Документація до лабораторної роботи №1  
з дисципліни Основи Об’єктно орієнтоване програмування

Студента факультету   
комп'ютерних наук та кібернетики  
Групи ІПС-22  
Дехтяренка Михайла

Зміст

1. Завдання 3

2. Клас Shape. 4

3. Клас triangle 5

4. Quadrangle 6

5. pentagon 7

6. polygon 8

7. node 9

8. List 9

9. Використанні джерела 10

1. Завдання

Для виконання лабораторної роботи, було взяте перше завдання з першого списку та додатково реалізовано сортування з другого завдання того ж списку, з другого списку було взято перше завдання (реалізоване виділено жовтим кольором):  
   
1.1 (\*) Список, циклічний список. Реалізації на основі зв’язних списків, масивів змінної довжини, бібліотечних засобів. Пошук елемента за значенням, за індексом, першого елемента за заданою умовою.   
  
2.1 Фігури на площині: трикутники, чотирикутники, п’ятикутники. Обчислення периметра та площі фігури. Перевірка на спеціальні фігури – прямокутні, рівнобедрені, рівносторонні трикутники; трапеції, паралелограми, ромби, прямокутники, квадрати; правильні п’ятикутники.

* 1. +\* за інші спеціальні типи фігур;
  2. +\* за підтримку n-кутників для довільного n;
  3. +\* за перевірку опуклості фігур.

Мова програмування С++.

2. Клас Shape

Клас Shape або фігура був створений як батьківський клас для всіх фігур, які будуть реалізовані далі в проекті. Цей клас має 5 захищених(protected) полів та 11 відкритих(public) методів.  
  
Поля:  
square – поле, в якому зберігається площа фігури   
perimeter – поле, в якому зберігається периметр фігури  
points – поле двовимірного масиву для збереження координат точок фігури   
sides – поле одновимірного масиву для збереження довжини сторін  
vectors – поле двовимірного масиву для збереження векторів фігури

Методи (віртуальні, абстрактні):  
getSquare() – метод для підрахунку площі фігури.  
  
getPerimeter() – метод для підрахунку периметру фігури.  
  
calculateSides() – метод для підрахунку периметру фігури.  
  
calculateVectors() – метод для підрахування векторів фігури.  
  
isConvex() – метод для перевірки фігури на опуклість.  
  
operator>(shape&), operator< (shape&, shape&), operator==(shape&, shape&) ,  
operator!= (shape&, shape&) – оператори потрібні для порівняння фігур за площою.  
  
operator<<(ostream& c, const shape& s) – перевантаження оператора «<<», щоб виводити в консольплощину фігури.  
  
swap (shape&, shape&) – метод для обміну фігур.

3. Клас Triangle

Triangle, або трикутник, є класом, що наслідує клас фігуру, тобто має перевизначені методи батьківського класу Shape (динамічний поліморфізм):  
getSquare() – метод для підрахунку площі фігури.  
  
getPerimeter() – метод для підрахунку периметру фігури.  
  
calculateSides() – метод для підрахунку периметру фігури.  
  
calculateVectors() – метод для підрахування векторів фігури.  
  
isConvex() – метод для перевірки фігури на опуклість  
  
 але при цьому має і власні методи:  
  
setPoints() – метод, який викликає цикл, щоб встановити точки.  
  
Далі йдуть булвеські методи, для визначення до якого типу відноситься  
трикутник   
  
isRectangular() - перевіряє чи є трикутник прямокутним.  
  
isIsosceles() – перевіряє чи є трикутник рівнобедреним.  
  
isEquilateral() – перевіряє чи є трикутник рівностороннім.  
  
В кінці класу реалізовані конструктори:  
  
triangle() – конструктор без параметрів.  
  
triangle(const triangle& x) – конструктор копіювання.

~triangle() – деструктор.

4. Клас quadrangle

Quadrangle, або чотирикутник, є класом, що наслідує клас фігуру, тобто має методи батьківського Shape:  
getSquare() – метод для підрахунку площі фігури.  
  
getPerimeter() – метод для підрахунку периметру фігури.  
  
calculateSides() – метод для підрахунку периметру фігури.  
  
calculateVectors() – метод для підрахування векторів фігури.  
  
isConvex() – метод для перевірки фігури на опуклість  
  
Але при цьому має і власні методи:

setPoints() – метод, який викликає цикл, щоб встановити точки.  
  
Далі йдуть булвеські методи, для визначення до якого типу відноситься  
трикутник   
  
isTrapezium () - перевіряє чи є чотирикутник трапецією  
  
isParallelogram () – перевіряє чи є чотирикутник паралелограмом   
  
isDiamond() – перевіряє чи є чотирикутник ромбом.

IsRectangular() – перевіряє чи є чотирикутник прямокутником

isSquare() – перевіряє, чи я чотирикутник квадратом  
  
В кінці класу реалізовані конструктори:  
  
quadrangle () –конструктор без параметрів.  
  
quadrangle (const quadrangle&) – конструктор копіювання.

~ quadrangle () – деструктор.

5. Клас pentagon

Pentagon, є класом, що наслідує клас фігуру, тобто має методи батьківського Shape:  
  
getSquare() – метод для підрахунку площі фігури.  
  
getPerimeter() – метод для підрахунку периметру фігури.  
  
calculateSides() – метод для підрахунку периметру фігури.  
  
calculateVectors() – метод для підрахування векторів фігури.  
  
isConvex() – метод для перевірки фігури на опуклість

setPoints() – метод, який викликає цикл, щоб встановити точки.

В кінці класу реалізовані конструктори:  
  
pentagon() –конструктор без параметрів.  
  
pentagon (const pentagon &) – конструктор копіювання.

~ pentagon () – деструктор.

6. Клас polygon

Polygon або багатокутник є шаблонним класом, для якого вхідним параметром є кількість вершин, оскільки нам невідома кількість потенціальних кутів. Цей клас, як і інші, наслідує клас фігуру, тобто має такі методи:

getSquare() – метод для підрахунку площі фігури.  
  
getPerimeter() – метод для підрахунку периметру фігури.  
  
calculateSides() – метод для підрахунку периметру фігури.  
  
calculateVectors() – метод для підрахування векторів фігури.  
  
isConvex() – метод для перевірки фігури на опуклість

setPoints() – метод, який викликає цикл, щоб встановити точки  
  
В кінці класу реалізовані конструктори:  
  
polygon() –конструктор без параметрів.  
  
polygon (const pentagon &) – конструктор копіювання.

~ polygon() – деструктор.

7. Структура node

Ця шаблонна структура задає вузол однозв’язного списку і має два відкриті поля:

val – данні

next – покажчик на наступний вузол.

template <typename T>

struct node {

T val;

node<T>\* next;

}

Таким чином, лабораторна робота включає елементи статичного поліморфізму.

8. Клас List

Клас Lіst має такі приховані поля

node<T>\* head

і такі відкриті методи:

AddNode (T val) – створює нові вузли в списку

Operator()(int i) – шукає вузол з індексом «і» та повертає його значення, або видає помилку

certainSearch(T val) – якщо існує вузлом з заданим значенням, виводить індекс вузла з цим значенням, інакше видає помилку

sort() – сортує список у порядку зростання

show() – виводе список

Джерела

Під час виконання цієї лабораторної роботи було використано:

Sololearn C++ intermidiare Course для здобуття загальних знань з ООП

Математичні форуми для знаходження формул