt program:

$y := < >$
$sx, dx, x : \text{read}$
$sx = \text{norm}$
$y : \text{hiext}(\text{távirat}(dx))$
$sx, dx, x : \text{read}$


:

C++ program:

Egész karaktereket tartalmazó szekvenciális inputfájl

A fájl deklarációja <code>ifstream x("inp.txt");</code>	 :.....
Egy elem kiolvasása <code>char dx;</code> <code>x.get(dx);</code> vagy <code>#include <iomanip></code> <code>x.unsetf(ios::skipws);</code> <code>x>>dx;</code>	
Fájlvége figyelése <code>x.eof()</code>	

Egész karaktereket tartalmazó szekvenciális outputfájl

A fájl deklarációja <code>ofstream y("out.txt");</code>	 :.....
Egy elem beírása <code>char dx;</code> <code>y.put(dx);</code> vagy <code>y<<dx;</code>	

Switch:

```
switch (<kifejezés>) {  
    case <konstans1> : <utasítássorozat1> ; break;  
    case <konstans2> : <utasítássorozat2> ; break;  
    case <konstans3> :  
    case <konstans4> : <utasítássorozat34> ; break;  
    default          : <utasítássorozat5> ;  
}
```



Absztrakt program kódja

```
char dx;  
x.get(dx);  
while (!x.eof()) {  
    switch (dx) {  
        case 'á' : y<<"ae"; break;  
        case 'é' : y<<"ee"; break;  
        case 'í' : y<<"i"; break;  
        case 'ö' :  
        case 'ő' : y<<"oe"; break;  
        case 'ü' :  
        case 'ű' : y<<"ue"; break;  
        case 'Ä' : y<<"Ae"; break;  
        case 'É' : y<<"Ee"; break;  
        case 'Í' : y<<"I"; break;  
        case 'Ö' :  
        case 'Ő' : y<<"Oe"; break;  
        case 'Ü' :  
        case 'Ű' : y<<"Ue"; break;  
        default : y<<dx;  
    }  
    x.get(dx);  
}
```



A teljes megoldóprogram:

```
/* **** */
/* Feladat: Ékezetes magánhangzókat is tartalmazó szöveg */
/* távirati stílusúvá alakítása */
/* szöveges állományból szöveges állományba */
/* **** */
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
    // Szöveges állományok előkészítése
    ifstream x("inp.txt");
    if ( x.fail() ){
        cerr<<"Nincs input fájl"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }
    ofstream y("out.txt");
    if ( y.fail() ){
        cerr<<"Nem lehet létrehozni az output fájlt"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }
    // Elemenkénti feldolgozás
    char dx;
    x.get(dx);
    while (!x.eof()) {
        switch (dx) {
            case 'á' : y<<"ae"; break;
            case 'é' : y<<"ee"; break;
            case 'í' : y<<"i"; break;
            case 'ö' :
            case 'ő' : y<<"oe"; break;
            case 'ü' :
            case 'ű' : y<<"ue"; break;
            case 'Á' : y<<"Ae"; break;
            case 'É' : y<<"Ee"; break;
            case 'Í' : y<<"I"; break;
            case 'Ö' :
            case 'Ő' : y<<"Oe"; break;
            case 'Ü' :
            case 'Ű' : y<<"Ue"; break;
            default : y<<dx;
        }
        x.get(dx);
    }
    return 0;
}
```


3. Feladat

Az „inp.txt” szöveges állomány egy könyvtár nyilvántartását tartalmazza. Ebben egy könyv adatait szóközzel (1 pozíció) elválasztva egy sorban helyeztük el: azonosító(4 pozíció), szerző(14), cím(19), kiadó(14), kiadás éve(4), aktuális példányszám(3), ISBN szám(14). Válogassuk ki a nulla példányszámú könyvek azonosítóját, szerzőjét és címét, és a fenti formában helyezzük el őket az „out.txt” szöveges állományban!

Specifikáció:

$A = \text{infile}(\text{könyv}) \times \text{outfile}(\text{könyv2})$

$\begin{matrix} x & y \\ \text{könyv} = \text{rec}(\text{azon}:N, \text{szerző}:K^*, \text{cím}:K^*, \text{kiadó}:K^*, \text{év}:K^*, \text{darab}:N_0, \text{isbn}:K^*) \\ \text{könyv2} = \text{rec}(\text{azon}:N, \text{szerző}:K^*, \text{cím}:K^*) \end{matrix}$

$B = \text{infile}(\text{könyv})$

$\begin{matrix} x' \\ Q = (x = x') \end{matrix}$

$R = (y = \text{conc}_{i=1}^{\text{dom}(x')} f(x_i'))$

$f(dx) = \begin{cases} < \text{rec}(dx.\text{azon}, dx.\text{szerző}, dx.\text{cím}) > & \text{ha } dx.\text{darab} = 0 \\ < > & \text{kül} \end{cases}$

Absztrakt program:

$y := < >$	
$sx, dx, x := \text{read}$	
$sx = \text{norm}$	
$dx.\text{darab} = 0$	
$y := \text{hiext}(dx.\text{azon}, dx.\text{szerző}, dx.\text{cím})$	SKIP
$sx, dx, x := \text{read}$	

C++ program:

Absztrakt program kódja

```
könyv dx;
status sx;

olv(x, dx, sx);
while (sx == norm) {
    if (dx.darab == 0) {
        ir(y, dx);
    }
    olv(x, dx, sx);
}
```

Szekvenciális inputfájl

Szükséges típusok

```
struct konyv{
    int azon;
    string szerzo;
    string cim;
    string kiado;
    string ev;
    int darab;
    string isbn;
};

enum status {abnorm, norm};
```



Olvasó művelet


```
void olv(ifstream &x, konyv &dx, status &sx)
{
    string sor;

    getline(x,sor,'\n');
    if (!x.eof()) {
        sx=norm;
        dx.azon  =atoi(sor.substr( 0, 4).c_str());
        dx.szerzo =    sor.substr( 5,14);
        dx.cim    =    sor.substr(20,19);
        dx.kiado  =    sor.substr(40,14);
        dx.ev     =    sor.substr(55, 4);
        dx.darab  =atoi(sor.substr(60, 3).c_str());
        dx.isbn   =    sor.substr(64,14);
    }
    else sx=abnorm;
}
```



Egész karaktereket tartalmazó szekvenciális outputfájl

Manipulátorok, formátumjelző flagek:

Formátumozott kiírás <pre>cout<<endl; cout.setf(ios::fixed ios::showpos ios::showpoint ios::left); int n=32.4; cout<<setw(10)<< setprecision(4)<< n;</pre>	
Manipulátorok <pre>setw(int w), setprecision(int p), setfill(char c) endl width(int w), precision(int p) setf(), unsetf()</pre>	
Formátumjelző flagek <pre>scientific, fixed, right, left, dec, hex, oct, showpoint, showpos skipws</pre>	

Író művelet

```
void ir(ofstream &y, const konyv dy)
{
    y<<setw(4)<<dy.azon<<' '<<dy.szerzo<<' '<<dy.cim<<endl;
}
```



A teljes megoldóprogram:

```
/* **** */
/* Feladat: egy könyvek adatait tartalmazó szöveges állományból */
/*          válogassuk ki a nulla darabszámú könyvek azonosítóját, */
/*          szerzőjét, címét egy szöveges állományba */
/* **** */
#include <fstream>
#include <iomanip>
#include <string>
using namespace std;
struct konyv {
    int azon;
    string szerzo;
    string cim;
    string kiado;
    string ev;
    int darab;
    string isbn;
} konyv;
enum status {abnorm, norm};
void olv(ifstream &x, konyv &dx, status &sx);
void ir(ofstream &x, const konyv dx);

int main()
{
    //Szöveges állományok előkészítése
    ifstream x("inp.txt");
    if ( x.fail() ){
        cerr<<"Nincs input fájl"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }
    ofstream y("out.txt");
    if ( y.fail() ){
        cerr<<"Nem lehet létrehozni az output fájlt"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }
    // Elemenkénti feldolgozás
    konyv dx;
    status sx;
    olv(x,dx,sx);
    while (sx==norm) {
        if (dx.darab==0) {
            ir(y,dx);
        }
        olv(x,dx,sx);
    }
    return 0;
}
```

```

// Olvasó és író eljárások
void olv(ifstream &x, konyv &dx, status &sx)
{
    string sor;

    getline(x,sor,'\n');
    if (!x.eof()) {
        sx=norm;
        dx.azon  =atoi(sor.substr( 0, 4).c_str());
        dx.szerzo =    sor.substr( 5,14);
        dx.cim    =    sor.substr(20,19);
        dx.kiado  =    sor.substr(40,14);
        dx.ev     =    sor.substr(55, 4);
        dx.darab  =atoi(sor.substr(60, 3).c_str());
        dx.isbn   =    sor.substr(64,14);
    }
    else sx=abnorm;
}

void ir(ofstream &y, const konyv dy)
{
    y<<setw(4)<<dy.azon<<' '<<dy.szerzo<<' '<<dy.cim<<endl;
}

```

:

4. Feladat

Az „nyilv.txt” szöveges állomány egy könyvtár nyilvántartását tartalmazza. Ebben egy könyv adatait egy sorban helyeztük el szóközzel (1 pozíció) elválasztva : az azonosítót (4 pozíción), a szerzőt (14), a címet (19), a kiadót (14), a kiadás évét (4), az aktuális példányszámot (3), az ISBN számot (14). A könyvek azonosító szerint szigorúan növekvően rendezettek.

Egy másik „mod.txt” szöveges állomány az aznapi könyvtári forgalmat mutatja: melyik könyvből hányat vittek el, illetve hoztak vissza. Minden sorban egy azonosítót (4 pozíció) és egy előjeles egészszámot (4) - ha elvitték: negatív, ha visszahozták: pozitív - találunk szóközzel elválasztva . A sorok azonosító szerint szigorúan növekvően rendezettek.

Aktualizáljuk a könyvtári nyilvántartást, és az eredeti formában helyezzük el az „ujnyilv.txt” szöveges állományban! Ha kell, írjunk hiba jelzéseket a „hiba.txt” szöveges állományba!

Specifikáció:

$$A = \underset{t}{\text{infile}}(\underset{m}{\text{könyv}}) \times \underset{u}{\text{infile}}(\underset{h}{\text{forg}}) \times \underset{u}{\text{outfile}}(\underset{h}{\text{könyv}}) \times \underset{h}{\text{outfile}}(\underset{h}{\text{hiba}})$$

$$\text{könyv} = \text{rec}(\text{azon}:N, \text{szerző}:K^*, \text{cím}:K^*, \text{kiadó}:K^*, \text{év}:K^*, \text{darab}:N_0, \text{isbn}:K^*)$$

$$\text{forg} = \text{rec}(\text{azon}:N, \text{db}:Z)$$

$$\text{hiba} = \text{rec}(\text{azon}:N, \text{uzen}:K^*)$$

$$B = \underset{t'}{\text{infile}}(\underset{m'}{\text{könyv}}) \times \underset{m'}{\text{infile}}(\underset{m'}{\text{forg}})$$

$$Q = (t = t' \wedge m = m' \wedge t \uparrow_{\text{azon}} \wedge m \uparrow_{\text{azon}})$$

$$R = ((u, h) = f(t', m'))$$

$$\forall a \in \text{azon}(t') \cup \text{azon}(m') :$$

$$f(a, \emptyset) = (a(t'), \emptyset)$$

$$f(\emptyset, a) = (\emptyset, (a, \text{"Nem létezik az könyv azonosító"}))$$

$$f(a, a) = \begin{cases} \text{ha } a(t').\text{darab} + a(m').\text{darab} < 0 \text{ akkor} \\ \quad (a(t'), (a, \text{"Hibás darabszám"})) \\ \text{ha } a(t').\text{darab} + a(m').\text{darab} \geq 0 \text{ akkor} \\ \quad (a(t')\text{darab} \leftarrow a(t').\text{darab} + a(m').\text{darab}, \emptyset) \end{cases}$$

Absztrakt program:

<i>st,dt,t:read</i>		<i>sm,dm,m:read</i>	<i>z,h:=< >, < ></i>
<i>st=norm</i> \vee <i>sm=norm</i>			
<i>sm=abnorm</i> \vee <i>st=norm</i> \wedge <i>dt.azon</i> < <i>dm.azon</i>	<i>st=abnorm</i> \vee <i>sm=norm</i> \wedge <i>dt.azon</i> > <i>dm.azon</i>	<i>st=norm</i> \wedge <i>sm=norm</i> \wedge <i>dt.azon</i> = <i>dt.azon</i>	
<i>u:hiext(dt)</i>	<i>h:hiext((dt.azon,</i> „Nem létező könyv azonosító”))	<i>y:hiext(dx)</i> <i>h:hiext((dt.azon,</i> „Hibás darabszám”))	<i>dt.darab</i> + <i>dm.darab</i> < 0 <i>dt.darab</i> := <i>dt.darab</i> + <i>dm.darab</i> <i>u:hiext(dt)</i>
<i>st,dt,t:read</i>	<i>sm,dm,m:read</i>	<i>st,dt,t:readd</i> <i>sm,dm,m:read</i>	

C++ program:

Absztrakt program

```

tread(t,dt,st);
mread(m,dm,sm);
while (st==norm || sm==norm) {
    if (sm==abnorm || (st==norm && dt.azon<dm.azon) ) {
        uhiext(u,dt);
        tread(t,dt,st);
    }
    else if (st==abnorm || (sm==norm && dt.azon>dm.azon) ) {
        hhiext(h,dm.azon,"Nem létező könyv azonosító"<<endl;
        mread(m,dm,sm);
    }
    else if (st==norm && sm==norm && dt.azon==dm.azon ) {
        db=dt.darab+dm.darab;
        if (db<0) {
            hhiext(h,dt.azon,"Hibás darabszám");
            uhiext(u,dt);
        }
        else {
            dt.darab=db;
            uhiext(u,dt);
        }
        tread(t,dt,st);
        mread(m,dm,sm);
    }
}

```

:

Változók

```
konyv dt;  
valtozas dm;  
status st,sm;  
int db;
```



Típusok

```
struct konyv{  
    int azon;  
    string szerzo;  
    string cim;  
    string kiado;  
    string ev;  
    int darab;  
    string isbn;  
};  
  
struct forg{  
    int azon;  
    int darab;  
};  
  
enum status {abnorm, norm};
```



Függvények

```
void tread(ifstream &t, konyv &dt, status &st);  
void mread(ifstream &m, forg &dm, status &sm);  
void uhiext(ofstream &u, const konyv du);  
void hhiext(ofstream &h, const int azon,  
            const string uzen);
```



Szekvenciális fájl olvasó és író műveletei

```
void tread(istream &t, konyv &dt, status &st)
{
    string sor;
    getline(t,sor,'\n');
    if (!t.eof()) {
        st=norm;
        dt.azon  =atoi(sor.substr( 0, 4).c_str());
        dt.szerzo =    sor.substr( 5,14);
        dt.cim    =    sor.substr(21,19);
        dt.kiado  =    sor.substr(42,14);
        dt.ev      =    sor.substr(58, 4);
        dt.darab  =atoi(sor.substr(63, 3).c_str());
        dt.isbn   =    sor.substr(67,14);
    }
    else st=abnorm;
}

void mread(istream &m, valtozas &dm, status &sm)
{
    m>>dm.azon;
    if (!m.eof()) {
        sm=norm;
        m>>dm.darab;
    }
    else sm=abnorm;
}

void uhiext(ofstream &u, const konyv du)
{
    u<<setw(4) <<du.azon <<' ' <<
        setw(14)<<du.szerzo<<' ' <<
        setw(19)<<du.cim <<' ' <<
        setw(14)<<du.kiado <<' ' <<
        setw(4) <<du.ev <<' ' <<
        setw(3) <<du.darab <<' ' <<
        setw(14)<<du.isbn <<endl;
}

void hhiext(ofstream &h, const int azon,
            const string uzen)
{
    h<<setw(4)<<azon <<' ' << uzen <<endl;
}
```

:

A teljes megoldóprogram:

```
/* **** */
/* Feladat: Könyvtári nyilvántartás aktualizálása */
/*          2-2 elemenkénti feldolgozás */
/* **** */

//Fordítási direktívák
#include <fstream>
#include <iomanip>
#include <string>
using namespace std;

//Típus definíciói
struct konyv{
    int azon;
    string szerzo;
    string cim;
    string kiado;
    string ev;
    int darab;
    string isbn;
};

struct forg{
    int azon;
    int darab;
};

enum status {abnorm, norm};

// Függvény deklarációk
void tread(ifstream &t, konyv &dt, status &st);
void mread(ifstream &m, forg &dm, status &sm);
void uhiext(ofstream &u, const konyv du);
void hhiext(ofstream &h, const int azon,
            const string uzen);

int main()
{
    //Szöveges állományok előkészítése
    ifstream t("nyilv.txt");
    if (t.fail() ) {
        cerr<<"Nincs törzsfájl"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }
    ifstream m("mod.txt");
    if (t.fail() ) {
        cerr<<"Nincs módosító fájl"<<endl;
        char barmi;
        cin>>barmi;
        return 2;
    }
}
```

```

ofstream u("ujnyilv.txt");
if ( u.fail() ){
    cerr<<"Nem lehet létrehozni az új törzsfájlt"<<endl;
    char barmi;
    cin>>barmi;
    return 2;
}
ofstream h("hiba.txt");
if ( h.fail() ){
    cerr<<"Nem lehet létrehozni a hiba fájlt"<<endl;
    char barmi;
    cin>>barmi;
    return 2;
}
//Változó definíciók
konyv dt;
valtozas dm;
status st,sm;
int db;

//Elemenkénti feldolgozás
tread(t,dt,st);
mread(m,dm,sm);
while (st==norm || sm==norm) {
    if (sm==abnorm || (st==norm && dt.azon<dm.azon) ) {
        uhiext(u,dt);
        tread(t,dt,st);
    }
    else if (st==abnorm || (sm==norm && dt.azon>dm.azon) ) {
        hhiext(h,dm.azon,"Nem létező könyv azonosító"<<endl;
        mread(m,dm,sm);
    }
    else if (st==norm && sm==norm && dt.azon==dm.azon ) {
        db=dt.darab+dm.darab;
        if (db<0) {
            hhiext(h,dt.azon,"Hibás darabszám");
            uhiext(u,dt);
        }
        else {
            dt.darab=db;
            uhiext(u,dt);
        }
        tread(t,dt,st);
        mread(m,dm,sm);
    }
}

return 0;
}

```

:

```

// Függvény definíciók
void tread(istream &t, konyv &dt, status &st)
{
    string sor;
    getline(t,sor,'\n');
    if (!t.eof()) {
        st=norm;
        dt.azon  =atoi(sor.substr( 0, 4).c_str());
        dt.szerzo =    sor.substr( 5,14);
        dt.cim    =    sor.substr(21,19);
        dt.kiado  =    sor.substr(42,14);
        dt.ev     =    sor.substr(58, 4);
        dt.darab  =atoi(sor.substr(63, 3).c_str());
        dt.isbn   =    sor.substr(67,14);
    }
    else st=abnorm;
}

void mread(istream &m, valtozas &dm, status &sm)
{
    m>>dm.azon;
    if (!m.eof()) {
        sm=norm;
        m>>dm.darab;
    }
    else sm=abnorm;
}

void uhiext(ofstream &u, const konyv du)
{
    u<<setw(4) <<du.azon <<' ' <<
        setw(14)<<du.szerzo<<' ' <<
        setw(19)<<du.cim <<' ' <<
        setw(14)<<du.kiado <<' ' <<
        setw(4) <<du.ev <<' ' <<
        setw(3) <<du.darab <<' ' <<
        setw(14)<<du.isbn <<endl;
}

void hhiext(ofstream &h, const int azon,
            const string uzen)
{
    h<<setw(4)<<azon <<' ' << uzen <<endl;
}

```

:

Modularizáció

Modularizáció

```
// Fordítási direktívák
// Típusok definíciói
// Függvény deklarációk
int main( )
{
    //Szöveges állományok előkészítése
    //Változók definíciói
    //Elemenkénti feldolgozás

    return 0;
}
// Függvény definíciók
```

Közös típusok

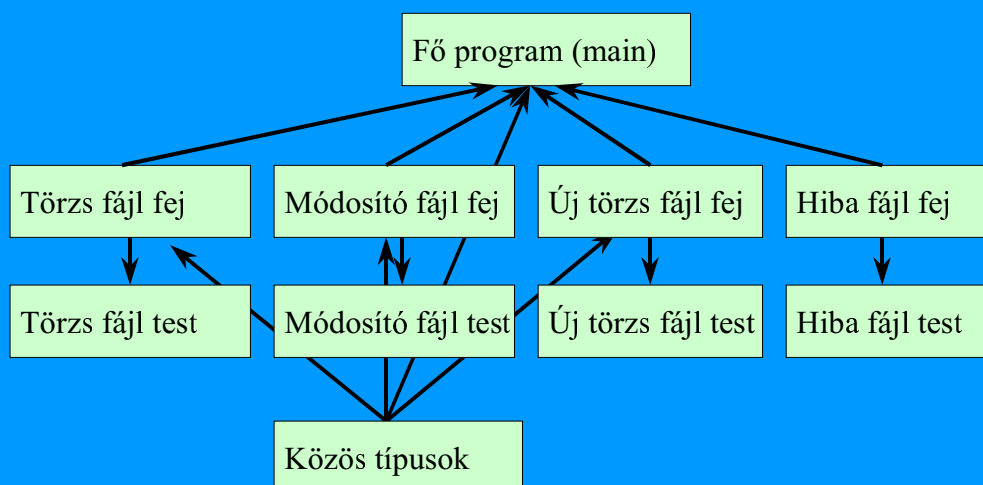
Törzs fájl típusa

Módosító fájl típusa

Új törzs fájl típusa

Hiba fájl típusa

Modul szerkezet



fo.cpp:

```
/* **** */
/* Feladat: Könyvtári nyilvántartás aktualizálása */
/* 2-2 elemenkénti feldolgozás */
/* **** */

//Fordítási direktívák
#include "kozos.h"
#include "torzs.h"
#include "mod.h"
#include "ujtorzs.h"
#include "hiba.h"
using namespace std;

int main() {
    //Változó definíciók
    torzs t("nyilv.txt");
    mod m("mod.txt");
    ujtors u("ujnyilv.txt");
    hiba h("hiba.txt");
    konyv dt;
    valtozas dm;
    status st,sm;
    int db;

    //Elemenkénti feldolgozás
    t.read(dt,st);
    m.read(dm,sm);
    while (st==norm || sm==norm) {
        if (sm==abnorm || (st==norm && dt.azon<dm.azon) ) {
            u.hiext(dt);
            t.read(dt,st);
        }
        else if (st==abnorm || (sm==norm && dt.azon>dm.azon) ) {
            h.hiext(dm.azon,"Nem létező könyv azonosító"<<endl;
            m.read(dm,sm);
        }
        else if (st==norm && sm==norm && dt.azon==dm.azon ) {
            db=dt.darab+dm.darab;
            if (db<0) {
                h.hiext(dt.azon,"Hibás darabszám");
                u.hiext(dt);
            }
            else {
                dt.darab=db;
                u.hiext(dt);
            }
            t.read(dt,st);
            m.read(dm,sm);
        }
    }

    return 0;
}
```

kozos.h:

```
#ifndef KOZOS_H
#define KOZOS_H

#include <string>

struct konyv{
    int azon;
    string szerzo;
    string cim;
    string kiado;
    string ev;
    int darab;
    string isbn;
};

struct forg{
    int azon;
    int darab;
};

enum status {abnorm, norm};

#endif
```



torzs.h:

```
#ifndef TORZS_H
#define TORZS_H

#include <fstream>
#include <string>
#include "kozos.h"

class torzs
{
    ifstream f;
public:
    torzs(string fnev="");
    void read(konyv &dt, status &st);
};

#endif
```

torzs.cpp:

```
#include "torzs.h"

torzs::torzs(string fnev="")
{
    if ( fnev.size()<1 ) {
        cout<<"Add meg a törzsfájl nevét:";
        cin>>fnev;
    }
    f.open(fnev.c_str());
    if (f.fail()) {
        cerr<<"Nincs törzsfájl"<<endl;
        char barmi;
        cin>>barmi;
        exit(2);
    }
}

void torzs::read(konyv &dt, status &st)
{
    string sor;
    getline(f,sor,'\n');
    if (!f.eof()) {
        st=norm;
        dt.azon  =atoi(sor.substr( 0, 4).c_str());
        dt.szerzo =    sor.substr( 5,14);
        dt.cim   =    sor.substr(21,19);
        dt.kiado =    sor.substr(42,14);
        dt.ev    =    sor.substr(58, 4);
        dt.darab =atoi(sor.substr(63, 3).c_str());
        dt.isbn  =    sor.substr(67,14);
    }
    else st=abnorm;
}
```


mod.h:

```
#ifndef MOD_H
#define MOD_H

#include <fstream.h>
#include " kozos.h"

class mod
{
    ifstream f;
public:
    mod(string fnev="");
    void read(forg &dm, status &sm);
};

#endif
```

mod.cpp:

```
#include "mod.h"

mod::mod(string fnev="")
{
    if ( fnev.size()<1 ) {
        cout<<"Add meg a módosító fájl nevét:";
        cin>>fnev;
    }
    f.open(fnev.c_str());
    if (f.fail()) {
        cerr<<"Nincs módosító fájl"<<endl;
        char barmi;
        cin>>barmi;
        exit(2);
    }
}

void mod::read(forg &dm, status &sm)
{
    f>>dm.azon;
    if (f.eof()) {
        sm=abnorm;
    }
    else {
        sm=norm;
        f>>dm.darab;
    }
}
```

ujtorzs.h:

```
#ifndef UJTORSZS_H
#define UJTORSZS_H

#include <fstream.h>
#include <string>
#include "kozos.h"

class ujtorszs
{
    ofstream f;
public:
    ujtorszs(string fnev="");
    void hiext(const konyv du);
};
#endif
```

ujtorzs.cpp:

```
#include <stdio.h>
#include <iomanip>
#include "ujtorzs.h"

ujtorszs::ujtorszs(string fnev="")
{
    if ( fnev.size()<1 ) {
        cout<<"Add meg az új törzsfájl nevét:";
        cin>>fnev;
    }
    f.open(fnev.c_str());
    if (f.fail()) {
        cerr<<"Nem lehet létrehozni az új törzsfájlt"<<endl;
        char barmi;
        cin>>barmi;
        exit(2);
    }
}

void ujtorszs::hiext(const konyv du)
{
    f<<setw(4) <<du.azon <<' ' <<
        setw(14)<<du.szerzo<<' ' <<
        setw(19)<<du.cim <<' ' <<
        setw(14)<<du.kiado <<' ' <<
        setw(4) <<du.ev <<' ' <<
        setw(3) <<du.darab <<' ' <<
        setw(14)<<du.isbn <<endl;
}
```

hiba.h:

```
#ifndef HIBA_H
#define HIBA_H

#include <fstream.h>
#include <string>
#include "kozos.h"

class hiba
{
    ofstream f;
public:
    hiba(string fnev="");
    void hiext(const int azon, const string uzen);
};

#endif
```

hiba.cpp:

```
#include <stdio.h>
#include <iomanip>
#include "hiba.h"

hiba::hiba(string fnev="")
{
    if ( fnev.size()<1 ) {
        cout<<"Add meg a hiba fájl nevét:";
        cin>>fnev;
    }
    f.open(fnev.c_str());
    if (f.fail()) {
        cerr<<"Nem lehet létrehozni a hiba fájlt"<<endl;
        char barmi;
        cin>>barmi;
        exit(2);
    }
}

void hiba::hiext(const int azon, const string uzen)
{
    f<<setw(4)<<azon<<' '<<uzen<<endl;
}
```