

Типичные ошибки и то, что обычно забывается

1. В Visual Studio навигацию по программе лучше осуществлять с помощью вкладки Class View.
2. Если в программе используется наследование, то вместо доступа (по умолчанию) `private` используйте `protected`
3. Параметр в конструкторах копирования – `const Type&`
4. В конструкторах лучше элементарные инициализации делать до `{}`: `prob(p)` и т.д.
`random_search(const vect& v, ... ): p(prob),...`
`{...}`
Но если инициализацию надо проверить на правильность (а ее нужно проверять), то тогда делайте это в фигурных скобках. Например, нужно проверить вероятность на принадлежность $[0,1]$. Обязательно везде нужно проверять размерности на соответствие: чтобы не получилось, что точка одной размерности, функция другой, а область поиска - третьей. При нарушении условий это нужно либо автоматически исправлять (например, возвращать `prob` в отрезок $[0,1]$), либо при критической ситуации вроде несоответствия размерностей выбрасывать исключение.
5. Передавайте структуры/классы в функции по ссылке. Если это входной параметр, то ставьте `const`. Это касается и параметров `catch` в исключениях.
6. Также, нужно ставить `const` в прототипе тех методов класса, которые не меняют данных класса (делать константные функции).
7. Математические функции не должны ничего печатать и пр. Взаимодействие с внешним миром должно быть отделено от всего остального.
8. Не забывайте о ключевом слове `inline` для коротких функций – это сильно убыстряет их использование.
9. Не забывайте о ключевом слове `explicit` для конструкторов классов, у которых может быть один параметр и которые не годятся для приведения типа по умолчанию. Отсутствие `explicit` означает, что этот конструктор может быть неявно использован для приведения типов.
10. Не забывайте в классах типа Вектор (если он ваш собственный) переопределять операцию `[]` (возврат значения по ссылке), а в классе типа Функция переопределить операцию `()`.
11. При заполнении `vector` из STL, если вы знаете заранее его размерность, то быстрее работает заполнение вектора через `reserve` и `push_back`, чем через `resize` и обращение по индексу через `[]`.
12. Явно выписанных констант вроде 0.001 или 1000 внутри функций быть не должно. Вводите переменные-константы. Лучше константы локализовать в классах.
13. Если вы видите, что кусок кода у вас повторяется (или даже почти повторяется), то, значит, что-то не так. Например, часто повторяющийся кусок можно вынести из `if`. Если повтор идет в наследниках, то его можно вынести в базовый класс.
14. Не забывайте про часто возникающую при наследовании необходимость виртуальных деструкторов.
15. Все элементы класса должны быть инициализированы во всех конструкторах. В частности, указатели инициализируются нулем.
16. Если в классе есть указатель/ссылка, под который динамически отводится память, то необходимо реализовать конструктор копирования, деструктор и переопределить оператор `=`.
17. При переопределении конструктора копирования (аналогично для оператора присваивания) не забывайте о вызове конструктора копирования для базового класса.
18. Не забывайте в файлах `.h` писать `#pragma once` (или использовать конструкцию `#ifndef ... #define ...`)