# Feladat

Egy szöveges fileból írjuk ki a képernyőre (soronként egyet) a pontosan 5 karakterből álló szavakat. A szavak a file-ban egy vagy több szóközzel illetve soremelés karakterrel vannak elválasztva.

# Megoldás

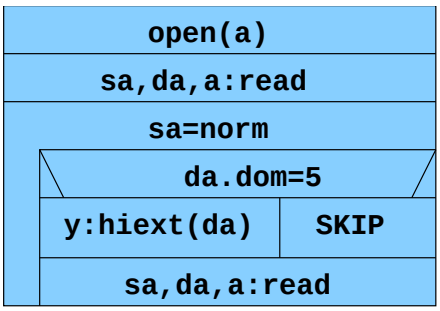
Közvetlenül a feladat nem vezethető vissza egyetlen ismert programozási tételre sem. Tegyük fel, hogy az eredeti szövegfile helyett egy olyan file-unk van amiben az eredeti file szavai vannak.

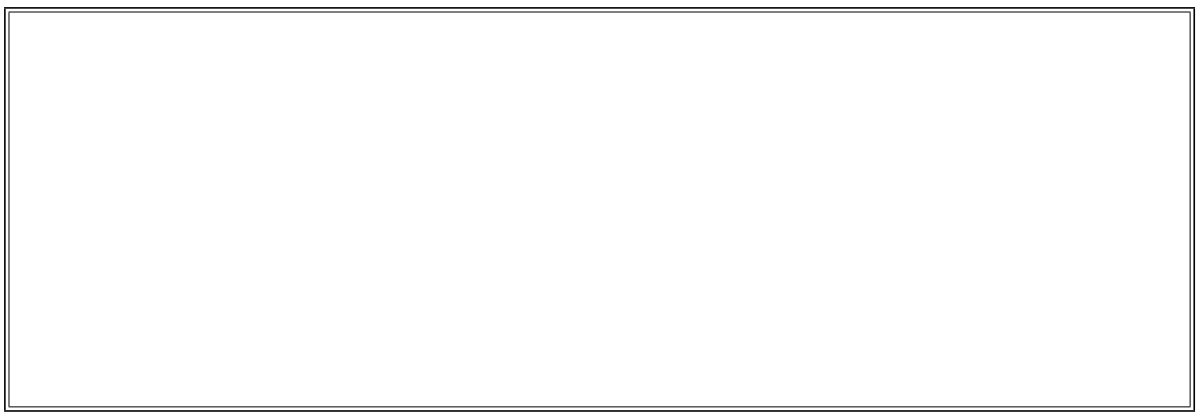
Legyen az eredeti file `x:F` (amiből karaktereket olvasunk), és legyen az absztrakt file `a:A` (amiből szavakat olvasunk).

Ekkor a feladat visszavezethető az egyváltozós, egyértékű elementi feldolgozás tételére, az `a:A` absztrakt file-ból való olvasással. A `A` egy új típus, a szóolvasás műveletével. Ám a műveletet meg kell valósítani az eredeti `x:F` file-on. Az absztrakt file-ból egy szóolvasása meg fog felelni az eredeti file-ból több karakter olvasásának. Az absztrakt olvasó művelet (az absztrakt READ) ezért egy összetettebb program lesz.

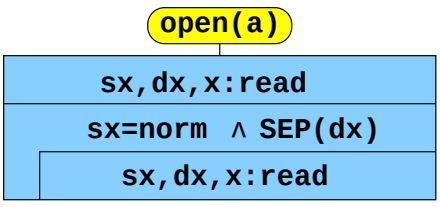
# Absztrakt program

Az absztrakt `a:A` file-on az elemenkénti feldolgozás:

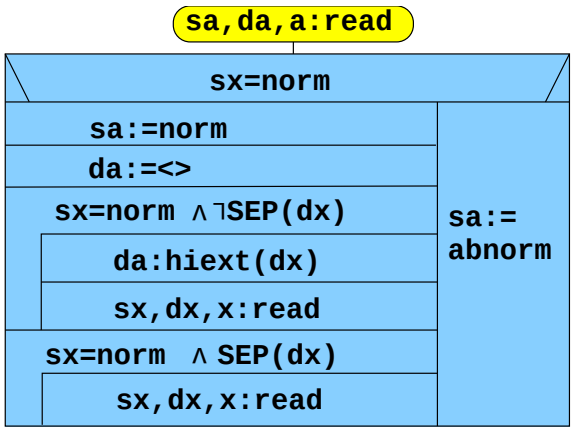


Az `open` művelet rááll az első értékes karakterre:



Az `read` művelet elolvassa a szót, és rááll az következő értékes karakterre:




## Megoldás C++-ban

Lévén az `A` egy absztrakt file típus, C++-ban osztállyal fogjuk megvalósítani. Reprezentációjában az eredeti `x` file, valamint a műveletek során használt (és a műveletek között megosztott) `sx` és `dx` változók szerepelnek. A típusnak (a konstruktoron kívül) két művelete lesz: az tt open és a `read`.

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

class A{
   ifstream x;
   bool     sx;
   char     dx;
public:
   A(const string& fn);
   void open();
   void read(bool& sa, string& da);
};
```

Az osztály ismeretében a főprogram implementációja:

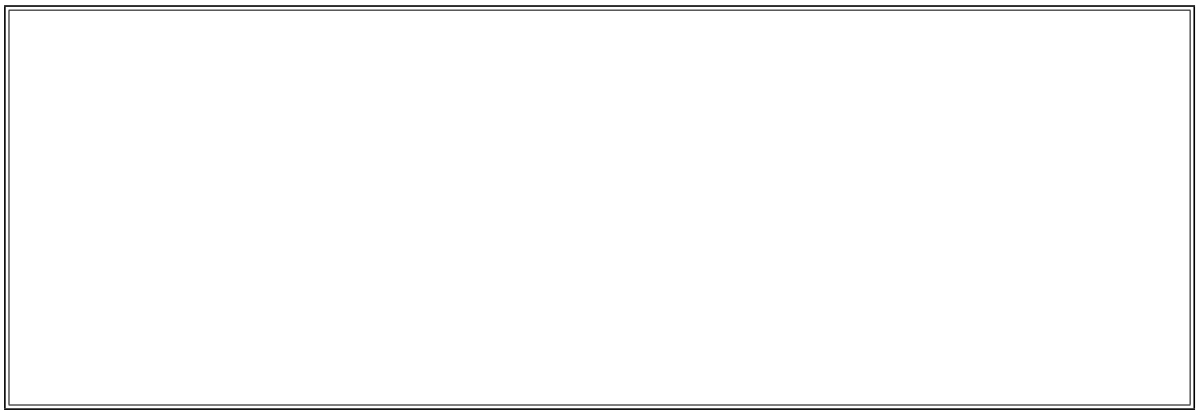
```
int main(int argc, char *argv[])
{
  bool sa;
  string da;
  A a("x.txt");

  a.open();
  a.read(sa, da);
  while(sa){
     if(da.length()==5){
        cout << da << endl;
     }
     a.read(sa, da);
  }
  return 0;
}
```

Hátra van még az A osztály műveleteinek megvalósítása. A konstruktor az argumentumként kapott filenévvel inicializálja az x input streamet:

```
A::A(const string& fn):
   x(fn.c_str())
{
   //empty
}
```

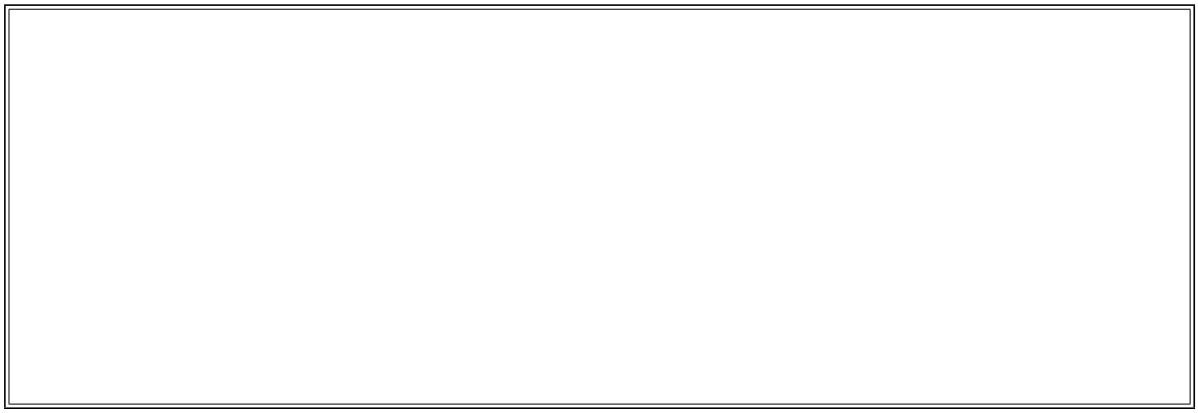
Az `open` műveletben a struktogramot kódoljuk:

```
void A::open()
{
   x.get(dx);
   sx = !x.eof();
   while(sx && (dx==' ' || dx=='\n')){
      x.get(dx);
      sx = !x.eof();
   }
}
```



A `read` művelet struktogramjának kódolása:

```
void A::read(bool& sa, string& da)
{
   if(sx){
      sa = true;
      da = "";
      while(sx && dx!=' ' && dx!='\n'){
         da+=dx;
         x.get(dx);
         sx = !x.eof();
      }
      while(sx && (dx==' ' || dx=='\n')){
         x.get(dx);
         sx = !x.eof();
      }
   }else{
      sa=false;
   }
}
```



Mivel az elkészített típus nem általános célú, hanem a konkrét feladat megoldásához használtuk, az `A` osztályt nem tesszük külön modulba. A teljes `main.cpp` file tehát:

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


class A{
   ifstream x;
   bool     sx;
   char     dx;
public:
   A(const string& fn);
   void open();
   void read(bool& sa, string& da);
};

A::A(const string& fn):
  x(fn.c_str())
{
   //empty
}

void A::open()
{
   x.get(dx);
   sx = !x.eof();
   while(sx && (dx==' ' || dx=='\n')){
      x.get(dx);
      sx = !x.eof();
   }
}

void A::read(bool& sa, string& da)
{
   if(sx){
      sa = true;
      da = "";
      while(sx && dx!=' ' && dx!='\n'){
```

```
         da+=dx;
         x.get(dx);
         sx = !x.eof();
      }
      while(sx && (dx==' ' || dx=='\n')){
         x.get(dx);
         sx = !x.eof();
      }
   }else{
      sa=false;
   }
}


int main(int argc, char *argv[])
{
  bool sa;
  string da;
  A a("x.txt");

  a.open();
  a.read(sa, da);
  while(sa){
     if(da.length()==5){
        cout << da << endl;
     }
     a.read(sa, da);
  }
  return 0;
}
```