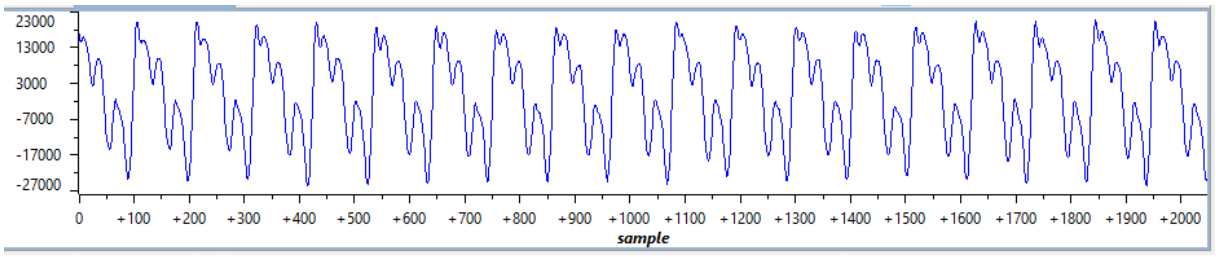
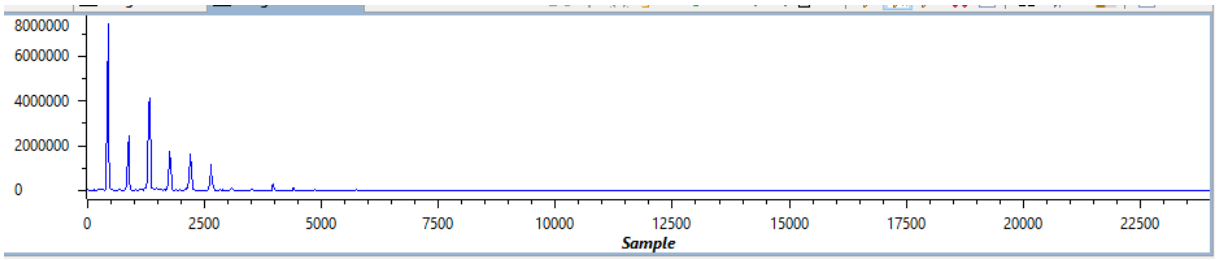
Zastosowania Procesorów Sygnałowych
Raport z zadania projektowego nr 3
Analiza widmowa
Mateusz Miler 171577
31.05.2020

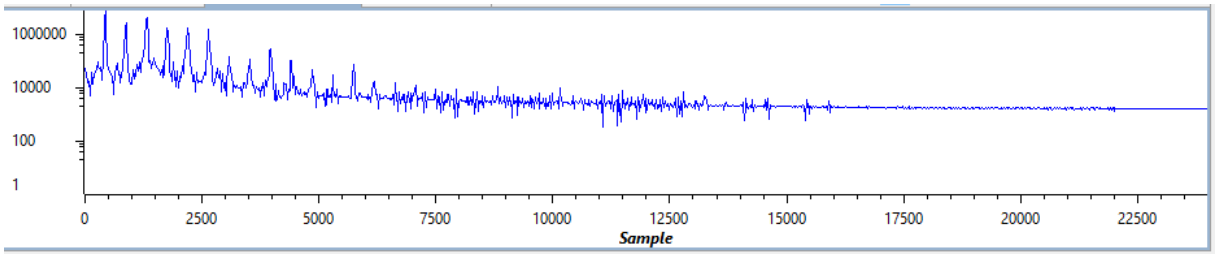
1. Przygotowanie sygnału testowego



Wyk.1. Wykres czasowy zawartości tablicy testsignal zawierającej próbki fragmentu nagrania klarnetu



Wyk.2. Widmo zawartości tablicy testsignal zawierającej próbki fragmentu nagrania klarnetu w skali liniowej (okno Hamminga) wykreślone przez CCS



Wyk.3. Widmo zawartości tablicy testsignal zawierającej próbki fragmentu nagrania klarnetu w skali logarytmicznej (okno Hamminga) wykreślone przez CCS

Na wykresie czasowym można zauważyć, że sygnał jest okresowy, czego się spodziewano. Pseudo-okres składa się z około 100 próbek, co przy szybkości próbkowania 48000 próbek/sekundę daje pseudo-okres T=100/48000=1/480=0,00208(3) sekundy. Częstotliwość podstawowa instrumentu wynosi zatem około f=480 Hz. Na wykresie widmowym można zauważyć prążek w okolicy 480 Hz, jednak wcale nie jest to prążek wiodący. Największy jest prążek pierwszy w częstotliwości około 100 Hz. Dla uzyskania dokładniejszych wyników wysokości instrumentu potrzebne są inne metody obliczeń.

2. Analiza widmowa sygnału

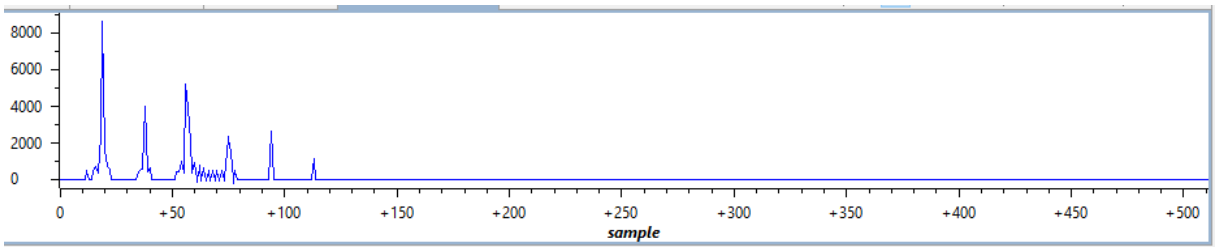
```
326     //zad 2 Analiza widmowa sygnału
327     copy((int*)testsignal, buffer_fft, N);
328     rfft((DATA*)buffer_fft, N, SCALE);
```

Rys.1. Kod kopiujący próbki do nowego buforu i wykonujący na nim FFT

```
326     //zad 2 Analiza widmowa sygnału
327     copy((int*)testsignal, buffer_fft, N);
328     rfft((DATA*)buffer_fft, N, SCALE);
329     Amrfft(buffer_fft, N);
```

```
206 //Oblicza widmo amplitudowe na podstawie zespolonego widma
207 //Nadpisuje probki widma z buffer
208 //buflen musi byc potega liczby 2
209 //STRUKTURA BUFFER:
210 //dwie pierwsze probki to: skladowa stala, skladowa Nyquista: y(0)Re, y(nx/2)im
211 //pozostałe: y(1)Re, y(1)Im, y(2)Re, y(2)Im...
212 void Amrfft(int* buffer, unsigned int buflen)
213 {
214     //uzyc w przypadku liczenia dokladnej amplitudy na probke
215     //opcja 2. w skalowaniu
216     //int pow;
217     //log_2((DATA*)buflen, (LDATA*)pow, 1);
218
219     int limit = buflen>>1;
220     int re, im;
221     int i;
222     for(i = 0; i < limit; i++)
223     {
224         //buffer[0] to usredniona wartosc sygnalu, skladowa stala
225         re = _smpy(buffer[2*i], buffer[2*i]);
226         im = _smpy(buffer[2*i+1], buffer[2*i+1]);
227         buffer[i] = re + im; //mozemy dodac bez obawy przepelnienia, poniewaz sa to kwadraty liczb <1
228     }
229
230     sqrt_16((DATA*)buffer, (DATA*)buffer, limit); //po tej operacji w temp mamy modul widma (jena polowa)
231
232     //Skalowanie
233     for(i = 0; i < limit; i++)
234     {
235         //1. musimy uwzglednic druga (symetryczna) czesc widma
236         buffer[i] = buffer[i]<<1;
237
238         //2. ponizsze jesli chcielibysmy znac dokladna amplitude przypadająca na próbke
239         //tutaj za mala precyzja i ucina 6 prazek z widma (dla danych klarnet)
240         //buffer[i] = buffer[i]>>(pow-1);
241     }
242
243     //zerwowanie nadmiarowych probek
244     for(i = limit; i < buflen; i++)
245         buffer[i] = 0;
246 }
```

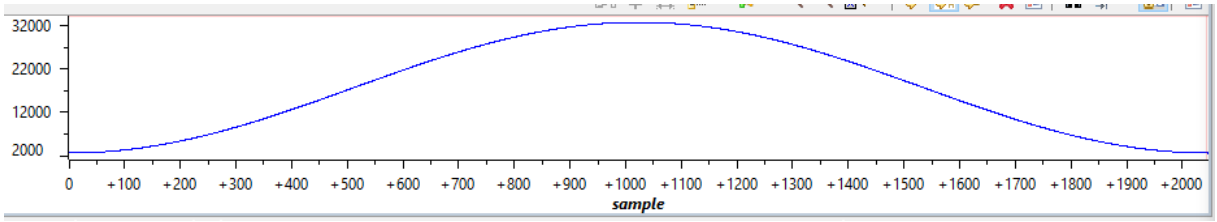
Rys.2. Kod kopiujący próbki do nowego buforu i wykonujący na nim FFT, a następnie za pomocą funkcji Amrfft zamienia widmo zespolone na widmo amplitudowe



Wyk.4. Widmo amplitudowe zawartości tablicy testsignal zawierającej próbki fragmentu nagrania klarnetu w skali liniowej wykreślone na podstawie „ręcznie” wypełnionej tablicy

Ilość prążków (6) jest taka sama jak w przypadku wykreślenia ich przez CCS. Rozdzielczość częstotliwościowa wynosi df = fs/N= 48000/2048 = 23,4375 Hz/próbkę. Pierwszy prążek widoczny jest na około 20 próbce, co daje częstotliwość f = 20*df = 468,75 Hz. Jest to zbliżona częstotliwość do odczytanej z wykresu widmowego wykreślonego przez CCS. Zauważono zniekształcenia pomiędzy 3 i 4 próbką. Prawdopodobnie ich przyczyną jest nieciągłość funkcji (po jej zapętleniu) i może być wygładzona odpowiednim oknem.

3. Zastosowanie funkcji okna

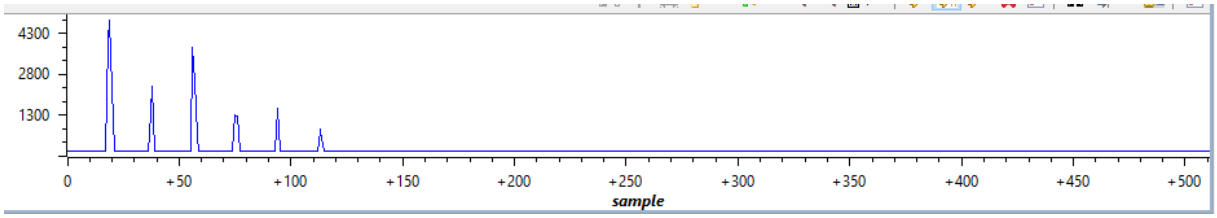


Wyk.5. Wykres czasowy okna Hamminga

```
336     //zad 2 Analiza widmowa sygnału
337     copy((int*)testsignal, buffer_fft, N);
338     multiply(buffer_fft, (int*)hamming, N);
339     rfft((DATA*)buffer_fft, N, SCALE);
340     Amrfft(buffer_fft, N);
```

```
249 //Mnozy bufor inOut przez in i zapisuje w inOut
250 //Oba bufory maja dlugosc buflen
251 void multiply(int* inOut, int* in, unsigned int buflen)
252 {
253     int i;
254     for(i = 0; i < buflen; i++)
255         inOut[i] = _smpy(inOut[i], in[i]);
256 }
```

Rys.3. Procedura wyznaczania widma amplitudowego rozszerzona o przemnożenie przez okno Hamminga



Wyk.6. Widmo amplitudowe zawartości tablicy testsignal zawierającej próbki fragmentu nagrania klarnetu w skali liniowej wykreślone na podstawie „ręcznie" wypełnionej tablicy oraz przemnożone wcześniej przez okno Hamminga

Wykres okna Hamminga wygląda na wykonany prawidłowo. Wartości nie przekraczają 1, skrajne krańce okna są większe od 0 oraz wstęgi rozkładają się wyraźnie płasko, co jest charakterystyczne dla tego okna. Pozbyto się „rozlewania widma" w okolicach prążków. Ilość prążków się zgadza, natomiast

ich amplituda zmalała, co skutkiem użycia okna. Jeśli do dalszych operacji potrzebne by były ich większe amplitudy, próbki można by przemnożyć jeszcze przez wartość większą od 1.

4. Wyszukiwanie maksimów w widmie

Nr	deltaO	deltaN	summit	Ilustracja	reakcja
1	+	+	0	//	
2	+	+	+	nie ma	
3	+	-	0	/\	prążek?
4	+	-	+	nie ma	
5	+	0	0	/--	summit++
6	+	0	+	nie ma	
7	-	+	0	\/	
8	-	+	+	nie ma	
9	-	-	0	\\	
10	-	-	+	nie ma	
11	-	0	0	\--	
12	-	0	+	nie ma	
13	0	+	0	\--/	
14	0	+	+	/--/	summit = 0
15	0	-	0	\--\	
16	0	-	+	/--\	prążek? Else summit = 0
17	0	0	0	\----	
18	0	0	+	/----	summit++

Tab.1. Tabela prawdy ułatwiająca wyszukiwanie maksimum lokalnego

```
259 //Zwraca indeks pierwszego maksimum lokalnego odrozniajacego sie od szumu z tablicy buffer o dlugosci buflen
260 //maksimum lokalne musi byc wieksze od threshold (odrzucanie szumu)
261 //maksymalna szerokosc prążka - maxWidth
262 int maxIndex(int* buffer, unsigned int buflen, int threshold, int maxWidth)
263 {
264     int i;
265     int deltaO = 0; //poprzednia delta
266     int deltaN = 0; //obecna delta
267     int summit = 0; //licznik szerokosci szczytu
268     for(i = 2; i < buflen; i++)
269     {
270         deltaO = buffer[i-1] - buffer[i-2];
271         deltaN = buffer[i] - buffer[i-1];
272 
273         if(deltaO > 0)
274         {
275             if(summit == 0)
276             {
277                 if(deltaN == 0) summit++;
278                 if (deltaN < 0)
279                 {
280                     if(buffer[i-1] > threshold) return i-1;
281                 }
282             }
283         }
284 
285         if(deltaO == 0)
286         {
287             if(deltaN > 0 && summit > 0) summit = 0;
288             if(summit > 0)
289             {
290                 if(deltaN == 0) summit++;
291                 if(deltaN < 0)
292                 {
293                     if(summit <= maxWidth && buffer[i-1] > threshold) return i-1-(summit>>1);
294                     else summit = 0;
295                 }
296             }
297         }
298     }
299     return 0;
300 }
```

Rys.4. Kod realizujący wyszukiwanie indeksu prążka zgodnie z wyżej przedstawioną tabelą prawdy

W pierwszej kolejności napisano funkcję odpowiedzialną za wyszukiwania pierwszego maksimum lokalnego (nie muszącym być koniecznie maksimum globalnym). Funkcję napisano na podstawie rozpisanej wyżej tabeli prawdy. Dzięki temu uniknięto „plątaniny w ifach" i poprawiono czytelność. Oprócz bufora i jego długości, funkcja jako dwa dodatkowe parametry przyjmuje *threshold* – wartość progową, poniżej której sygnał jest traktowany jako szum i nie jest możliwe przyjęcie go jako ekstremum – oraz maxWidth, która to określa maksymalną szerokość prążka na jego płaskim szczycie.

Kolejną funkcją była metoda zwracająca częstotliwość próbki na podstawie indeksu próbki w przebiegu widmowym. Funkcja ta działa jedynie dla szybkości próbkowania 48 kHz oraz stałego rozmiaru okna 2048. Dzięki takim ograniczeniom, udało się ją całkiem dobrze zoptymalizować, mimo procesora stałoprzecinkowego. Na początku próbowano zdefiniować funkcję jako typ *long*. Niestety nie działała ona poprawnie. Kolejnym etapem rozważań co do wyboru pojemnika na dane miał być *unsigned int*. Liczono tutaj na dodatkowy bit zapisu uwolniony z wartości ujemnych. Niestety, podobnie jak w przypadku *longa,* próba zakończyła się niepowodzeniem. Ostatecznie mnożenie kolejnych próbek przez fs/N = 48000/2048 = 375/16 [Hz/próbkę] zrealizowano na *intach* poprzez rozbicie iloczynów i ilorazów na czynniki pierwsze. Dzielenia wykonywano poprzez przesunięcia bitowe, tym samym oszczędzając na cyklach zegara. Drabinka warunków *else if* pozwoliła uniknąć zbyt szybkich przepełnień oraz zachować dokładność wyników. Nie dało się tego zrobić za pomocą jednej komendy, co przedstawiono w późniejszej części raportu na fragmentach tabel porównawczych opracowanych w excelu. Tabele utworzono na podstawie wydruków *(printf)* specjalnie napisanej w tym celu funkcji testowej. Na załączonych zdjęciach widać porównanie 3 różnych kryteriów kwalifikacyjnych z funkcji *freqIndex* (REGUŁY). Czerwonym kolorem zaznaczono pozycje, w których odstępstwo wyliczonej częstotliwości od rzeczywistej jest większe niż 1Hz. Lewa tabela przedstawia warunki z początku „drabinki if" – sprawdzają się one dobrze dla małych indeksów próbek, jednak dla dużych, odchyłki sięgają nawet 300 Hz. Środkowa tabela przedstawia klasyfikator przystosowany do indeksów o większych wartościach. Wyraźnie widać na nim zaznaczone na czerwono odchyłki już na początku listy, za to przy dużych indeksach odchyłki rzędu 300 Hz zmalały do około 10 Hz. Ostatnia, prawa tabela, przedstawia drabinkę warunków w całości. Pierwsza błędna wartość ma indeks aż 103. Cały arkusz kalkulacyjny załączono również osobno do sprawozdania.

```
303 //Zwraca czestotliwosc prazka widma o indeksie index
304 //Do obliczen zalozono:
305 //szybkosc probkowania 48 kHz
306 //staly rozmiar FFT 2048
307 int freqIndex(int index)
308 {
309     if (index <= 87) return (index*375)>>4;
310     else if (index <= 174) return (((index*125)>>1)*3)>>3;
311     else if (index <= 436) return (((index*75)>>3)*5)>>1;
312     else if (index <= 699) return ((((index*25)>>1)*3)>>2)*5>>1;
313     else if (index <= 1048) return ((((index*25)>>2)*5)>>2)*3;
314     else if (index <= 1310) return ((((index*25)>>2)*3)>>2)*5;
315     else if (index <= 1398) return ((((index*15)>>2)*5)>>2)*5;
316     else return ((((index*15)>>3)*5)>>1)*5;
317 }
```

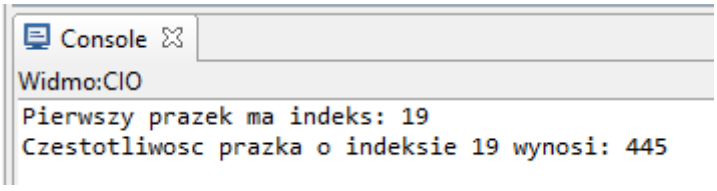
Rys.5. Funkcja zwracająca częstotliwość prążka na podstawie jego indeksu

```
319 //Drukuje czestotliwosc prazka o indksie indeks
320 //Do obliczen zalozono:
321 //szybkosc probkowania 48 kHz
322 //staly rozmiar FFT 2048
323 void printFreq(int index)
324 {
325     int freq = freqIndex(index);
326     if (freq < 0)
327     {
328         freq -= 32767;
329         printf("Czestotliwosc prazka o indeksie %d wynosi: 32767+%d\n", index, freq);
330         //printf("=32767+%d\n", freq); //do porownania w excelu
331     }
332     else
333     {
334         printf("Czestotliwosc prazka o indeksie %d wynosi: %d\n", index, freq);
335         //printf("%d\n", freq); //do porownania w excelu
336     }
337 }
```

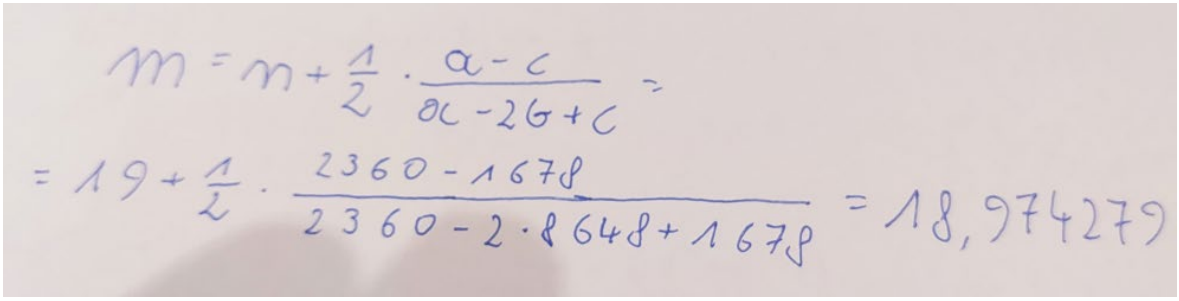
Rys.6. Funkcja do testowania zwracanej częstotliwości próbki o indeksie index przez funkcję freqIndex

```
430     //zad 4 Wyszukiwanie maksimów w widmie
431     copy((int*)testsignal, buffer_fft, N);
432     rfft((DATA*)buffer_fft, N, SCALE);
433     Amrfft(buffer_fft, N);
434
435     //Wypisanie pierwszego prążka
436     int index = maxIndex(buffer_fft, N<<2, 987, 5);
437     printf("Pierwszy prazek ma indeks: %d \n", index);
438     int freq = freqIndex(index);
439     printFreq(index);
440
441     //Sprawdzenie pozostalych czestotliwosci
442     int i;
443     printf("\n");
444     for(i = 0; i < 2048; i++)
445         printFreq(i);
```

Rys.7. Fragment kodu testujący napisane funkcje – wywołanie



Rys.8. Zwrócony przez program indeks pierwszego prążka oraz jego częstotliwość

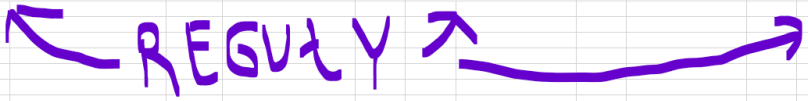


Rys.9. Ręcznie obliczona częstotliwość pierwszego prążka na podstawie próbek sąsiadujących

Częstotliwość zwrócona przez funkcję programu jest bardzo zbliżona do oczekiwanych 440 Hz – różnica 5 Hz. Po tak dokładnych obliczeniach możemy stwierdzić, że na klarnecie wybrzmiewał dźwięk A. Różnice mogą wynikać chociażby z ograniczonej rozdzielczości częstotliwościowej. Według obliczeń teoretycznych, rzeczywiste maksimum znajduje się między 18, a 19 próbką w punkcie 18,97 próbki licząc od początku sygnału. Należałoby jeszcze wspomnieć, że wyliczona parabola mogła się nieznacznie różnić od rzeczywistej, ze względu na wycieki widma (nie użyto okna, aby nie zniekształcić amplitudy w obliczeniach) oraz zaokrąglenia procesora (ograniczona precyzja Q15).

Poniżej przedstawiono fragmenty wcześniej omawianych tabel sporządzonych w excelu [załącznik].

return (index*375)>>4;



return ((((index*15)>>3)*5)>>1)*5;

if (index <= 87) return (index*375)>>4;
else if (index <= 174) return (((index*125)>>1)*3)>>3;
else if (index <= 436) return (((index*75)>>3)*5)>>1;
else if (index <= 699) return ((((index*25)>>1)*3)>>2)*5>>1;
else if (index <= 1048) return ((((index*25)>>2)*5)>>2)*3;
else if (index <= 1310) return ((((index*25)>>2)*3)>>2)*5;
else if (index <= 1398) return ((((index*15)>>2)*5)>>2)*5;

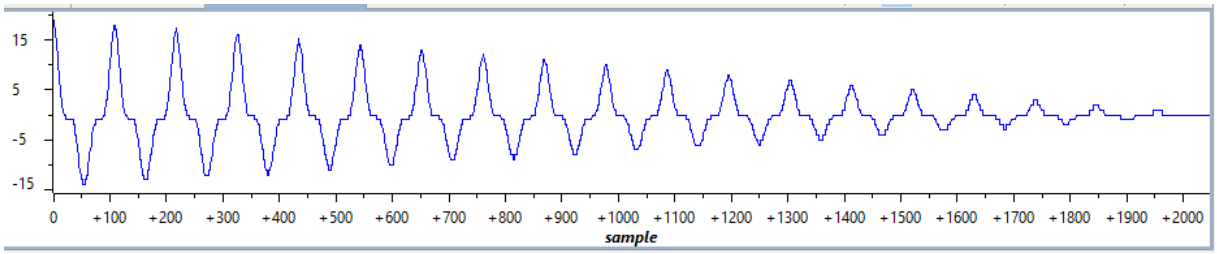
else return ((((index*15)>>3)*5)>>1)*5;

nr próbki	częstotliwość dokładna	różnica	częstotliwosc program
1	23,4375	0,4375	23
2	46,875	11,875	35
3	70,3125	10,3125	60
4	93,75	8,75	85
5	117,1875	7,1875	110
6	140,625	5,625	135
7	164,0625	4,0625	160
8	187,5	2,5	185
9	210,9375	10,9375	200
10	234,375	9,375	225
11	257,8125	7,8125	250
12	281,25	6,25	275
13	304,6875	4,6875	300
14	328,125	3,125	325
15	351,5625	1,5625	350
16	375	0	375
17	398,4375	13,4375	385
18	421,875	11,875	410
19	445,3125	10,3125	435
20	468,75	8,75	460
21	492,1875	7,1875	485
22	515,625	5,625	510
23	539,0625	4,0625	535
24	562,5	2,5	560
25	585,9375	10,9375	575
26	609,375	9,375	600
27	632,8125	7,8125	625
28	656,25	6,25	650
84	1968,75	8,75	1960
85	1992,1875	7,1875	1985
86	2015,625	5,625	2010
87	2039,0625	4,0625	2035
88	2062,5	2,5	2060
89	2085,9375	10,9375	2075
90	2109,375	9,375	2100
91	2132,8125	7,8125	2125
92	2156,25	6,25	2150
93	2179,6875	4,6875	2175
94	2203,125	3,125	2200
95	2226,5625	1,5625	2225
96	2250	0	2250
97	2273,4375	13,4375	2260
98	2296,875	11,875	2285
99	2320,3125	10,3125	2310
100	2343,75	8,75	2335
101	2367,1875	7,1875	2360
102	2390,625	5,625	2385
103	2414,0625	4,0625	2410
104	2437,5	2,5	2435
105	2460,9375	10,9375	2450
106	2484,375	9,375	2475
107	2507,8125	7,8125	2500
108	2531,25	6,25	2525
109	2554,6875	4,6875	2550
110	2578,125	3,125	2575
111	2601,5625	1,5625	2600
112	2625	0	2625
113	2648,4375	13,4375	2635
114	2671,875	11,875	2660
115	2695,3125	10,3125	2685
116	2718,75	8,75	2710
117	2742,1875	7,1875	2735
118	2765,625	5,625	2760
119	2789,0625	4,0625	2785
120	2812,5	2,5	2810
121	2835,9375	10,9375	2825

nr próbki	częstotliwość dokładna	różnica	częstotliwosc program
1	23,4375	0,4375	23
2	46,875	0,875	46
3	70,3125	0,3125	70
4	93,75	0,75	93
5	117,1875	0,1875	117
6	140,625	0,625	140
7	164,0625	0,0625	164
8	187,5	0,5	187
9	210,9375	0,9375	210
10	234,375	0,375	234
11	257,8125	0,8125	257
12	281,25	0,25	281
13	304,6875	0,6875	304
14	328,125	0,125	328
15	351,5625	0,5625	351
16	375	0	375
17	398,4375	0,4375	398
18	421,875	0,875	421
19	445,3125	0,3125	445
20	468,75	0,75	468
21	492,1875	0,1875	492
22	515,625	0,625	515
23	539,0625	0,0625	539
24	562,5	0,5	562
25	585,9375	0,9375	585
26	609,375	0,375	609
27	632,8125	0,8125	632
28	656,25	0,25	656
84	1968,75	0,75	1968
85	1992,1875	0,1875	1992
86	2015,625	0,625	2015
87	2039,0625	0,0625	2039
88	2062,5	0,5	2062
89	2085,9375	0,9375	2085
90	2109,375	0,375	2109
91	2132,8125	0,8125	2132
92	2156,25	0,25	2156
93	2179,6875	0,6875	2179
94	2203,125	0,125	2203
95	2226,5625	0,5625	2226
96	2250	0	2250
97	2273,4375	0,4375	2273
98	2296,875	0,875	2296
99	2320,3125	0,3125	2320
100	2343,75	0,75	2343
101	2367,1875	0,1875	2367
102	2390,625	0,625	2390
103	2414,0625	1,0625	2413
104	2437,5	0,5	2437
105	2460,9375	0,9375	2460
106	2484,375	0,375	2484
107	2507,8125	0,8125	2507
108	2531,25	0,25	2531
109	2554,6875	0,6875	2554
110	2578,125	0,125	2578
111	2601,5625	0,5625	2601
112	2625	0	2625
113	2648,4375	0,4375	2648
114	2671,875	0,875	2671
115	2695,3125	0,3125	2695
116	2718,75	0,75	2718
117	2742,1875	0,1875	2742
118	2765,625	0,625	2765
119	2789,0625	1,0625	2788
120	2812,5	0,5	2812
121	2835,9375	0,9375	2835

2010	47109,375	234,375	46875	2010	47109,375	9,375	47100	2010	47109,375	9,375	47100
2011	47132,8125	257,8125	46875	2011	47132,8125	7,8125	47125	2011	47132,8125	7,8125	47125
2012	47156,25	281,25	46875	2012	47156,25	6,25	47150	2012	47156,25	6,25	47150
2013	47179,6875	304,6875	46875	2013	47179,6875	4,6875	47175	2013	47179,6875	4,6875	47175
2014	47203,125	328,125	46875	2014	47203,125	3,125	47200	2014	47203,125	3,125	47200
2015	47226,5625	351,5625	46875	2015	47226,5625	1,5625	47225	2015	47226,5625	1,5625	47225
2016	47250	0	47250	2016	47250	0	47250	2016	47250	0	47250
2017	47273,4375	23,4375	47250	2017	47273,4375	13,4375	47260	2017	47273,4375	13,4375	47260
2018	47296,875	46,875	47250	2018	47296,875	11,875	47285	2018	47296,875	11,875	47285
2019	47320,3125	70,3125	47250	2019	47320,3125	10,3125	47310	2019	47320,3125	10,3125	47310
2020	47343,75	93,75	47250	2020	47343,75	8,75	47335	2020	47343,75	8,75	47335
2021	47367,1875	117,1875	47250	2021	47367,1875	7,1875	47360	2021	47367,1875	7,1875	47360
2022	47390,625	140,625	47250	2022	47390,625	5,625	47385	2022	47390,625	5,625	47385
2023	47414,0625	164,0625	47250	2023	47414,0625	4,0625	47410	2023	47414,0625	4,0625	47410
2024	47437,5	187,5	47250	2024	47437,5	2,5	47435	2024	47437,5	2,5	47435
2025	47460,9375	210,9375	47250	2025	47460,9375	10,9375	47450	2025	47460,9375	10,9375	47450
2026	47484,375	234,375	47250	2026	47484,375	9,375	47475	2026	47484,375	9,375	47475
2027	47507,8125	257,8125	47250	2027	47507,8125	7,8125	47500	2027	47507,8125	7,8125	47500
2028	47531,25	281,25	47250	2028	47531,25	6,25	47525	2028	47531,25	6,25	47525
2029	47554,6875	304,6875	47250	2029	47554,6875	4,6875	47550	2029	47554,6875	4,6875	47550
2030	47578,125	328,125	47250	2030	47578,125	3,125	47575	2030	47578,125	3,125	47575
2031	47601,5625	351,5625	47250	2031	47601,5625	1,5625	47600	2031	47601,5625	1,5625	47600
2032	47625	0	47625	2032	47625	0	47625	2032	47625	0	47625
2033	47648,4375	23,4375	47625	2033	47648,4375	13,4375	47635	2033	47648,4375	13,4375	47635
2034	47671,875	46,875	47625	2034	47671,875	11,875	47660	2034	47671,875	11,875	47660
2035	47695,3125	70,3125	47625	2035	47695,3125	10,3125	47685	2035	47695,3125	10,3125	47685
2036	47718,75	93,75	47625	2036	47718,75	8,75	47710	2036	47718,75	8,75	47710
2037	47742,1875	117,1875	47625	2037	47742,1875	7,1875	47735	2037	47742,1875	7,1875	47735
2038	47765,625	140,625	47625	2038	47765,625	5,625	47760	2038	47765,625	5,625	47760
2039	47789,0625	164,0625	47625	2039	47789,0625	4,0625	47785	2039	47789,0625	4,0625	47785
2040	47812,5	187,5	47625	2040	47812,5	2,5	47810	2040	47812,5	2,5	47810
2041	47835,9375	210,9375	47625	2041	47835,9375	10,9375	47825	2041	47835,9375	10,9375	47825
2042	47859,375	234,375	47625	2042	47859,375	9,375	47850	2042	47859,375	9,375	47850
2043	47882,8125	257,8125	47625	2043	47882,8125	7,8125	47875	2043	47882,8125	7,8125	47875
2044	47906,25	281,25	47625	2044	47906,25	6,25	47900	2044	47906,25	6,25	47900
2045	47929,6875	304,6875	47625	2045	47929,6875	4,6875	47925	2045	47929,6875	4,6875	47925
2046	47953,125	328,125	47625	2046	47953,125	3,125	47950	2046	47953,125	3,125	47950
2047	47976,5625	351,5625	47625	2047	47976,5625	1,5625	47975	2047	47976,5625	1,5625	47975

5. Znajdywanie częstotliwości podstawowej metodą autokorelacji



Wyk.7. Wykres funkcji autokorelacji dla sygnału testsignal (klarnet) podzielonego przez 16

```
441
442     //zad 5 Znajdywanie czestotliwości podstawowej metodą autokorelacji
443     int i;
444     for (i = 0; i < N; i++)
445         samples[i] = testsignal[i]>>4;
446     acorr((DATA*)samples, (DATA*)samples2, 2048, 2048, bias);
447     int index = maxIndex(samples2, N, 0, 5);
448     printf("Pierwsze maksimum koleracji wystepuje dla indeksu: %d \n", index);
449
```

Console | Single Time -7 | Single Time -8

Widmo:CIO

```
Pierwsze maksimum koleracji wystepuje dla indeksu: 109
```

Rys.10. Implementacja funkcji autokorelacji, znajdywanie jej maksimum oraz wypisanie indeksu wartości maksymalnej

Wykres funkcji autokorelacji składa się z symetrycznych powtarzalnych okresów – wraz z przesuwaniem się po sygnale sygnał jest coraz mniej skorelowany z poprzednim okresem i coraz mocniej skorelowany z następnym – stąd symetryczny kształt. Wartości mają tendencję malejącą,

ponieważ wyjeżdżamy poza granicę tablicy – kończą się próbki i sygnał zaczyna się coraz mocniej różnić – wartości spadają do zera. Wartość 109 próbek wydaje się być prawdziwa i jest zbliżona do tej szacowanej w pierwszym ćwiczeniu. Pseudo-okres składa się z 109 próbek, co przy szybkości próbkowania 48000 próbek/sekundę daje pseudo-okres T=109/48000=0,0022708(3) sekundy. Częstotliwość podstawowa instrumentu wynosi zatem około f = 48000/109 = 440,36 Hz, czyli dokładnie tyle ile wynosi częstotliwość dźwięku A, którego się spodziewaliśmy. Wyniki metodą autokorelacji okazały się dokładniejsze od metody widmowej.