Objektum elvű alkalmazások fejlesztése

tárgyfelelős: Gregorics Tibor

Tantárgy

- Cél: objektum elvű alkalmazások fejlesztéséhez szükséges kódolási készség kialakítása.
- Előfeltétel:
  - Programozás
- Eszköz: C++
- Kapcsolat:
  - Szoftvertechnológia
  - Algoritmusok és adatszerkezetek 1.
  - Programnyelvek C++
- Előadás (heti 1 óra)
  - Mintapéldák megoldása
  - Legszükségesebb háttérismeretek
- Konzultációs gyakorlat (heti 1 óra)
  - Házi feladatok bemutatása
  - Géptermi zárthelyi

Összevont számonkérés

- Kötelező házi feladatok:
  - 4 darab feladat max. 5-5 pontért
  - határidőre (legfeljebb 4 hét késés, de szorgalmi időszakban)
- Géptermi zárthelyik:
  - félév közi és félév végi max 5-5 pontért
  - félév végi zárthelyi egyszer ismételhető
- Gyakorlati jegy: megszerzett pontok átlaga
  - zárthelyik duplán számítanak
  - ha az összes házi feladat legalább 1 pontos
  - ha a félév végi zárthelyi legalább 2 pontos

Háttér anyagok

- http://people.inf.elte.hu/gt/oaf
  - Követelmények
  - Házi feladatok
  - Egyéb segédanyagok
- Az előadások diáit a Neptunból tölthetik le
  - Meet Street / Virtuális tér / Tantárgy név / Dokumentumok
- Gregorics Tibor: Programozás (2.kötet) – Megvalósítás. ELTE-Eötvös Kiadó, 2013.

Objektum elvű alkalmazások fejlesztése

Típus és osztály

Készítette: Gregorics Tibor
Sike Sándor

Feladat

Készítsünk olyan programot, amely sokszögekkel dolgozik. Egy sokszög csúcspontjainak koordinátái legyenek egész számok.
Feltöltünk egy tömböt különféle sokszögekkel, majd mindegyiket eltoljuk ugyanazon mértékkel, és kiszámoljuk az így nyert sokszögek súlypontjait.
Ehhez bevezetjük a sokszögek, illetve a sokszögeket alkotó síkbeli pontok osztályát. Szükség lesz továbbá egy sokszög eltolására, ehhez egy pont eltolására, valamint egy sokszög súlypontjának kiszámítására.

Specifikáció és algoritmus

$A = (t : \text{Sokszog}^n,\ p : \text{Pont},\ sp : \text{Pont}^n)$

$Ef = (t = t' \wedge p = p')$

$Uf = (p = p' \wedge \overset{n}{\underset{i=1}{\forall}} t[i] = \text{eltol}(t'[i], p) \wedge sp[i] = \text{súlypont}(t[i]))$

$Uf = (p = p' \wedge t = \overset{n}{\underset{i=1}{\oplus}} < \text{eltol}(t'[i], p) > \wedge sp = \overset{n}{\underset{i=1}{\oplus}} < \text{súlypont}(t[i]) >)$

i = 1 .. n

t[i] := eltol(t[i] , p)

sp[i] := súlypont(t[i])

Pont típusa

típus-értékek | típus-műveletek

típus-specifikáció

síkbeli pontok	b := eltol(a,p) (a,b,p:Pont)
x,y : ℤ	b.x, b.y := a.x+p.x, a.y+p.y

reprezentáció | implementáció

típus-megvalósítás

Pont
- x : int
- y : int

+eltol(Pont p) : Pont ○---- **return** Pont(x+p.x, y+p.y)

Pont osztály C++ kódja

```
#ifndef _PONT_H
#define _PONT_H

class Pont
{
   public:

      Pont eltol(const Pont &p) const;
   private:
      int _x;
      int _y;
};

#endif
```

Pont a,b,p;

b = a.eltol(p)

a this (a) nem változik

Pont
- x : int
- y : int

+eltol(Pont) : Pont

pont.h

Pont bővített típusa

síkbeli pontok	b:=eltol(a,p) (a,b,p:Pont)	újX(a,i) újY(a,i) (a:Pont, i:ℤ)	i:=x(a) i:=y(a) (a:Pont, i:ℤ)
x,y : ℤ	b.x,b.y:=a.x+p.x,a.y+p.y	a.x:=i a.y:=i	i:=a.x i:=a.y

a "setter"-ek egy pont attribútumait (rejtett adattagjait) módosítják

a "getter"-ekkel a rejtett adatok kérdezhetők le

Pont
- x : int
- y : int

+újX(int) : void
+újY(int) : void
+x() : int
+y() : int
+eltol(Pont) : Pont

Pont bővített osztályának C++ kódja

```
#ifndef _PONT_H
#define _PONT_H

class Pont
{
   public:
      void ujX(int x);
      void ujY(int y);
      int  x() const;
      int  y() const;
      Pont eltol(const Pont &p) const;
   private:
      int _x;
      int _y;
};

#endif
```

a.ujX(-3);

int i = a.x();

Pont
- x : int
- y : int

+újX(int) : void
+újY(int) : void
+x() : int
+y() : int
+eltol(Pont) : Pont

pont.h

Alapértelmezett konstruktor felüldefiniálása

```
#ifndef _PONT_H
#define _PONT_H

class Pont
{
   public:
      Pont();     // konstruktor

      void ujX(int x);
      void ujY(int y);
      int  x() const;
      int  y() const;
      Pont eltol(const Pont &p) const;
   private:
      int _x;
      int _y;
};

#endif
```

Egy origóbeli pontot hoz létre

Pont
- x : int
- y : int

+Pont()
+újX(int) : void
+újY(int) : void
+x() : int
+y() : int
+eltol(Pont) : Pont

pont.h

Pont osztály metódusainak C++ kódja

```
#include "pont.h"

Pont::Pont()
{
    _x = _y = 0;
}

void Pont::ujX(int x)
{
    _x = x;
}

void Pont::ujY(int y)
{
    _y = y;
}
```

```
int Pont::x() const
{
    return _x;
}

int Pont::y() const
{
    return _y;
}

Pont Pont::eltol(const Pont &p) const
{
    Pont c;
    c.ujX(_x + p._x); c.ujY(_y + p._y);
    return c;
}
```

return Pont(x+p.x, y+p.y)

pont.cpp

eltol() metódus másként – új konstruktor

```
class Pont
{
   public:
       Pont();
       Pont(int x, int y);      // új konstruktor
       …
};

Pont::Pont(int x, int y)
{
    _x = x;
    _y = y;
}

Pont Pont::eltol(const Pont &p) const
{
   return Pont(_x + p._x, _y + p._y );
}
```

eltol() metódus használata

```
Pont a(3,4);
Pont b;
b = a.eltol(Pont(-1,-1));
```

konstans kifejezés átadható az Eltol() p konstans paraméterváltozójának

eltol() előtt:

b:Pont	=	a:Pont	.eltol(	:Pont	)
x = 0 y = 0		x = 3 y = 4		x = -1 y = -1	

return Pont(x+p.x, y+p.y)

eltol() után:

b:Pont	értékadás	:Pont
x = 2 y = 3		x = 2 y = 3

Alapértelmezett másoló konstruktor és értékadás operátor

Minden osztály rendelkezik az alábbiakhoz hasonló metódusokkal:

```
Pont(const Pont &p);
Pont& operator=(const Pont &p);
```

Pont a(1,2);
Pont b(a); vagy Pont b = a;

Pont a(1,2);
Pont b,c;
b = a; vagy akár c = b = a;

tagonkénti másolás

b:Pont	a:Pont
x = 1 y = 2	x = 1 y = 2

eltol() metódus egy kifejezőbb formája

```
Pont a(3,4);
Pont b;
b = a + Pont(1,1);        // b = a.operator+(Pont(1,1))
```

```
class Pont
{
   public:
       …
       Pont operator+(const Pont &p) const;
       …
};

Pont Pont::operator+(const Pont &p) const
{
   return Pont(_x + p._x, _y + p._y );
}
```

Új operátor a súlypont számításhoz: Osztás

A sokszög csúcspontjait ez a tömb tartalmazza: ennek elemeit összegezzük az „eltolás” műveletével

```
Pont sp;
for(int i=0; i<n ; ++i)  sp = sp + s[i];
sp.ujX(sp.x()/n);  sp.ujY(sp.y()/n);
```

Szebb lenne:
sp = sp / n;

```
class Pont
{
   public:
       …
       Pont operator/(int f) const;
       …
};

Pont Pont::operator/(int f) const
{
   return Pont(_x / f, _y / f );
}
```

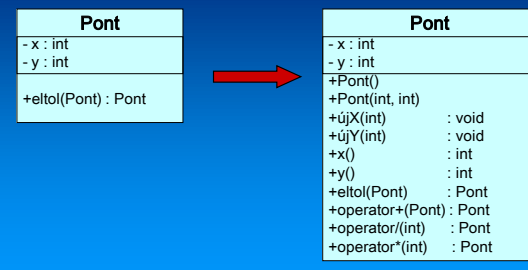
Új operátor csak úgy: Nyújtás

```
class Pont
{
public:
    …
    Pont operator*(int f) const;
    friend Pont operator*(const int f, const Pont &p);
    …
};

Pont Pont::operator*(int f) const
{
    return Pont(_x * f, _y * f );
}

Pont operator*(const int f, const Pont &p)
{
    return Pont( f * p._x, f * p._y );
}
```

Pont a, b;
b = a * 5;

Pont a, b;
b = 5 * a;

Pont osztálya



Sokszög típusa

sokszögek	s := eltol(s,p) (s:Sokszog, p:Pont) s-re vonatkozó művelet: s.eltol(p)	sp := súlypont(s) (s:Sokszog, sp:Pont)
pontok : Pont* |pontok| ≧ 3 invariáns	∀i∈[1 .. |pontok|]: s.pontok[i] := eltol(s.pontok[i],p)	$sp := (\sum_{i=1}^{\|pontok\|} s.pontok[i]) / \|pontok\|$

Sokszog
- pontok : int[]

+eltol(Pont) : void
+súlypont() : Pont

i = 1 .. n
t[i].eltol(p)
sp[i]:=t[i].súlypont()