



Bazy Danych II
Dokumentacja Projektu

Obsługa danych przestrzennych dwuwymiarowych

Patryk Śledź

303806

12.06.2021

Informatyka Stosowana

Wydział Fizyki i Informatyki Stosowanej

Akademia Górniczo-Hutnicza w Krakowie

Spis treści

1.	Opis problemu.....	3
2.	Opis funkcjonalności udostępnianej przez API.	3
2.1.	Aplikacja terminalowa.	4
3.	Opis typów danych oraz metod (funkcji) udostępnionych w ramach API.....	7
3.1.	Klasy UDT.....	7
3.2.	Klasy aplikacji konsolowej.....	8
4.	Opis implementacji.....	9
5.	Prezentacja przeprowadzonych testów jednostkowych.....	11
6.	Podsumowanie i wnioski.	13
7.	Uruchomienie.....	13
8.	Literatura.....	14

1. Opis problemu.

Celem projektu było przygotowanie API i jego implementacji umożliwiającej przetwarzanie danych dwuwymiarowych z wykorzystaniem własnych typów danych **CLR** (Common Language Runtime), **UDT** (User Defined Types) oraz metod.

Aplikacja powinna umożliwić:

- wyznaczenie odległości pomiędzy punktami,
- wyznaczenie pola określonego obszaru,
- sprawdzenie czy punkt należy do danego obszaru.

2. Opis funkcjonalności udostępnianej przez API.

Stworzone API umożliwia przeprowadzanie operacji na punktach oraz wielokątach.

Aplikacja pozwala dodawać dowolną ilość punktów określonych przez współrzędne całkowite X oraz Y (dwa wymiary) oraz dowolną ilość wielokątów wypukłych określonych przez kombinację punktów.

Kolejność dodawanych punktów wielokąta nie ma znaczenia – zaimplementowano algorytm sortujący wierzchołki.

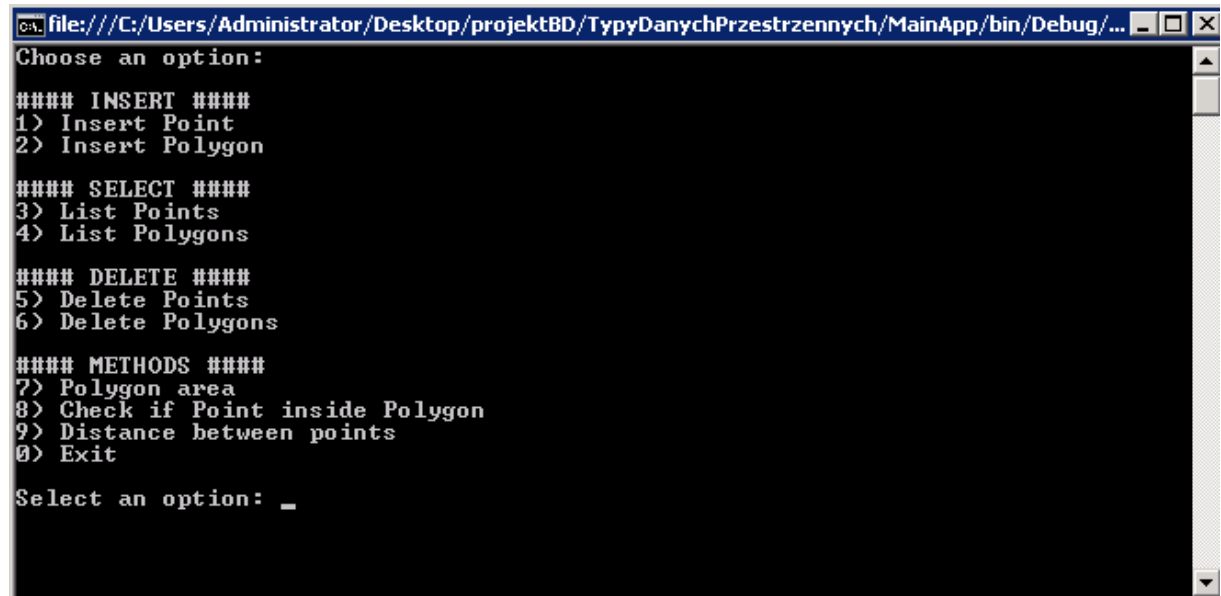
Dla punktów, udostępniono możliwość wyznaczania odległości między nimi oraz sprawdzenia czy punkt należy do obszaru wyznaczonego przez wielokąt.

Dla wielokątów umożliwiono wyznaczanie pola obszaru.

2.1. Aplikacja terminalowa.

W celu wygodnej obsługi złożonych typów danych udostępniono interfejs konsolowy.

W aplikacji konsolowej poruszamy się wybierając odpowiednie opcje, a następnie zatwierdzając klawiszem Enter.



```
file:///C:/Users/Administrator/Desktop/projektBD/TypyDanychPrzestrzennych/MainApp/bin/Debug/...
Choose an option:

#### INSERT ####
1> Insert Point
2> Insert Polygon

#### SELECT ####
3> List Points
4> List Polygons

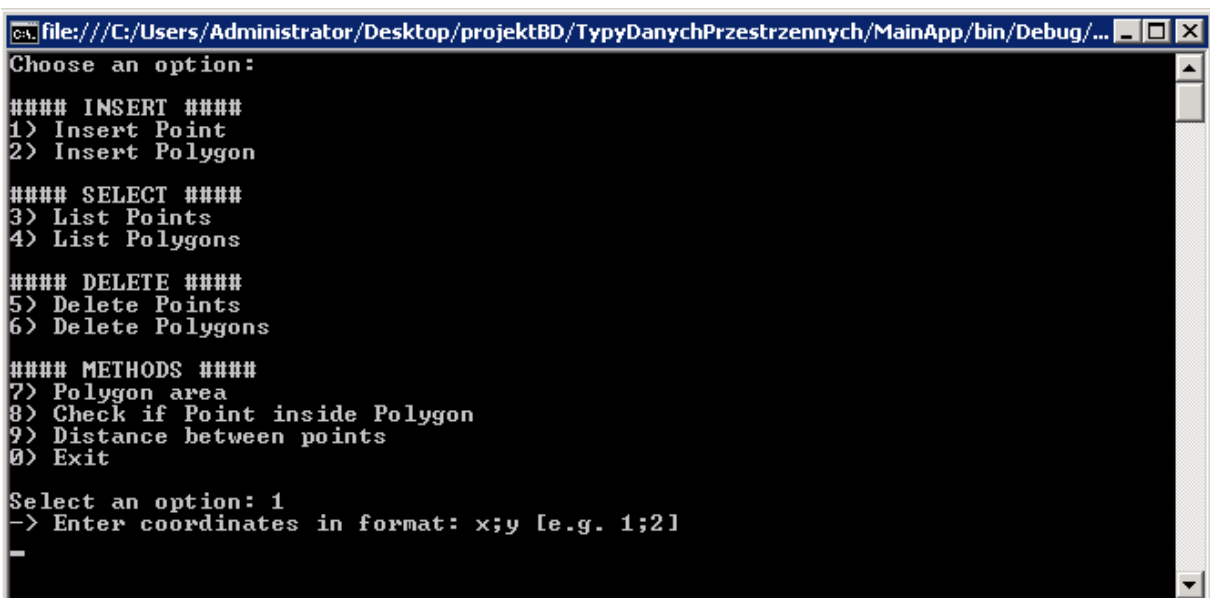
#### DELETE ####
5> Delete Points
6> Delete Polygons

#### METHODS ####
7> Polygon area
8> Check if Point inside Polygon
9> Distance between points
0> Exit

Select an option: _
```

Rysunek 1 Menu główne.

Wybierając opcję 1 lub 2 zostaniemy poproszeni o podanie odpowiednich danych w celu zapisania punktu/wielokąta do bazy danych. Przykładowe wejście jest podane w nawiasach kwadratowych.



```
file:///C:/Users/Administrator/Desktop/projektBD/TypyDanychPrzestrzennych/MainApp/bin/Debug/...
Choose an option:

#### INSERT ####
1> Insert Point
2> Insert Polygon

#### SELECT ####
3> List Points
4> List Polygons

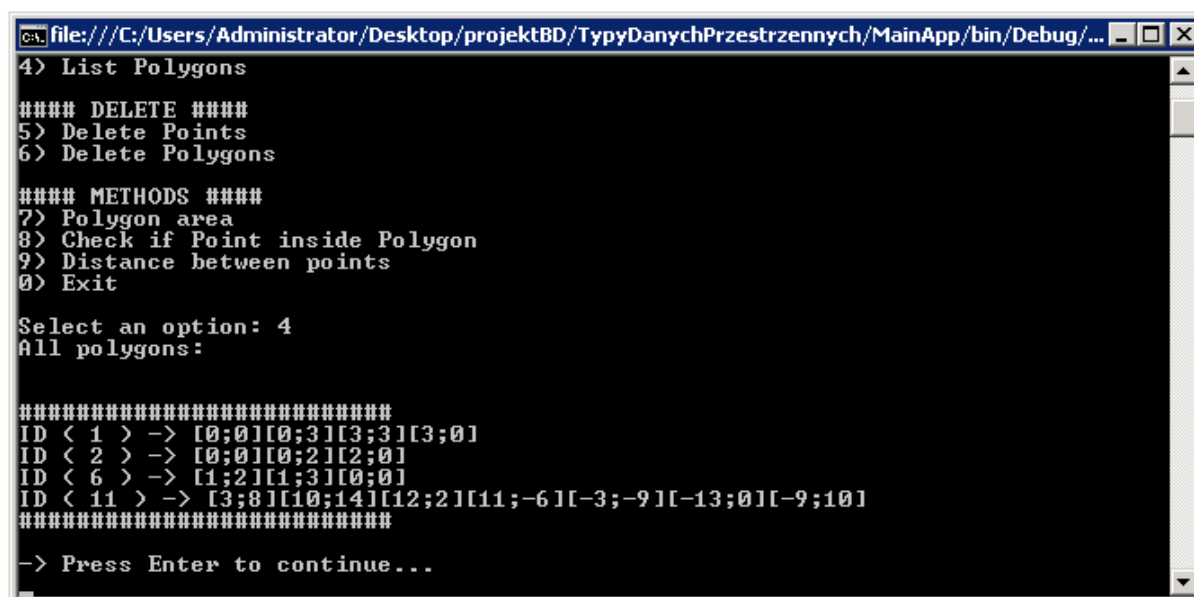
#### DELETE ####
5> Delete Points
6> Delete Polygons

#### METHODS ####
7> Polygon area
8> Check if Point inside Polygon
9> Distance between points
0> Exit

Select an option: 1
-> Enter coordinates in format: x;y [e.g. 1;2]
_
```

Rysunek 2 Menu wprowadzania punktu/wielokąta.

Wybierając opcje 3 lub 4 zostanie wyświetlona lista dostępnych punktów/wielokątów w bazie danych.

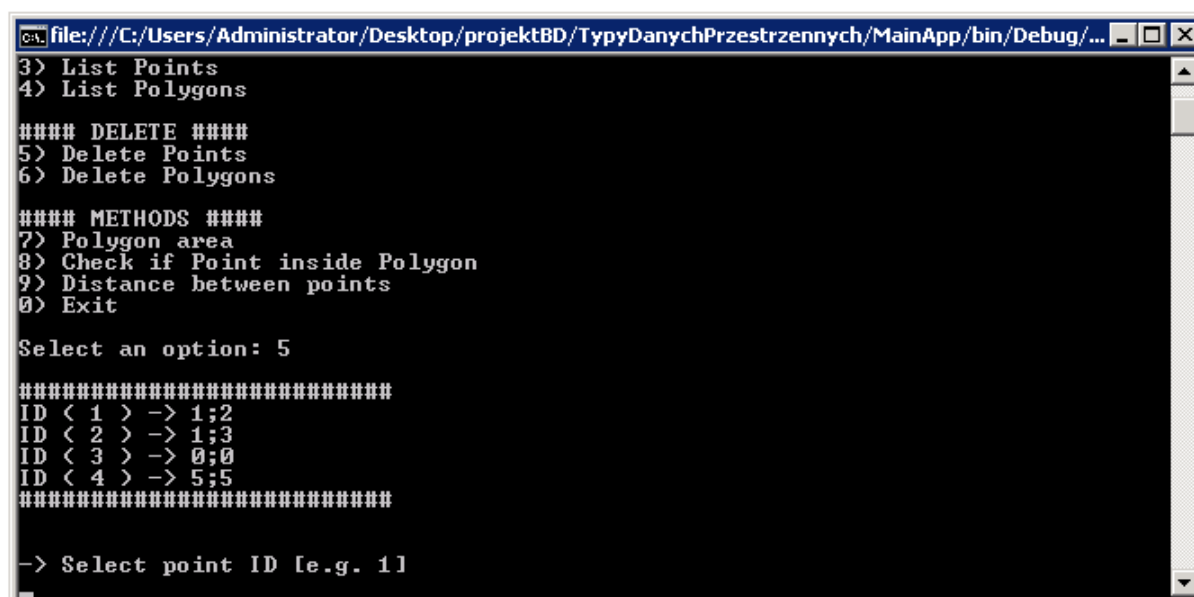


```
file:///C:/Users/Administrator/Desktop/projektBD/TypyDanychPrzestrzennych/MainApp/bin/Debug/...
4) List Polygons
#### DELETE ####
5) Delete Points
6) Delete Polygons
#### METHODS ####
7) Polygon area
8) Check if Point inside Polygon
9) Distance between points
0) Exit
Select an option: 4
All polygons:

#####
ID < 1 > -> [0;0][0;3][3;3][3;0]
ID < 2 > -> [0;0][0;2][2;0]
ID < 6 > -> [1;2][1;3][0;0]
ID < 11 > -> [3;8][10;14][12;2][11;-6][1;-3][9;-13][0;-9][10;-9]
#####
-> Press Enter to continue...
```

Rysunek 3 Lista dostępnych punktów/wielokątów.

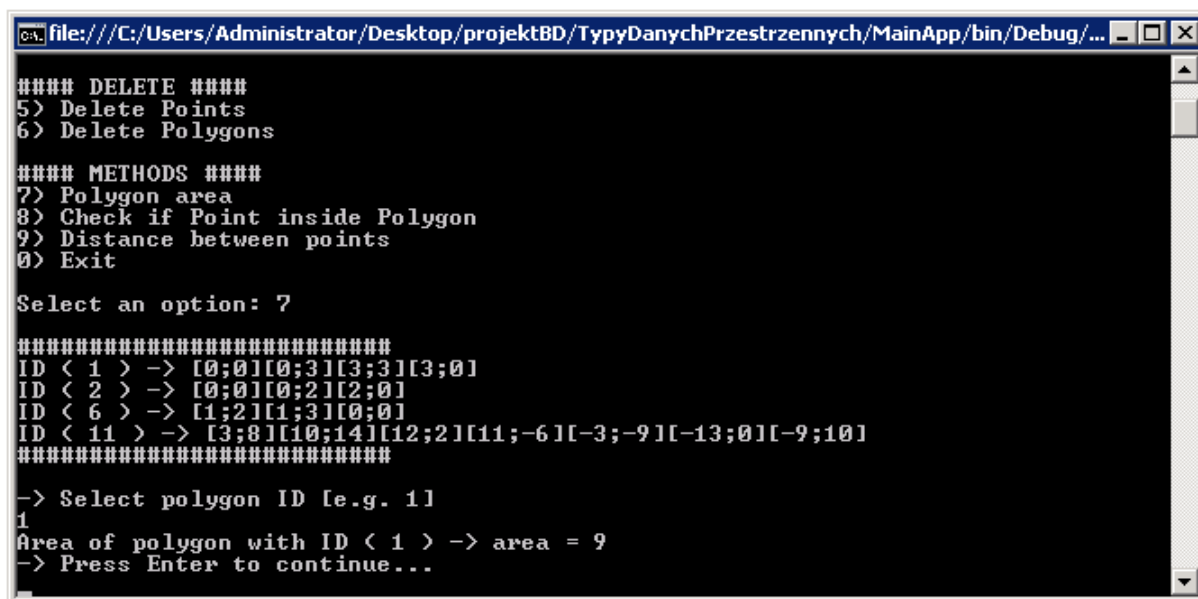
Wybierając opcje 5 lub 6 zostaje wyświetlona aktualna lista elementów z bazy, które możemy usunąć podając ID rekordu.



```
file:///C:/Users/Administrator/Desktop/projektBD/TypyDanychPrzestrzennych/MainApp/bin/Debug/...
3) List Points
4) List Polygons
#### DELETE ####
5) Delete Points
6) Delete Polygons
#### METHODS ####
7) Polygon area
8) Check if Point inside Polygon
9) Distance between points
0) Exit
Select an option: 5
#####
ID < 1 > -> 1;2
ID < 2 > -> 1;3
ID < 3 > -> 0;0
ID < 4 > -> 5;5
#####
-> Select point ID [e.g. 1]
```

Rysunek 4 Usuwanie rekordu z bazy danych.

Wybierając opcję 7 zostaje wyświetlona lista dostępnych wielokątów, a po wyborze ID wielokąta zostaje wyświetlona obliczona wartość pola.



```
file:///C:/Users/Administrator/Desktop/projektBD/TypyDanychPrzestrzennych/MainApp/bin/Debug/...
#### DELETE ####
5> Delete Points
6> Delete Polygons

#### METHODS ####
7> Polygon area
8> Check if Point inside Polygon
9> Distance between points
0> Exit

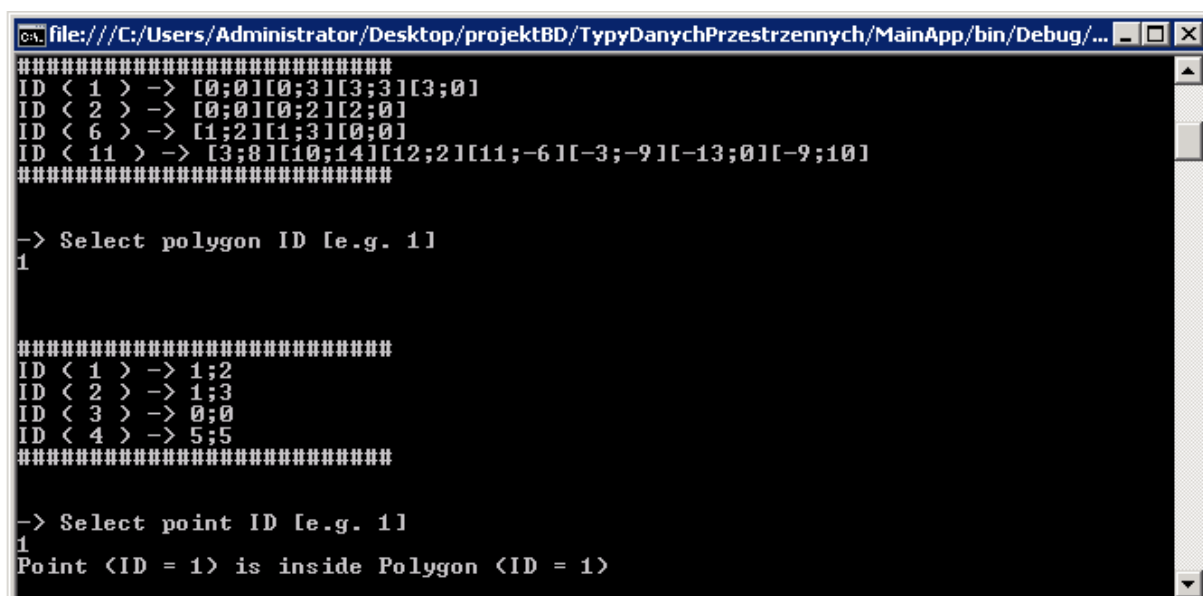
Select an option: 7

#####
ID < 1 > -> [0;0][0;3][3;3][3;0]
ID < 2 > -> [0;0][0;2][2;0]
ID < 6 > -> [1;2][1;3][0;0]
ID < 11 > -> [3;8][10;14][12;2][11;-6][-3;-9][-13;0][-9;10]
#####

-> Select polygon ID [e.g. 1]
1
Area of polygon with ID < 1 > -> area = 9
-> Press Enter to continue...
```

Rysunek 5 Obliczanie pola wielokąta.

Wybierając opcję 8 w pierwszej kolejności zostaje wyświetlona lista dostępnych wielokątów, a po wyborze ID wielokąta, zostaje wyświetlona lista punktów dostępnych do wyboru.



```
file:///C:/Users/Administrator/Desktop/projektBD/TypyDanychPrzestrzennych/MainApp/bin/Debug/...
#####
ID < 1 > -> [0;0][0;3][3;3][3;0]
ID < 2 > -> [0;0][0;2][2;0]
ID < 6 > -> [1;2][1;3][0;0]
ID < 11 > -> [3;8][10;14][12;2][11;-6][-3;-9][-13;0][-9;10]
#####

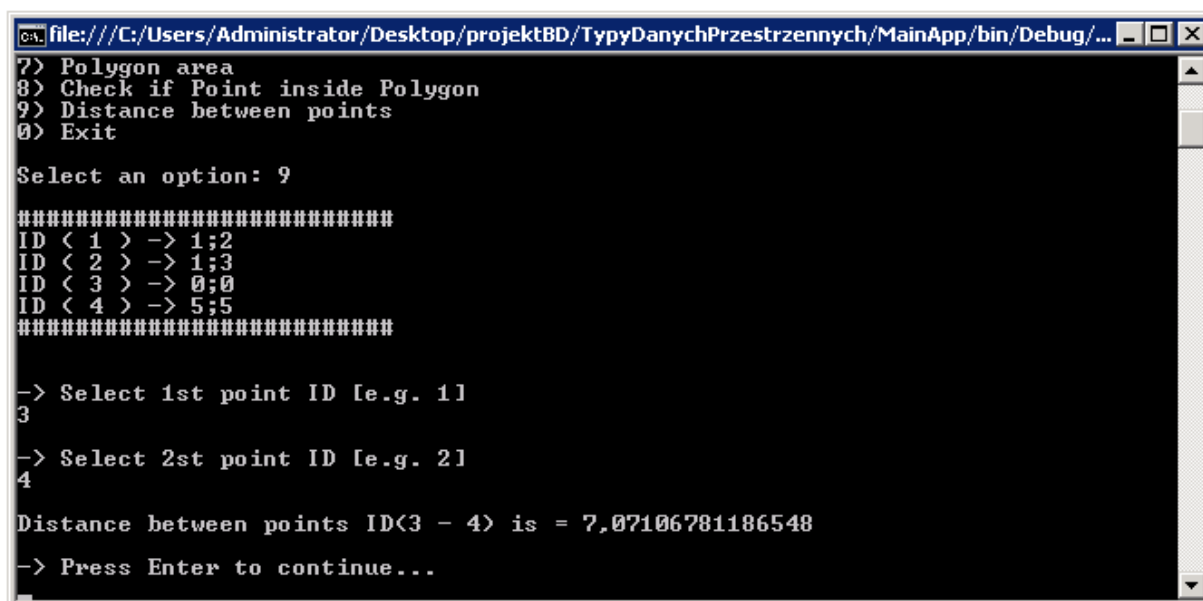
-> Select polygon ID [e.g. 1]
1

#####
ID < 1 > -> 1;2
ID < 2 > -> 1;3
ID < 3 > -> 0;0
ID < 4 > -> 5;5
#####

-> Select point ID [e.g. 1]
1
Point <ID = 1> is inside Polygon <ID = 1>
```

Rysunek 6 Sprawdzanie czy punkt należy do obszaru zadanego przez wielokąt.

Wybierając opcję 9 zostanie wyświetlona lista dostępnych punktów i zostaniemy poproszeni o wybranie dwóch punktów, aby policzyć dystans między nimi.



```
C:\file:///C:/Users/Administrator/Desktop/projektBD/TypyDanychPrzestrzennych/MainApp/bin/Debug/...
7> Polygon area
8> Check if Point inside Polygon
9> Distance between points
0> Exit

Select an option: 9

#####
ID < 1 > -> 1;2
ID < 2 > -> 1;3
ID < 3 > -> 0;0
ID < 4 > -> 5;5
#####

-> Select 1st point ID [e.g. 1]
3
-> Select 2st point ID [e.g. 2]
4
Distance between points ID<3 - 4> is = 7.07106781186548
-> Press Enter to continue...
```

Rysunek 7 Obliczanie dystansu pomiędzy dwoma punktami.

Wybranie opcji 0 zamyka aplikację.

3. Opis typów danych oraz metod (funkcji) udostępnionych w ramach API.

3.1. Klasy UDT.

Każda poniżej wymieniana klasa posiada metody takie jak:

- `public void Read(System.IO.BinaryReader r)` – metoda deserializująca obiekt, dla klasy Point wygenerowana automatycznie, dla Polygon zaimplementowano własną,
- `public void Write(System.IO.BinaryWriter w)` – metoda serializująca obiekt, dla klasy Point wygenerowana automatycznie, dla Polygon zaimplementowano własną,
- `public override string ToString()`,
- `public static Point/Polygon Parse(SqlString s)`
- konstruktory,
- zestaw getterów oraz setterów.

Zdefiniowano dwie klasy reprezentujące odpowiednio punkt oraz wielokąt:

Point – klasa reprezentująca typ złożony – punkt

Pola klasy:

- `private int _x` – współrzędna X punktu,
- `private int _y` – współrzędna Y punktu,
- `private bool isNull` - wartość definiująca czy obiekt jest zainicjalizowany.

Metody klasy:

- `public SqlDouble distance(Point p)` – metoda obliczająca odległość między dwoma punktami.

Polygon – klasa reprezentująca typ złożony – wielokąt

Pola klasy:

- `private List<Point> points` - lista punktów tworzących wielokąt,
- `private bool isNull` - wartość definiująca czy obiekt jest zainicjalizowany.

Metody klasy:

- `private bool CheckPolygon()` – metoda walidująca poprawność wielokąta,
- `private Point AveragePointInside()` – metoda zwracająca punkt referencyjny znajdujący się we wnętrzu wielokąta,
- `private void SortAngular()` – metoda sortuje punkty tworzące wielokąt,
- `public SqlDouble area()` – metoda obliczająca pole wielokąta,
- `public SqlBoolean IsPointInside(Point P)` – metoda sprawdzająca czy punkt należy do wielokąta.

3.2. Klasy aplikacji konsolowej.

Aplikacja konsolowa składa się z jednej klasy - **Program**.

Pola klasy:

- `static String sqlconnection` – zawiera connection string do bazy danych.

Metody klasy:

- `static void Main(string[] args)` – główna metoda klasy,
- `private static bool MainMenu()` – metoda implementująca interfejs konsolowy,
- `private static void getAllPoints()` – metoda wyświetlająca punkty dostępne w bazie danych,
- `private static void getAllPolygons()` – metoda wyświetlająca wielokąty dostępne w bazie,

- `private static void InsertPoint(string value)` – metoda wprowadzająca rekord do tabeli punktów,
- `private static void InsertPolygonManually(string value)` – metoda wprowadzająca rekord do tabeli wielokątów,
- `private static void InsertPolygonFromPointsTable(string value)` – metoda wprowadzająca rekord do tabeli wielokątów,
- `private static void DeletePoint(string pointID)` – metoda usuwająca punkt,
- `private static void DeletePolygon (string polygonID)` – metoda usuwająca wielokąt,
- `private static void getPolygonArea(string id)` – metoda wyświetlająca pole wielokąta,
- `private static void checkIfPointInsidePolygon(string idPolygon, string idPoint)` – metoda wyświetlająca informację o przynależności punktu do wielokąta,
- `private static void distanceBetweenPoints(string firstPoint, string secondPoint)` – metoda wyświetlająca dystans pomiędzy dwoma punktami.
-

4. Opis implementacji.

Wymaganiem projektu było udostępnienie możliwości obliczania pola obszaru, odległości między punktami, sprawdzenia przynależności punktu do obszaru wyznaczonego przez wielokąt.

W celu spełnienia zadanych wymagań zaimplementowano metody, które będą zajmowały się obliczaniem wyników po stronie bazy danych.

a) Odległość pomiędzy dwoma punktami

Skorzystano z odległości euklidesowej między dwoma punktami. Zakładając, że mamy dane punkty: A(x1, y1) oraz B(x2, y2) odległość między A i B możemy obliczyć korzystając z następującego wzoru:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

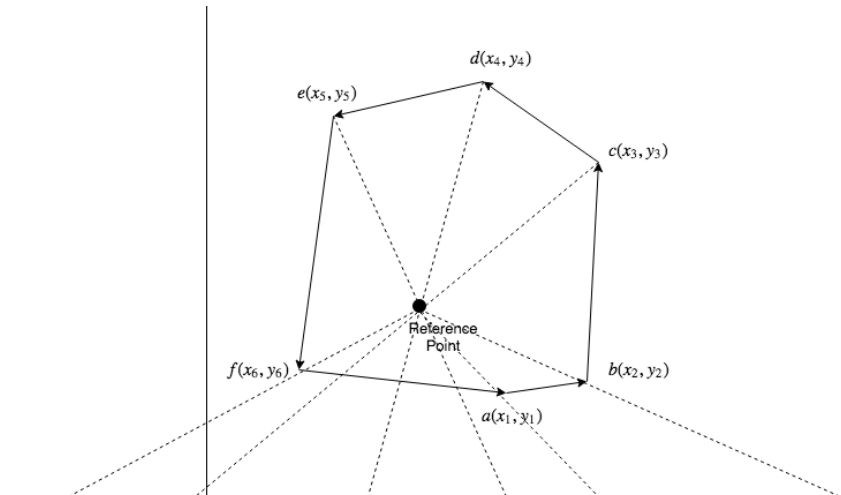
b) Pole obszaru zadanego przez wielokąt.

Jeśli wierzchołki wielokąta są posortowane zgodnie z kierunkiem wskazówek zegara lub przeciwnie to pole wielokąta możemy otrzymać stosując **Algorytm Gaussa** nazywany również **Shoelace Formula**:

$$A = \frac{1}{2} \left| \sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n \right|$$

Dlatego, że wymogiem algorytmu jest to, aby wierzchołki były posortowane zgodnie lub przeciwnie do ruchu wskazówek zegara zaimplementowano sortowanie wierzchołków.

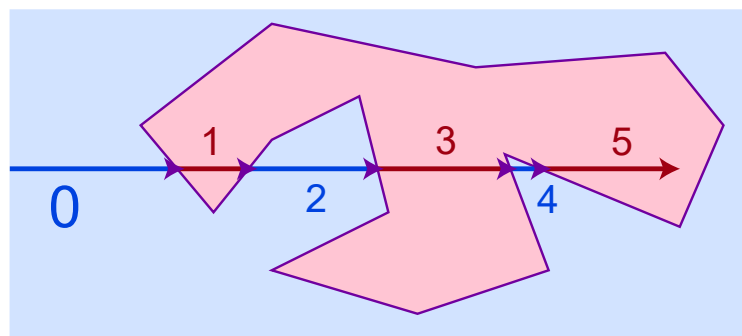
Do sortowania wierzchołków używany jest punkt referencyjny, który znajduje się we wnętrzu wielokąta. Założeniem programu jest przechowywanie wielokątów wypukłych więc współrzędne punktu referencyjnego będą średnią arytmetyczną współrzędnych wierzchołków.



Rysunek 8 Sortowanie wierzchołków wielokąta według punktu referencyjnego.

c) Sprawdzanie czy punkt należy do wnętrza wielokąta.

W celu sprawdzenia czy punkt należy do wnętrza wielokąta skorzystano z algorytmu **Ray Casting**. Algorytm ten sprawdza ilość przecięć pomiędzy promieniem przechodzącym przez punkt, a krawędziami wielokąta. Jeśli punkt znajduje się na zewnątrz wielokąta, promień przetnie jego krawędzie parzystą liczbę razy, a jeśli punkt znajduje się wewnątrz wielokąta – nieparzystą liczbę razy.

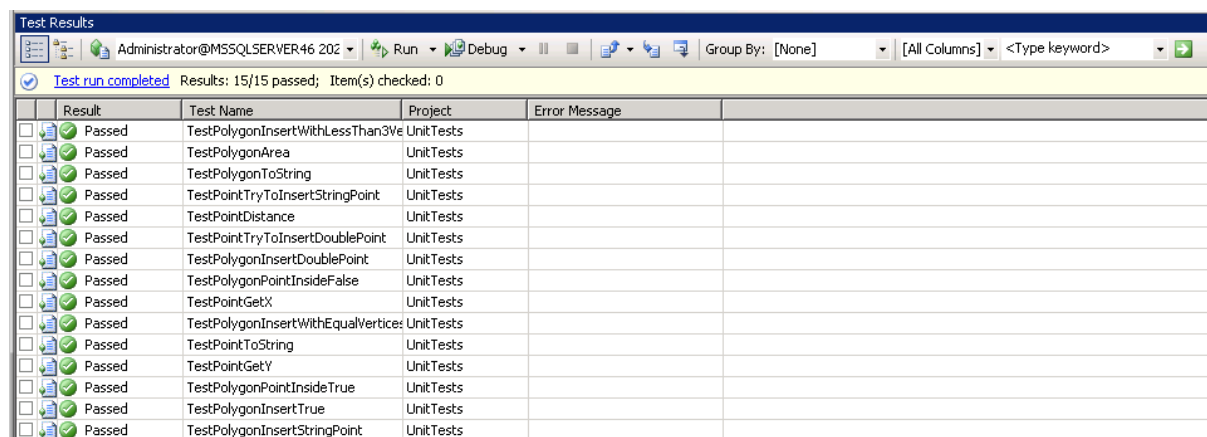


Rysunek 9 Wizualizacja działania algorytmu Ray Casting (crossing number).

5. Prezentacja przeprowadzonych testów jednostkowych.

W celu sprawdzenia poprawnego działania aplikacji zostały przygotowane testy jednostkowe.

Zostało wykonanych 15 testów jednostkowych testujących utworzone typy UDT wraz z ich metodami.



	Result	Test Name	Project	Error Message
<input type="checkbox"/>	Passed	TestPolygonInsertWithLessThan3Ve	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonArea	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonToString	UnitTests	
<input type="checkbox"/>	Passed	TestPointTryToInsertStringPoint	UnitTests	
<input type="checkbox"/>	Passed	TestPointDistance	UnitTests	
<input type="checkbox"/>	Passed	TestPointTryToInsertDoublePoint	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonInsertDoublePoint	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonPointInsideFalse	UnitTests	
<input type="checkbox"/>	Passed	TestPointGetX	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonInsertWithEqualVertices	UnitTests	
<input type="checkbox"/>	Passed	TestPointToString	UnitTests	
<input type="checkbox"/>	Passed	TestPointGetY	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonPointInsideTrue	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonInsertTrue	UnitTests	
<input type="checkbox"/>	Passed	TestPolygonInsertStringPoint	UnitTests	

Rysunek 10 Wykonane testy jednostkowe.

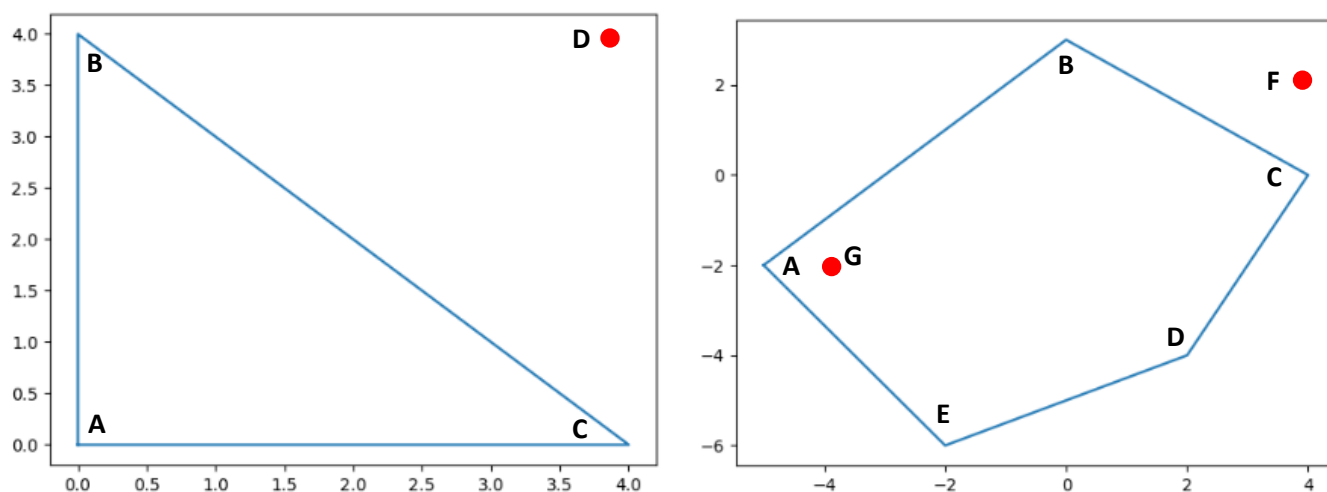
W celu uruchomienia testów należy otworzyć projekt TypyDanychPrzestrzennych i wybrać opcję:

Test -> Run -> All Tests in Solution.

W przypadku aplikacji konsolowej, która wyświetla interfejs użytkownika skupiono się na testach manualnych. Aplikacja ta wywołuje odpowiednie metody typów złożonych UDT, które są obsługiwane przez bazę danych, a testy tych metod zostały wykonane wcześniej (rys. 9).

Dodatkowo wykonano manualne testy polegające na porównaniu obliczonych wartości pola i dystansu pomiędzy punktami z innymi implementacjami algorytmów w środowisku Python.

Utworzono trójkąt o wierzchołkach: A(0,0), B(0,4), C(4,0)
oraz wielokąt o wierzchołkach: A(-5,-2), B(0, 3), C(4,0), D(2, -4), E(-2, -6) obie figury są widoczne na rysunku 10 poniżej.



Rysunek 11 Wykresy utworzonych figur.

Testy trójkąta A(0,0), B(0,4), C(4,0).

Test pola: pole oczekiwane: 8, obliczone: 8.

Test odległości pomiędzy punktami A oraz B: odległość oczekiwana: 4, obliczona: 4.

Test odległości pomiędzy punktami B oraz C: odległość oczekiwana: 5,66, obliczona: 5,66.

Test zawierania się punktu D(4,4) w trójkącie: wartość oczekiwana: False, obliczona: False.

Testy wielokąta A(-5,-2), B(0,3), C(4,0), D(2,-4), E(-2,-6).

Test pola: pole oczekiwane: 44.5, obliczone: 44.5.

Rysunek 11 Wykresy uzyskanych wielokątów.

Test odległości pomiędzy punktami A oraz B: odległość oczekiwana: 7.07, obliczona: 7.07.

Test odległości pomiędzy punktami B oraz E: odległość oczekiwana: 9.22, obliczona: 9.22.

Test zawierania się punktu F(4,2) w wielokącie: wartość oczekiwana: False, obliczona: False.

Test zawierania się punktu G(-4,-2) w wielokącie: wartość oczekiwana: True, obliczona: True.

Wyniki testów manualnych są pozytywne.

6. Podsumowanie i wnioski.

Przy pomocy UDT w łatwy i intuicyjny sposób można tworzyć rozbudowane typy danych. Ciekawą funkcjonalnością typów definiowanych przez użytkownika jest możliwość zdefiniowania metod i możliwość komunikacji z innymi obiektami. Daje to duże możliwości bazodanowe.

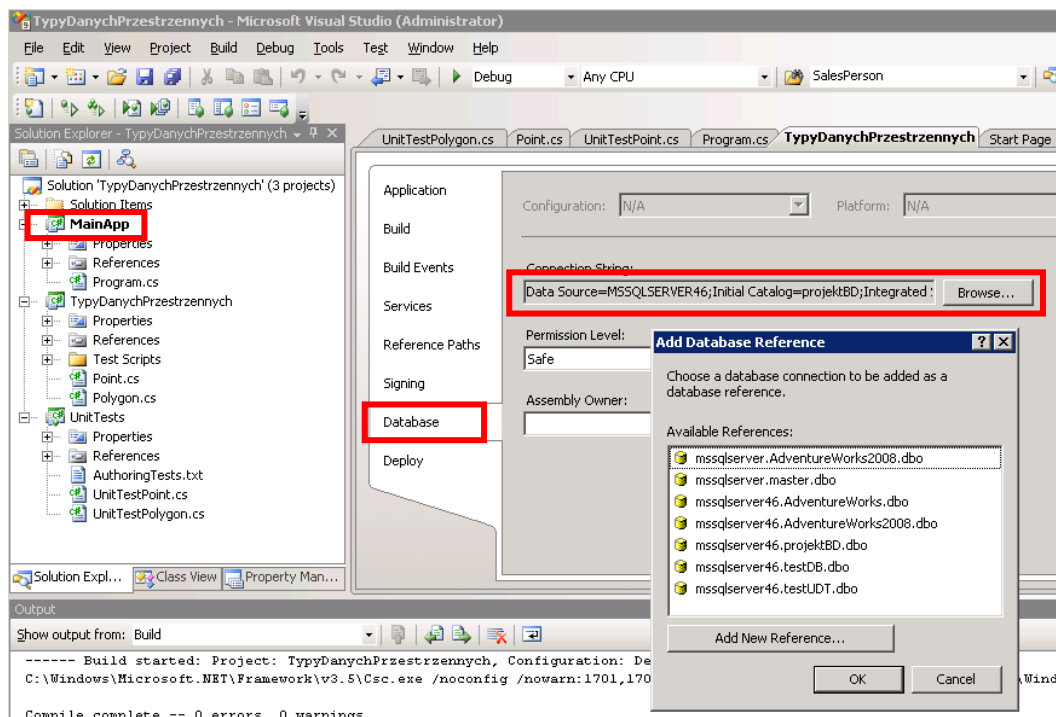
Warto również wspomnieć o tym, że SQL Server udostępnia predefiniowane obiekty przechowywujące dane przestrzenne, niektóre z nich:

- **Point** – reprezentuje punkt na płaszczyźnie, posiada dwa pola, które reprezentują długość oraz szerokość geograficzną.
- **Polygon** – reprezentuje dwuwymiarową powierzchnię zdefiniowaną jako sekwencję punktów, które są wierzchołkami.
Punkty te formują zewnętrzną granicę oraz zero lub więcej wewnętrznych pierścieni. Porównując obiekt z biblioteki SQL Server do tego utworzonego na potrzeby projektu, stworzony obiekt nie posiada możliwości definiowania wewnętrznych pierścieni.

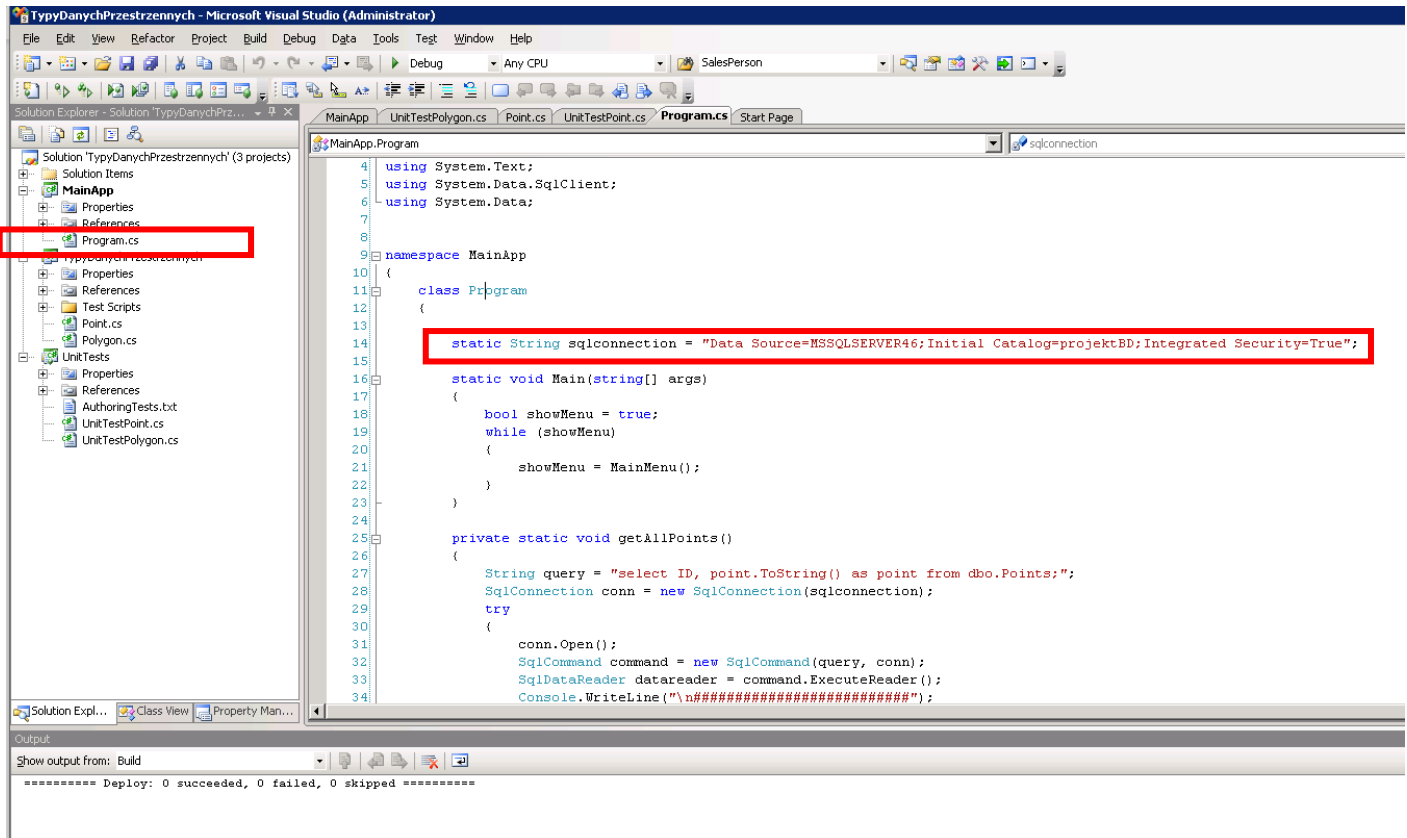
7. Uruchomienie.

Projekt został przygotowany w języku C# przy użyciu Microsoft Visual Studio 2008 (.NET framework 3.5).

1. Utworzenie bazy danych. Należy uruchomić skrypt zamieszczony w pliku SQLScripts/**DBCreate.sql**.
2. Otworzenie projektu Visual Studio. Należy otworzyć plik **TypyDanychPrzestrzennych.sln** z folderu TypyDanychPrzestrzennych.
3. Wybranie bazy do deployu. Klikamy prawym przyciskiem myszy na TypyDanychPrzestrzennych w Solution Explorer, wybieramy Properties i wybieramy odpowiednią bazę danych.



4. Uruchamiamy opcję **Build**, a następnie **Deploy** UDT.
5. Tworzymy tabele, pomocnicze procedury oraz opcjonalnie insert przykładowych wartości. Pliki z folderu SQLScripts w kolejności:
1. TablesCreate.sql, 2. ProceduresCreate.sql, 3. SqlInsert.sql.
6. Przechodzimy do Visual Studio i otwieramy plik **Program.cs** znajdujący się w podprojekcie MainApp. Należy zmienić wartość obiektu **sqlconnection** na odpowiedni connection string do bazy danych.



7. W celu uruchomienia aplikacji konsolowej przechodzimy do **Debug -> Start Without Debugging**.
8. W celu uruchomienia testów należy zmienić wartości obiektów **sqlconnection** na odpowiednie wartości connection string w plikach **UnitTestPoint.cs** oraz **UnitTestPolygon.cs**. Następnie uruchamiamy test wybierając opcję **Test -> Run -> All Tests in Solution**.

8. Literatura.

- https://newton.fis.agh.edu.pl/~antek/index.php?sub=db2_doc
- https://en.wikipedia.org/wiki/Shoelace_formula
- https://en.wikipedia.org/wiki/Point_in_polygon
- <http://www.geomalgorithms.com/code.html>
- <https://algorithmmtutor.com/Computational-Geometry/Area-of-a-polygon-given-a-set-of-points/>
- <https://stackoverflow.com/>