Hikari9's blog

Easy geometry using std::complex

By Hikari9, history, 4 years ago,

It's always a hassle to define our 2D Geometry library during a contest. Is there a way to make our computational geometry lives easier in any way? Fortunately for us, there is, at least in C++, using complex numbers.

Complex numbers are of the form a + bi, where a is the real part and b imaginary. Thus, we can let a be the x-coordinate and b be the y-coordinate. Whelp, complex numbers can be represented as 2D vectors! Therefore, we can use complex numbers to define a point instead of defining the class ourselves. You can look at std::complex reference here.

Defining our point class

We can define our point class by typing `typedef complex<double> point;` at the start of our program. To access our x- and y-coordinates, we can macro the `real()` and `imag()` functions by using `#define`. Of course, don't forget to `#include <complex>` before anything.

```
#include <iostream>
#include <complex>
using namespace std;
```

```
// define x, y as real(), imag()
typedef complex<double> point;
#define x real()
#define y imag()

// sample program
int main() {
        point a = 2;
        point b(3, 7);
        cout << a << ' ' << b << endl; // (2, 0) (3, 7)
        cout << a + b << endl;         // (5, 7)
}
```

Oh goodie! We can use `std:cout` for debugging! We can also add points as vectors without having to define `operator+`. Nifty. And apparently, we can overall add points, subtract points, do scalar multiplication **without defining any operator**. Very nifty indeed.

Example

```
point a(3, 2), b(2, -7);

// vector addition and subtraction
cout << a + b << endl;   // (5,-5)
cout << a - b << endl;   // (1,9)

// scalar multiplication
cout << 3.0 * a << endl; // (9,6)
cout << a / 5.0 << endl; // (0.6,0.4)
```

Functions using std::complex

What else can we do with complex numbers? Well, there's a lot that is really easy to code.

#	User	Contrib.
1	Errichto	193
2	Radewoosh	179
3	tourist	172
4	Vovuh	168
5	antontrygubO_o	167
6	PikMike	166
7	rng_58	160
8	majk	157
9	Um_nik	156
10	300iq	154

View all →

→ Find user

Handle:

Find

→ Recent actions

sysjuruo → Hack Method About 1242C (#599 div.1C)

gen → Codeforces Round #599

thomas_muller → need help to solve this problem

gen → Codeforces Round #599 Editorial

shourov.sust → A Problem on Perfect Matching

PikMike → Educational Codeforces Round 74 Editorial

ko_osaga → [Gym] XX Open Cup. Grand Prix of Korea

1. Vector addition: `a + b`
2. Scalar multiplication: `r * a`
3. Dot product: `(conj(a) * b).x`
4. Cross product: `(conj(a) * b).y`

   notice: *conj(a) * b = (ax*bx + ay*by) + i (ax*by — ay*bx)*

5. Squared distance: `norm(a - b)`
6. Euclidean distance: `abs(a - b)`
7. Angle of elevation: `arg(b - a)`
8. Slope of line (a, b): `tan(arg(b - a))`
9. Polar to cartesian: `polar(r, theta)`
10. Cartesian to polar: `point(abs(p), arg(p))`
11. Rotation about the origin: `a * polar(1.0, theta)`
12. Rotation about pivot p: `(a-p) * polar(1.0, theta) + p`

    UPD: added more useful functions

13. Angle ABC: `abs(remainder(arg(a-b) - arg(c-b), 2.0 * M_PI))`

    remainder normalizes the angle to be between [-PI, PI]. Thus, we can get the positive non-reflex angle by taking its abs value.

14. Project p onto vector v: `v * dot(p, v) / norm(v);`
15. Project p onto line (a, b): `a + (b - a) * dot(p - a, b - a) / norm(b - a)`
16. Reflect p across line (a, b): `a + conj((p - a) / (b - a)) * (b - a)`
17. Intersection of line (a, b) and (p, q):

hmehta → Topcoder SRM 769 — Registration Open

justHusam → [Gym] The 2018 Arabella Collegiate Programming Contest

Flappy_Bird → What're your future plans in the job sector?

-Morass- → Problem Topics

Errichto → Randomized algorithms lecture

Vovuh → Codeforces Round #598 (Div. 3) Editorial

UnstoppableSolveMachine → MEX Foundation Contest #1 (supported by AIM Tech)

Zlobober → Checkers with testlib.h

robertthethirtieth → CHADFORCES > CODEFORCES

fchirica → Codeforces Round #245 — Editorial

Tobby_And_Friends → List of interesting problems solved so far

hmehta → Topcoder SRM 770 — Sponsored by Google

PikMike → Educational Codeforces Round 73 Editorial

NoureIdin → Who is going to Africa and Arab Championship — ACPC 2019 ?

MikeMirzayanov → Rule about third-party code is changing

champion812r → Time limit exceeded

DanAlex → How to sweep like a Sir

riela → Gotta Solve Them All

Detailed →

```
point intersection(point a, point b, point p, point q) {
  double c1 = cross(p - a, b - a), c2 = cross(q - a, b - a);
  return (c1 * q - c2 * p) / (c1 - c2); // undefined if parallel
}
```

Drawbacks

Using std::complex is very advantageous, but it has one disadvantage: you can't use `std::cin` or `scanf` . Also, if we macro x and y, we can't use them as variables. But that's rather minor, don't you think?

EDIT: Credits to **Zlobober** for pointing out that `std::complex` has issues with integral data types. The library will work for simple arithmetic like vector addition and such, but not for `polar` or `abs` . It will compile but there will be some errors in correctness! The tip then is to rely on the library only if you're using floating point data all throughout.

geometry, complex, stl

+202 Hikari9 4 years ago 20

Comments (20)

Write comment?

Zlobober

4 years ago, # | +31

I've used std::complex as a geometry primitive before, but there is a serious disatvantage: by using it, you can't perform operations in integral data types. I always prefer using my own integral vectors when the task allows it (also some tasks simply can't be solved in doubles).

→ Reply

Laakeri

4 years ago, # ^ | ← Rev. 2 0

What you mean by `"you can't perform operations in integral data types"` ? You can use `typedef complex<long long> point;` and every operation which should work with integral data types will work.

→ Reply

4 years ago, # ^ | +39

According to C++ standard, instantiating complex with any data type other than standard float-point types is unspecified. This means that although it may even be compiled, it may work not as you expect it to work.

Some links related to this issue:

Zlobober

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2012/n3337.pdf (section 26.4.2)

http://stackoverflow.com/questions/11108587/why-does-abscomplexint-always-return-zero

http://stackoverflow.com/questions/11108743/why-does-c-mandate-that-complex-only-be-instantiated-for-float-double-or-lon

I believe that using unspecified behaviour that depends much on compiler and platform is a bad idea.

→ Reply

4 years ago, # ^ | ← Rev. 2 0

Laakeri

Thank you, I didn't know that. I have used `complex<long long>` a lot so I guess it should work in expected way with most of the modern compilers (you shouldn't use complex abs with integers anyway). I will continue using `complex<long long>` but this is a good reason to not use it.

→ Reply

Timur_Sitdikov

4 years ago, # | ← Rev. 2 +8

A good article on TopCoder about geometry with complex numbers

→ Reply

Hikari9

4 years ago, # ^ | 0

The intersection part is interesting. Thanks for sharing!

→ Reply

Hikari9

4 years ago, # | 0

*Auto comment: topic has been updated by **Hikari9** (previous revision, new revision, compare).*

→ Reply

Hikari9

4 years ago, # | 0

*Auto comment: topic has been updated by **Hikari9** (previous revision, new revision, compare).*

→ Reply

dj3500

4 years ago, # | 0

Really nice reference list! :)

→ Reply

mogers

4 years ago, # | +6

To use cin, we just need to overload the operator. I don't see a problem with this in competitions:

```
template<class T>
istream& operator>> (istream& is, complex<T>& p) {
  T value;
  is >> value;
  p.real(value);
  is >> value;
  p.imag(value);
  return is;
}
```

```
int main() {
  point a;
  cin >> a;
  cout << a << endl;
  return 0;
}
```
→ Reply

Hikari9

4 years ago, # ^ | ▲ 0 ▼

I think `p.real(value)` and `p.imag(value)` only works for C++11, but yeah, most competitions have C++11 compilers anyway so this is a good suggestion.

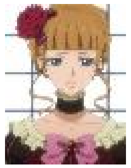
→ Reply

Hikari9

4 years ago, # | ▲ 0 ▼

Just for reference, `polar(r, theta)` is equal to `r * exp(point(0, theta))`, which is cool because it obeys the Euler's identity.

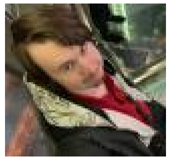
→ Reply


beatoriche

4 years ago, # | ▲ 0 ▼

どもうありがとう

→ Reply


adamant

3 years ago, # | ▲ 0 ▼

1. Angle of elevation

I don't get this one... What is it?

→ Reply

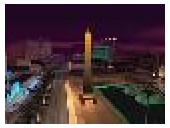
3 years ago, # ^ | ▲ 0 ▼

Slope of line b/w points a and b.

→ Reply

→ Reply

downvoteplz



Alpha_Q

3 years ago, # ^ | +3

The arctan of the slope of the line connecting a and b. In other words, the angle that the line connecting a and b forms with the x-axis in positive direction.

→ Reply

Tim510

3 years ago, # | ← Rev. 2 +8

Suggestion: Instead of using `#define x real()` and `#define y imag()` it's easier to use `#define x() real()` and `#define y() imag()` so that you can use `x()` and `y()` as getter methods without having to remember to always avoid using the variables `x` and `y` in other contexts, especially if you include this code in a template.

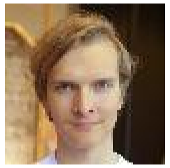
→ Reply

Xellos

3 years ago, # | +29

The problem with geometry isn't in such technical details, it's the thousands of special cases you have to watch out for :D.

→ Reply



yeputons

22 months ago, # ^ | 0

Well, you just have to get rid of special cases with a good general approach :)

→ Reply

sameer_hack

22 months ago, # | 0

Is there any similar type of class lies in Java for complex number or geometry?

→ Reply