

# Vecta: описание за потребител

**Vecta** е програмна библиотека, с помощта на която могат да се представят вектори и точки в равнината и в пространството и да се извършват основни пресмятания с тях в програми на езика C++. Библиотеката е параметризирана чрез типа на координатите, използвани за задаване на векторите. По този начин може да се работи с целочислени (`int`, `long`, ...), реалночислени (`float`, `double`) или от друг тип вектори, каквито се окажат нужни. В дадено пресмятане могат да участват вектори от различни типове.

Библиотеката се състои от три класа и известен брой функции, определени като шаблони (templates). Всички те са поместени в самостоятелно пространство от имена (namespace), наречено `vecta`. Повечето функции са реализирани като операции.

За да се използва библиотеката в дадена програма, файлът `vecta.h` трябва да се цитира в команда `#include` на предпроцесора.

Следва описание на съдържанието на библиотеката.

## Векторна аритметика в равнината

`vec2d<тип>`

Клас за представяне на типа *вектор* или *точка* в равнината. С оглед на естественото еднозначно съответствие между двата вида геометрични обекти, а и за простота на програмирането, между тях не се прави разлика.

Типът-параметър трябва да бъде числов и се отнася до координатите на вектор. Може да не се посочва, подразбира се `double`.

Класть `vec2d` разполага с два конструктора. Единият създава вектор по посочена двойка декартови координати, а другият – от друг вектор, възможно от различен координатен тип.

Стойност-вектор може да се създава и чрез присвояване, при което се създава копие на даден вектор, също както при втория вид конструктор. И в този случай присвояваната стойност може да бъде от различен тип – тогава се прави привеждане, което на свой ред може да води до загуба на точност (напр. при привеждане на `double` в `int`) и затова се сигнализира от компилатора.

Примери:

```
vec2d<int> p1(2,3); // p1 е целочислен, с координати 2 и 3  
vec2d<> p2(p1); // p2 е от вид double, с координати като на p1  
vec2d<> p3 = p1; // ... също и p3
```

`vec2d<double> polar(дължина, посока)` – функция

Вектор по зададени полярни координати

`double len(вектор)`

`double dir(вектор)` – функции

Дължина и посока (в радиани) на вектор

`число norm(вектор)` – функция

Норма (дължина на квадрат) на вектор – число от същия тип като координатите на вектора

`vec2d<double> unit(вектор)` – функция

Единичен вектор с посоката на даден ненулев вектор

- *вектор* – операция

Противоположен, равен по големина на даден вектор

**! вектор** – операция

„Комплексно спрегнат“ вектор: симетричен на дадения относно хоризонталата  
**вектор + вектор**

**вектор - вектор** – операции

Сбор и разлика на вектори

**~ вектор** – операция

„Перп“ на вектор: вектор със същата като на дадения дължина, завъртян спрямо  
него в положителна посока на прав ъгъл

**число \* вектор**

**вектор \* число** – операции

Произведение на вектор с число

**вектор / число** – операция

Произведение на вектор с реципрочното на дадено число

**вектор += вектор**

**вектор -= вектор** – операции

Прибавяне и изваждане на вектор към/от даден вектор

**вектор \*= число**

**вектор /= число** – операции

Умножаване на вектор с число или с реципрочното му

**вектор ^ вектор**

**вектор \* вектор** – операции

Лицеvo и скаларно произведения на вектори

**вектор < вектор**

**вектор <= вектор**

**вектор >= вектор**

**вектор > вектор** – операции

Предшестване между вектори:  $u < v$ , ако посоката на  $v$  е наляво от тази на  $u$ . За  
 $u > v$  – надясно. При  $<=$  и  $>=$  сравнението е вярно и ако посоките съвпадат. Всяко  
от сравненията е вярно, ако някой от векторите е нулев.

**вектор || вектор** – операция

Проверка за успоредност между вектори. (Успоредност е налице и ако поне единият  
вектор е нулев)

**вектор == вектор**

**вектор != вектор** – операции

Проверка за равенство и неравенство между вектори

**angle(вектор, вектор)** – функция

Ориентиран – измерен в положителна посока – ъгъл от първия вектор към  
втория  $((-\pi, \pi])$

**вектор & згбл**

**вектор &= згбл** – операции

Завъртане на вектор на даден ъгъл около началото на координатната система.

При  $\&=$  се променя самият вектор

**вектор & вектор**

**вектор &= вектор** – операции

„Комплексно произведение“ на вектори – резултатът е вектор, съответен на произведението на комплексните числа, съответни на дадените вектори. Може да се използва за завъртане на единия вектор с големина ъгъла на посоката на другия, и по-общо – за хомотетично въртене. При `&=` се променя левият аргумент

`вектор / вектор`

`вектор /= вектор` – операции

„Комплексно частно“ на вектори – резултатът е вектор, съответен на частното на комплексните числа, съответни на дадените вектори. Може да се тълкува като обратна на `&` (`&=`) операция по два начина: намиране на въртенето или хомотетичното въртене, което довежда десния аргумент в левия или прилагане към левия аргумент на въртене или хомотетично въртене, обратно на зададеното чрез комплексното число, съответно на десния аргумент. При `/=` се променя левият аргумент

## Векторна аритметика в пространството

`vec3d<тип>`

Клас за представяне на типа `вектор` или `точка` в пространството. Използва се подобно на `vec2d` за равнинни точки и вектори

`double len(вектор)` – функция

Дължина на вектор

`число norm(вектор)` – функция

Норма (дължина на квадрат) на вектор – число от същия тип като координатите на вектора

`vec3d<double> unit(вектор)` – функция

Единичен вектор с посоката на даден ненулев вектор

`- вектор` – операция

Противоположен, равен по големина на даден вектор

`вектор + вектор`

`вектор - вектор` – операции

Сбор и разлика на вектори

`число * вектор`

`вектор * число` – операции

Произведение на вектор с число

`вектор / число` – операция

Произведение на вектор с реципрочното на дадено число

`вектор += вектор`

`вектор -= вектор` – операции

Прибавяне и изваждане на вектор към/от даден вектор

`вектор *= число`

`вектор /= число` – операции

Умножаване на вектор с число или с реципрочното му

`вектор ^ вектор`

`вектор ^= вектор` – операция

Векторно произведение на вектори. При `^=` се променя левият аргумент

`вектор * вектор` – операция

Скаларно произведение на вектори

*~ вектор* – операция

Двойка (pair) вектори  $\mathbf{v}$  и  $\mathbf{w}$ , перпендикулярни на дадения вектор  $\mathbf{u} \neq \mathbf{0}$  и помежду си, при което е изпълнено  $(\mathbf{u} \times \mathbf{v}) \cdot \mathbf{w} > 0$ . По-точно,  $\mathbf{v} = \mathbf{e} \times \mathbf{u}$ , където  $\mathbf{e}$  е единичен вектор с посоката на някоя от координатните оси, а  $\mathbf{w} = \mathbf{u} \times \mathbf{v}$

*вектор || вектор* – операция

Проверка за успоредност между вектори. (Успоредност е налице и ако поне единият вектор е нулев)

*вектор == вектор*

*вектор != вектор* – операции

Проверка за равенство и неравенство между вектори

*angle(вектор, вектор)* – функция

Големината на ъгъла от първия вектор към втория  $([0, \pi])$

*вектор & двойка*

*вектор &= двойка* – операции

Завъртане на точка (вектор) около ос през началото на координатната система. Въртенето се задава чрез двойка (pair) от число и вектор: посоката на вектора определя тази на оста, а числото – големината на ъгъла на въртене. Ъгълът се отмерва в радиани обратно на часовника, гледано срещу посоката на оста. При **&=** се променя левият аргумент

*вектор & кватернион*

*вектор &= кватернион* – операции

Завъртане на точка (вектор), зададено чрез кватернион. Векторът  $\mathbf{t}$  ( $|\mathbf{t}|=1$ ) и числото  $\varphi$  в представянето  $\mathbf{q} = \cos \frac{\varphi}{2} + \sin \frac{\varphi}{2} \mathbf{t}$  на кватерниона ( $q$  е с единична дължина) дават съответно посоката на оста и големината и посоката на ъгъла на въртене. При **&=** се променя самият вектор

*вектор / вектор* – операция

Двойка (pair) от число – коефициентът на разтягане, с който дълчината на десния аргумент достига тази на левия – и единичен кватернион, представляящ въртенето (както при **&**), което довежда десния аргумент в левия.

*quatrn< mun >*

Клас за представяне на типа *кватернион*. В настоящата библиотека кватернионната аритметика е представена частично – само дотолкова, доколкото тези алгебрични обекти имат помощна роля за реализиране на пространствените ротации. (Кватернионите се използват включително и в тези случаи, в които оста и ъгълът на въртене се дават чрез вектор и число.)

Кватернионите от класа *quatrn* имат реалночислови (*double*) компоненти и биват създавани чрез конструктори – по компоненти, от друг кватернион или от двойка от число и пространствен вектор, както и чрез присвояване на стойност от тип *quatrn*.

При образуването на кватернион от число и вектор последният задава ос на пространствено въртене, числото – ъгъл на въртене, а образуваният кватернион представля същото въртене. За извършване на въртене се посочват също кватернион или двойка от число и вектор

*кватернион \* кватернион*

*кватернион \*= кватернион* – операции

Кватернионно произведение. При **\*=** се променя левият аргумент.

*Забележка.* В алгебрата на кватернионите умножаването обикновено се записва с обратен ред на аргументите! Тук, когато двата кватерниона представлят ротации, тяхното произведение представя композиция на тези ротации, в която първа е съответната на *първия* (а не втория) аргумент.

## Разни

Библиотеката **vecta** има за цел да направи програмирането на векторни пресмятания в равнината и пространството просто и удобно. Изразите с участие на вектори се записват почти така, както в математиката. Предоставят се само основни, най-необходими действия с вектори, повечето във вид на операции – инфиксни или префиксни.

Придържането към простота и удобство в случая изисква невключване на средства за осигуряване на надеждност по отношение на числените пресмятания – ако последното е нужно, за него програмистът трябва да се погрижи изрично. Например сравняването за равенство или успоредност на реалночислови вектори може да изисква използване на допуск за точност („ $\varepsilon$ -точност“) вместо да се прави непосредствено с `==` или `||`. Препълване, загуба на значещи цифри или на точност на резултат са други потенциални проблеми на пресмятанията, които във **vecta** не се решават автоматично.

Бойко Банчев