Feladat:

Számoljuk meg, hogy egy szöveges file hány olyan bekezdést tartalmaz, ami legalább 1 sorból áll, és a leghoszabb sora legalább 10 betűből áll. A sorok egymástól sorvége jellel vannak elválasztva. A bekezdések egymástól egy vagy több üres (nulla hosszúságú) sorral vannak elválasztva.

Megoldás[1]:

Közvetlenül a feladat nem vezethető vissza egyetlen ismert programozási tételre sem.

Tegyük fel hogy az eredeti szövegfile helyett egy olyan absztrakt file-unk van amiben az eredeti file bekezdései sorainak a száma és a bekezdés leghosszabb sorának a hossza van. Ekkor a feladat visszavezethető a számlálás tételére.

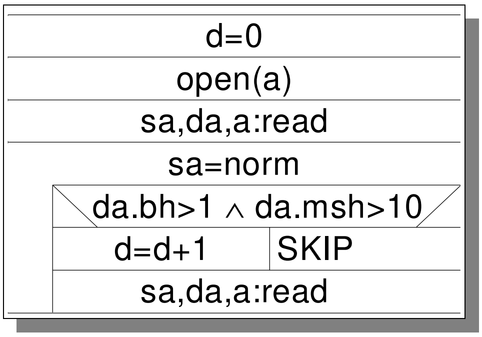
Legyen az eredeti file $x : F$ (amiből karaktereket olvasunk), és legyen az absztrakt file $b : B$ (amiből a bekezdések sorainak a számát és a leghoszabb sorának a hosszát olvassuk). Az eredeti $x : F$ file-t és az absztrakt $b : B$ file-t közvetlenül nehéz lenne összekapcsolni, ezért bevezetünk egy ujabb absztrakt file-t $a : A$ ami az eredeti $x : F$ file sorainak a hosszát fogja tartalmazni. Ezt a közbülső absztrakt file-t könnyen elő tudjuk állítani az eredeti $x : F$ file-ból, és segítségével könnyen elő tudjuk állítani az absztrakt $b : B$ file-t (a sorok számlálásával és a bekezdés leghosszabb sorának a keresésével).

A megoldás menete:

- Készítük egy absztrakt programot, ami egy számlálás lesz az $a : A$ absztrakt file-n.
  Az absztrakt file-okat C++-ban osztállyal fogjuk ábrázolni.
- Készítünk egy ("B" nevü) osztályt ami az eredeti file-ből fog olvasni soronként (open és read), és a beolvasott sor hosszát adja meg (számlálással).
- Készítünk egy ("A" nevü) osztályt ami az előzőleg definiált absztrakt file-ból fog olvasni bekezdésenként. A beolvasott bekezdésben lévő sorok számát (számlálás) és a leghosszabb sor hosszát (maximumkeresés) fogja megadni.

1 Ezt a feladatot rekurzív függvénnyel már igencsak körülményes lenne megoldani...

Az absztrakt a file-t használó absztrakt megoldó program struktogramja



A főprogram C++ programrészlete

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

  a.open();
  a.read(sa, da);
  while(sa==norm){
     if(da.bh > 1 && da.msh > 10){
        d++;
     }
     a.read(sa, da);
  }
  cout << d << endl;
  return 0;
}
```

Az absztrakt "B" file típusának a megvalósítása

A "B" nevü osztály mezőinek, konstruktorának és két műveletének a deklarációja C++-ban

```
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);
};
```

A "B" nevü osztály konstruktorát megvalósító C++ programrészlet

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

Az eredeti x file-t megnyitó open(b) művelet struktogramja

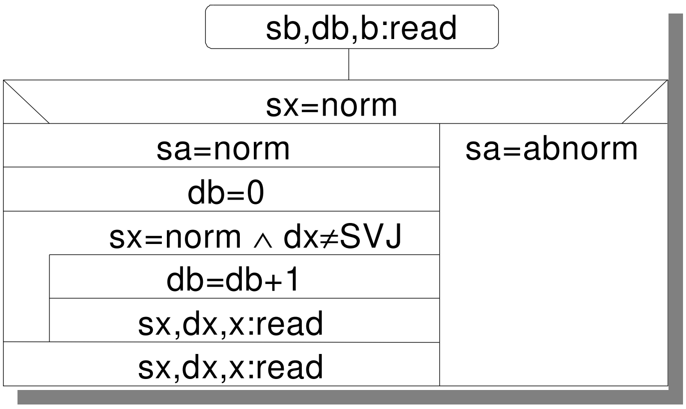
open(b)

sx,dx,x:read

A "B" nevü osztály open műveletét megvalósító C++ programrészlet

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

Az eredeti x file-t olvasó read(b) művelet struktogramja



A "B" nevü osztály read műveletét megvalósító C++ programrészlet

```
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 absztrakt "A" file típusának a megvalósítása

Az "A" nevü osztály mezőinek, konstruktorának és két műveletének a deklarációja C++-ban

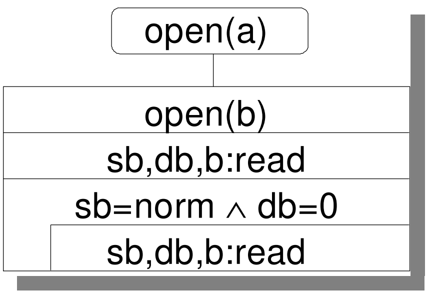
```
struct Rec{
   int bh;
   int msh;
};

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

Az "A" nevü osztály konstruktorát megvalósító C++ programrészlet

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

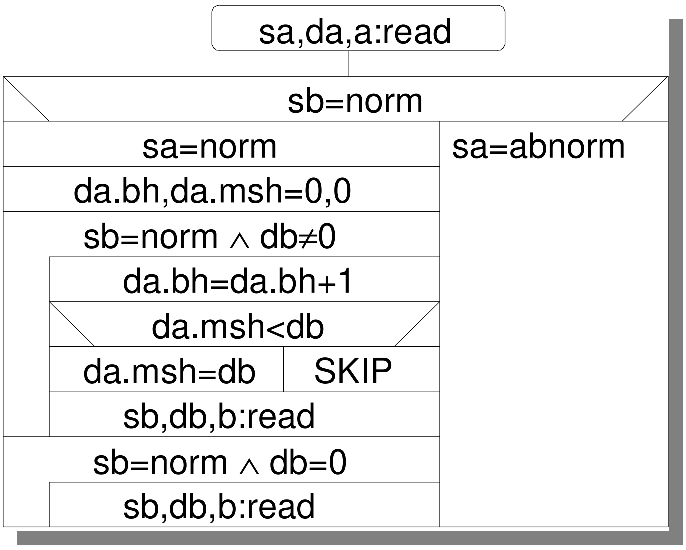
Az absztrakt b file-t megnyitó open(a) művelet struktogramja



Az "A" nevü osztály open műveletét megvalósító C++ programrészlet

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

Az absztrakt b file-t olvasó read(a) művelet struktogramja



Az "A" nevü osztály read műveletét megvalósító C++ programrészlet

```
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 teljes megoldóprogram

```
#include <fstream>
#include <iostream>
#include <cstdlib>
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);
};

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

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

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;
   }
}

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);
};

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);
   }
}

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;
   }
}


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

  a.open();
  a.read(sa, da);
  while(sa==norm){
     if(da.bh > 1 && da.msh > 10){
        d++;
     }
     a.read(sa, da);
  }
  cout << d << endl;
  return 0;
}
```

✎ ....

....

....

....

....

....

....

....

....

....

....