Комментарии к коду и синтаксису

1. Пробел после запятой

```
dsp::instfreq(dst,sig,prev_sample);
dsp::instfreq(dst, sig, prev_sample);
```

2. Ссылка прикрепляется к типу данных

```
std::vector<float> demod_fm(const std::vector<pmath::IQF> & sig);
std::vector<float> demod_fm(const std::vector<pmath::IQF>& sig);
```

3. Блок выделяется скобками даже если в блоке одна строчка кода

```
if (src.size() == 0) return;
if (src.size() == 0) {return;}
```

4. Использовать const/constexpr выражения там, где возможно

```
size_t nsamp = 5;
const size_t nsamp = 5;
```

5. Использование тернарного оператора в случае крайней необходимости и если выражение предельно простое.

6. Не использовать комбинированные неочевидные приведения типов.

```
unsigned overlap = 0;
...
overlap += (unsigned) (i >= size ? !pmath::compare(dst_full[i], dst2[i - size]) :
!pmath::compare(dst_full[i], dst1[i]));
...
```

```
...
if (i >= size) {
      if (!pmath::compare(dst_full[i], dst2[i - size])) {
            ++overlap;
      }
} else {
      if (!pmath::compare(dst_full[i], dst1[i])) {
            ++overlap;
      }
}
```

...

7. Использовать форматировние проекта в своем коде, чтобы все в проекте было в одном стиле:

7.1 Фигурные скобки (выделение блока)

```
if (i >= size)
{
      ...
}

if (i >= size){
      ...
}
```

7.2 Передача параметров в функцию

Как при присваивании (memcpy like):

```
void demod_fm(std::vector<float>& dst, const std::vector<pmath::IQF>& src,
pmath::IQF& prev_sample);
```

Обратное:

```
void demod_fm(cobst std::vector<pmath::IQF>& src,std::vector<float>& dst,
pmath::IQF& prev_sample);
```

7.3 Не использовать указатель на массив и вектор, например, при передаче в функцию
7.4 Использовать стиль имен перменных/функций как в проекте.

8. Все преобразования типов следует делать явными

```
unsigned M = 8;
...
float angle = src[i] * M_PIf * 2.0f / M;
float angle = src[i] * M_PIf * 2.0f / (float) M;
```

9. Перенос строки даже для простых операций:

```
if(src[i] > val) { ++c; }

if(src[i] > val) {
      ++c;
}
```

10. Неочевидная логика работы указывается в комментариях к функции

11. Создаваемые переменные должны соответствовать типу при присваивании:

```
std::vector<float> vec;

...
int size = vec.size();

...
size_t size = vec.size();
```

12. Включение заголовков напрямую, а не через другие заголовочные файлы.

13. Коммиты нужно разделять по логике исправлений (основной критерий не количество кода, а логика изменений).

14. Отбивать одну строку в конце файла.

15. Всегда указывать тип вещественной константы.

```
float x = 5.0;
float x = 5.0f;
```

16. Слово const, если оно относится к типу, ставить в начало.

```
void fun(float const& val);
void fun(const float& val);
```

17. Не использовать слишком длинные имена переменных и функций, где в контексте и так понятно, зачем нужна переменная.

Например, при передаче в функцию demod_am нет смысла передавать исходные вектор с именем demod_am_src, достаточно src.

18. Использовать строки инициализации в конструкторе.

19. Не исползовать инклюды вида "../"

20. Считывать из файла одним блоком, а не по одному байту за вызов.

21. Использовать пробелы после if, for, перед фигурными скобками:

```
for(size_t  i = 0; i != vec.size(); ++i){
        int a = 0;
        if (vec[i] == 0){
                ++a;
        }
}
```

```
for (size_t  i = 0; i != vec.size(); ++i) {
        int a = 0;
        if (vec[i] == 0) {
                ++a;
        }
}
```