Dokumentáció
Balogh Ferenc
LNBZVX
2. Beadandó

1. Feladat

Valósítsuk meg az azonos állású négyzetek típusát. Ennek értékei a sík olyan négyzetei, amelyek közül bármelyik kettőnek egy-egy oldala vagy párhuzamos, vagy merőleges. A reprezentációhoz képzeljen el a négyzetek oldalaival párhuzamos, illetve merőleges tengelyű derékszögű koordináta rendszert, és tegye fel, hogy csak az első sík-negyedbe (ahol a pontok koordinátái nem negatívok) eső négyzetekkel van dolgunk. Egy négyzetet létre lehet hozni a két ellentétes csúcsa alapján, továbbá le lehet kérdezni a területét és kerületet, valamint meg lehet vizsgálni, hogy egy adott pont benne van-e. Ehhez valósítsuk meg a síkbeli pont típusát is. A főprogram rögzítsen egy síkbeli pontot, töltsön fel egy tömböt négyzetekkel, majd minden olyan négyzet paraméterét írja ki, amely a rögzített pontot tartalmazza.

2. Megoldások és meggondolások

- A pontot egy struktúrában valósítottam meg, amiben az adattagok is publikusak maradtak. Így számos standard művelet maradt elérhető, amelyekkel éltem is a program többi részében.
- A pont struktúra integer típusú pont koordinátákat tárol. Mivel a feladat nem követelte meg a sablonok használatát ezt elhagytam, ugyanis meglehetősen sok és fáradságos határmunkálatot igényel az egyes műveletek biztosítása és ellenőrzése (pl. Rendezési relációk lebegő pontos számokon).
- A struktúra implementációja egy az egyben egy header-fileba került, amit egy makróval biztosítottam a re-definíciós veszélyek elhárítása miatt.
- Megvalósítottam egy Rectangle osztályt melynek példányait a feladatkiíráshoz hűen a négyszögek két átellenes csúcsával lehet deklarálni.
- A Rectangle osztály egy headerben van elhelyezve, azonban egy másik állományba tettem a függvénydefiníciókat.
- A könnyebb kezelhetőség érdekében operátor túlterhelésekkel bővítettem ki a struktúrát és az osztályt is. Elsősorban az írás-olvasás-lekérdezés műveletek egyszerűbb elérésére (bár született több olyan tagfüggvény, ami a későbbiek során mégsem került felhasználásra).
- Az infrastruktúra kialakítása után a Main-nek lényegében két feladata maradt: az adatok beolvasása és feldolgozása. Természetesen mindezt a rendelkezésre álló függvényhívásokkal.

3. Implementáció

A most következő két táblázatban egységesen áttekinthetőek a Point és Rectangle szerkezetek elemei és azok főbb jellemzőik:

ss_point.h	Point
ELEM	RENDELTETÉS
x, y	Adattagok, a koordináták tárolására
ss_point(int,int)	Konstruktor
ss_point()	Default konstruktor, az érvénytelen adatok kivédésére
Operator ==	Azonos pontok összehasonlítása
Operator >>	Könnyű olvasás
Operator <<	Könnyű írás
float ss_tav(const ss_point&)	Ponttávolság terület és kerület számításhoz

1. táblázat: Pont megvalósítása

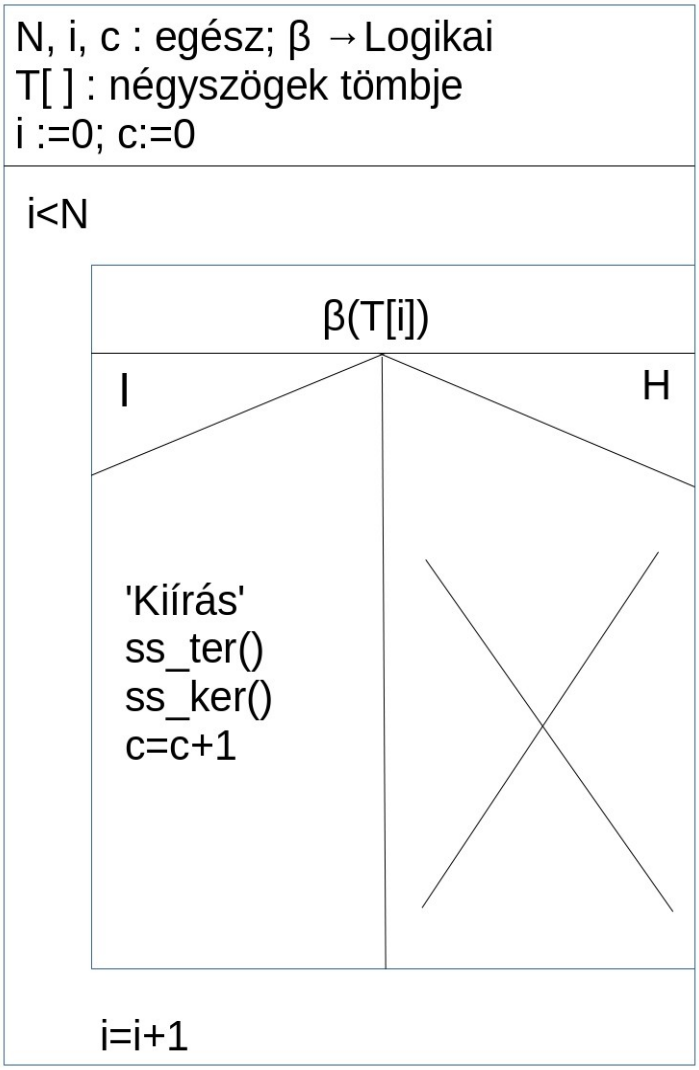
Rectangle	ss_rectangle.h
A, B, C, D	Privát adattagok, ss_point típusúak
ss_rectangle(const ss_point&, const ss_point&)	Konstruktor
ss_rectangle(const ss_rectangle&)	Copy-konstruktor
ss_rectangle()	Default konstruktor, az érvénytelen adatok kivédésére
~ss_rectangle()	Destruktor
Operator ()	A könnyebb adatelérésért
Operator >>	Könnyű olvasás
Operator <<	Könnyű írás
const float ss_ter()	Teület számítás
const float ss_ker()	Kerület számítás
const bool ss_p_benne(const ss_point&)	Pont illeszkedésének egyszerű vizsgálata

2. táblázat: Négyzet megvalósítása

A teljes program a következő egyszerű szerkezetbe foglalható: két absztrakt típus kiszolgál egy programot. Az állományok egymáshoz való viszonya megtekinthető az alábbi ábrán:

1. ábra: szerkezeti felépítés

A beolvasás csak fileból lehetséges és meglehetősen instabil. Ha figyelembe vesszük, hogy a könnyebb beolvasás érdekében a beolvasó operátorokat túlterheltem nyilvánvaló, hiszen a flag-ek megvalósítása körülményes ezért azt el is hagytam a programból ezáltal nem is olyan jól ellenőrizhetőek ezek az operátor műveletek.
A program lényegi rész viszont meglehetősen rövid, azaz kevés sort tesz ki a teljes implementációból. A megadott pontra illeszkedő téglalapok kereséséhez elkészítettem a stuktogramot, amit a 2. ábrán lehet megtekinteni.



2. ábra: Stuktogram

4. Tesztek

A program tesztelésére a 'b.txt' nevű mintafile áll rendelkezésre. A pontok száma páratlan, amit a program kezel és a pár nélküli pontot figyelmen kívül hagyja. A file pont koordináta párokat tartalmaz, melyekből a program téglalapokat készít, majd kiírja a kapott téglalapokat:
$
A filebol kinyert negyszogek:
1 1
4 1
4 5
1 5

1 3
10 3
10 9
1 9

3 2
7 2
7 8
3 8

5 3
9 3
9 6
5 6

9 1
11 1
11 2
9 2

Négy teszt pontot jelöltem ki, az illeszkedés vizsgálatra és keresésre:
(5,1) : 0 db pont illeszkedik
(9,2) : 1 db pont illeszkedik (határeset, az illeszkedés a csúcson van)
(8,4) : 2 db pont illeszkedik
(4,3) : 3 db pont illeszkedik (határeset, az illeszkedés oldallapoknál történik)

A második teszt teljes kimenete megtekinthető:
[sergei@localhost v07 (master=)]$ ./a.out b.txt
Alkalmazott Modul 2. beadando
Keszitette: Balogh Ferenc (LNBZVX)
Az adatfile nevet programinditaskor argumentumkent kell megadni.
Ellenkezo esetben es hibas filenev eseten a program leall
A file tartalmanak helyesseget a program nem ellenorzi!
Hibas megis olvashato adat eseten a program fut tovabb.

A fileban az elso adat egy egesz legyen ami a teglalapok szamanak ketszereset adja meg illetve a definialo csucsok osszes szamat.
A negyzeteket ket atellenes csucsanak koordinataja definialja, szokozzel vagy ujsorral elvalasztva.

A filebol kinyert pontok:
1 1
4 5
1 3
10 9
3 2
7 8
5 3
9 6
9 1
11 2

A filebol kinyert negyszogek:
1 1
4 1
4 5
1 5

1 3
10 3
10 9
1 9

3 2
7 2
7 8
3 8

5 3
9 3
9 6
5 6

9 1
11 1
11 2
9 2

Keressuk azon teglalapokat, amelyekre illeszkedik a megadott pont.
Hibas koordinata ertekek eseten a program a default ertekekkel fog dolgozni.
Kerem adja meg az adott pont koordinatait (ket egesz szam): 9 2
A kovetkezo negyzetek illeszkednek 9 2 pontra.
9 1
11 1
11 2
9 2
Terulete: 2
Kerulete: 6
Az osszes negyzet, amelyre a pont illeszkedik: 1
[sergei@localhost v07 (master=)]$