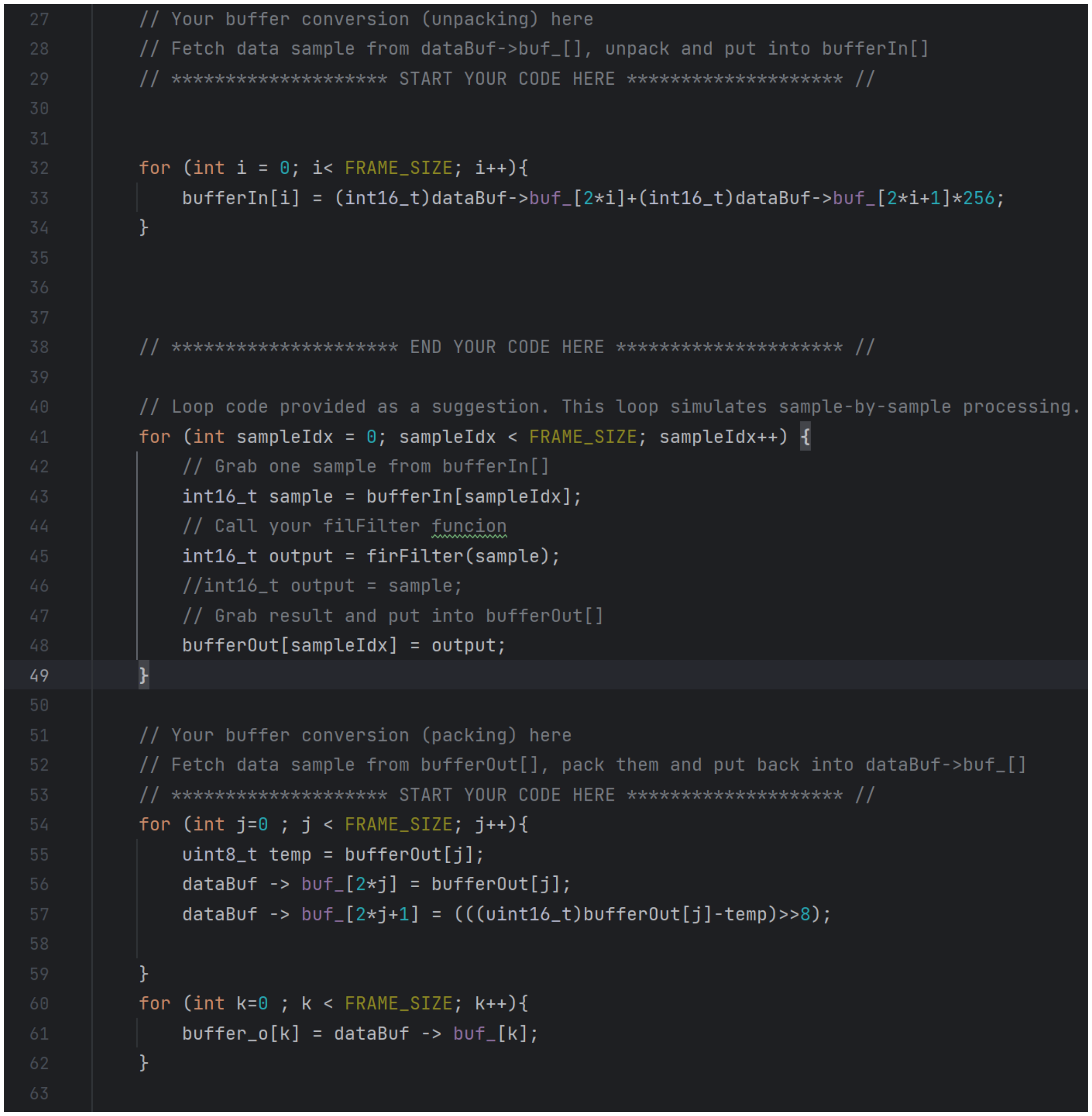
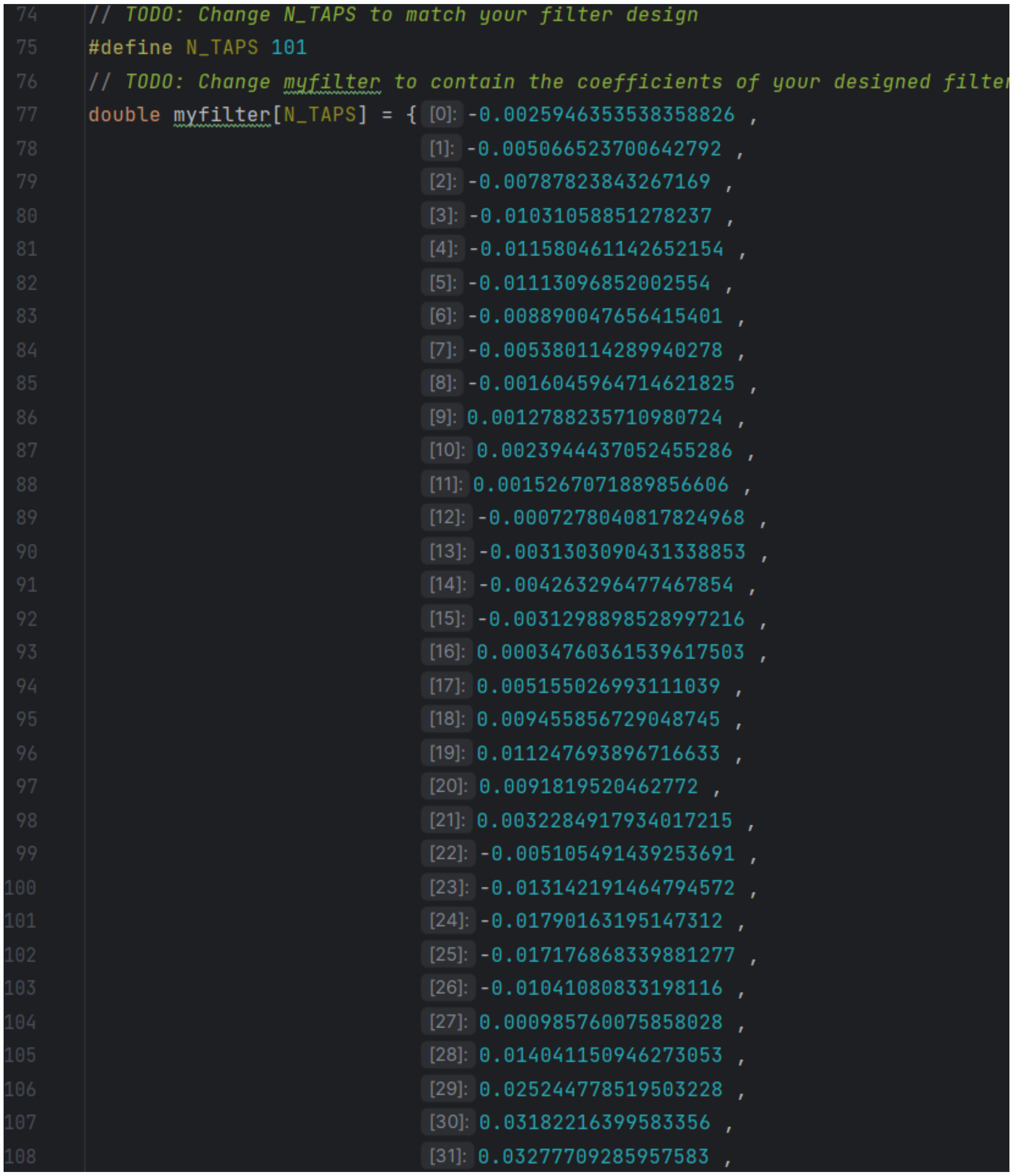
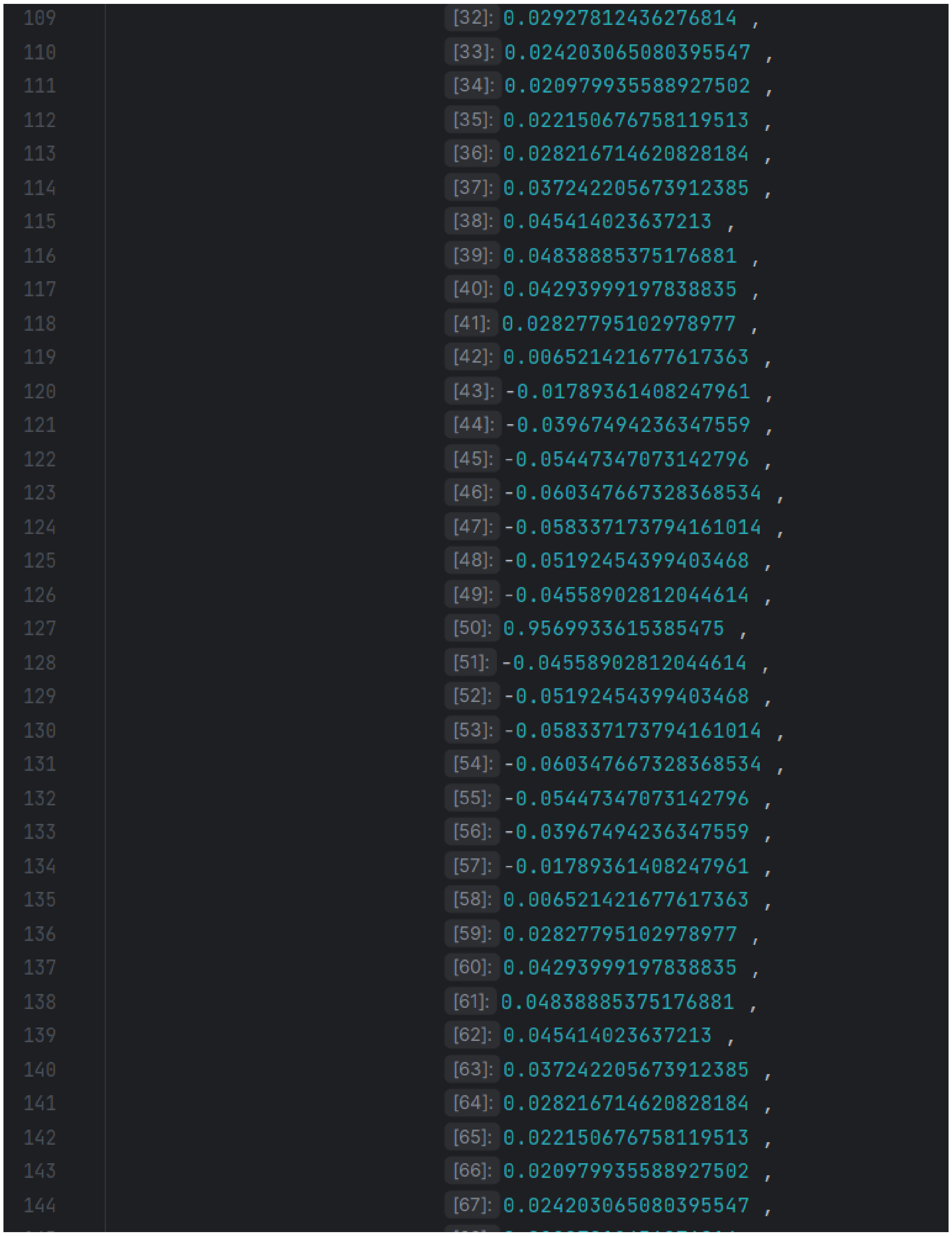


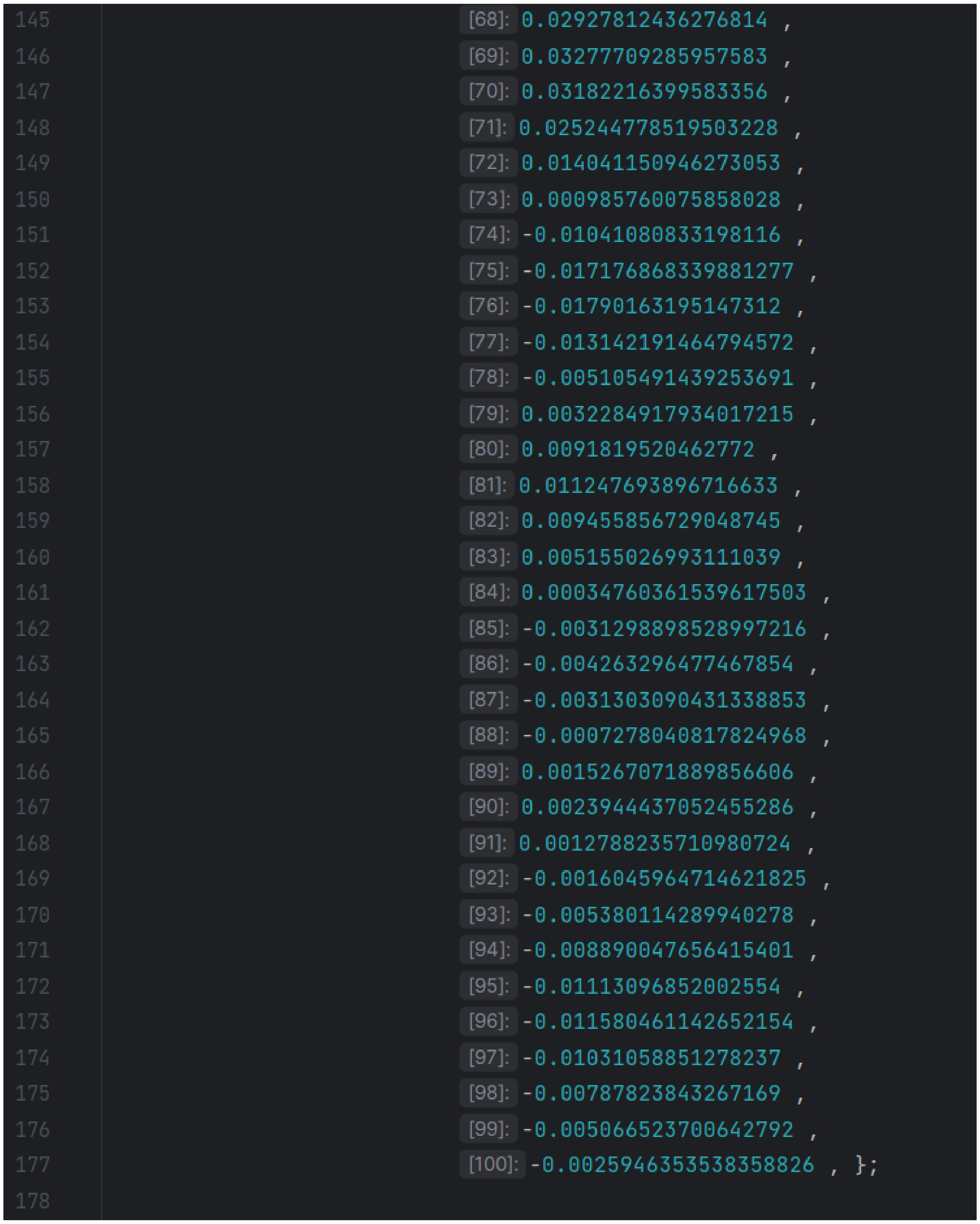
```
    // Your buffer conversion (unpacking) here
    // Fetch data sample from dataBuf->buf_[], unpack and put into bufferIn[]
    // ******************** START YOUR CODE HERE ******************** //


    for (int i = 0; i< FRAME_SIZE; i++){
        bufferIn[i] = (int16_t)dataBuf->buf_[2*i]+(int16_t)dataBuf->buf_[2*i+1]*256;
    }



    // ********************* END YOUR CODE HERE ********************* //

    // Loop code provided as a suggestion. This loop simulates sample-by-sample processing.
    for (int sampleIdx = 0; sampleIdx < FRAME_SIZE; sampleIdx++) {
        // Grab one sample from bufferIn[]
        int16_t sample = bufferIn[sampleIdx];
        // Call your filFilter funcion
        int16_t output = firFilter(sample);
        //int16_t output = sample;
        // Grab result and put into bufferOut[]
        bufferOut[sampleIdx] = output;
    }

    // Your buffer conversion (packing) here
    // Fetch data sample from bufferOut[], pack them and put back into dataBuf->buf_[]
    // ******************** START YOUR CODE HERE ******************** //
    for (int j=0 ; j < FRAME_SIZE; j++){
        uint8_t temp = bufferOut[j];
        dataBuf -> buf_[2*j] = bufferOut[j];
        dataBuf -> buf_[2*j+1] = (((uint16_t)bufferOut[j]-temp)>>8);

    }
    for (int k=0 ; k < FRAME_SIZE; k++){
        buffer_o[k] = dataBuf -> buf_[k];
    }

```



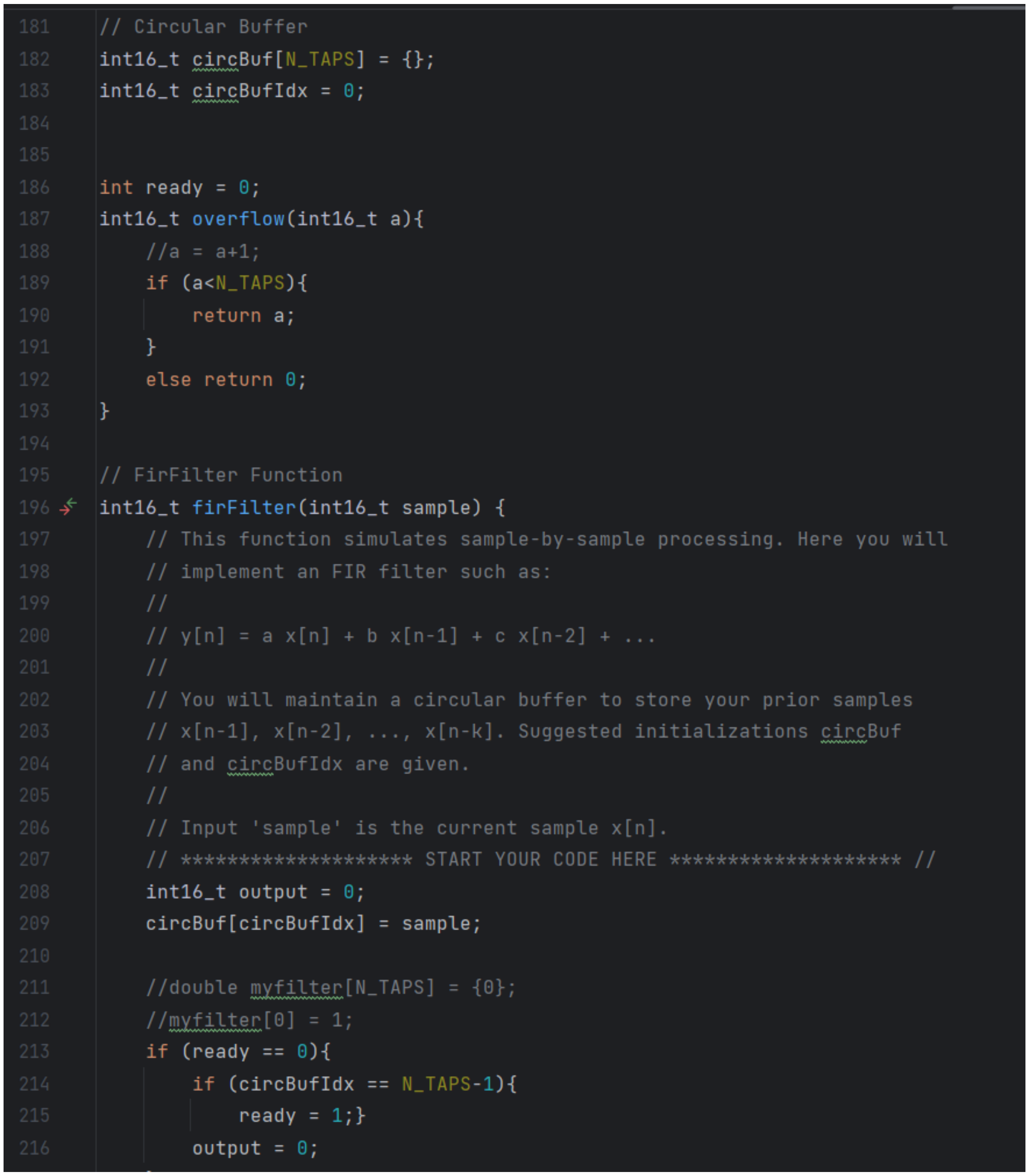
```
// TODO: Change N_TAPS to match your filter design
#define N_TAPS 101
// TODO: Change myfilter to contain the coefficients of your designed filte
double myfilter[N_TAPS] = { [0]: -0.0025946353538358826 ,
                            [1]: -0.005066523700642792 ,
                            [2]: -0.00787823843267169 ,
                            [3]: -0.01031058851278237 ,
                            [4]: -0.011580461142652154 ,
                            [5]: -0.01113096852002554 ,
                            [6]: -0.008890047656415401 ,
                            [7]: -0.005380114289940278 ,
                            [8]: -0.0016045964714621825 ,
                            [9]: 0.0012788235710980724 ,
                            [10]: 0.0023944437052455286 ,
                            [11]: 0.0015267071889856606 ,
                            [12]: -0.0007278040817824968 ,
                            [13]: -0.0031303090431338853 ,
                            [14]: -0.004263296477467854 ,
                            [15]: -0.0031298898528997216 ,
                            [16]: 0.00034760361539617503 ,
                            [17]: 0.005155026993111039 ,
                            [18]: 0.009455856729048745 ,
                            [19]: 0.011247693896716633 ,
                            [20]: 0.0091819520462772 ,
                            [21]: 0.0032284917934017215 ,
                            [22]: -0.005105491439253691 ,
                            [23]: -0.013142191464794572 ,
                            [24]: -0.01790163195147312 ,
                            [25]: -0.017176868339881277 ,
                            [26]: -0.01041080833198116 ,
                            [27]: 0.000985760075858028 ,
                            [28]: 0.014041150946273053 ,
                            [29]: 0.025244778519503228 ,
                            [30]: 0.03182216399583356 ,
                            [31]: 0.03277709285957583 ,
```

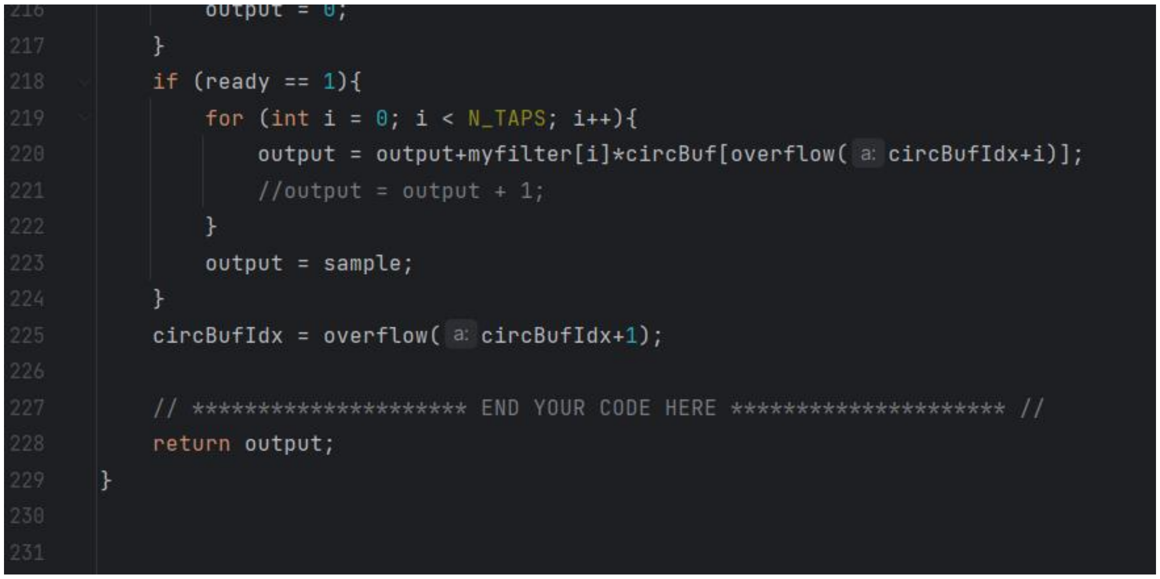


```
[32]: 0.02927812436276814 ,
[33]: 0.024203065080395547 ,
[34]: 0.020979935588927502 ,
[35]: 0.022150676758119513 ,
[36]: 0.028216714620828184 ,
[37]: 0.037242205673912385 ,
[38]: 0.045414023637213 ,
[39]: 0.04838885375176881 ,
[40]: 0.04293999197838835 ,
[41]: 0.02827795102978977 ,
[42]: 0.006521421677617363 ,
[43]: -0.01789361408247961 ,
[44]: -0.03967494236347559 ,
[45]: -0.05447347073142796 ,
[46]: -0.060347667328368534 ,
[47]: -0.058337173794161014 ,
[48]: -0.05192454399403468 ,
[49]: -0.04558902812044614 ,
[50]: 0.9569933615385475 ,
[51]: -0.04558902812044614 ,
[52]: -0.05192454399403468 ,
[53]: -0.058337173794161014 ,
[54]: -0.060347667328368534 ,
[55]: -0.05447347073142796 ,
[56]: -0.03967494236347559 ,
[57]: -0.01789361408247961 ,
[58]: 0.006521421677617363 ,
[59]: 0.02827795102978977 ,
[60]: 0.04293999197838835 ,
[61]: 0.04838885375176881 ,
[62]: 0.045414023637213 ,
[63]: 0.037242205673912385 ,
[64]: 0.028216714620828184 ,
[65]: 0.022150676758119513 ,
[66]: 0.020979935588927502 ,
[67]: 0.024203065080395547 ,
```



```
[68]: 0.02927812436276814 ,
[69]: 0.03277709285957583 ,
[70]: 0.03182216399583356 ,
[71]: 0.025244778519503228 ,
[72]: 0.014041150946273053 ,
[73]: 0.000985760075858028 ,
[74]: -0.01041080833198116 ,
[75]: -0.017176868339881277 ,
[76]: -0.01790163195147312 ,
[77]: -0.013142191464794572 ,
[78]: -0.005105491439253691 ,
[79]: 0.0032284917934017215 ,
[80]: 0.0091819520462772 ,
[81]: 0.011247693896716633 ,
[82]: 0.009455856729048745 ,
[83]: 0.005155026993111039 ,
[84]: 0.00034760361539617503 ,
[85]: -0.0031298898528997216 ,
[86]: -0.004263296477467854 ,
[87]: -0.0031303090431338853 ,
[88]: -0.0007278040817824968 ,
[89]: 0.0015267071889856606 ,
[90]: 0.0023944437052455286 ,
[91]: 0.0012788235710980724 ,
[92]: -0.0016045964714621825 ,
[93]: -0.005380114289940278 ,
[94]: -0.008890047656415401 ,
[95]: -0.01113096852002554 ,
[96]: -0.011580461142652154 ,
[97]: -0.01031058851278237 ,
[98]: -0.00787823843267169 ,
[99]: -0.005066523700642792 ,
[100]: -0.0025946353538358826 , };
```


```c
// Circular Buffer
int16_t circBuf[N_TAPS] = {};
int16_t circBufIdx = 0;


int ready = 0;
int16_t overflow(int16_t a){
    //a = a+1;
    if (a<N_TAPS){
        return a;
    }
    else return 0;
}

// FirFilter Function
int16_t firFilter(int16_t sample) {
    // This function simulates sample-by-sample processing. Here you will
    // implement an FIR filter such as:
    //
    // y[n] = a x[n] + b x[n-1] + c x[n-2] + ...
    //
    // You will maintain a circular buffer to store your prior samples
    // x[n-1], x[n-2], ..., x[n-k]. Suggested initializations circBuf
    // and circBufIdx are given.
    //
    // Input 'sample' is the current sample x[n].
    // ******************** START YOUR CODE HERE ******************** //
    int16_t output = 0;
    circBuf[circBufIdx] = sample;

    //double myfilter[N_TAPS] = {0};
    //myfilter[0] = 1;
    if (ready == 0){
        if (circBufIdx == N_TAPS-1){
            ready = 1;}
        output = 0;
```



```
        output = 0;
    }
    if (ready == 1){
        for (int i = 0; i < N_TAPS; i++){
            output = output+myfilter[i]*circBuf[overflow( a: circBufIdx+i)];
            //output = output + 1;
        }
        output = sample;
    }
    circBufIdx = overflow( a: circBufIdx+1);

    // ******************** END YOUR CODE HERE ******************** //
    return output;
}
```