УДК 004.415.3
Пех П.А., Середа А.О., Калінін Б.Ю., Лавренюк О.С.
Луцький національний технічний університет

ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ ДОСЛІДЖЕННЯ ПРОЦЕСУ ОБРОБЛЕННЯ ВИМОГ У СИСТЕМАХ МАСОВОГО ОБСЛУГОВУВАННЯ (СМО)

Пех П. А., Середа А. О., Калінін Б. Ю., Лавренюк О. С. Програмний комплекс для дослідження процесу оброблення вимог у системах масового обслуговування (СМО). *В статті запропоновано програмний комплекс засобами C++Builder для визначення на базі експериментальних даних параметрів інтенсивності та стабільності процесу оброблення вимог в СМО. Перевіряється також гіпотеза про експоненціальний чи Ерлангівський характер розподілу часу оброблення вимог за* χ^2 *–критерієм.*

Ключові слова: *C++Builder проект, процес оброблення вимог, експоненціальний розподіл, розподіл Ерланга.*

Пех П. А., Середа А. О., Калинин Б. Ю., Лавренюк А. С.Программный комплекс для исследования процесса обработки требований в системах массового обслуживания (СМО). *В статье предложено программный комплекс для определения на базе экспериментальных данных параметров интенсивности и стабильности процесса обработки требований в СМО. Проверяется также гипотеза об экспоненциальном или Эрланговском характере распределения времени обработки требований по* χ^2 *–критерию.*

Ключевые слова: *C++Builder проект, процесс обработки требований, экспоненциальное распределение, распределение Эрланга.*

Pekh P., Sererda A., Kalinin B., Lavrenjuk A. Software system for research of processing requirements in queuing systems (QS). *The paper proposed software system means C++Builder to determine the parameters of intensity and stability of processing requirements QS based on experimental data. Also checked the hypothesis of exponential or Erlang the distribution of time processing requirements in QS by* χ^2 *-test.*

Keywords: *C + + Builder project, the processing requirements, the exponential distribution, Erlang distribution*

Постановка задачі. Розглянемо випадковий процес оброблення вимог в СМО з метою дослідження його природи та параметрів. Тривалість часу оброблення вимог в СМО будемо вважати випадковою величиною *T*. Імовірність того, що випадкова величина *T* прийме значення, менше деякого довільного значення *t*, описується інтегральною функцією розподілу, тобто $F(t) = P(T < t)$.

Процес оброблення вимог, для якого властиві ознаки стаціонарності, ординарності і відсутності післядії, прийнято називати найпростішим [1,2]. Для такого процесу інтегральна функція розподілу часу оброблення вимог має експоненціальний вигляд :

$$F(t) = P(T < t) = 1 - e^{-\lambda t}.$$

Інтегральна функція розподілу імовірностей проміжків часу в процесі Ерланга Е [1,2] з інтенсивністю Λ та параметром стабільності *k* має вигляд:

$$F^k(t) = 1 - e^{-k\Lambda t} \sum_{n=0}^{k-1} \frac{(k\Lambda t)^n}{n!}$$

Задача, яка розв'язується у даній роботі, полягає в тому, щоб на базі експериментальних досліджень встановити вид розподілу процесу оброблення вимог і розрахувати його параметри. Інакше кажучи, необхідно, виходячи з аналізу експериментальних даних і встановлення кількісних параметрів інтенсивності та стабільності процесу оброблення вимог, сформулювати гіпотезу про вид розподілу цього процесу, і за фактичним значенням χ^2-критерію прийняти або відкинути її. Розроблення C++Builder проекту для вирішення цих завдань і становить предмет розгляду даної роботи.

Основна частина. Нехай за результатами експериментальних досліджень отримані n=100 значень часу оброблення вимог t_i в СМО (табл.1). Звичайно, і обсяг вибірки n, і значення спостережених даних t_i можуть бути і іншими. Для вирішення всіх вище зазначених завдань стосовно такого роду емпіричних даних нами розроблено C++Builder проект. Він складається з головної та восьми підлеглих форм (табл.2), кожна з яких вирішує ту чи іншу частину задачі. Кількість форм за потреби може бути збільшена.

Таблиця 1. Результати спостережень часу оброблення вимог t_i у СМО.

i	t_i	i	t_i	i	t_i	i	t_i	i	t_i
1	50	21	67	41	83	61	19	81	182
2	172	22	40	42	23	62	76	82	30
3	17	23	86	43	36	63	33	83	48
4	56	24	7	44	50	64	90	84	118
5	13	25	17	45	10	65	19	85	10
6	128	26	96	46	30	66	72	86	69
7	26	27	14	47	83	67	13	87	140
8	66	28	160	48	24	68	37	88	78
9	198	29	22	49	188	69	45	89	5
10	44	30	35	50	8	70	32	90	93
11	52	31	108	51	16	71	113	91	16
12	7	32	28	52	176	72	130	92	103
13	82	33	60	53	12	73	9	93	38
14	32	34	18	54	34	74	116	94	58
15	18	35	99	55	72	75	21	95	10
16	42	36	50	56	37	76	147	96	30
17	58	37	73	57	46	77	24	97	49
18	6	38	88	58	6	78	62	98	16
19	54	39	9	59	64	79	38	99	124
20	25	40	29	60	203	80	11	100	146

Таблиця 2. Найменування та призначення форм і відповідних їм файлів

№ з/п	**Найменування форми**	**Найменування файла**	**Призначення форми**
1	`Form1`	`MainFormUnit1`	Головна форма проекту
2	`Form2`	`GrupDanUnit2`	Групування даних
3	`Form3`	`TabErl1Unit3`	Наближення розподілу часу оброблення експоненціальним розподілом (k=1)
4	`Form4`	`GrafApr1Unit4`	Графік функції експоненціального розподілу
5	`Form5`	`TabErl13Unit5`	Наближення розподілу часу оброблення розподілом Ерланга (k=3)
6	`Form6`	`GrafErl3Unit6`	Графік функції розподілу Ерланга (k=3)
7	`Form7`	`TabErl15Unit7`	Наближення розподілу часу оброблення розподілом Ерланга (k=5)
8	`Form8`	`GrafErl5Unit8`	Графік функції розподілу Ерланга (k=5)
9	`Form9`	`ContentUnit9`	Зміст задач проекту

Вигляд головної форми показаний на рис.2. У ній використано компонент `MainMenu`, кожна команда якого (пункт меню) забезпечує перехід на відповідну підлеглу форму, а з кожної підлеглої форми можна повернутись лише на головну.

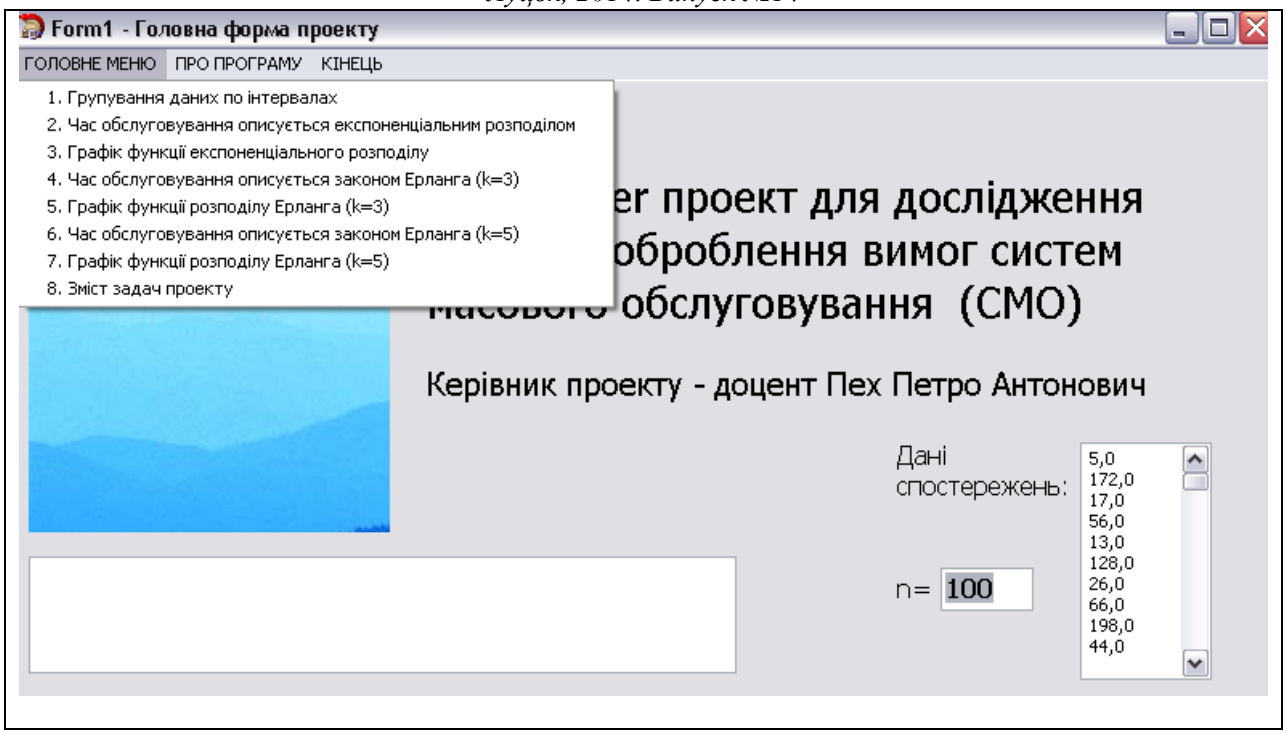


Рис.1 – Вигляд головної форми проекту

Зв'язування головної та всіх підлеглих форм досягається тим, що у файлі головної форми `MainFormUnit1.cpp` включені директиви з іменами файлів усіх підлеглих форм:

```
#include "MainFormUnit1.h"
#include "GrupDanUnit2.h"
#include "TabErl1Unit3.h"
#include "GrafErl1Unit4.h"
#include "TabErl3Unit5.h"
#include "GrafErl3Unit6.h"
#include "TabErl5Unit7.h"
#include "GrafErl5Unit8.h"
#include "ContentUnit9.h"
#include "ContentUnit9.h
```

а у всіх файлах підлеглих формах включена директива з іменем `MainFormUnit1.h` головної форми. Наприклад, для форми `Form2` це виглядає так:

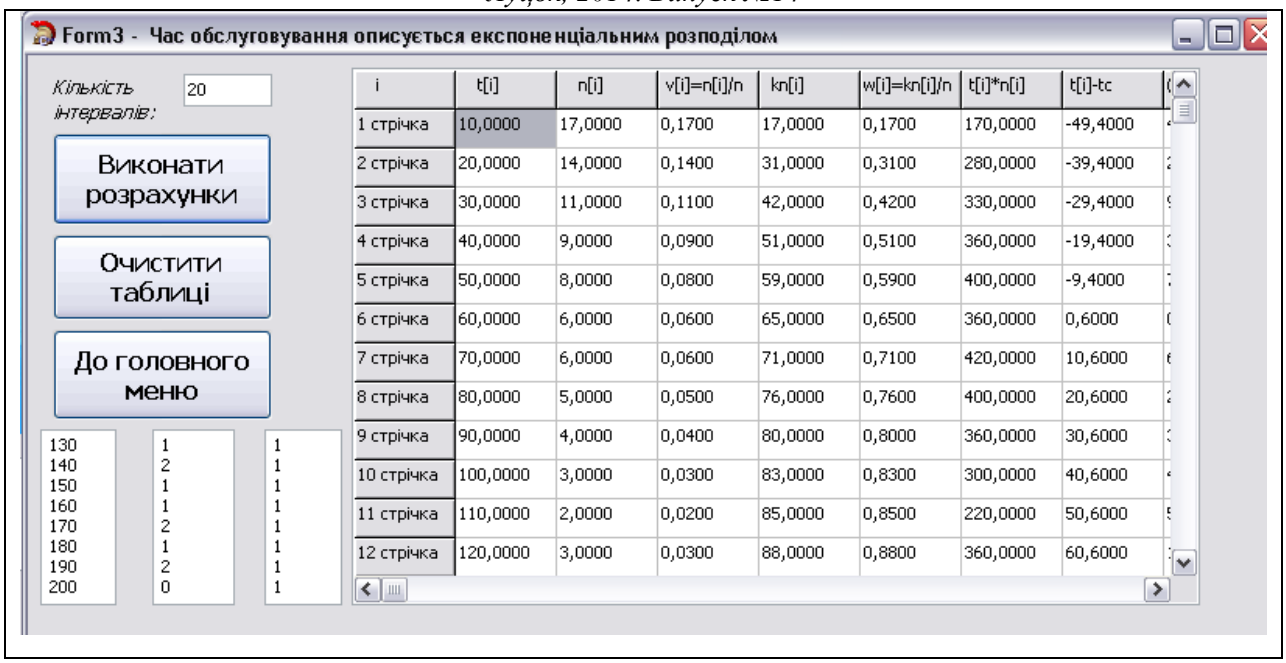
```
#include "GrupDanUnit2.h"
#include "MainFormUnit1.h"
```

З головною формою ми зв'язали також глобальний клас `data`

```
class DATA
 {public:
   int n;
   float a[250];
   float s[20];
 };
extern DATA data;
```

який містить масив `a[250]` для зберігання спостережених даних та масив `s[20]` для формування ряду розподілу `s[20]`. У такий спосіб ми досягли того, що масив вхідних даних буде досяжний в кожній підлеглій формі.

Розглянемо більш детально форму `Form3`, з допомогою якої здійснюється апроксимація експериментальних даних експоненціальним розподілом (рис.2). Обробник подій кнопки `Button1 (Виконати розрахунки)` форми `Form3` розпочинається описом змінних та масивів,



i	t[i]	n[i]	v[i]=n[i]/n	kn[i]	w[i]=kn[i]/n	t[i]*n[i]	t[i]-tc
1 стрічка	10,0000	17,0000	0,1700	17,0000	0,1700	170,0000	-49,4000
2 стрічка	20,0000	14,0000	0,1400	31,0000	0,3100	280,0000	-39,4000
3 стрічка	30,0000	11,0000	0,1100	42,0000	0,4200	330,0000	-29,4000
4 стрічка	40,0000	9,0000	0,0900	51,0000	0,5100	360,0000	-19,4000
5 стрічка	50,0000	8,0000	0,0800	59,0000	0,5900	400,0000	-9,4000
6 стрічка	60,0000	6,0000	0,0600	65,0000	0,6500	360,0000	0,6000
7 стрічка	70,0000	6,0000	0,0600	71,0000	0,7100	420,0000	10,6000
8 стрічка	80,0000	5,0000	0,0500	76,0000	0,7600	400,0000	20,6000
9 стрічка	90,0000	4,0000	0,0400	80,0000	0,8000	360,0000	30,6000
10 стрічка	100,0000	3,0000	0,0300	83,0000	0,8300	300,0000	40,6000
11 стрічка	110,0000	2,0000	0,0200	85,0000	0,8500	220,0000	50,6000
12 стрічка	120,0000	3,0000	0,0300	88,0000	0,8800	360,0000	60,6000

130	1	1
140	2	1
150	1	1
160	1	1
170	2	1
180	1	1
190	2	1
200	0	1

Рис.2 – Вигляд форми проекту, на якій подається розрахункова таблиця для перевірки гіпотези про експоненціальний (параметр стабільності k=1) розподіл часу оброблення вимог

формуванням ряду розподілу, конструюванням таблиці `StringGrid1`, вибраної нами для виведення всієї розрахункової таблиці:

```
void __fastcall TForm3::Button1Click(TObject *Sender)
{int nn;    //Кількість інтервалів групування даних
 int i;     //Робоча змінна
 int j;     //Робоча змінна
 int k;     //Робоча змінна
 int l;     //Робоча змінна
 int kr;    //Кількість рядків таблиці
 float t[30][30];    //Масив  t[25][10] для зберігання значень
                     //оброблення ряду розподілу
 float s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11; //Суми по кожному
                     //стовпчику масиву t[20][10]
 float d0,d1,d2,d,p,m,tc,dc,kc; //Допоміжні змінні
 float amin,amax;
 float h;                   //Крок таблиці
 //Таблиця StringGrid1 містить результати розрахунків параметрів
 //процесу обслуговування
 StringGrid1->RowCount=30; //Гранична кількість рядків таблиці
                           //StringGrid1
 StringGrid1->ColCount=14; //Гранична кількість стовпців таблиці
                           //StringGrid1
 amin=data.a[0];  //Найбільший елемент масиву a[i] спостережених
даних
 amax=data.a[0];  //Найменший елемент масиву a[i] спостережених даних
 for (i = 0; i< data.n; i++){
   if (data.a[i]<amin)
      amin=data.a[i];
   if (data.a[i]>amax)
      amax=data.a[i];
  }
 h=5;
 kr=amax/(2*h); //Кількість інтервалів групування даних
```

```
 nn=kr;
 Edit1->Text=IntToStr(kr);
 //Обнулення значень таблиці StringGrid1
  for (i = 0; i<kr+4; i++)
   for (j = 0; j<14; j++)
      StringGrid1->Cells[j+1][i+1]="0,0";

 //Таблиця StringGrid1 містить результати розрахунків параметрів
 //вхідного потоку
 StringGrid1->RowCount=kr+4; //Фактична кількість рядків таблиці
                             //StringGrid1
 // Іменування стрічок таблиці StringGrid1
  for (i=1; i<=kr+3; i++)
   StringGrid1->Cells[0][i] =IntToStr(i)+" стрічка";
 //Фактична кількість стовпців таблиці StringGrid1
  StringGrid1->ColCount=13;
 // Іменування стовпців таблиці StringGrid1
   StringGrid1->Cells[0][0] ="    i";
   StringGrid1->Cells[1][0] ="    t[i]";
   StringGrid1->Cells[2][0] ="    n[i]";
   StringGrid1->Cells[3][0] =" v[i]=n[i]/n";
   StringGrid1->Cells[4][0] ="  kn[i]";
   StringGrid1->Cells[5][0] ="w[i]=kn[i]/n";
   StringGrid1->Cells[6][0] =" t[i]*n[i]";
   StringGrid1->Cells[7][0] =" t[i]-tc";
   StringGrid1->Cells[8][0] ="((t[i]-tc)^2)/v[i]";
   StringGrid1->Cells[9][0] =" F(t[i])";
   StringGrid1->Cells[10][0] =" n*F(t[i])";
   StringGrid1->Cells[11][0] =" nт[i]";
   StringGrid1->Cells[12][0] ="((n[i]-nт[i])^2)/(nт[i])";
   for (i = 0; i<30; i++)
      for (j = 0; j<30; j++)
         t[i][j]=0.0;
  //Формування значень стовпчика t[i] таблиці StringGrid1
  for (i = 0; i<kr; i++){
     Memo1->Lines-
>Strings[i]=FloatToStrF((amin+h)*(i+1),ffFixed,4,0);
  }
  //Обнулення комірок масиву  s[i]
  for (i = 0; i < 30; i++) data.s[i]=0;
  //Формування значень масиву  s[i]
  for (i = 0; i <  data.n; i++)
   for (j=1; j<=kr; j++)
   if ((data.a[i]>=amin +(j-1)*2*h) && (data.a[i]<amin+j*2*h))
     data.s[j-1]=data.s[j-1]+1;

 //Занесення значень масиву s[i] в стовпчик n[i] таблиці StringGrid1
  for (j=1; j<=kr; j++){
        StringGrid1->Cells[2][j]=FloatToStrF(data.s[j],ffFixed,8,0);
        Memo2->Lines->Strings[j]=FloatToStrF(data.s[j],ffFixed,8,0);
  }
```

Далі формуються або розраховуються всі стовпчики таблиці `StringGrid1`:

1. Стовпчик `t[i]`, у якому записуються середини інтервалів можливих значень часу оброблення вимог:

```
  //Зчитування даних з компонента Memo1 у стовпчик t[i]
```

```
//та їх запис у масив t[i][j] та таблицю StringGrid1
s0=0;
for (i = 0; i<nn; i++){
 j=0;
 t[i][j]=(amin+h)*(i+1);
 s0=s0+t[i][j];
 StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,6,0);
};
 t[nn][0]=s0;
 t[nn+1][0]=s0/data.n;
```

2. Стовпчик `n[i]`, у якому записуються частоти, з якими зустрічалися значення `t[i]`:

```
//Зчитування даних з компонента Memo2 у стовпчик
// емпіричних частот n[i]
//та їх запис у масив t[i][j] та таблицю StringGrid1
s1=0;
for (i = 0; i<nn; i++) {
    j=1;
    t[i][j]=data.s[i];
    s1=s1+t[i][j];
    StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
 };
 t[nn][1]=s1;
 t[nn+1][1]=s1/data.n;
```

3. Стовпчик `v[i]=n[i]/n`, у якому записуються частості, з якими зустрічалися значення `t[i]`:

```
//Обчислення значень стовпчика частостей v[i]=n[i]/n
  s2=0;
  for (i = 0; i < nn; i++) {
     j=2;
     t[i][j]= t[i][1]/data.n;
     s2=s2+t[i][1]/data.n;
   }
  t[nn][2]=s2;
  t[nn+1][2]=s2/data.n;
```

4. Стовпчик `kn[i]`, у якому записуються кумулятивні частоти, з якими зустрічалися значення `t[i]`:

```
//Обчислення значень стовпчика kn[i] v[i]=n[i]/n    t[i]*n[i]
  t[0][3]=t[0][1];
  s3=t[0][1];;
  for (i = 1; i < nn; i++) {
     j=3;
     t[i][3]= s3+t[i][1];
     s3=t[i][3];
   }
  t[nn][3]=s3;
  t[nn+1][3]=s3/data.n;
```

5. Стовпчик `w[i]=kn[i]/n`, у якому записуються кумулятивні частості, з якими зустрічалися значення `t[i]`:

```
//Обчислення значень стовпчика w[i]=kn[i]/n
  s4=0;
  for (i = 0; i<nn; i++) {
     j=4;
     t[i][4]=t[i][3]/data.n;
     s4=s4+t[i][3]/data.n;
```

```
      StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
    };
    t[nn][4]=s1;
    t[nn+1][4]=s1/data.n;
```

6. Стовпчик `t[i]*n[i]`, у якому розраховуються добутки елементів відповідних стовпчиків:

```
//Обчислення значень стовпчика t[i]*n[i]
  s5=0;
  for (i = 0; i<nn; i++) {
      j=5;
      t[i][j]=t[i][0]*t[i][1];
      s5=s5+t[i][0]*t[i][1];
      StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
    };
    t[nn][5]=s5;
    t[nn+1][5]=s5/data.n;
    tc=t[nn+1][5];
```

7. Стовпчик `t[i]-tc`, у якому розраховуються різниці елементів стовпчика `t[i]` та середнього значення часу оброблення вимог tc, обчисленого у попередньому стовпчику:

```
//Обчислення значень стовпчика t[i]-tc
  s6=0;
  for (i = 0; i<nn; i++) {
      j=6;
      t[i][j]=t[i][0]-tc;
      s6=s6+t[i][0]-tc;
      StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
    };
    t[nn][6]=s6;
    t[nn+1][6]=s6/data.n;
```

8. Стовпчик `((t[i]-tc)^2)/v[i]`, у якому розраховуються елементи за даними двох відповідних попередніх стовпчиків:

```
//Обчислення значень стовпчика ((t[i]-tc)^2)/v[i]
  s7=0;
  for (i = 0; i<nn; i++) {
      j=7;
      t[i][j]=t[i][6]*t[i][6]*t[i][2];
      s7=s7+t[i][6]*t[i][6]*t[i][2];
      StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
    };
    t[nn][7]=s7;
    t[nn+1][7]=sqrt(s7);
    dc=t[nn][7];
    kc=tc*tc/dc;
    t[nn+2][7]=kc;
```

9. Стовпчик `F(t[i])`, у якому розраховуються значення теоретичних імовірностей

```
//Обчислення значень стовпчика F(t[i])
  s8=0;
  for (i = 0; i<nn; i++) {
      j=8;
      t[i][j]=1-exp(-t[i][0]/tc);
      StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
    };
```

10. Стовпчик `n*F(t[i])`, у якому розраховуються значення теоретичних частот

```
//Обчислення значень стовпчика n*F(t[i])
  s9=0;
  for (i = 0; i<=nn; i++) {
     j=9;
     t[i][j]=t[i][8]*data.n;
     StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
   };
```

11. Стовпчик `nт[i])`, у якому розраховуються значення накопичених теоретичних частот

```
//Обчислення значень стовпчика nт[i])
  t[0][10]=t[0][9];
  s10=t[0][9];
  for (i = 1; i<nn; i++) {
     j=10;
     t[i][j]=t[i][9]-t[i-1][9];
     s10=s10+ t[i][9]-t[i-1][9];
     StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
   };
  t[nn][10]=s10;
```

12. Стовпчик `((n[i]-nт[i])^2 )/nт[i]` , у якому розраховуються значення відхилень

```
//Обчислення значень стовпчика ((n[i]-nт[i])^2 )/nт[i]
  s11=0;
  for (i = 0; i<nn; i++) {
     j=11;
     t[i][j]=(t[i][1]-t[i][10])*(t[i][1]-t[i][10])/t[i][10];
     s11=s11+ (t[i][1]-t[i][10])*(t[i][1]-t[i][10])/t[i][10];;
     StringGrid1->Cells[j+1][i+1]=FloatToStrF(t[i][j],ffFixed,8,2);
   };
  t[nn][11]=s11;
```

Завершується програмний код виведенням на екран всіх вхідних та розрахованих величин. За значенням суми останнього стовпчика, а це є фактичне значення χ^2-критерію, яке порівнюється з гранично допустимим і приймається або відкидається гіпотеза про те, що процес оброблення вимог описується експоненціальним розподілом.

У проекті передбачено також побудову графіків емпіричної та теоретичної інтегральних функцій розподілу часу оброблення вимог у випадку k=1 (рис.3).

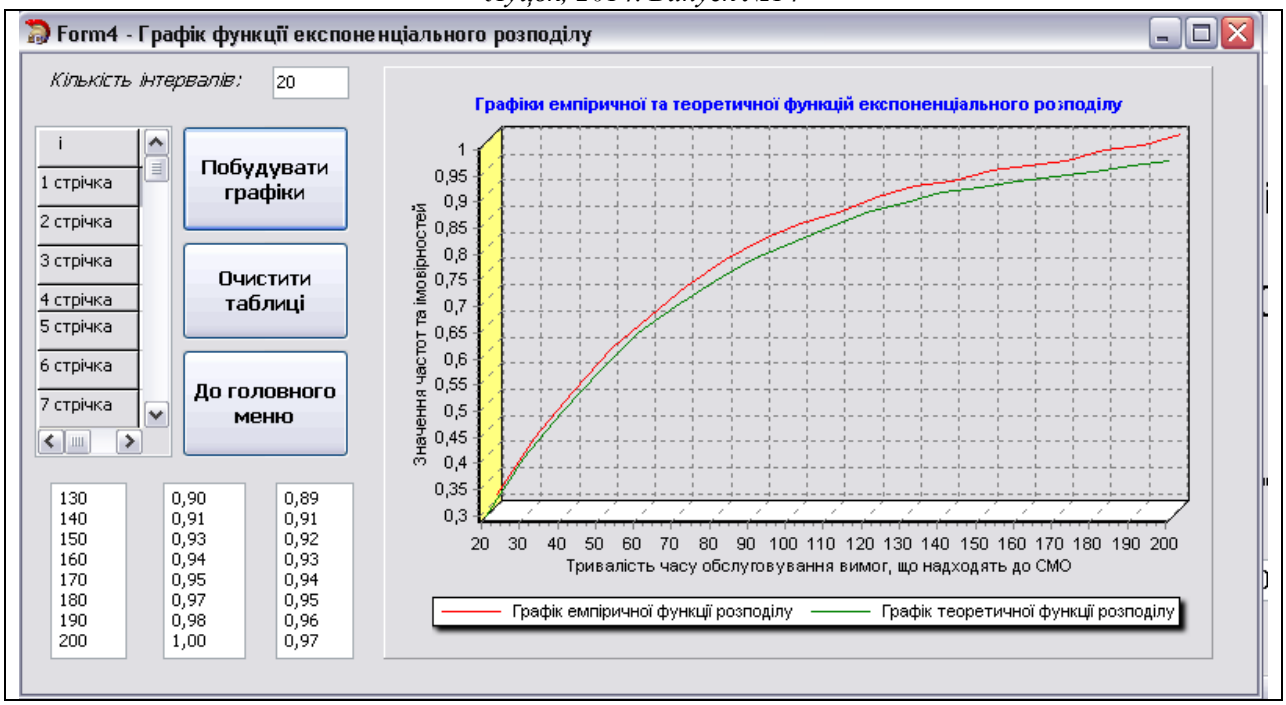


Рис.3 – Вигляд форми проекту, на якій будуються графіки емпіричної та теоретичної інтегральних функцій розподілу часу оброблення вимог у випадку k=1

Висновки

В роботі розроблено C++Builder проект, за допомогою якого засобами сучасних інформаційних технологій розв'язується актуальна задача дослідження процесу оброблення вимог у СМО. Ця задача полягає у тому, щоб, виходячи з аналізу експериментальних даних і встановлення кількісних параметрів інтенсивності та стабільності процесу оброблення вимог, сформулювати гіпотезу про вид розподілу, і за фактичним значенням χ^2-критерію прийняти або відкинути її.

1. Д.Л. Дудюк та інші. Моделювання та оптимізація об'єктів і систем керування: Навчальний посібник для студентів вищих навчальних закладів. –К.:ІЗМН, 1998.-248 с.
2. Дудюк Дмитро та інші. Елементи теорії автоматичних ліній: Навчальний посібник для студентів вищих навчальних закладів. –Київ-Львів:ІЗМН, 1998.-192 с.