Feladat:

Egy szöveges fileból írjuk ki a képernyőre (soronként egyet) az 5 betü hosszú szavakat. A szavak egy vagy több szóközzel, vagy soremelés karakterrel vannak elválasztva.

Megoldás[1]:

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 : G* (amiből szavakat olvasunk).

Ekkor a feladat visszavezethető az egyváltozós, egyértékű elementi feldolgozás tételére, a *a : G* absztrakt file-ból való olvasással.

Az absztrakt *a : G* file-t össze kell kapcsolni az eredeti *x : F* file-val. Az absztrakt file-ból egy szó olvasása meg fog felelni az eredeti file-ból több karakter olvasásának. Az absztrakt olvasó művelet (az absztrakt READ[2]) ezért egy összetettebb program lesz.

A megoldás menete:

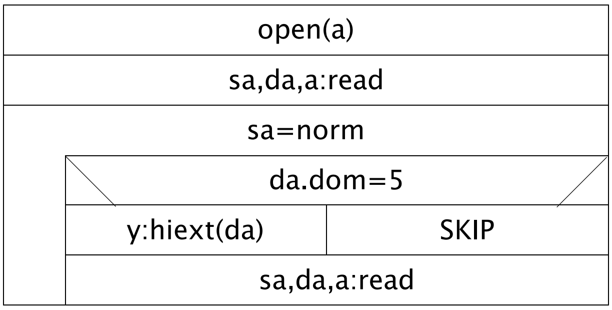
Készítsük el az absztrakt programot. Az absztrakt file-t C++-ban osztállyal fogjuk ábrázolni, ezért definiáljuk az absztrakt file osztályt a hozzá tartozó `open` és `read` műveletekkel.

1 Egy másik lehetséges megoldás a rekurzív függvény használata... (de azt mindenki ismeri...)

2 Innét az "absztrakt READ" elnevezés

Az absztrakt megoldó program struktogramja

Egyváltozós egyértékü elementkénti feldolgozás, az olvasott elemet akkor írjuk ki, ha a hossza 5.



A struktogramhoz tartozó C++ program:

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

A főprogram:

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

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

Az absztrakt file típusának a megvalósítása

Az `A` nevü osztály mezőinek a definíciója:

```
class A{
   ifstream x;
   status   sx;
   char     dx;
```

Osztály konstruktorának és a két műveletének a deklarációja:

```
public:
   A(const string& fn);
   void open();
   void read(status& sa, string& da);
```

Az absztrakt típus megvalósítása:

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

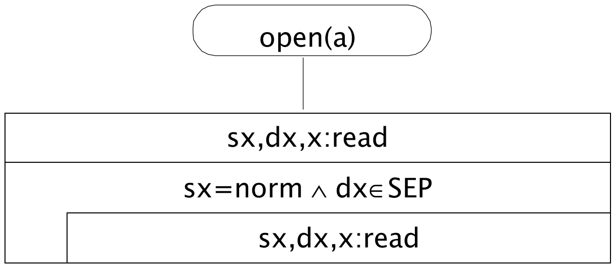
A konstruktor megvalósítása:

Az eredeti és az absztrakt file-nak azonos a neve.

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

Open művelet

Az absztrakt open művelet biztosítja, hogy a read elején igaz legyen az az invariáns tulajdonság, hogy vagy `sx=abnorm` (vége az eredeti file-nak) vagy `dx` az eredeti file első karakterét tartalmazza. Ezt úgy éri el hogy "levágja" (beolvassa de nem írja ki) az eredeti file elejéről az elválasztó (szóköz vagy soremelés) karaktereket.



Az absztrakt open-t megvalósító C++ programrész:

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

Három operandusú operátor ("makró kifejezés")

```
<kifejezés> ? <igaz_kif._esetén_az_érték> : <hamis_kif._esetén_az_érték>
```

Példa:

"rövidített írásmód"

```
sx = x.eof() ? abnorm : norm;
```

If-then-else szerkezetes írásmód

```
if (x.eof()) {
   sx = abnorm;
}else{
   sx = norm;
}
```

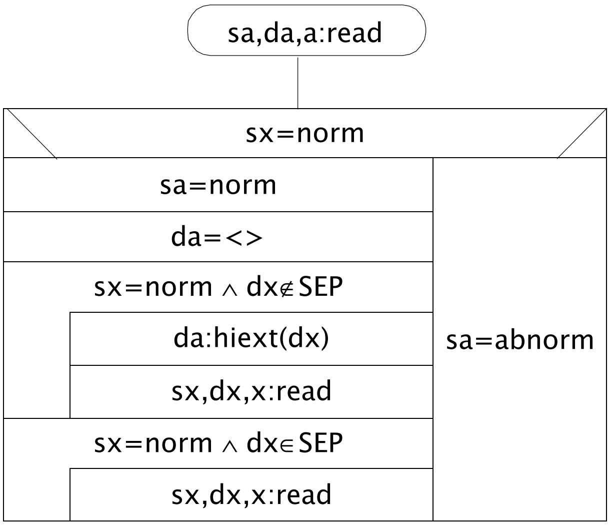
A struktogramban szereplő absztrakt olvasás

```
sx,dx,x:read
```

C++ megfelelő

```
x.get(dx);
sx = x.eof() ? abnorm : norm;
```

A read művelet



A read műveletet megvalósító C++ programrész:

```
void A::read(status& sa, string& da)
{
   if(sx==norm){
      sa = norm;
      da = "";
      while(sx==norm && dx!=’ ’ && dx!=’\n’){
         da+=dx;
         x.get(dx);
         sx = x.eof() ? abnorm : norm;
      }
      while(sx==norm && (dx==’ ’ || dx==’\n’)){
         x.get(dx);
         sx = x.eof() ? abnorm : norm;
      }
   }else{
      sa=abnorm;
   }
}
```

A teljes megoldóprogram

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

enum status {norm, abnorm};

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

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

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

void A::read(status& sa, string& da)
{
   if(sx==norm){
      sa = norm;
      da = "";
      while(sx==norm && dx!=' ' && dx!='\n'){
         da+=dx;
         x.get(dx);
         sx = x.eof() ? abnorm : norm;
      }
      while(sx==norm && (dx==' ' || dx=='\n')){
         x.get(dx);
         sx = x.eof() ? abnorm : norm;
      }
   }else{
      sa=abnorm;
   }
}

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

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