Algorithm

1) We sort all points according to x coordinates.
2) Divide all points in two halves.
3) Recursively find the smallest distances in both subarrays.
4) Take the minimum of two smallest distances. Let the minimum be d.
5) Create an array strip[] that stores all points which are at most d distance away from the middle line dividing the two sets.
6) Find the smallest distance in strip[].
7) Return the minimum of d and the smallest distance calculated in above step 6.
The great thing about the above approach is, if the array strip[] is sorted according to y coordinate, then we can find the smallest distance in strip[] in O(n) time.

Formula for calculating the distance

$$\|pq\| = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>


#define MAX_POINTS (4U)


struct Point

{

   int x;

   int y;

};
```

```
struct PointPair
{
    struct Point a;
    struct Point b;
};

double getDistance(const struct PointPair pair)
{
    return sqrt((pair.a.x - pair.b.x) * (pair.a.x - pair.b.x) +
                (pair.a.y - pair.b.y) * (pair.a.y - pair.b.y));
}

void readPoints(struct Point points[const])
{
    for (unsigned i = 0; i < MAX_POINTS; i++)
    {
        printf("Enter coordinate of point %u: ", i);
        scanf("%d %d", &(points[i].x), &(points[i].y));
    }
}

bool checkForShorterDistance(const struct PointPair pair, double *const p_minDistance)
{
    double tempDistance = getDistance(pair);
```

```
    if (tempDistance < *p_minDistance)
    {
        *p_minDistance = tempDistance;
        return true;
    }

    return false;
}

struct PointPair getClosestPair(const struct Point points[const])
{
    struct PointPair result =
    {
        .a = points[0],
        .b = points[1]
    };
    double minDistance = getDistance(result);

    struct PointPair tempPair;

    unsigned i, j;

    for (i = 0; i < MAX_POINTS; i++)
    {
        tempPair.a = points[i];

        for (j = 0; j < MAX_POINTS; j++)
```

```
        {
            if (i == j)
            {
                continue;
            }

            tempPair.b = points[j];

            if (checkForShorterDistance(tempPair, &minDistance))
            {
                result = tempPair;
            }
        }
    }

    return result;
}

int main(void)
{
    struct Point points[MAX_POINTS];

    readPoints(points);

    struct PointPair pair = getClosestPair(points);

    printf("Closest pair is (%d, %d) and (%d, %d)\n",
```

```
            pair.a.x,
            pair.a.y,
            pair.b.x,
            pair.b.y);

    return 0;
}
```