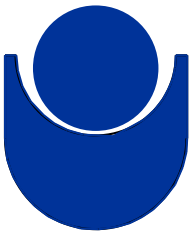


НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

Департамент Информатика

Проект по *аналитична геометрия* към курс CSCB209 Обектно-ориентирано програмиране

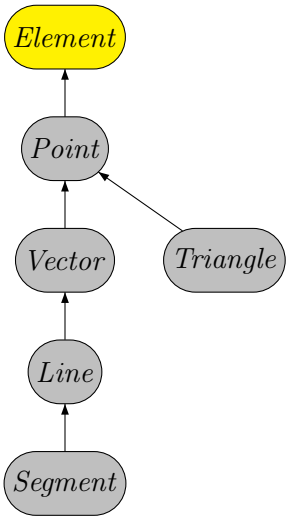
гл. ас. д-р Марияна Райкова mariana_sokolova@yahoo.com
гл. ас. д-р Стоян Боев stoyan@nbu.bg

Въведение

За разлика от синтетичната геометрия, която е изградена върху набор от непротиворечащи си аксиоми, аналитичната геометрия изучава геометричните обекти с помощта на т.нар. *Декартова координатна система*, използвайки принципите на алгебрата и анализа. Аналитичната геометрия е основоположна за различни области на информатиката като например компютърна графика, линейно програмиране и др. Тя се изгражда върху основите на векторната алгебра и на координатния метод. Този метод се прилага за аналитичното представяне на точките, правите и равнините на тримерното евклидово пространство и за изследване на взаимното им разположение.

Постановка на задачата

Да се напише програма на C++, която реализира функционалност за следните обекти в пространството: точка (*Point*), вектор (*Vector*), права линия (*Line*), отсечка (*Segment*) и триъгълник (*Triangle*), чрез класове и наследяване. Да се реализира йерархията показана на Фиг. 1, в която базов клас е абстрактен клас *Element*.



Фигура 1. Йерархия от геометрични обекти

Точка

Всяка точка се описва в пространството, чрез тройка реални числа (x, y, z). Да се реализира клас *Point*, който да предоставя средства за създаване на точка. За класа *Point* да се предефинира оператора ==, който проверява дали две точки съвпадат и ако съвпадат да връща true, а в противен случай false.

Вектор

В пространството един вектор се описва с тройка реални числа (x, y, z) или чрез две точки - начало A и край B, които еднозначно определят вектор с компоненти

$$(x_B - x_A, y_B - y_A, z_B - z_A).$$

Да се реализира клас *Vector*, чиито обект може да се инициализира по два начина – или чрез три реални стойности или чрез две точки. Класът *Vector* да предоставя член-функции за:

- изчисляване на дължина на вектор, която връща реално число;
- изчисляване на посока на вектор, която връща единичен вектор или хвърля изключение от потребителски дефиниран клас за изключения *VectorLengthException* (наследява стандартния клас за изключения), ако дължината на вектора е нула;
- проекция на вектор върху друг вектор, която връща вектор;
- проверка за нулев вектор. Член функцията връща true, ако $x = y = z = 0$ и false в противен случай;
- проверка за успоредност на текущия вектор (x, y, z) и зададен вектор (v_1, v_2, v_3). Член-функцията връща true, ако $x : v_1 = y : v_2 = z : v_3$ и false в противен случай. Тя

хвърля изключение от тип *VectorLengthException*, ако един от двата вектора е с нулева дължина.

- проверка за перпендикулярност на текущия вектор (x, y, z) и зададен вектор (v_1, v_2, v_3). Член-функцията връща true, ако $x.v_1 + y.v_2 + z.v_3 = 0$ и false в противен случай. Тя хвърля изключение от тип *VectorLengthException*, ако един от двата вектора е нулев вектор.

Нека с $a = (x, y, z)$ означим текущия вектор от класа (this). Следните операции с вектори да се реализират като предефиниране на операция, като сами определите дали ще се предефинират като член-функции или като външни функции:

- събиране на два вектора, чрез операция $+$, като резултат се получава отново вектор:

$$a + v = (x, y, z) + (v_1, v_2, v_3) = (x + v_1, y + v_2, z + v_3);$$

- изваждане на два вектора, чрез операция $-$, като резултат се получава отново вектор:

$$a - v = (x, y, z) - (v_1, v_2, v_3) = (x - v_1, y - v_2, z - v_3);$$

- умножение на вектор с реално число, чрез операция $*$, като резултат се получава отново вектор:

$$r * a = (rx, ry, rz);$$

- скаларно произведение на два вектора, чрез операция $*$, като резултат се получава реално число:

$$a * v = x.v_1 + y.v_2 + z.v_3;$$

- векторно произведение на два вектора, чрез операция $\wedge$ като резултатът е вектор.

$$a \wedge v = (y.v_3 - z.v_2, -x.v_3 + z.v_1, x.v_2 - y.v_1).$$

- смесено произведение на три вектора, чрез операция () с аргументи на операцията два вектора и резултат реално число. Например: Ако имаме $(u \times v).w$, ще извикваме операцията по следния начин $u(v, w)$, където

$$(u \times v) \cdot w = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}.$$

Права линия

Друг геометричен обект в пространството е правата линия. Да се реализира клас *Line*, който наследява класа *Vector* и който дава възможност за инициализация на обект от класа, чрез точка и вектор или чрез две точки. Ако се използва конструктор с две точки, то за да се инициализира отсечката, трябва да се образува вектор от две точки, след което на база на получения вектор и първата от двете точки да се инициализира отсечка.

За класа права линия да се реализират следните член-функции:

- за намиране на посока на правата - връща вектор, успореден на правата;

- за намиране на нормален вектор - връща вектор, перпендикулярен на правата;
- за намиране на ъгъл между две прави - връща стойността на ъгъла в радиани.

За класа Line да се предефинират следните оператори, като сами определите дали ще се предефинират като член-функции или като външни функции:

- за проверка дали дадена точка лежи на дадената права, чрез оператора +, като връща true ако лежи и false в противен случай;
- за проверка дали дадена права е успоредна на друга права, чрез оператора ||, като връща true ако е успоредна и false в противен случай;
- за проверка дали права съвпада с друга права, чрез оператора ==, като връща true ако е съвпада и false в противен случай;
- за проверка дали права пресича друга права, чрез оператора &&, като връща true ако пресича и false в противен случай;
- за проверка дали права е кръстосана с друга права, чрез оператора ! =, като връща true ако е кръстосана и false в противен случай;
- за проверка дали права е перпендикулярна на друга права, чрез оператора |, като връща true ако е перпендикулярна и false в противен случай.

Отсечка

Друг геометричен обект в пространството е отсечката. Да се реализира клас *Segment*, който разширява класа *Line*, като допълнително предоставя член-данни за начало и край на интервал. Нека имаме права линия, дефинирана чрез точка $A(a_1, a_2, a_3)$ и вектор $b(b_1, b_2, b_3)$, и нека имаме интервал $t \in [0, 1]$. Тогава можем да получим отсечка, дефинирана параметрично като

$$x = a_1 + b_1 t,$$
$$y = a_2 + b_2 t,$$
$$z = a_3 + b_3 t,$$

където $t \in [0, 1]$. Следователно, за да се инициализира една отсечка ние се нуждаем от две точки - начало и край на отсечката, с помощта на които ще определим и интервала в параметричното представяне на отсечката.

За всяка отсечка да бъдат декларирани член-функции за:

- намиране на дължина на отсечка, която връща положително реално число;
- намиране на среда на отсечка, която връща точка с координати $((x_A + x_B)/2, (y_A + y_B)/2, (z_A + z_B)/2)$, където A и B са начална и крайна точка за отсечката.

За отсечката да се предефинира оператора ==, който проверява дали една точка лежи на дадената отсечка и връща true, ако лежи и false, в противен случай.

Триъгълник

Друг геометричен обект е триъгълникът, който се дефинира еднозначно чрез три точки в пространството. При опит да се инициализира триъгълник, чрез точки, някои от които съвпадат, да се изхвърля изключение от тип *EqualPointException* - потребителски дефиниран клас, който наследява стандартния клас за изключения. *EqualPointException* съобщава на потребителя, кои са съвпадащите точки, причинили изключението.

За класа *Triangle* да се дефинират член-функции за:

- определяне вида на триъгълника
  (равнобедрен/равностранен/остроъгълен/правоъгълен/тъпоъгълен);
- намиране лицето на триъгълника;
- намиране периметъра на триъгълника;
- намиране медицентъра на триъгълника, като връща обект от тип Point.

За класовете *Triangle* и *Point* да се предефинират операторите $<$, $>$ и $==$ по следния начин:

1. ако сравняваме точка и триъгълник с оператор $<$, то резултатът от операцията ще бъде true само ако точката е от вътрешността на триъгълника, във всички останали случай ще връща false;
2. ако сравняваме точка и триъгълник с оператор $>$, то резултатът от операцията ще бъде true само ако точката е извън триъгълника, във всички останали случай ще връща false;
3. ако сравняваме точка и триъгълник с оператор $==$, то резултатът от операцията ще бъде true само ако точката лежи върху някоя от страните на триъгълника, във всички останали случай ще връща false,

Създаване и отпечатване

За всеки обект от йерархията, трябва да можем да печатаме и четем данни в/от конзолата и файл, чрез предефиниране на операторите за вмъкване и четене в зададен поток.

При взаимодействие с потребителя той трябва да избере от меню какъв геометричен елемент иска да въведе, след което системата да изисква от него въвеждане на необходимите данни. За целта трябва да имаме създаден указател към абстрактния клас и след това да „прикрепяме“ към него обекти от Например:

1. Моля изберете вид геометричен обект:

   1 – Точка

   2 – Вектор

   3 – Линия

   4 – Отсечка

   5 – Триъгълник

   /* потребителят въвежда */

   2

2. Моля въведете стойност x на вектора: **1**
3. Моля въведете стойност y на вектора: **1**
4. Моля въведете стойност z на вектора: **2**

   /* след въвеждане на желания обект се появява меню с всички възможни операции които могат да се извършват с този обект */
5. Моля изберете операция за обекта:

   /* изброяват се всички операции, които можем да изпълняваме */

   1 - изчисляване на дължина на вектор

   2 - изчисляване на посока на вектор

   3 - проекция на вектор върху друг вектор

   4 - проверка за нулев вектор

   5 - проверка за успоредност на два вектора

   6 - проверка за перпендикулярност на два вектора

   7 - събиране на два вектора

   8 - умножение на вектор с реално число

   9 - скаларно произведение на два вектора

   10 - векторно произведение на два вектора

   11 - смесено произведение на три вектора

   /* в зависимост от броя и вида на аргументите след избор на операция потребителя ще се подкани да въведе необходимите данни */

   7
6. Моля въведете стойност x на вектора: **2**
7. Моля въведете стойност y на вектора: **2**
8. Моля въведете стойност z на вектора: **2**

   /* тук ще се извика и изпълни операция +, за събиране на два вектора (1, 1, 2) + (2, 2, 2) и резултатът е нов вектор (3, 3, 4), който ще се отпечата*/
9. (1, 1, 2) + (2, 2, 2) = (3, 3, 4)

   /* аналогично потребителя ще се подкани да продължи или не с въвеждането на операции */
10. Желаете ли да изберете нова операция (y/n)? **n**
11. Желаете ли да изберете нов геометричен обект(y/n)? **n**

    /* програмата приключва */

Данни и тип за геометричен обект могат да се четат и от файл. За фигура с горните характеристики ще имаме следния текстов файл:

Пример:

```
2
1 1 2
7
2 2 2
```

На първи ред от файла се записва вида на фигурата.

На втори ред от файла се записват последователно данни за фигурата разделени с интервал. Ако данните са комплекси всеки отделен обект се записва разделен със запетая.

На трети ред е операцията, която трябва да се приложи върху обекта.

На четвърти ред ще са всички допълнителни обекти, които са необходими за изпълнението на операцията.

Всички изключения да се прихващат чрез механизмите за обработка на изключения, така че да може програмата ни да продължи изпълнението си дори и при възникване на аварийни ситуации.

За всички класове да бъде реализирано правилото на големите пет. Нямате право за използвате string обекти, освен ако не напишете свой собствен потребителски дефиниран такъв клас и да ползвате него. Можете да ползвате шаблони.

В допълнение

Да се разшири йерархията от Фиг. 1 с геометрична фигура тетраедър, като се реализира клас *Tetrahedron*. Обект от този клас се инициализира чрез четири точки (точка A, точка B, точка C, точка D). Ако някоя от точките съвпадат при опит на инициализация на обект, то да се изхвърля изключение от тип *EqualPointException*.

За тетраедъра да се реализират следните член-функции за:

- проверка дали е правилен (всички ръбове са равни);
- проверка дали е ортогонален (всеки два срещуположни ръба са перпендикулярни);
- намиране на околна повърхнина;
- намиране на обем;

Да се предефинират операторите $<$, $>$, $==$ за взаимно положение на точка и тетраедър, подобно на тези от класа Triangle.