

## Feladat

Egy szekvenciális inputfájlban egy vállalat anyagkészlet-nyilvántartásának nyitóegyenlegeit tároljuk. Minden anyagról ismerjük a főkönyvi számát (természetes szám) és a nyitó egyenlegét (pozitív egész szám). A fájl főkönyvi szám szerint rendezett és egyértelmű. Egy másik szekvenciális input fájl az eltelt időszak mozgásait tartalmazza. Egy mozgás egy adott főkönyvi számú anyagra vonatkozó ki- vagy bevételnek megfelelő előjeles egész szám. A fájl főkönyvi szám szerint rendezett. Állítsuk elő főkönyvi számonként rendezett és egyértelmű módon a záró egyenlegeket egy szekvenciális outputfájlban! Ha egy főkönyvi számhoz nem tartozik nyitóegyenleg, akkor azt automatikusan nulla nyitóegyenlegűnek kell tekinteni.

## Specifikáció

A feladat lényegében az időszerűsítés egy speciális esete. Állapottér-transzformációval biztosítjuk, hogy a módosító-fájl is egyértelmű kulcsú legyen.

$$\begin{aligned} A &= T \times M \times T \\ &\quad i \quad m \quad o \\ B &= T \times M \\ &\quad i' \quad m' \\ Q &= i' = i \wedge m' = m \wedge \pi_K(i') \wedge \pi_K(m') \wedge \pi_V(i', m') \\ R &= Q \wedge o = \mathcal{A}_{m'}(i') \end{aligned}$$

ahol

$$\begin{aligned} T &:= \text{seq}(K : \mathbb{Z}, D : \mathbb{N}) \\ M &:= \text{seq}(K : \mathbb{Z}, C : \mathbb{Z}) \\ \pi_K(x) &:= \forall i, j \in x.\text{dom} : i < j \Rightarrow x_i.K < x_j.K \\ \pi_V(x, m) &:= \forall k \in \{m.K\} \setminus \{x.K\} : m_k.C \geq 0 \\ &\quad \wedge \forall k \in \{m.K\} \cap \{x.K\} : m_k.C + x_k.D \geq 0 \\ \{\mathcal{A}_m(x).K\} &:= \{m.K\} \cup \{x.K\}; \pi_K(\mathcal{A}_m(x)) \\ \forall k \in \{\mathcal{A}_m(x).K\} : \mathcal{A}_m(x)_k.D &:= \begin{cases} x_k.D & \text{ha } k \notin \{m.K\} \\ m_k.C & \text{ha } k \notin \{x.K\} \wedge m_k.C \geq 0 \\ x_k.D + m_k.C & \text{ha } k \in \{m.K\} \cap \{x.K\} \wedge x_k.D + m_k.C \geq 0 \end{cases} \end{aligned}$$

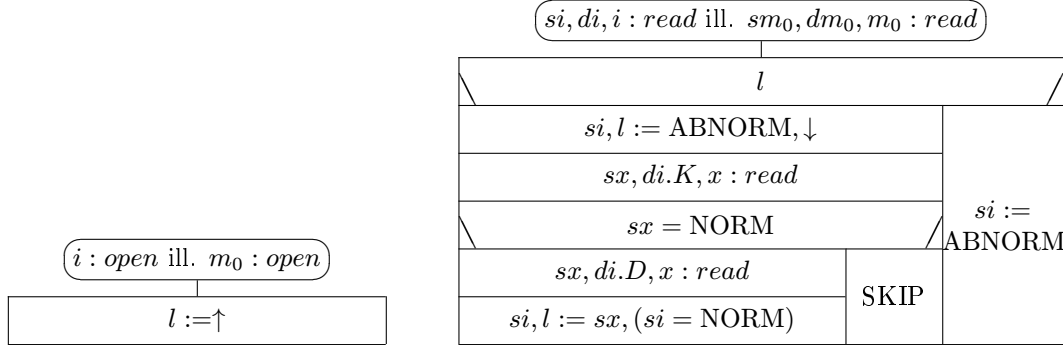
Az egyértelmű kulcsú  $m \in M$  módosítófájlt az eredeti  $m_0 \in M_0$ -ból egy egyszerű invariánssal kapjuk:

$$\begin{aligned} \{m.K\} &= \{m_0.K\} \\ \forall k \in \{m.K\} : m_k.C &= \sum_{i \in m_0.\text{dom}} \chi(m_{0_i}.K = k) \cdot m_{0_i}.C \end{aligned}$$

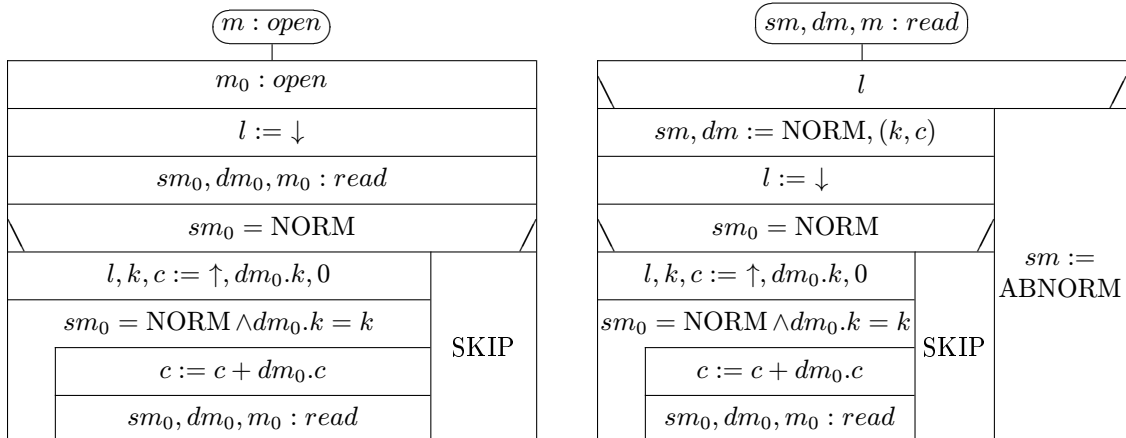
## Absztrakt program

Az egyértelmű kulcsú módosítófájllal működő programunk az általános időszersítés egy igencsak leegyszerűsített változata.

$i : open$		
$m : open$		
$o := \langle \rangle$		
$si, di, i : read$		
$sm, dm, m : read$		
$si = NORM \vee sm = NORM$		
$sm \neq NORM \vee si = NORM \wedge di.k < dm.k$	$si = sm \wedge di.k = dm.k$	$si \neq NORM \vee sm = NORM \wedge dm.k < di.k$
$o : hievt(di.K, di.D)$	$o : hievt(di.K, di.D + dm.C)$	$o : hievt(dm.K, dm.C)$
$si, di, i : read$	$i, si, di : read$	$sm, dm, m : read$
	$sm, dm, m : read$	



Világos, hogy  $M$  típusműveleteinek megvalósításához az szükséges, hogy összegezzük az azonos kulcsú (és ezért egymást követő) módosítások  $C$  mezőjét. Ezt előreolvasással tesszük meg:



## C++ implementáció

Az absztrakt fájlokat C++-os osztályok segítségével implementáljuk.

### Programváz

```
// Fordítási direktívák

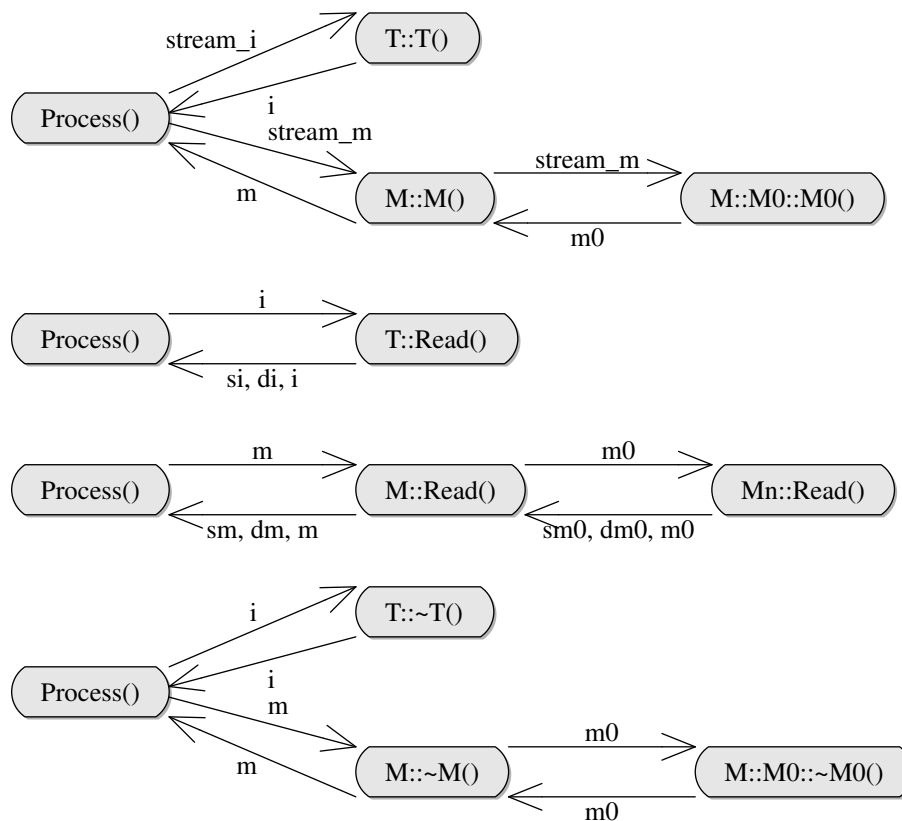
// Típusdefiníciók
// Osztálydefiníciók

// Osztályok tagfüggvényeinek definíciói
// Függvénydefiníciók

int main (int argc, char **argv)
{
    // A főprogram
}
```

### Adatforgalom a függvények között

A nyitó egyenleg igazi fájlja	$stream\_i \in seq(\mathbb{Z})$
A nyitó egyenleg absztrakt fájlja	$i \in T$
A módosítás igazi fájlja	$stream\_m \in seq(\mathbb{Z})$
A módosítás nem egyértelmű kulcsú absztrakt fájlja	$m0 \in M_0$
A módosítás absztrakt fájlja	$m \in M$



## Megvalósított típusok

### IOState

FELADAT:	Az absztrakt szekvenciális fájlok olvasási státusza.
TÍPUSÉRTÉK-HALMAZ:	{NORM, ABNORM}
TÍPUSMŰVELETEK:	Értékadás, egyenlőség-vizsgálat
MEGVALÓSÍTÁS:	Felsorolási (enum) típussal.

```
enum IOState
{
    Norm,
    Abnorm
};
```

### Elem\_T, Elem\_M

FELADAT:	Az absztrakt szekvenciális fájlok elemtípusai
TÍPUSÉRTÉK-HALMAZ:	$(k : \mathbb{Z}, d : \mathbb{N})$ ill. $(k : \mathbb{Z}, c : \mathbb{Z})$
TÍPUSMŰVELETEK:	Szelektorok
MEGVALÓSÍTÁS:	Rekord (struct) típussal.

```
struct Elem_T
{
    int k;
    unsigned int d;
};
```

```
struct Elem_M
{
    int k;
    int c;
};
```

### T, M0

FELADAT:	A kezdeti egyenleget, illetve a változtatásokat tartalmazó absztrakt szekvenciális fájl típus
TÍPUSÉRTÉK-HALMAZ:	$seq(Elem\_T)$ ill. $seq(Elem\_M)$
TÍPUSMŰVELETEK:	open, read
REPREZENTÁCIÓ:	Egy C++-os bemeneti stream (std::istream) a valódi fájl
MEGVALÓSÍTÁS:	Osztály (class) típussal. A típusműveletek megvalósítása az absztrakt programok alapján egyértelműen következik.

```
class T
{
    std::istream &stream;
    bool l;

public:
    T (std::istream &stream_);
    IOState Read (Elem_T &di);
};
```

```

};

T::T(std::istream &stream_):
    stream(stream_),
    l (true)
{
}

IOState T::Read(Elem_T &di)
{
    if (!l)
        return Abnorm;

    l = false;
    stream >> di.k;
    if (!stream.eof ())
    {
        stream >> di.d;
        if (!stream.eof ())
        {
            l = true;
            return Norm;
        }
    }

    return Abnorm;
}

```

**M**

- FELADAT: A változtatásokat tartalmazó, már egyértelmű kulcsúvá tett absztrakt szekvenciális fájl típus
- TÍPUSÉRTÉK-HALMAZ:  $seq(Elem\_M)$
- TÍPUSMŰVELETEK: `open`, `read`
- REPREZENTÁCIÓ: A nem egyértelmű kulcsú, kulcs szerint rendezett  $Elem\_M$  értékeket egy  $M\_0$  típusú absztrakt szekvenciális fájl tartalmazza. Az előolvasás eredményét két tagváltozóban tároljuk.
- MEGVALÓSÍTÁS: Osztály (`class`) típussal. A típusműveletek megvalósítása az absztrakt programok alapján egyértelműen következik.

```

class M
{
    class M0
    {
        std::istream &stream;
        bool l;
    public:
        M0 (std::istream &stream);
        IOState Read (Elem_M &dm0);
    };

    M0 m0;
    bool l;

    Elem_M dm0;
}

```

```
IOState sm0;

int k;
int c;

public:
    M (std::istream &stream);
    IOState Read (Elem_M &sm);
};

M::M (std::istream &stream):
    m0 (stream),
    l (false)
{
    sm0 = m0.Read (dm0);
    if (sm0 == Norm)
    {
        l = true; k = dm0.k; c = 0;

        while (sm0 == Norm && dm0.k == k)
        {
            c += dm0.c;
            sm0 = m0.Read (dm0);
        }
    }
}

IOState M::Read (Elem_M &dm)
{
    if (!l)
        return Abnorm;

    dm.k = k; dm.c = c;

    l = false;
    if (sm0 == Norm)
    {
        l = true; k = dm0.k; c = 0;
        while (sm0 == Norm && dm0.k == k)
        {
            c += dm0.c;
            sm0 = m0.Read (dm0);
        }
    }

    return Norm;
}
```

## Alprogramok

### Főprogram

- FELADAT:** A tesztelő keretrendszer része; egy keret a `Process` alprogram köré.
- BEMENŐ ADATOK:** A feldolgozandó fájlok nevei:  $f_i, f_m \in string$   
 A feldolgozandó konkrét fájlok:  $stream_i, stream_m \in seq(\mathbb{Z})$   
 A std. kimenet:  $out \in seq(\mathbb{Z})$
- KIMENŐ ADATOK:** A feldolgozandó konkrét fájlok:  $stream_i, stream_m \in seq(\mathbb{Z})$   
 A std. kimenet:  $out \in seq(\mathbb{Z})$
- TEVÉKENYSÉG:** A parancssori kapcsolókból megállapítja a nyitóegyenleget, és a módosításokat tartalmazó fájlok neveit, majd az ezekből olvasó konkrét fájlokat, és a std. kimenetet átadja a `Process` alprogramnak.

```
int main (int argc, char **argv)
{
    if (argc != 3)
    {
        std::cerr << "Meg kell adnod a nyitóegyenleg és a módosítófájl nevét!" << std::endl;
        return -1;
    }

    std::ifstream stream_i (argv[1]);
    if (!stream_i)
    {
        std::cerr << "Fájl megnyitása nem sikerült:_" << argv[1] << std::endl;
        return -1;
    }

    std::ifstream stream_m (argv[2]);
    if (!stream_m)
    {
        std::cerr << "Fájl megnyitása nem sikerült:_" << argv[2] << std::endl;
        return -1;
    }

    Process (stream_i, stream_m, std::cout);
}
```

### Process

- FELADAT:** Elvégzi az időszerűsítést.
- BEMENŐ ADATOK:** A feldolgozandó konkrét fájlok:  $stream_i, stream_m \in seq(\mathbb{Z})$   
 A kezdőegyenleg absztrakt fájlja:  $i \in T$   
 A módosítások absztrakt fájlja:  $m \in M$   
 A kimenet:  $stream_o \in seq(\mathbb{Z})$
- KIMENŐ ADATOK:** A feldolgozandó konkrét fájlok:  $stream_i, stream_m \in seq(\mathbb{Z})$   
 A kezdőegyenleg absztrakt fájlja:  $i \in T$   
 A módosítások absztrakt fájlja:  $m \in M$   
 A kimenet:  $stream_o \in seq(\mathbb{Z})$
- TEVÉKENYSÉG:** Létrehozza az absztrakt szekvenciális fájlokat, ezután az időszerűsítést az absztrakt programnak megfelelően hajtja végre.

```
void Process (std::istream &stream_i, std::istream &stream_m, std::ostream &stream_o)
```

```
{
    T i (stream_i);
    M m (stream_m);

    Elem_T di;
    IOState si = i.Read (di);

    Elem_M dm;
    IOState sm = m.Read (dm);

    while (si == Norm || sm == Norm)
    {
        if (sm != Norm || si == Norm && di.k < dm.k)
        {
            stream_o << di.k << "_" << di.d << std::endl;

            si = i.Read (di);
        }
        else if (si == sm && di.k == dm.k)
        {
            assert (di.d + dm.c >= 0);

            stream_o << di.k << "_" << int (di.d + dm.c) << std::endl;

            si = i.Read (di);
            sm = m.Read (dm);
        }
        else if (si != Norm || sm == Norm && dm.k < di.k)
        {
            assert (dm.c >= 0);

            stream_o << dm.k << "_" << int (dm.c) << std::endl;

            sm = m.Read (dm);
        }
    }
}
```