

```
1. #include <stdlib.h>
2. #include <stdio.h>
3. #include <fcntl.h>
4. #include <math.h>
5. #include <string.h>
6.
7. struct dpoint{
8.     double a;
9.     double b;
10.    double cent;
11.    int n;
12.    double f;
13.    double pbin;
14. };
15.
16. void
17. genGistDataAbs(struct dpoint* mas, int n)
18. {
19.     FILE* out = fopen("Abs.txt", "w");
20.     for (int i = 0; i < n; i++){
21.         fprintf(out, "0 %.3lf\n", mas[i].a);
22.         fprintf(out, "%d %.3lf\n", mas[i].n, mas[i].a);
23.         fprintf(out, "%d %.3lf\n", mas[i].n, mas[i].b);
24.         fprintf(out, "0 %.3lf\n", mas[i].b);
25.     }
26.     fclose(out);
27. }
28.
29. int
30. generateMas(struct dpoint* mas, char* name, int flg)
31. {
32.     int n=0;
33.     FILE* in = fopen(name, "r");
34.     n = 0;
35.     double tmp;
36.     if(flg){
37.         int sum = 0;
38.         while(fscanf(in, "%lf %lf %d", &mas[n].a, &mas[n].b, &mas[n].n) != EOF){
39.             n++;
40.         }
41.         for(int i = 0; i < n; i++){
42.             mas[i].cent = (mas[i].a + mas[i].b)/2;
43.             sum += mas[i].n;
44.         }
45.         for(int i = 0; i < n; i++){
46.             mas[i].f = (double)mas[i].n/sum;
47.         }
48.     }else{
49.         fscanf(in, "%lf", &tmp);
50.         mas[0].cent = tmp;
51.         mas[0].n = 1;
52.         n = 1;
53.         while(fscanf(in, "%lf", &tmp) != EOF){
54.             int flg = 1;
55.             for(int i = 0; i < n; i++){
56.                 if(tmp == mas[i].cent){
57.                     mas[i].n++;
58.                     flg = 0;
59.                     break;
60.                 }
61.             }
62.             if(flg){
```

```
63.         mas[n].cent = tmp;
64.         mas[n].n = 1;
65.         n++;
66.     }
67. }
68. int sum = 0;
69. for (int i = 0; i < n; ++i){
70.     sum += mas[i].n;
71. }
72. for (int i = 0; i < n; ++i){
73.     mas[i].f = (double)mas[i].n/sum;
74. }
75. }
76. fclose(in);
77. return n;
78. }
79.
80. int
81. HandleInput(struct dpoint* mas, int n, int flg)
82. {
83.     int sum = 0;
84.     int size = 1;
85.     if(flg){
86.         for (int i = 0; i < n; i++){
87.             printf("(a,b),n [%d]:\n",i);
88.             scanf("%lf %lf %d",&mas[i].a,&mas[i].b,&mas[i].n);
89.         }
90.         for(int i = 0; i < n; i++){
91.             mas[i].cent = (mas[i].a + mas[i].b)/2;
92.             sum += mas[i].n;
93.         }
94.         for(int i = 0; i < n; i++){
95.             mas[i].f = (double)mas[i].n/sum;
96.         }
97.     }else{
98.         printf("Mas of data is:\n");
99.         double tmp;
100.        scanf("%lf",&mas[0].cent);
101.        mas[0].n = 1;
102.        for(int i = 1; i < n; i++){
103.            scanf("%lf",&tmp);
104.            int flg1 = 1;
105.            for(int j = 0; j < size; j++){
106.                if(tmp == mas[j].cent){
107.                    mas[j].n++;
108.                    flg1 = 0;
109.                    break;
110.                }
111.            }
112.            if(flg1){
113.                mas[size].cent = tmp;
114.                mas[size].n = 1;
115.                size++;
116.            }
117.        }
118.        for(int i = 0; i < size; i++){
119.            sum += mas[i].n;
120.        }
121.        for(int i = 0; i < size; i++){
122.            mas[i].f = (double)mas[i].n/sum;
123.        }
124.    }
```

```
125.     return (flg)?n:size;
126. }
127.
128. double
129. F0(double a)
130. {
131.     double res = erf(a/sqrt(2))/2;
132.     return res;
133. }
134.
135. int
136. isA(double a)
137. {
138.     if(a == 0.995)
139.         return 0;
140.     if(a == 0.99)
141.         return 1;
142.     if(a == 0.975)
143.         return 2;
144.     if(a == 0.95)
145.         return 3;
146.     if(a == 0.9)
147.         return 4;
148.     if(a == 0.75)
149.         return 5;
150.     if(a == 0.5)
151.         return 6;
152.     if(a == 0.25)
153.         return 7;
154.     if(a == 0.1)
155.         return 8;
156.     if(a == 0.05)
157.         return 9;
158.     if(a == 0.025)
159.         return 10;
160.     if(a == 0.01)
161.         return 11;
162.     if(a == 0.005)
163.         return 12;
164.     return -1;
165. }
166.
167. double
168. GetXiSq(double a, int k)
169. {
170.     FILE* in = fopen("xi2Table.txt", "r");
171.     int key = 0;
172.     double xi[13];
173.     while(key != k){
174.         fscanf(in, "%d %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf",
%lf", &key, &xi[0], &xi[1], &xi[2], &xi[3], &xi[4], &xi[5], &xi[6], &xi[7], &xi[8], &xi[9], &xi[10], &xi[11], &xi[12]);
175.     }
176.     int num = isA(a);
177.     fclose(in);
178.     return (num == -1)?-1:xi[num];
179. }
180.
181. void
182. genPolDataAbs(struct dpoint* mas, int n)
183. {
184.     FILE* out = fopen("Abskr.txt", "w");
185.     for (int i = 0; i < n; ++i){
```

```
186.     fprintf(out, "%d %.3lf\n", mas[i].n, mas[i].cent);
187. }
188. fclose(out);
189. }
190.
191. void
192. genPolDataAbs1(struct dpoint* mas, int n)
193. {
194.     FILE* out = fopen("Abskr.txt", "w");
195.     for (int i = 0; i < n; ++i){
196.         fprintf(out, "%.3lf %.3lf\n", mas[i].pbin, mas[i].cent);
197.     }
198.     fclose(out);
199. }
200.
201. void
202. genPolDataRel(struct dpoint* mas, int n)
203. {
204.     FILE* out = fopen("Absrel.txt", "w");
205.     for (int i = 0; i < n; ++i){
206.         fprintf(out, "%.3lf %.3lf\n", mas[i].f, mas[i].cent);
207.     }
208.     fclose(out);
209. }
210.
211. void
212. genScr(double a, double b, int y, int tmp2)
213. {
214.     FILE* scr = fopen("scr_1.txt", "w");
215.     fprintf(scr, "set terminal png\n");
216.     fprintf(scr, "set output 'Abs.png'\n");
217.     fprintf(scr, "set xrange [%lf:%lf]\n", a, b);
218.     fprintf(scr, "set yrange [0:%ld]\n", y);
219.     fprintf(scr, "set multiplot\n");
220.     if(tmp2)
221.         fprintf(scr, "plot \"Abs.txt\" u 2:1 w l lw 4\n");
222.     fprintf(scr, "plot \"Abskr.txt\" u 2:1 w l lw 4 lt rgb 'red'\n");
223.     fclose(scr);
224. }
225.
226. void
227. genScrN(double a, double sig, double x, double y)
228. {
229.     FILE* scr = fopen("scr_2.txt", "w");
230.     fprintf(scr, "set terminal png\n");
231.     fprintf(scr, "set output 'Abs1.png'\n");
232.     fprintf(scr, "set xrange [%lf:%lf]\n", x, y);
233.     fprintf(scr, "plot 1/(sqrt(2*pi)*%lf)*exp(-(x-%lf)**2/(2*%lf**2)) lw 4 lt rgb 'red'", sig, a, sig);
234.     fprintf(scr, "plot \"Abskr.txt\" u 2:1 w l lw 4 lt rgb 'red'\n");
235.     fclose(scr);
236. }
237.
238.
239. void
240. genScrBin(double a, double b)
241. {
242.     FILE* scr = fopen("scr_bin.txt", "w");
243.     fprintf(scr, "set terminal png\n");
244.     fprintf(scr, "set output 'TeorBin.png'\n");
245.     fprintf(scr, "set xrange [%lf:%lf]\n", a, b);
246.     // fprintf(scr, "set yrange [0:1]\n");
247.     fprintf(scr, "plot \"Abskr.txt\" u 2:1 w l lw 4 lt rgb 'red'\n");
```

```
248.     fclose(scr);
249. }
250.
251. void
252. genScrBinT(double a, double b)
253. {
254.     FILE* scr = fopen("scr_bint.txt", "w");
255.     fprintf(scr, "set terminal png\n");
256.     fprintf(scr, "set output 'bin.png'\n");
257.     fprintf(scr, "set xrange [%lf:%lf]\n", a, b);
258.     // fprintf(scr, "set yrange [0:1]\n");
259.     fprintf(scr, "plot \"Absrel.txt\" u 2:1 w l lw 4 lt rgb 'red'\n");
260.     fclose(scr);
261. }
262.
263. void
264. swap(struct dpoint* mas, int i, int j)
265. {
266.     struct dpoint tmp = mas[i];
267.     mas[i] = mas[j];
268.     mas[j] = tmp;
269. }
270.
271. void
272. sort(struct dpoint* mas, int n)
273. {
274.     for (int i = 0; i < n-1; i++){
275.         for (int j = i+1; j < n; j++){
276.             if (mas[j].cent < mas[i].cent){
277.                 swap(mas, i, j);
278.             }
279.         }
280.     }
281. }
282.
283. void
284. genPiData(struct dpoint* mas, int n, double* p, int tmp2)
285. {
286.     FILE* out = fopen("Pi.txt", "w");
287.     if(tmp2){
288.         for(int i = 0; i < n; i++){
289.             fprintf(out, "(%.3lf, %.3lf) | %.3lf\n", mas[i].a, mas[i].b, p[i]);
290.         }
291.     }else{
292.         for(int i = 0; i < n; i++){
293.             fprintf(out, "%.3lf | %.3lf\n", mas[i].cent, p[i]);
294.         }
295.     }
296.     fclose(out);
297.     system("subl Pi.txt");
298. }
299.
300. void
301. CheckNorm(struct dpoint* mas, int n, double a, int tmp2)
302. {
303.     double p[n];
304.     double np[n];
305.     double row[n];
306.     double x = 0.0;
307.     double D = 0.0;
308.     double sig = 0.0;
309.     int sum = 0;
```

```
310.     int mx = 0;
311.     for(int i = 0; i < n; i++){
312.         x += mas[i].cent * mas[i].n;
313.         D += mas[i].cent * mas[i].n * mas[i].cent;
314.         sum += mas[i].n;
315.         if(mas[i].n > mx){
316.             mx = mas[i].n;
317.         }
318.     }
319.     x /= sum;
320.     D /= sum;
321.     D -= x*x;
322.     sig = sqrt(D);
323.     printf("Xв = sum(XiNi)/N, (i=1,%ld) = %.3lf\n", n, x);
324.     printf("Dв = sum(Xi^2Ni)/N - Xв^2, (i=1,%ld) = %.3lf\n", n, D);
325.     printf("sig = sqrt(Dв) = %.3lf\n\n", sig);
326.     printf("Предположительное мат ожидание = %.3lf\n", x);
327.     printf("Предположительное ср.кв отклонение = %.3lf\n", sig);
328.     if(tmp2){
329.         p[0] = F0((mas[0].b-x)/sig) + 0.5;
330.         for(int i = 1; i < n-1; i++){
331.             p[i] = F0((mas[i].b-x)/sig) - F0((mas[i].a-x)/sig);
332.         }
333.         p[n-1] = 0.5 - F0((mas[n-1].a-x)/sig);
334.     }else{
335.         p[0] = F0((mas[1].cent - x)/sig) + 0.5;
336.         for(int i = 1; i < n-1; i++){
337.             p[i] = F0((mas[i+1].cent - x)/sig) - F0((mas[i].cent - x)/sig);
338.         }
339.         p[n-1] = 0.5 - F0((mas[n-1].cent - x)/sig);
340.     }
341.     double X2 = 0;
342.     for(int i = 0; i < n; i++){
343.         np[i] = p[i]*sum;
344.         row[i] = (mas[i].n-np[i])*(mas[i].n-np[i])/np[i];
345.         X2 += row[i];
346.     }
347.     printf("X2: %.5lf\n", X2);
348.     double sr = GetXiSq(a, n-3);
349.     printf("xi(table) %.5lf\n", sr);
350.     if(tmp2)
351.         genGistDataAbs(mas, n);
352.     genPolDataAbs(mas, n);
353.     if(tmp2)
354.         genScr(mas[0].a, mas[n-1].b, mx, tmp2);
355.     else
356.         genScr(mas[0].cent, mas[n-1].cent, mx, tmp2);
357.     if(tmp2)
358.         genScrN(x, sig, mas[0].a, mas[n-1].b);
359.     else
360.         genScrN(x, sig, mas[0].cent, mas[n-1].cent);
361.     system("gnuplot scr_1.txt");
362.     system("gnuplot scr_2.txt");
363.     genPiData(mas, n, p, tmp2);
364.     if(sr > 0){
365.         if(X2 <= sr){
366.             printf("X2 <= XiTable\n");
367.             printf("Гипотеза подтверждена\n");
368.         }else{
369.             printf("X2 > XiTable\n");
370.             printf("Гипотеза опровергнута\n");
371.         }
372.     }
```

```
372.     }else{
373.         printf("a не найдена в таблице или количество интервалов/точек меньше 4\n");
374.     }
375.     system("ristretto Abs.png");
376. }
377.
378. void
379. CheckBin(struct dpoint* mas, int n, double a, double cnt, int tmp2)
380. {
381.     double x = 0.0;
382.     double D = 0.0;
383.     double sig = 0.0;
384.     double np[n];
385.     double row[n];
386.     double X2 = 0;
387.     int sum = 0;
388.     int mx = 0;
389.     for(int i = 0; i < n; i++){
390.         x += mas[i].cent * mas[i].n;
391.         D += mas[i].cent * mas[i].n * mas[i].cent;
392.         sum += mas[i].n;
393.         if(mas[i].n > mx){
394.             mx = mas[i].n;
395.         }
396.     }
397.     x /= sum;
398.     D /= sum;
399.     D -= x*x;
400.     sig = sqrt(D);
401.     double p = x/cnt;
402.     printf("p = Xв/N, N - число испытаний\n");
403.     printf("Xв = sum(XiNi)/N, (i=1,%ld) = %.3lf\n", n, x);
404.     printf("Dв = sum(Xi^2Ni)/N - Xв^2, (i=1,%ld) = %.3lf\n", n, D);
405.     printf("sig = sqrt(Dв) = %.3lf\n", sig);
406.     printf("Pi = C_from_n_by_i * p^i * q^(N-i)\n");
407.     double q = -p + 1;
408.     for(int i = 0; i < n; i++){
409.         mas[i].pbin = tgamma(cnt+1)/(tgamma(cnt-i+1)*tgamma(i+1))*pow(p,i)*pow(q,cnt-i);
410.     }
411.     // reconstr(mas,n,start,finish);
412.     for(int i = 0; i < n; i++){
413.         np[i] = mas[i].pbin*sum;
414.         row[i] = (mas[i].n-np[i])*(mas[i].n-np[i])/np[i];
415.         X2 += row[i];
416.         printf("%.3lf\n", mas[i].pbin);
417.     }
418.     genPolDataAbs1(mas,n);
419.     genPolDataRel(mas,n);
420.     genScrBin(mas[0].cent,mas[n-1].cent);
421.     genScrBinT(mas[0].cent,mas[n-1].cent);
422.     system("gnuplot scr_bin.txt");
423.     system("gnuplot scr_bint.txt");
424.     system("ristretto bin.png");
425.     printf("X2: %.5lf\n", X2);
426.     double sr = GetXiSq(a,n-2);
427.     printf("xi(table) %.5lf\n", sr);
428.     if(sr > 0){
429.         if(X2 <= sr){
430.             printf("X2 <= XiTable\n");
431.             printf("Гипотеза подтверждена\n");
432.         }else{
433.             printf("X2 > XiTable\n");
```

```
434.         printf("Гипотеза опровергнута\n");
435.     }
436. }else{
437.     printf("a не найдена в таблице или количество интервалов/точек меньше 4\n");
438. }
439. }
440.
441. int
442. main(void)
443. {
444.     struct dpoint mas[100];
445.     printf("0 - ввод из файла\n");
446.     printf("1 - ручной ввод\n");
447.     int tmp;
448.     int tmp1;
449.     int tmp2;
450.     int n = 0;
451.     scanf("%d",&tmp1);
452.     printf("0 - массив данных\n");
453.     printf("1 - интервальный ряд\n");
454.     scanf("%d",&tmp2);
455.     if(tmp1){
456.         if(tmp2){
457.             printf("Введите количество интервалов\n");
458.         }else{
459.             printf("Введите количество чисел\n");
460.         }
461.         scanf("%d",&n);
462.         n = HandleInput(mas,n,tmp2);
463.     }else{
464.         n = generateMas(mas,"data.txt",tmp2);
465.     }
466.     sort(mas,n);
467.     printf("N is %d\n",n);
468.     printf("Проверка гипотезы о виде распределения по критерию Пирсона\n");
469.     printf("0 - о нормальном распределении\n");
470.     printf("1 - о биномиальном распределении\n");
471.     scanf("%d",&tmp);
472.     printf("Введите a:\n");
473.     double a;
474.     scanf("%lf",&a);
475.     if (tmp){
476.         double cnt;
477.         printf("Введите N - число испытаний\n");
478.         scanf("%lf",&cnt);
479.         CheckBin(mas,n,a,cnt,tmp2);
480.     }else{
481.         CheckNorm(mas,n,a,tmp2);
482.     }
483.     printf("Введите целое число для завершения\n");
484.     scanf("%d",&n);
485.     // delData(tmp);
486.     return 0;
487. }
```