# Feladat:

5. Keressük egy valós számokat tartalmazó tömbnek azt az indexét, amelyre teljesül, hogy v[k]=(v[k-1]+ v[k+1])/2.

# Specifikáció:

Az értékék tömbbe kerülnek, a tömb méretének tisztázása után.

*A* = ( x : ^n l: s :)

*Ef* = (x = x’)

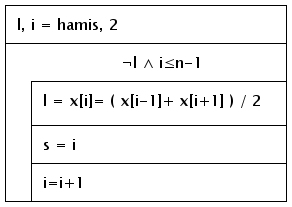
*Uf* = ( Ef (s,l) = x[i]= ( x[i-1]+ x[i+1] ) / 2 )  
Algoritmus:

A feladatot a lineáris keresés programozási tételének „pesszimista” változatára vezetjük vissza. Mivel a tétel kimenetei között szereplő index a feladat szempontjából a legfontosabb, ezért azt semmiképp sem hagyhatjuk ki az értékadásból. Maga az implementált program ezen felül közli a tömbben tárolt értéket is.

*Keresés Tétele: | Feladat:*

*m..n ~ 2..n-1*

*f(i) ~ x[i]= ( x[i-1]+ x[i+1] ) / 2*



# Implementáció :

Adattípusok:

Egy n hosszú vektorban (tömbben) [n természetes szám (0 is), a felhasználó határozza meg], tároljuk a valós számokat. Továbbá deklarálunk egy l (*bool*  isExist) logikai változót, a keresés sikeressége végett, egy integert az index ( *int* index),és egy valós változót, ami a több egy értékére fog mutatni. Továbbá segédváltozó a program újrafutatásához (IO műveletekhez : *char* a).

Program menete:

1. **Deklaráció:**

Bekéri a tömb méretét a felhasználótól (n), majd n hosszúra méretezi a valós vektort.

1. **Inicializálás:**

Feltölteti a felhasználóval a tömböt.

1. **Keresés:**

Végig megy a tömbön és megkeresi az első adott tulajdonságú elemet, ha van, ha nincs hamissal tér vissza.

1. **Kiértékelés:**

Tájékozatja a felhasználót a keresés sikerességéről, és közli a számításait,ezután felajánlja a program újrafuttatását.

# Tesztterv:

1. A feladat specifikációjára épülő (fekete doboz) tesztesetek:
2. Van a keresett elemből
3. Nincs a keresett elemből
4. A megoldó programra épülő (fehér doboz) tesztesetek:
5. Felhasználói kapcsolat: (újrafuttatás, beolvasás)
6. 0 méretű tömb esete
7. 1,2 mérető tömbök esetei
8. 3 és nagyobb méretű tömbök esete