Elemi alkalmazások fejlesztése I/11.
Absztrakt file típus
2.rész

Steingart Ferenc
stengi@inf.elte.hu

2002. május 6.

Feladat

Számoljuk meg, hogy egy szöveges file hány olyan bekezdést tartalmaz, ami legalább 5 sorból áll, és a leghoszabb sora legalább 10 betűből áll. A sorok egymástól sorvége jellel vannak elválasztva. A bekezdések egymástól egy vagy több üres sorral vannak elválasztva (az üres sor csak a sorvége jelet tartalmazza).

Megoldás

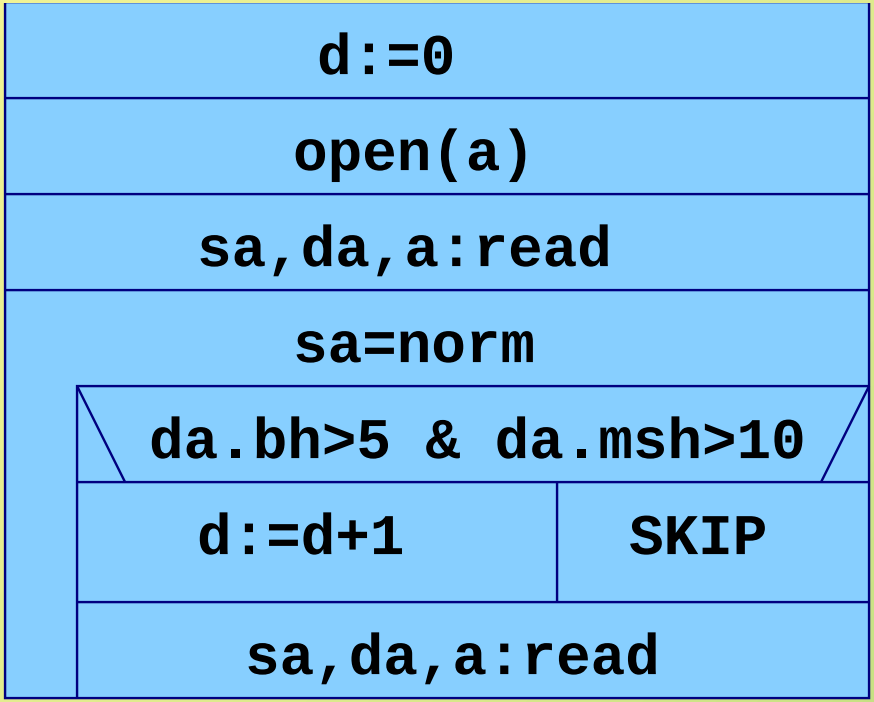
Absztrakt file típus bevezetése:

```
Egy,
megérett a meggy.

Kettő,
csipkebokor vessző.
Három,
Te vagy
az én
párom


Négy,
egy kiló döglött légy.
```

(2, 17)
(6, 19)
(2, 22)

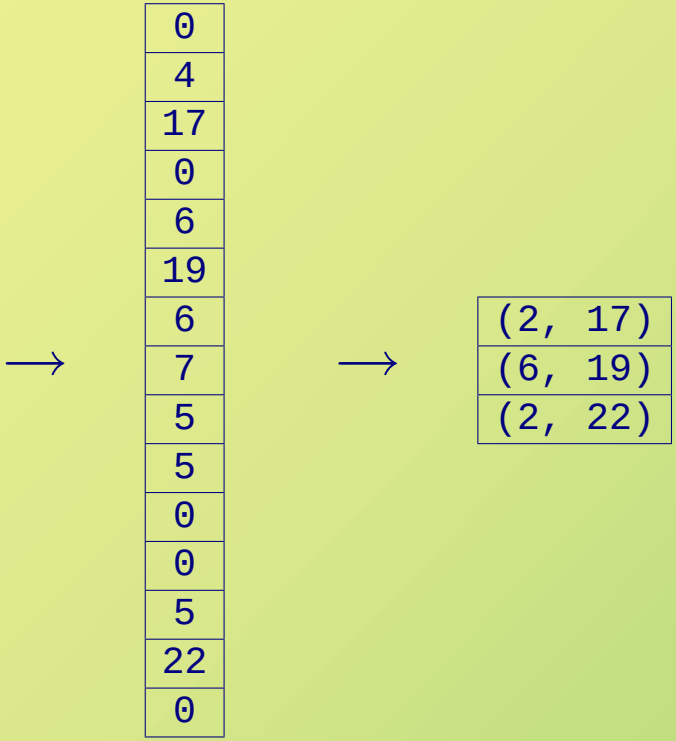
Absztrakt megoldóprogram

Számlálás az absztrakt a file-ra:



A típusműveletek megvalósítása

```
Egy,
megérett a meggy.

Kettő,
csipkebokor vessző.
Három,
Te vagy
az én
párom


Négy,
egy kiló döglött légy.
```



Olvasóművelet az "A" file-ból

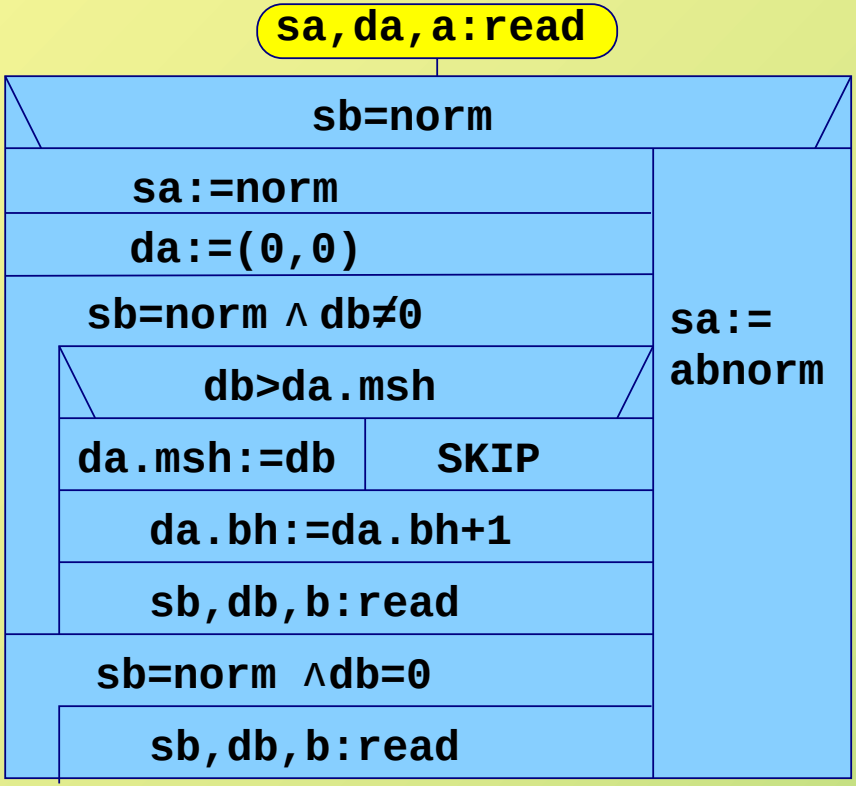
Az a:A file olvasóműveletét a b:B absztrakt file felett definiáljuk:
Az olvasó művelet feltételezi az alábbi "típusinvariáns" meglétét:

`sb=abnorm`
vagy
db a következő bekezdés első sorának hossza

A read az alábbi rekurzív formulával definiált függvény értékeit számítja ki a bekezdés végén:

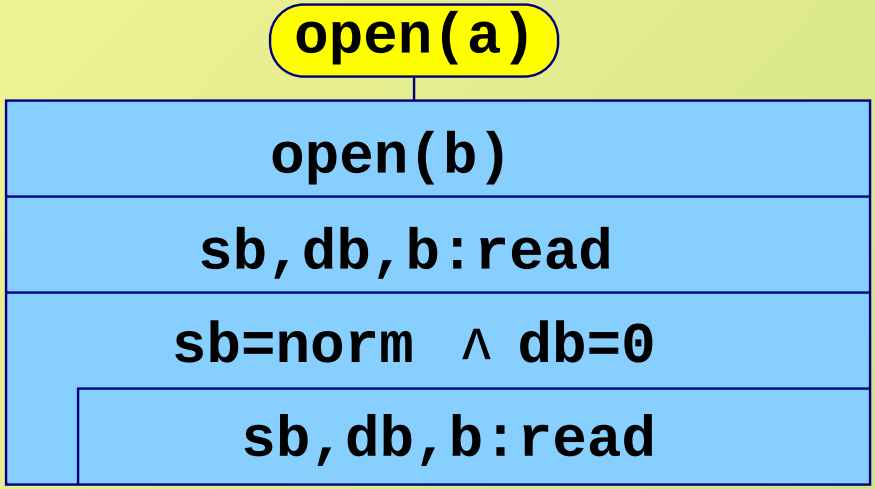
$$
\begin{aligned}
f(0) &= (0,0) \\
f(i+1) &= \begin{cases} f_1(i)+1, b_{i+1} & \text{ha } b_{i+1} > f_2(i) \\ f_1(i)+1, f_2(i) & \text{különben} \end{cases}
\end{aligned}
$$

Olvasóművelet az "A" file-ból: absztrakt program



Az open művelet

A "típusinvariáns"-t először az open művelettel teljesítjük:



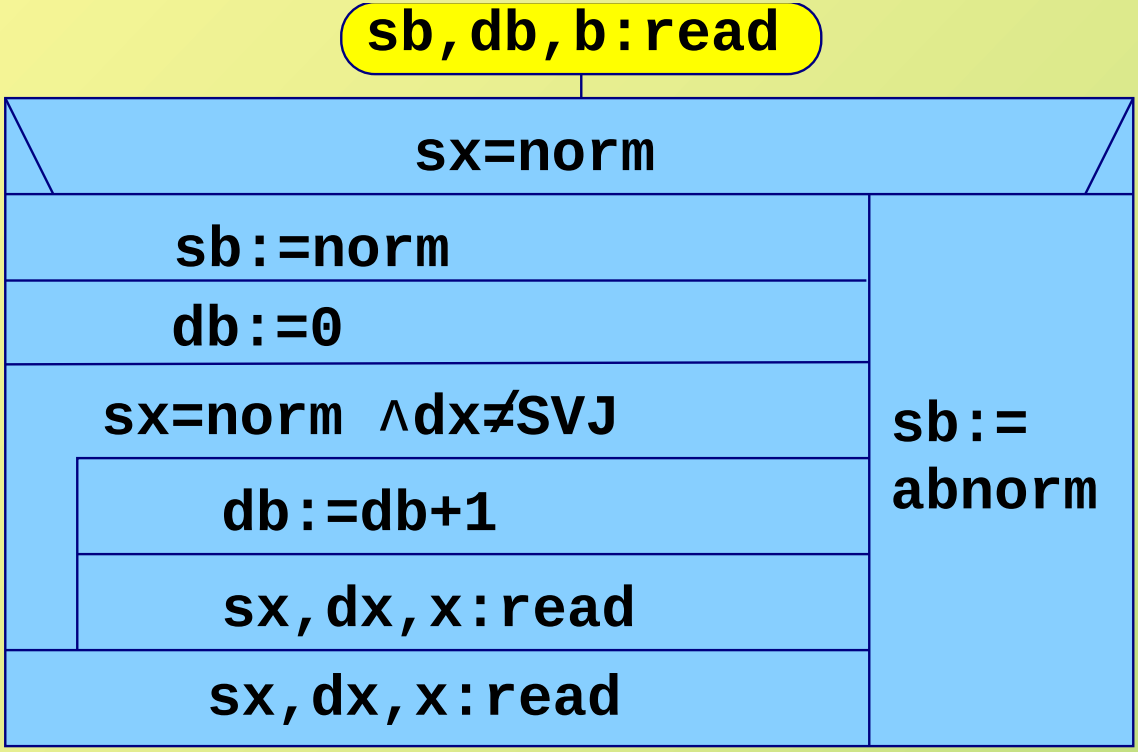
Olvasóművelet a "B" file-ból

Az b:B file olvasóműveletét az eredeti x szövegfile felett definiáljuk:
Az olvasó művelet feltételezi az alábbi "típusinvariáns" meglétét:

sx=abnorm
vagy
dx a következő sor első karaktere

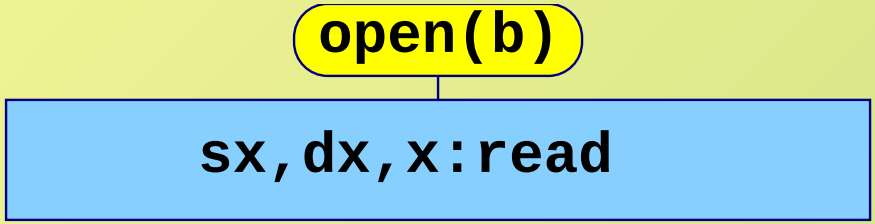
A read megszámolja a sorban található karaktereket és rááll a következő sor elejére.

Olvasóművelet a "B" file-ból: absztrakt program



Az open művelet

A "típusinvariáns"-t először az open művelettel teljesítjük:



Megoldás C++-ban

A kódolás során elvégzendő feladatok:

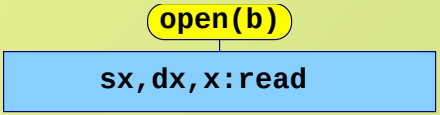
- az absztrakt file típusok definiálása osztállyal
- a főprogram (`main` függvény) kódolása
- a típusműveletek implementációja

A B file típus

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

enum status {norm, abnorm};

const char SVJ='\n';

class B{
   ifstream x;
   char     dx;
   status   sx;
public:
   B(const char* fn);
   void open();
   void read(status& sb, int& db);
};
```

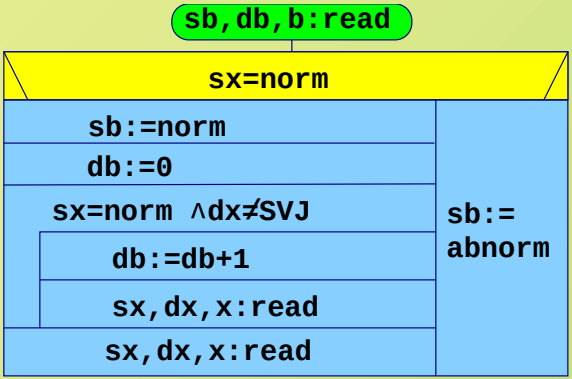
Az open művelet

```
B::B(const char* fn):
  x(fn),dx(0),sx(norm)
{
}

void B::open()
{
   x.get(dx);
   sx = x.eof() ? abnorm: norm;
}
```



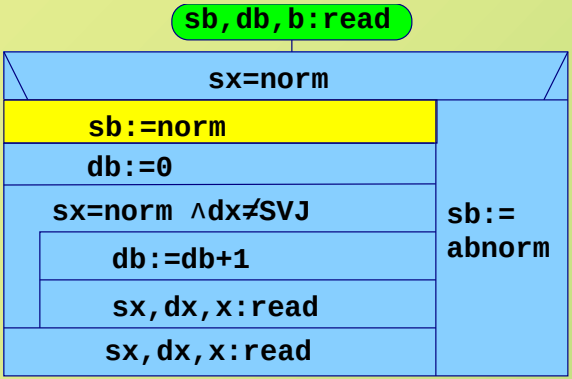
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



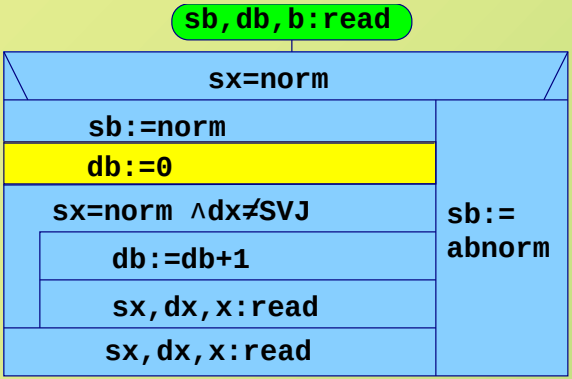
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



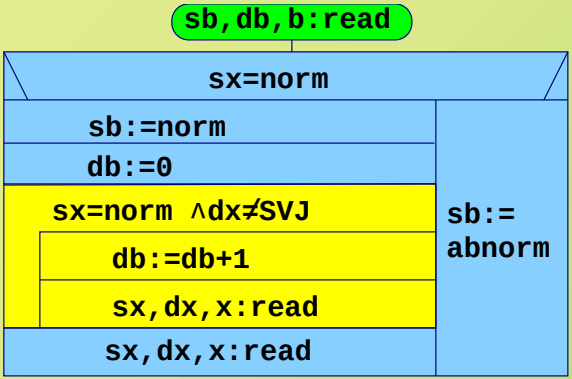
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



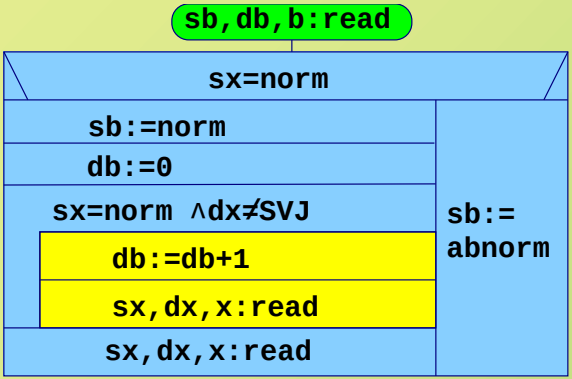
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



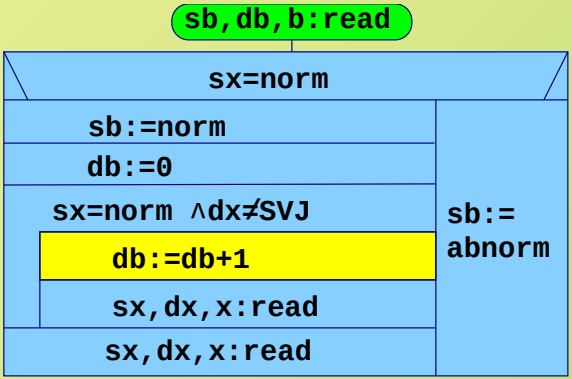
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



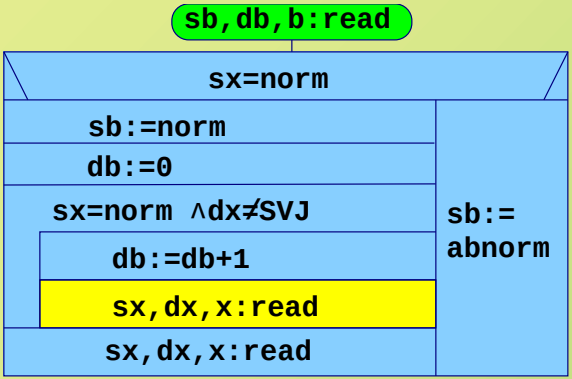
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



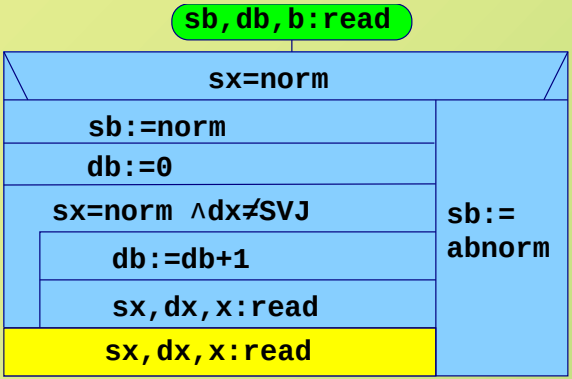
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



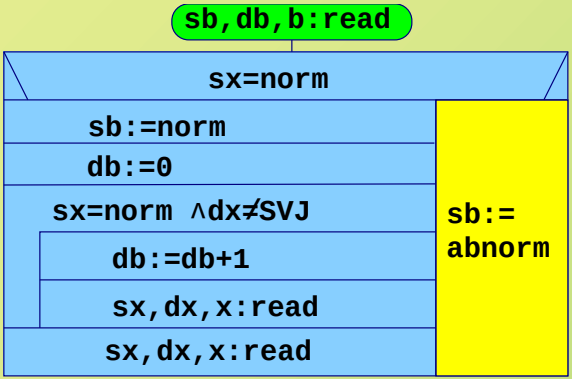
A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```



A `read` művelet

```
void B::read(status& sb, int& db)
{
   if(sx == norm){
      sb=norm;
      db=0;
      while(sx==norm && dx!=SVJ){
         db++;
         x.get(dx);
         sx = x.eof() ? abnorm: norm;
      }
      x.get(dx);
      sx = x.eof() ? abnorm: norm;
   }else{
      sb=abnorm;
   }
}
```

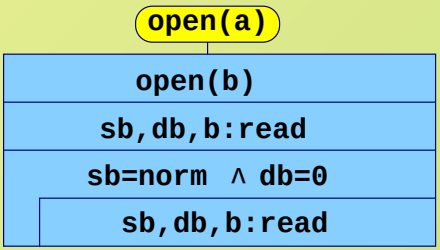


Az A file típus

```
struct Rec{
   int bh;
   int msh;
};

class A{
   B        b;
   int      db;
   status   sb;
public:
   A(const char* fn);
   void open();
   void read(status& sa, Rec& da);
};
```

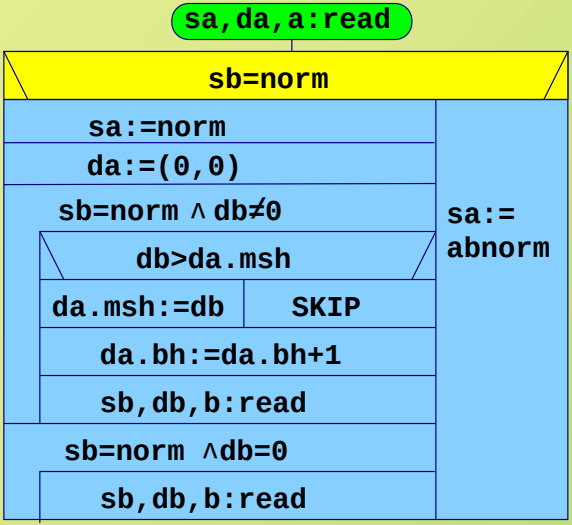
Az open művelet

```
A::A(const char* fn):
  b(fn),db(0),sb(norm)
{
}

void A::open()
{
   b.open();
   b.read(sb,db);
   while(sb==norm && db==0){
      b.read(sb,db);
   }
}
```



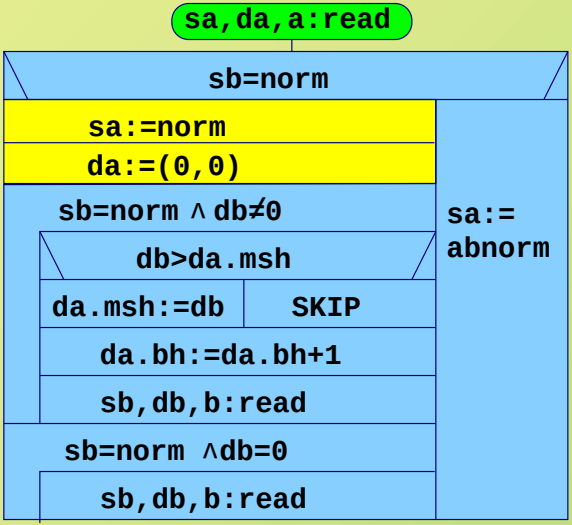
A `read` művelet

```
void A::read(status& sa, Rec& da)
{
   if(sb == norm){
      sa=norm;
      da.bh=0;
      da.msh=0;
      while(sb==norm && db!=0){
         da.bh++;
         if(da.msh < db){
            da.msh=db;
         }
         b.read(sb,db);
      }
      while(sb==norm && db==0){
         b.read(sb,db);
      }
   }else{
      sa=abnorm;
   }
}
```



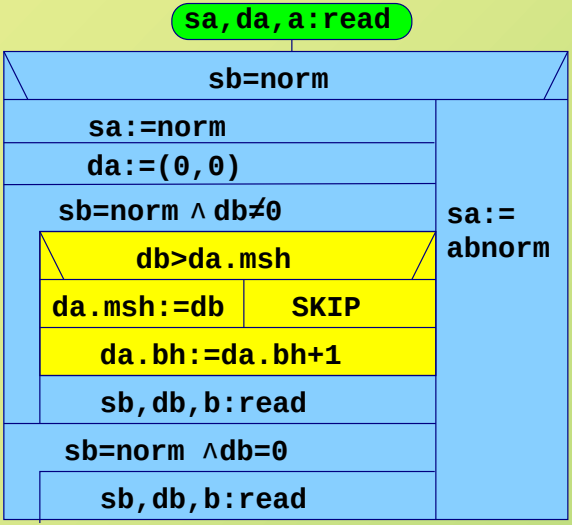
A `read` művelet

```
void A::read(status& sa, Rec& da)
{
   if(sb == norm){
      sa=norm;
      da.bh=0;
      da.msh=0;
      while(sb==norm && db!=0){
         da.bh++;
         if(da.msh < db){
            da.msh=db;
         }
         b.read(sb,db);
      }
      while(sb==norm && db==0){
         b.read(sb,db);
      }
   }else{
      sa=abnorm;
   }
}
```



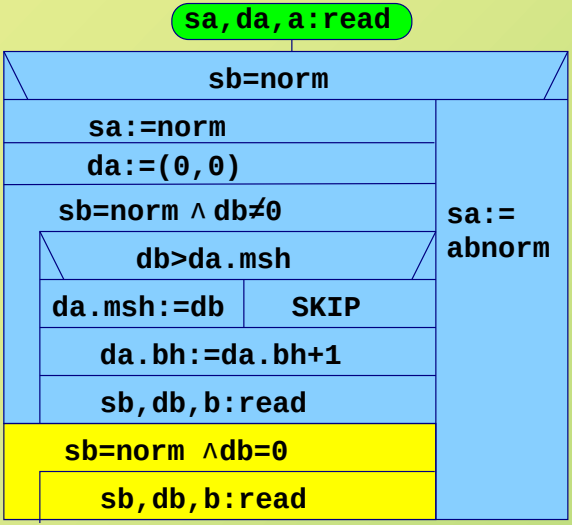
A `read` művelet

```
void A::read(status& sa, Rec& da)
{
   if(sb == norm){
      sa=norm;
      da.bh=0;
      da.msh=0;
      while(sb==norm && db!=0){
         da.bh++;
         if(da.msh < db){
            da.msh=db;
         }
         b.read(sb,db);
      }
      while(sb==norm && db==0){
         b.read(sb,db);
      }
   }else{
      sa=abnorm;
   }
}
```



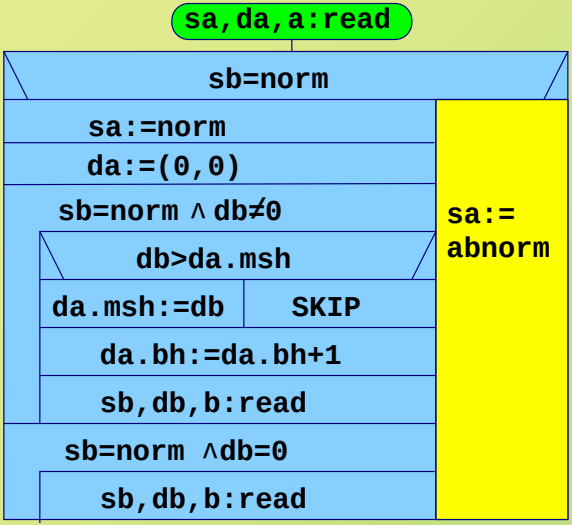
A `read` művelet

```
void A::read(status& sa, Rec& da)
{
   if(sb == norm){
      sa=norm;
      da.bh=0;
      da.msh=0;
      while(sb==norm && db!=0){
         da.bh++;
         if(da.msh < db){
            da.msh=db;
         }
         b.read(sb,db);
      }
      while(sb==norm && db==0){
         b.read(sb,db);
      }
   }else{
      sa=abnorm;
   }
}
```



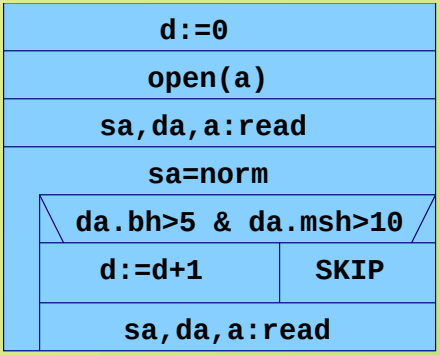
A `read` művelet

```
void A::read(status& sa, Rec& da)
{
   if(sb == norm){
      sa=norm;
      da.bh=0;
      da.msh=0;
      while(sb==norm && db!=0){
         da.bh++;
         if(da.msh < db){
            da.msh=db;
         }
         b.read(sb,db);
      }
      while(sb==norm && db==0){
         b.read(sb,db);
      }
   }else{
      sa=abnorm;
   }
}
```



A főprogram

```
int main(int argc, char *argv[])
{
  int d = 0;
  A   a("x.txt");
  Rec da;
  status sa;

  a.open();
  a.read(sa, da);
  while(sa==norm){
     if(da.bh > 5 && da.msh > 10){
        d++;
     }
     a.read(sa, da);
  }
  cout << d << endl;
  return 0;
}
```



Feladatok beadási határideje

A beadandó programokat
CSAK
a szorgalmi időszakban
lehet bemutatni!

(Kivétel azok, akiknek az utolsó beadási határ-idő – pünkösdhétfő miatt – május 27.)

VÉGE