מימוש שטח קמור

מטרתנו בחלק זה הינה לקרוא קלט של נקודות, זאת דרך **stdin** , לחשב את מעטפת הנקודות הקמורה, שזה הקמור שנוצר דרך הנקודות (Convex Hull) , ולהחזיר את שטחה המתקבל.

נזכיר מה זה Convex Hull :

נדמיין שכל הנקודות שלנו הינן מסמרים על לוח עץ, והקמור שלנו הינו הגומי הכי מתוח שיכול לעטוף את כולן מבחוץ, זה הינו הפוליגון הקמור הקטן ביותר שכולל את כל הנקודות.

ראשית, נבנה את הקובץ **main.cpp** , ששם נקלוט את הכל:

הקלט הראשון שלנו הינו מספר הנקודות, ודרך שימוש ב-**cin.ignore()** אנו מסירים את ה-newline מהזיכרון הזמני (buffer), אשר כולל את התו \n :

```
int n; //Number of the points
cout << "Enter the number of points: ";
cin >> n;
cin.ignore() ; //Read and ignore '/n' after the number
```

כעת, נקרא כל נקודה בפורמט של **x,y** , ונשמור אותה בוקטור:

השימוש בפסיק בין ה-x לבין ה-y מתבצע דרך המילה **comma :**

```
vector<pair<float,float>> points;
for(int i=0;i<n;i++){
    float x;
    float y;
    char comma;

    cout << "Point " << i + 1 << ": ";
    if (!(cin >> x >> comma >> y) || comma != ',') {
        cerr << "Error: Invalid input format. Expected format x,y (e.g. 0,1)." << endl;
        return 1;
    }
    points.emplace_back(x,y); //Adding it to the vector of the points

}
```

כעת, נקרא למתודה שמחשבת את שטח המעטפת הקמורה שלנו, והמדפיסה את התוצאה:

```
float area=CHArea(points); //Commpute the area of the Convex Hull
cout << "The area of the Convex Hull is : " << area << endl;
```

כעת, נבנה את הקובץ **ConvexHull.cpp** , שהיא מנתחת את התוכן שלנו לטובת חישוב שטח הקמור.

ראשית, נבנה מתודה ששמה **cross** , אשר מחשבת את המכפלה הוקטורית של הוקטורים CA ו-CB .

כלומר, אם:

AC=A-C

BC=B-C

אזי ← ((A(x)-C(x))*(B(y)-C(y)))-(A(y)-C(y))*(B(x)-C(x)))

המכפלה הזו נותנת לנו ← ערך חיובי אם הזווית בין AC לBC הינה פנייה שמאלה, וערך שלילי אם זו פנייה ימינה.

ערך 0 אם שלושת הנקודות הינן על קו ישר

```
//Computes the cross product of three points A, B, C:
static float cross(const pair<float,float>& A, const pair<float,float>& B,const pair<float,float>& C){
    return (A.first-C.first)*(B.second-C.second)-(A.second-C.second)*(B.first-C.first);
}
```

היא מבצעת את האלגוריתם המבוסס של Graham , ביעילות של O(nlog(n)) .

היא תבצע מיון לנקודות שבתוך הוקטור לפי הסדר הלקסיקוגרפי .

משמע← קודם משווים לפי הערך ה-x של כל נקודה ף ואם שני ערכי ה-x הינם שווים, אזי נשווה לפי ערך ה-y.

למשל← אם pts מכיל את הנקודות (1,1),(2,2),(1,5),(3,1)

אז לאחר :

```
sort(pts.begin(),pts.end()); //first by x, and after by y
```

אז יהיה כך: (3,1) , (2,2) , (1,5) , (1,1)

זה חיוני מאוד לטובת בניית המעטפת בסדר עולה.

כעת, נבנה את 2 המעטפות:

מעטפת תחתונה:

מעבירה את כל הנקודות משמאל לימין, אם יש פנייה ימינה או קו ישר← נמחק את הנקודה הקודמת:

```
for(int i=0;i<n;i++){
    while(k>=2 && cross(hull[k-2],hull[k-1],pts[i])<=0){
        k--;
    }
    hull[k++]=pts[i];
}
```

מעטפת עליונה:

דומה מאוד למעטפת התחתונה, אבל הולך אחורה ומוסיף שוב נקודות .

t שלנו שומר את הגבול בין העליון לתחתון:

```
for (int i = n-2, t = k+1; i >= 0; --i) {
    while (k >= t && cross(hull[k-2], hull[k-1], pts[i]) <= 0)
        k--;
    hull[k++] = pts[i];
}
```

נבצע גם הסרת כפילויות← מסירים את הנקודה הסוגרת שכבר הופיעה בתחילת המעטפת:

```
hull.resize(k-1); //The first snd the last point returns itself=>remove one of them.
```

כעת, נבצע את חישוב שטח הפוליגון , היא מחושבת בסיבוב על הקודקודים, ולבסוף נחלק ב-2:

```
//Compute the area:
float area=0.0;
for(size_t i=0;i<hull.size();i++){
    auto& p1=hull[i];
    auto& p2=hull[(i+1) % hull.size()];
    area=area+(p1.first*p2.second-p2.first*p1.second);
}

//Returnn the absolute value of the area divide 2==>the geometric formula for polygon's area:
return fabs(area)/2.0 ;
```

נראה כעת הרצה :

```
roy3177@LAPTOP-QCUJB7RP:~/Convex Hull server/part1-Area$ ./main
Enter the number of points: 4
Point 1: 0,0
Point 2: 0,1
Point 3: 1,1
Point 4: 2,0
The area of the Convex Hull is : 1.5
roy3177@LAPTOP-QCUJB7RP:~/Convex Hull server/part1-Area$
```