

# XJTLU Entrepreneur College (Taicang) Cover Sheet

|  |   |  |            |
|--|---|--|------------|
| Module code and Title  | <b>DTS102TC Programming with C++</b>                            |  |            |
| School Title   | <b>School of Artificial Intelligence and Advanced Computing</b> |  |            |
| Assignment Title   | <b>Coursework 1 (Assignment)</b>                                |  |            |
| Submission Deadline  | <b>5 pm China time (UTC+8 Beijing) on Fri. 28th. Nov. 2025</b>  |  |            |
| Final Word Count   | <b>NA</b>   |  |            |
| If you agree to let the university use your work anonymously for teaching and learning purposes, please type “yes” here. |   |  | <b>yes</b> |

I certify that I have read and understood the University’s Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
- My work does not contain any fabricated data.

**By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.**

| Scoring – For Tutor Use    |                              |
|----------------------------|------------------------------|
| <b>Student ID: 2468293</b> | <b>Student Name: Aibo Ge</b> |

| Stage of Marking                             | Marker Code            | Learning Outcomes Achieved (F/P/M/D)<br>(please modify as appropriate)               |  |   | Final Score   |
|--|------------------------|--|--|---|---|
|  |                        | A  | B  | C |   |
| 1 <sup>st</sup> Marker – red pen             |                        |  |  |   |   |
| Moderation<br>– green pen                    | <b>IM<br/>Initials</b> | The original mark has been accepted by the moderator (please circle as appropriate): |  |   | Y / N   |
|  |                        | Data entry and score calculation have been checked by another tutor (please circle): |  |   | Y   |
| 2 <sup>nd</sup> Marker if needed – green pen |                        |  |  |   |   |
| <b>For Academic Office Use</b>               |                        | <b>Possible Academic Infringement (please tick as appropriate)</b>                   |  |   |   |
| <b>Date Received</b>                         | <b>Days late</b>       | <b>Late Penalty</b>  | <input type="checkbox"/> <b>Category A</b><br><input type="checkbox"/> <b>Category B</b><br><input type="checkbox"/> <b>Category C</b><br><input type="checkbox"/> <b>Category D</b><br><input type="checkbox"/> <b>Category E</b> |   | Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____ |
|  |                        |  |  |   |   |
|  |                        |  |  |   |   |
|  |                        |  |  |   |   |
|  |                        |  |  |   |   |

# DTS102TC Programming with C++

## Coursework 1 (Assignment)

### Question 1. Algebra: solve quadratic equations (5 marks)

```
1 #include <cmath>
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     // WRITE YOUR CODE HERE, DO NOT CHANGE THE TEMPLATE
7     double a, b, c;
8     double r1, r2, d;
9
10    cout << "Enter a, b, c: ";
11
12    cin >> a >> b >> c;
13
14    d = b * b - 4 * a * c;
15
16    if (d > 0) {
17        r1 = (-b + pow(d, 0.5)) / (2 * a);
18        r2 = (-b - pow(d, 0.5)) / (2 * a);
19        cout << "The roots are " << r1 << " and " << r2 << endl;
20    } else if (d == 0) {
21        r1 = -b / (2 * a);
22        cout << "The root is " << r1 << endl;
23    } else {
24        cout << "The equation has no real roots" << endl;
25    }
26
27    return 0;
28 }
```

```
(base) ~ C:\Source Code Template\git:(main) x cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 01-quadratic.cpp -o 01-quadratic && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"01-quadratic
Enter a, b, c: 1 -3 2
The roots are 2 and 1
(base) ~ C:\Source Code Template\git:(main) x cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 01-quadratic.cpp -o 01-quadratic && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"01-quadratic
Enter a, b, c: 1 -2 1
The root is 1
(base) ~ C:\Source Code Template\git:(main) x cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 01-quadratic.cpp -o 01-quadratic && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"01-quadratic
Enter a, b, c: 1 0 1
The equation has no real roots
```

The core functionality is implemented within the main function, which solves quadratic equations of the form  $ax^2 + bx + c = 0$ . The program first reads the coefficients  $a$ ,  $b$ , and  $c$ . The critical step is the calculation of the discriminant,  $d = b^2 - 4ac$ , which determines the nature of the roots.

The program uses a conditional structure (if-else if-else) to handle three

scenarios:

1. If  $d > 0$ , the equation has two distinct real roots. These are calculated using the quadratic formula  $(-b \pm \sqrt{d}) / 2a$ . The `pow(d, 0.5)` function is used to compute the square root.
2. If  $d = 0$ , the equation has exactly one real root, calculated as  $-b / 2a$ .
3. If  $d < 0$ , the equation has no real roots.

This logic ensures all possible cases for real coefficients are handled correctly.

### Question 2. Geometry: area of a regular polygon (5 marks)

```
1 #include <cmath>
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     // WRITE YOUR CODE HERE, DO NOT CHANGE THE TEMPLATE
7
8     int n;
9     double s, area;
10
11    cout << "Enter the number of sides: ";
12    cin >> n;
13
14    cout << "Enter the length of a side: ";
15    cin >> s;
16
17    const double PI = 3.14159;
18    area = (n * s * s) / (4 * tan(PI / n));
19
20    cout << "The area of the polygon is " << area << endl;
21
22    return 0;
23 }
24 }
```

```
(base) ~ C:\Source Code Template\git:(main) x cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 02-polygon-area.cpp -o 02-polygon-area && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"02-polygon-area
Enter the number of sides: 5
Enter the length of a side: 6.5
The area of the polygon is 72.6903
```

The program calculates the area of a regular polygon given the number of sides ( $n$ ) and the side length ( $s$ ). The mathematical formula used is  $\text{Area} = (n * s^2) / (4 * \tan(\pi / n))$ .

The implementation defines `PI` as a constant double for precision. It utilizes the `tan` function from the `<cmath>` library. The expression  $(n * s * s) / (4 * \tan(\pi / n))$  directly translates the mathematical formula into C++ code. The

result is stored in a double variable to accommodate floating-point values.

### Question 3. Count positive and negative numbers and compute the average of numbers (5 marks)

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // WRITE YOUR CODE HERE, DO NOT CHANGE THE TEMPLATE
6
7     int sum;
8     int countPositive = 0;
9     int countNegative = 0;
10    int totalCount = 0;
11    int num = 0;
12
13    cout << "Enter an integer value, the input ends if it is 0: ";
14    cin >> num;
15
16    while (num != 0) {
17        if (num > 0) {
18            countPositive++;
19        }
20        else {
21            countNegative++;
22        }
23        sum += num;
24        totalCount++;
25        cin >> num;
26    }
27
28    cout << "The number of positives is " << countPositive << endl;
29    cout << "The number of negatives is " << countNegative << endl;
30    cout << "The total value is " << sum << endl;
31    cout << "The average value is " << (totalCount == 0 ? 0.0 : static_cast<double>(sum) / totalCount) << endl;
32
33    return 0;
34 }
```

```
(base) ➤ CW1 Source Code Template git:(main) x cd "/Users/albert/Documents/CW1/CW1 Source Code Template/" && g++ 03-counter.cpp -o 03-counter && ./03-counter
Enter an integer value, the input ends if it is 0: 34 26 52 48 31 48 83 32 0
The number of positives is 8
The number of negatives is 0
The total value is 354
The average value is 44.25
```

The program analyzes a stream of integer inputs ending with 0. It employs a while loop that continues execution as long as the input is not 0 (the sentinel value).

Inside the loop, conditional statements classify each number as positive or negative, incrementing the corresponding counters (countPositive, countNegative). Every non-zero

number is added to a running sum.

After the loop terminates, the average is calculated. A `static_cast<double>(sum)` is used to convert the integer sum to a floating-point number before division, ensuring the average is computed with decimal precision rather than performing integer division.

### Question 4. Binary to decimal (5 marks)

```
1 #include <iostream>
2 #include <string>
3
4 int solve_bin_to_dec(const std::string& binaryString) {
5     // WRITE YOUR CODE HERE, DO NOT CHANGE THE TEMPLATE
6
7     int decimal = 0;
8     int power = 1;
9     for (int i = binaryString.length() - 1; i >= 0; i--) {
10        if (binaryString[i] == '1') {
11            decimal += power;
12        }
13        power *= 2;
14    }
15    return decimal;
16
17 int main() {
18    std::cout << "Enter a binary number: ";
19    std::string binaryString;
20    std::cin >> binaryString;
21    std::cout << solve_bin_to_dec(binaryString) << std::endl;
22
23    return 0;
24 }
```

```
(base) ➤ CW1 Source Code Template git:(main) x cd "/Users/albert/Documents/CW1/CW1 Source Code Template/" && g++ 04-bin2dec.cpp -o 04-bin2dec && ./04-bin2dec
Enter a binary number: 10001
17
(base) ➤ CW1 Source Code Template git:(main) x cd "/Users/albert/Documents/CW1/CW1 Source Code Template/" && g++ 04-bin2dec.cpp -o 04-bin2dec && ./04-bin2dec
Enter a binary number: 1101111101010010000000000
58631168
(base) ➤ CW1 Source Code Template git:(main) x
```

The function `solve_bin_to_dec` converts a binary string representation into a decimal integer. The algorithm iterates through the string from the last character (least significant bit) to the first (most significant bit).

A variable 'power' tracks the place value of the current bit (1, 2, 4, 8, ...). In

each iteration:

1. If the current character is '1', the current 'power' value is added to the total decimal result.
2. The 'power' is multiplied by 2 to prepare for the next bit position.

This approach efficiently converts the binary number in a single pass without the overhead of repeated calls to a power function.

### Question 5. Print distinct numbers (5 marks)

```

1 #include <iostream>
2
3 int main() {
4     // Write your code here. Do not change the template.
5
6     int numbers[10]; // Store distinct values
7     int size = 0; // Indicate how many distinct values are in the array numbers
8
9     std::cout << "Enter ten integers: ";
10
11     int temp;
12     for (int i = 0; i < 10; i++) {
13         std::cin >> temp;
14         bool isDistinct = true;
15         for (int j = 0; j < size; j++) {
16             if (numbers[j] == temp) {
17                 isDistinct = false;
18                 break;
19             }
20         }
21         if (isDistinct) {
22             numbers[size] = temp;
23             size++;
24         }
25     }
26
27     std::cout << "The number of distinct numbers is " << size << std::endl;
28     std::cout << "The distinct numbers are: "; // Note: Sample output shows space after colon! No, sample shows "The distinct numbers are: 1 2 3 4 5"
29     for (int i = 0; i < size; i++) {
30         std::cout << " " << numbers[i];
31     }
32     std::cout << std::endl;
33
34     return 0;
35 }

```

```

(base) ~ % cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 06-distinct-numbers.cpp -std=c++11 -o 06-distinct-numbers
Enter ten integers: 2 7 1 3 0 9 5 9 4 2 6 1 5 2
The number of distinct numbers is 8
The distinct numbers are: 2 7 1 3 0 9 5 4

```

The program reads ten integers and filters out duplicates to display only distinct numbers. It utilizes an array 'numbers' to store the unique values identified so far and a counter 'size' to track the count of these unique values.

For each new input, a nested loop performs a linear search through the 'numbers' array. A boolean flag 'isDistinct' is used to indicate if the

number is already present. If the loop completes without finding the number, it is considered distinct, added to the array, and the size counter is incremented. This ensures that the output contains each number exactly once, preserving their original relative order.

## Question 6. Sum elements in each column (10 marks)

```

1 #include <iostream>
2
3 const int SIZE = 4;
4
5 double sumColumn(const double m[SIZE][SIZE], int rowSize, int columnIndex) {
6     // Write your code here. Do not change the template.
7
8     double sum = 0;
9     for (int i = 0; i < rowSize; i++) {
10         sum += m[i][columnIndex];
11     }
12     return sum;
13 }
14
15 int main() {
16     double matrix[SIZE][SIZE];
17
18     std::cout << "Enter a 3-by-4 matrix row by row " << std::endl;
19
20     for (int i = 0; i < 3; i++) {
21         for (int j = 0; j < SIZE; j++) {
22             std::cin >> matrix[i][j];
23         }
24     }
25
26     for (int j = 0; j < SIZE; j++) {
27         std::cout << "Sum of the elements at column " << j << " is " << sumColumn(matrix, 3, j) << std::endl;
28     }
29
30     return 0;
31 }

```

```

(base) ~ % cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 06-sum-column.cpp -std=c++11 -o 06-sum-column
Enter a 3-by-4 matrix row by row:
72 69 75 53
75 1 13 2
22 50 92 17
Sum of the elements at column 0 is 169
Sum of the elements at column 1 is 120
Sum of the elements at column 2 is 180
Sum of the elements at column 3 is 72

```

The function sumColumn computes the sum of elements in a specific column of a matrix. It accepts a 2D array, the number of rows, and the target column index as parameters.

The logic involves a single for-loop that iterates through the rows (from 0 to rowSize - 1) while keeping the column index fixed. In each iteration, the value at the current row and specified column is added to an accumulator variable 'sum'. This effectively traverses the matrix vertically. The main function demonstrates this by calling sumColumn for each column index (0 to 3) of a 3x4 matrix.

## Question 7. The Rectangle class (10 marks)

```

1  #pragma once
2
3  //==
4  // = Design a class named Rectangle to represent a rectangle. The class contains:
5
6  // = 1. Two double data fields named width and height that specify the width and
7  //   height of the rectangle.
8  // = 2. A no-arg constructor that creates a rectangle with width 1 and height 1.
9  // = 3. A constructor that creates a rectangle with the specified width and
10 //   height.
11 // = 4. The accessor and mutator functions for all the data fields.
12 // = 5. A function named getArea() that returns the area of this rectangle.
13 // = 6. A function named getPerimeter() that returns the perimeter.
14
15 //==
16
17 class Rectangle {
18 public:
19
20     Rectangle();
21     Rectangle(double width, double height);
22     double getWidth() const;
23     void setWidth(double width);
24     double getHeight() const;
25     void setHeight(double height);
26     double getArea() const;
27     double getPerimeter() const;
28
29 private:
30     double width, height;
31 };
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965

```

*double variables, ensuring they are modified only through defined interfaces.*

2. **Constructors:** A no-argument constructor initializes a default rectangle (1x1), while a parameterized constructor allows creating a rectangle with specific dimensions.

3. *Accessors:* Public methods `getWidth` and `getHeight` provide read access to the private data.

4. *Computation:* The `getArea` method returns the product of width and height, and `getPerimeter` returns  $2 * (\text{width} + \text{height})$ . This design adheres to object-oriented principles by bundling data with the methods that operate on it.

**Question 8. Geometry: The Rectangle2D class (15 marks)**

```

1 // Rectangle2D.h
2 #ifndef RECTANGLE2D_H
3 #define RECTANGLE2D_H
4
5 #include <iostream>
6
7 class Rectangle2D {
8 private:
9     double x;
10    double y;
11    double width;
12    double height;
13 public:
14    // Constructor
15    Rectangle2D(double x, double y, double width, double height);
16    // Getters
17    double getX() const;
18    double getY() const;
19    double getWidth() const;
20    double getHeight() const;
21    // Setters
22    void setX(double x);
23    void setY(double y);
24    void setWidth(double width);
25    void setHeight(double height);
26    // Area and Perimeter
27    double area() const;
28    double perimeter() const;
29    // Containment
30    bool contains(double x, double y) const;
31    bool contains(const Rectangle2D &r) const;
32    bool overlaps(const Rectangle2D &r) const;
33};
34
35#endif
36
37// Rectangle2D.cpp
38#include "Rectangle2D.h"
39
40// Constructor
41Rectangle2D::Rectangle2D(double x, double y, double width, double height) {
42    this->x = x;
43    this->y = y;
44    this->width = width;
45    this->height = height;
46}
47
48// Getters
49double Rectangle2D::getX() const { return x; }
50double Rectangle2D::getY() const { return y; }
51double Rectangle2D::getWidth() const { return width; }
52double Rectangle2D::getHeight() const { return height; }
53
54// Setters
55void Rectangle2D::setX(double x) { this->x = x; }
56void Rectangle2D::setY(double y) { this->y = y; }
57void Rectangle2D::setWidth(double width) { this->width = width; }
58void Rectangle2D::setHeight(double height) { this->height = height; }
59
60// Area and Perimeter
61double Rectangle2D::area() const { return width * height; }
62double Rectangle2D::perimeter() const { return 2 * (width + height); }
63
64// Containment
65bool Rectangle2D::contains(double x, double y) const {
66    return x >= this->x && x <= this->x + this->width && y >= this->y && y <= this->y + this->height;
67}
68
69bool Rectangle2D::contains(const Rectangle2D &r) const {
70    return r.x >= this->x && r.x + r.width <= this->x + this->width && r.y >= this->y && r.y + r.height <= this->y + this->height;
71}
72
73bool Rectangle2D::overlaps(const Rectangle2D &r) const {
74    return !this->contains(r) && !r.contains(*this);
75}
76
77// Main
78int main() {
79    Rectangle2D rect(10, 20, 100, 50);
80    Rectangle2D rect2(50, 60, 100, 50);
81    Rectangle2D rect3(10, 60, 100, 50);
82
83    std::cout << "Area: " << rect.area() << "\n";
84    std::cout << "Perimeter: " << rect.perimeter() << "\n";
85    std::cout << "Contains (10, 20): " << rect.contains(10, 20) << "\n";
86    std::cout << "Contains (50, 60): " << rect.contains(50, 60) << "\n";
87    std::cout << "Contains rect2: " << rect.contains(rect2) << "\n";
88    std::cout << "Overlaps rect2: " << rect.overlaps(rect2) << "\n";
89}

```

The `Rectangle2D` class extends the rectangle concept to a 2D coordinate space, adding  $(x, y)$  coordinates for the center.

1. **Point Containment:** The `contains(double x, double y)` method determines if a point lies inside the rectangle. It checks if the point's coordinates fall within the ranges  $[\text{center\_x} - \text{width}/2,$

*center x + width/2] and [center y - height/2, center y + height/2].*

2. **Rectangle Containment:** The `contains(const Rectangle2D &r)` method checks if another rectangle 'r' is entirely within the current rectangle. This requires that all boundaries of 'r' are within the boundaries of the current rectangle.

3. **Overlap Detection:** The `overlaps(const Rectangle2D &r)` method determines if two rectangles intersect. It uses the logic that two rectangles overlap unless they are completely separated (e.g., one is entirely to the left, right, above, or below the other).

## Question 9. Geometry: find the bounding rectangle (10 marks)

```
1 #include "Rectangle2D.h"
2
3 #include <vector>
4 #include <iostream>
5
6 const int SIZE = 2;
7
8 Rectangle2D getRectangleOfPoints(const double points[][SIZE], int n) {
9     if (n == 0) return Rectangle2D(0, 0, 0, 0);
10
11     double minX = points[0][0], maxX = points[0][0];
12     double minY = points[0][1], maxY = points[0][1];
13
14     for (int i = 1; i < n; i++) {
15         if (points[i][0] < minX) minX = points[i][0];
16         if (points[i][0] > maxX) maxX = points[i][0];
17         if (points[i][1] < minY) minY = points[i][1];
18         if (points[i][1] > maxY) maxY = points[i][1];
19     }
20
21     double width = maxX - minX;
22     double height = maxY - minY;
23     double centerX = minX + width / 2;
24     double centerY = minY + height / 2;
25
26     return Rectangle2D(centerX, centerY, width, height);
27 }
28
29 Rectangle2D getRectangleOfPoints(const double points[][SIZE], int n) {
30     if (n == 0) return new Rectangle2D(0, 0, 0, 0);
31
32     double minX = points[0][0], maxX = points[0][0];
33     double minY = points[0][1], maxY = points[0][1];
34
35     for (int i = 1; i < n; i++) {
36         if (points[i][0] < minX) minX = points[i][0];
37         if (points[i][0] > maxX) maxX = points[i][0];
38         if (points[i][1] < minY) minY = points[i][1];
39         if (points[i][1] > maxY) maxY = points[i][1];
40     }
41
42     double width = maxX - minX;
43     double height = maxY - minY;
44     double centerX = minX + width / 2;
45     double centerY = minY + height / 2;
46
47     return new Rectangle2D(centerX, centerY, width, height);
48 }
49
50 int main() {
51     double points[2][2];
52     for (int i = 0; i < 2; i++) {
53         for (int j = 0; j < 2; j++) {
54             points[i][j] = points[i][j];
55         }
56     }
57
58     Rectangle2D boundingRectangle = getRectangleOfPoints(points, 4);
59     cout << "The bounding rectangle's center (" << boundingRectangle.getCenterX() << ", " << boundingRectangle.getCenterY() << ", " << boundingRectangle.getWidth() << ", " << boundingRectangle.getHeight() << endl;
60
61     Rectangle2D boundingRectangle = getRectangleOfPoints(points, 4);
62     cout << "The bounding rectangle's center (" << boundingRectangle.getCenterX() << ", " << boundingRectangle.getCenterY() << ", " << boundingRectangle.getWidth() << ", " << boundingRectangle.getHeight() << endl;
63
64     return 0;
65 }
```

Autograder Results

| Test Summary   |
|--|
| 1) Test with sample run (0.2, 2.5, 3.4, 5.6, 7.8, 9.10): passed (5.0/5.0)                  |
| 2) Test with an additional example (0.4, 1.2, 3.9, 10.7, 8.114, 514): passed (5.0/5.0)     |
| 1) Test with sample run (0.2, 2.5, 3.4, 5.6, 7.8, 9.10, (5/5))                             |
| Enter five points: The bounding rectangle's center (3, 6.25), width 8, height 7.5          |
| The bounding rectangle's center (3, 6.25), width 8, height 7.5                             |
| 2) Test with an additional example (0.4, 1.2, 3.9, 10.7, 8.114, 514, (5/5))                |
| Enter five points: The bounding rectangle's center (57.5, 258.25), width 113, height 513.5 |
| The bounding rectangle's center (57.5, 258.25), width 113, height 513.5                    |

Coursework 1 - Question 9

Student: Alex Chen

Start Time: / 10 pts

Autograder Score: 10/10 (5/5)

Passed Tests: 1) Test with sample run (0.2, 2.5, 3.4, 5.6, 7.8, 9.10, (5/5)) 2) Test with an additional example (0.4, 1.2, 3.9, 10.7, 8.114, 514, (5/5))

to identify the extreme coordinates:

- $\min X$ : the smallest x-coordinate
- $\max X$ : the largest x-coordinate
- $\min Y$ : the smallest y-coordinate
- $\max Y$ : the largest y-coordinate

These four values define the boundaries of the bounding box. The width is calculated as  $(\max X - \min X)$  and the height as  $(\max Y - \min Y)$ . The center of the bounding rectangle is then derived as the midpoint of these ranges:  $(\min X + \text{width} / 2, \min Y + \text{height} / 2)$ . This efficiently determines the smallest axis-aligned rectangle containing all points.

The `getRectangle` function calculates the minimum bounding rectangle that encloses a given set of 2D points.

The algorithm iterates through the entire array of points