**Міністерство освіти і науки, молоді та спорту України**
**Міжнародний економіко-гуманітарний університет**
**імені академіка Степана Дем'янчука**
**Факультет кібернетики**
**Кафедра математичного моделювання**

**Ціхоцька Катерина Валентинівна**

# ЗАСОБИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ НА МОВІ ПРОГРАМУВАННЯ С++ ПРИ ВИВЧЕННІ СКЛАДНИХ ЕКОНОМІЧНИХ ЯВИЩ


**Науковий керівник:**
**Р.М.Літнарович, доцент,**
**кандидат технічних наук**

**Рівне – 2012**

**УДК 004.42**
**Ціхоцька К.В.** Засоби комп'ютерного моделювання на мові програмування С++ при вивченні складних економічних явищ. Монографія. Науковий керівник Р.М.Літнарович. МЕГУ, Рівне, 2011.- 99 с.
. Робота виконана на кафедрі математичного моделювання Міжнародного економіко-гуманітарного університету імені академіка Степана Дем'янчука
Рецензенти:В.Г.Бурачек, доктор технічних наук, професор
В.О.Боровий, доктор технічних наук, професор
…….Є.С.Парняков, доктор технічних наук, професор
Відповідальний за випуск: Й.В.Джунь, доктор фіз.-мат. наук, професор

Об'єктом дослідження є сучасні методи та засоби комп'ютерного моделювання, а також мови програмування, що надають можливість розробляти власні програми для побудови математичних моделей.

**Ключові слова**: математична модель, мова програмування С++, економіка, програмний продукт.

Объектом исследования являются современные методы и средства компьютерного моделирования, а также языки программирования, которые предоставляют возможность разрабатывать собственные программы для построения математических моделей.

**Ключевые слова**: математическая модель, язык программирования С++, экономика, программный продукт.

A research object are modern methods and facilities of computer design, and also programming languages that give possibility to develop the own programs for the construction of mathematical models.

Keywords: mathematical model, programming of C++, economy, software product.

**ЗМІСТ**

# ВСТУП

З середини XX століття в найрізноманітніших областях людської діяльності стали широко застосовувати математичні методи і ЕОМ. Виникли такі нові дисципліни, як «математична економіка», «математична хімія», «математична лінгвістика» і т.і., що вивчають математичні моделі відповідних об'єктів і явищ, а також методи дослідження цих моделей.

Математична модель – це наближений опис якогось класу явищ або об'єктів реального світу на мові математики. Основною метою моделювання є дослідження цих об'єктів і прогнозування результатів майбутніх спостережень. Однак, моделювання – це ще й метод пізнання оточуючого світу, що дає можливість керувати ним.

Математичне моделювання та пов'язаний з ним комп'ютерний експеримент незамінні в тих випадках, коли натурний експеримент неможливий або викликає багато труднощів по тим чи іншим причинам. Наприклад, неможливо поставити натурний експеримент в історії, щоб перевірити, «що було б, якби…». Неможливо перевірити правильність тієї чи іншої космологічної теорії. В принципі, це можливо, але навряд чи розумно, поставити експеримент по розповсюдженню якоїсь хвороби, або створити ядерний вибух, щоб вивчити його наслідки. Однак, все це цілком можливо зробити побудувавши математичні моделі явищ, що підлягають вивченню.[1]

Відтоді як економіка стала самостійною наукою, дослідники намагаються спрогнозувати ту чи іншу ситуацію, передбачити майбутні значення економічних показників, запропонувати інструменти зміни ситуації в бажаному напрямку.

Одним з головних напрямів розвитку економіки є застосування ефективних наукових методів аналізу й оптимізації складних економіко-організаційних систем. Серед наукових методів, які застосовуються в економіці, науці і техніці, особливе місце займають методи моделювання.

Найзагальнішим методом досліджень, та таким, що найбільше використовується в науці, зокрема в кібернетиці і економіці, можна назвати процес створення математичної моделі, тобто, математичне моделювання. Неможливо уявити собі сучасну науку без широкого застосування математичного моделювання, суть якого полягає в заміні досліджуваного об'єкта його «образом» – математичною моделлю – і подальшому вивченні моделі за допомогою відповідних обчислювально-логічних алгоритмів на ЕОМ.

Робота не з моделлю об'єкта дає можливість без істотних затрат і відносно швидко дослідити його властивості і поведінку у різних ситуаціях. Обчислювальні (комп'ютерні , стимуляційні, імітаційні) експерименти з моделями об'єктів дозволяють вивчати об'єкти з достатньою повнотою, недоступною для чисто теоретичних досліджень.[2]

В економіці математичне моделювання застосовують при вивченні складних економічних явищ. Значно спрощує це завдання наявність сучасного програмного забезпечення. Проте більшість сучасних комп'ютерних програм для статистичної обробки даних дають надто узагальнені висновки і не заглиблюються в результати побудованих моделей в контексті конкретної галузі.

***Метою роботи*** *є написання програми, що дозволяє не лише побудувати модель економічного явища, а й обчислити такі важливі економічні характеристики, як прибуток, еластичність попиту, товарообіг, а також їх критичні значення.*

***Робота є актуальною*** *тому, що більшість сучасних програм для статистичної обробки даних дають надто узагальнені висновки і не заглиблюються в результати побудованих моделей в контексті конкретної галузі.*

***Наукова новизна*** *полягає в тому, що розроблена програма на ряду із математичними характеристиками змодельованого явища надає також основні економічні характеристики. Це дає змогу більш предметно оцінити результати проведених досліджень.*

***Об'єктом дослідження*** *є сучасні методи та засоби комп'ютерного моделювання, а також мови програмування, що надають можливість розробляти власні програми для побудови математичних моделей.* В результаті даного дослідження мною

було обрано середовище розробки програмного забезпечення Microsoft Visual Studio 2010 як одне з найсучасніших середовищ. Воно надає зручні інструменти для розробки сучасних програм, підвищення ефективності їх виконання та своєчасного виявлення помилок. Крім того засоби створення проекту, що вбудовані в середовище Visual C++ 2010, дозволяють автоматично створювати основу коду для широкого діапазону різноманітних прикладних програм.

***Методологічною основою*** дисертації є Інтегроване середовище розробки (Integrated Development Environment – IDE), яке надається разом із середовищем Visual C++ 2010 – це повністю самодостатнє середовище, призначене для створення, компіляції, компонування і перевірки програм на мові C++. Це середовище включає в себе велику кількість повністю інтегрованих інструментів, призначених для полегшення написання програм.

Список фундаментальних складових середовища розробки Visual C++ 2010, що надаються IDE, включає в себе редактор, компілятор, компоновщик і бібліотеки. Це основні інструменти, що є необхідними для розробки, і відповідно були використані мною для написання програми.

***Практична значимість*** *роботи полягає в створенні повноцінного програмного продукту, що може використовуватись на підприємствах що займаються торгівлею.*

***Апробація роботи.*** Окремі розділи дисертації доповідались і отримали одобрення на наукових конференціях студентів і аспірантів у 2010 і 2011 роках, а також на науковому семінарі кафедри математичного моделювання.

***Публікації.*** Основні положення дисертації опубліковані в монографії автора: Ціхоцька К. В. Засоби комп'ютерного моделювання на мові програмування C++ при вивченні складних економічних явищ. Монографія. Науковий керівник Р.М.Літнарович. МЕГУ, Рівне, 2012. - с.

***Основні положення дисертації, що виносяться на захист:***

- теоретичні основи побудови економіко-математичної моделі по способу найменших квадратів з повною оцінкою точності її елементів;

- аналіз середовища програмування Microsoft Visual Studio 2010;
- розробка допоміжних бібліотек;
- візуальне оформлення програми.

***Структура і об'єм роботи.*** Магістерська дисертація складається із вступу, трьох розділів, розбитих на підрозділи, висновків і списку використаної літератури та додатків. Обсяг дисертації 86 сторінок, 15 рис. Список використаних джерел із 13 найменувань, в тому числі 4 складають Інтернет-джерела.

# РОЗДІЛ 1. ПОБУДОВА МОДЕЛІ МЕТОДОМ НАЙМЕНШИХ КВАДРАТІВ

## 1.1. Теоретичні основи

Математична модель результатів економічного експерименту виражається квадратичним поліномом виду

$$y = ax^2 + bx + c, \qquad (1.1.1)$$

або

$$y = ax^2 + bx, \qquad (1.1.2)$$

або

$$y = ax^2. \qquad (1.1.3)$$

Утворимо систему n початкових рівнянь з трьома невідомими

$$\begin{cases} ax_1^2 + bx_1 + c - y_1 = \varepsilon_1, \\ ax_2^2 + bx_2 + c - y_2 = \varepsilon_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ ax_n^2 + bx_n + c - y_n = \varepsilon_n. \end{cases} \qquad (1.1.4)$$

Застосувавши вимогу найменших квадратів

$$[\varepsilon\varepsilon] = [(ax_i^2 + bx_i + c - y_i)^2] = \min,$$

будемо мати три нормальних рівняння з трьома невідомими:

$$\begin{aligned} &a[x^4] + b[x^3] + c[x^2] = [x^2 y], \\ &a[x^3] + b[x^2] + c[x] = [xy], \\ &a[x^2] + b[x] + c \cdot n = [y]. \end{aligned} \qquad (1.1.6)$$

Символом [ ] позначена сума по Гаусу.

Рішення цих рівнянь приводить до визначення невідомих коефіцієнтів

$$a=\frac{[x^2y](n[x^2]-\overset{A}{[x][x]})+[xy](\overset{B}{[x][x^2]}-n[x^2])+[y](\overset{C}{[x][x^3]}-[x^2][x^2])}{n(\overset{D}{[x^2][x^4]}-[x^3][x^3])+[x](\overset{E}{[x^2][x^3]}-[x][x^4])+[x^2](\overset{C}{[x][x^3]}-[x^2][x^2])},$$

$$b=\frac{[x^2y](n[x^2]-\overset{B}{n[x^3]})+[xy](n[x^4]-\overset{E}{[x^2][x^2]})+[y](\overset{E}{[x^2][x^3]}-[x][x^4])}{n(\overset{D}{[x^2][x^4]}-[x^3][x^3])+[x](\overset{E}{[x^2][x^3]}-[x][x^4])+[x^2](\overset{C}{[x][x^3]}-[x^2][x^2])}, \qquad (1.1.7)$$

$$c=\frac{[x^2y](\overset{C}{[x][x^3]}-[x^2][x^2])+[xy](\overset{E}{[x^2][x^3]}-[x][x^4])+[y](\overset{D}{[x^2][x^4]}-[x^3][x^3])}{n(\overset{D}{[x^2][x^4]}-[x^3][x^3])+[x](\overset{E}{[x^2][x^3]}-[x][x^4])+[x^2](\overset{C}{[x][x^3]}-[x^2][x^2])}.$$

Зрівноважене рівняння буде

$$\varphi(x) = ax^2 + bx + c. \qquad (1.1.8)$$

Склавши різниці $\varphi(x_i) - y_i = \varepsilon_i$ , де $y_i$ – визначені значення, а $\varepsilon_i$ – відхилення визначених значень $y_i$ від їх ймовірних значень, отримаємо перше представлення про точність виконаних робіт.

Контрольна формула обчислення коефіцієнтів легко виводиться із (1.1.4) (1.1.6) і умови $[\varepsilon\varepsilon] = \min$.

$$[y^2] - a[yx^2] - b[yx] - c[y] = [\varepsilon\varepsilon] \qquad (1.1.9)$$

Замітимо, що корені рівняння (1.1.8) не виражаються простими величинами, як для випадку прямолінійної залежності.

Обчислення коефіцієнтів b і c можна значно спростити, якщо виразити їх із (1.1.6) через коефіцієнт a:

$$b = \frac{n[yx] - [y][x]}{n[x^2] - [x][x]} + a\frac{[x][x^2] - n[x^3]}{n[x^2] - [x][x]}, \qquad (1.1.10)$$

$$c = \frac{[y][x^2] - [x][xy]}{n[x^2] - [x][x]} + a\frac{[x][x^3] - [x^2][x^2]}{n[x^2] - [x][x]}. \qquad (1.1.11)$$

Введемо позначення:

$$A=n[x^2]-[x][x]; B=[x][x^2]-n[x^2]; C=[x][x^3]-[x^2][x^2];$$
$$D=[x^2][x^4]-[x^3][x^3]; E=[x^2][x^3]-[x][x^4]; F=n[x^4]-[x^2][x^2]. \qquad (1.1.12)$$

Тоді формули (1.1.7) будуть мати наступний вигляд

$$a = \frac{[x^2 y]\cdot A + [xy]\cdot B + [y]\cdot C}{n\cdot D + [x]\cdot E + [x^2\cdot C]},$$

$$b = \frac{[x^2 y]\cdot B + [xy]\cdot F + [y]\cdot E}{n\cdot D + [x]\cdot E + [x^2]\cdot C}, \qquad (1.1.13)$$

$$c = \frac{[x^2 y]\cdot C + [xy]\cdot E + [y]\cdot D}{n\cdot D + [x]\cdot E + [x^2]\cdot C}.$$

Підставивши (1.1.10), (1.1.11) у (1.1.8), отримаємо

$$\varphi(x) = ax^2 + \left\{\frac{n[yx] - [x][y] + a([x][x^2] - n[x^3])}{n[x^2] - [x][x]}\right\}x + \frac{[y][x^2] - [x][yx] + a([x][x^3] - [x^2][x^2])}{n[x^2] - [x][x]}. \qquad (1.1.14)$$

Отримана крива завжди проходить через точки

$$\left(0; \frac{a[x][x^3] - [x^2][x^2] + [y][x^2] - [x][yx]}{n[x^2] - [x][x]}\right),$$

$$\left(\frac{[x]}{n}; a\frac{[x][x] - n[x^2]}{n^2} + \frac{[yx]}{x}\right), \qquad (1.1.15)$$

$$\left(\frac{[x2]}{[x]}; a\frac{[x2][x2] - [x][x3]}{[x][x]} + \frac{[yx]}{[x]}\right).$$

## 1.2. Поступальне переміщення початку координат в точку арифметичної середини

Всі приведені формули суттєво спрощуються, якщо перемістити початок координатної системи в точку

$$\left(\frac{[x]}{n}; \frac{[y]}{n}\right).$$

Нові координати будуть виражатися через старі наступним чином

$$x' = x - \frac{[x]}{n},$$
$$y' = y - \frac{[y]}{n}. \qquad (1.2.2)$$

При цьому $[x'] = [y'] = 0$

Формули (1.1.7) перетворюються до виду

$$a' = \frac{n([x'^2 \cdot y'][x'^2] - [x'y'][x'^3])}{n([x'^2][x'^4] - [x'^3][x'^3]) - [x'^2][x'^2][x'^2]},$$
$$b' = \frac{n([x' \cdot y'][x'^4] - [x'^2y'][x'^3]) - [x'y'][x'^2][x'^2]}{n([x'^2][x'^4] - [x'^3][x'^3]) - [x'^2][x'^2][x'^2]}, \qquad (1.2.3)$$
$$c' = \frac{[x'^2]([x'y'][x'^3] - [x'^2y'][x'^2])}{n([x'^2][x'^4] - [x'^3][x'^3]) - [x'^2][x'^2][x'^2]}.$$

Коефіцієнт b` і c` можна виразити також і через коефіцієнт a`

$$b' = \frac{[x'y'] - a'[x'^2]}{[x'^2]}, \quad c' = \frac{a'[x'^2]}{n}. \qquad (1.2.4)$$

Ймовірніша крива має вигляд

$$\varphi(x') = a'x'^2 + b'x' + c'.$$

Або в початкових координатах

$$\varphi(x) - \frac{[y]}{n} = a'\left(x - \frac{[x]}{n}\right)^2 + b'\left(x - \frac{[x]}{n}\right) + c'. \qquad (1.2.6)$$

Співставленням (1.2.5) з (1.1.1) находимо

$$a = a',$$
$$b = b' - 2a'\frac{[x]}{n}, \qquad (1.2.7)$$
$$c = c' + a'\left(\frac{[x]}{n}\right)^2 - b'\left(\frac{[x]}{n}\right) + \frac{[y]}{n}.$$

Крива (1.2.5) проходить через точки

$$\left(0;-a'\frac{[x'^2]}{n}\right),$$

$$\left(-\frac{[x]}{n};a'\left(\frac{[x]}{n}\right)^2-b'\frac{[x]}{n}+c'\right), \qquad (1.2.8)$$

$$\left(\frac{[x]}{n};a'\left(\frac{[x]}{n}\right)^2+b'\frac{[x]}{n}+c'\right).$$

Обчислення контролюються формулою

$$[y^2]-\left([x'^2y']+[x'^2]\frac{[y]}{n}+[x'y']\frac{[x]}{n}+\left(\frac{[x]}{n}\right)^2[y]\right)a- \qquad .(1.2.9)$$

$$-\left([x'y']+\frac{[x][y]}{n}b-c[y]\right)b-c[y]=[\varepsilon\varepsilon]$$

Якщо одна із величин [x`], [y`] або обидві точно не рівні нулю, а є невеликою величиною нехтувати якою не можна, то a`, b`, c` обраховуються за формулами (1.1.13). При цьому переваги перетворень повністю не використовуються, але все ж досягаються полегшення у обчислювальних роботах.

## 1.3. Знаходження поправок до наближених значень коефіцієнтів

Один із прийомів раціоналізації обробки матеріалів полягає в тому, що знаходяться не повні значення коефіцієнтів, а поправки до наближених їх значень $a_l$, $b_l$, $c_l$, отриманих на основі даних експериментальних визначень.

В даному випадку будемо мати систему початкових рівнянь

$$y_i - y_{li} = (a-a_l)y_i^{\,2} + (b-b_l)y_i + (c-c_l)\,, \quad (1.3.1)$$

де

$$y_{li} = a_l x_i^{\,2} + b_l x_i + c_l\,, \qquad (1.3.2)$$

або

$$\begin{aligned}
&\delta a x_1^{\,2} + \delta b x_1 + \delta c - \delta y_1 = \varepsilon_1,\\
&\delta a x_2^{\,2} + \delta b x_2 + bc - \delta y_2 = \varepsilon_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\delta a x_n^{\,2} + \delta b x_n + \delta c - \delta y_n = \varepsilon_n.
\end{aligned} \qquad (1.3.3)$$

При цьому система нормальних рівнянь має вигляд:

$$\begin{aligned}
&\delta a[x^4] + \delta b[x^3] + \delta c[x^2] - [\delta y x^2] = 0,\\
&\delta a[x^3] + \delta b[x^2] + \delta c[x] - [\delta y x] = 0,\\
&\delta a[x^2] + \delta b[x] + [c]n - [\delta y] = 0.
\end{aligned} \qquad (1.3.4)$$

Звідси значення поправок δa, δb, δc будуть

$$\begin{aligned}
\delta a &= \frac{[x^2\delta y]([x][x^2]-[x][x])+[x\delta y]([x][x^2])-n[x^3]+[\delta y]([x][y^3]-[x^2][x^2])}{n([x^2][x^4]-[x^3][x^3])+[x]([x^2][x^3]-[x][x^4])+[x^2]([x][x^3]-[x^2][x^2])},\\
\delta b &= \frac{[x^2\delta y]([x][x^2]-n[x^3])+[x\delta y](n[x^4]-[x^2][x^2])+[\delta y]([x^2][x^3]-[x][x^4])}{n([x^2][x^4]-[x^3][x^3])+[x]([x^2][x^3]-[x][x^4])+[x^2]([x][x^3]-[x^2][x^2])},\\
\delta c &= \frac{[x^2 y]([x][x^3]-[x^2][x^2])+[x\delta y]([x^2][x^2])-[x][x^4]+[\delta y]([x^2][x^4]-[x^3][x^3])}{n([x^2][x^4]-[x^3][x^3])+[x]([x^2][x^3]-[x][x^4])+[x^2]([x][x^3]-[x^2][x^2])}.
\end{aligned} \qquad (1.3.5)$$

Поправки δb і δc можна підрахувати і по більш простим формулам

$$\delta b = \frac{n[\delta y \cdot x]-[x][\delta x]}{n[y^2]-[x][x]} + \delta a \frac{[x][x^2]-n[x^3]}{n[x^2]-[x][x]}, \qquad (1.3.6)$$

$$\delta c = \frac{[\delta y][x^2]-[y][\delta y \cdot x]}{n[x^2]-[x][x]} + \delta a \frac{[x][x^3]-[x^2][x^2]}{n[x^2]-[x][x]}. \qquad (1.3.7)$$

Додаючи поправки δa, δb, δc до наближених коефіцієнтів $a_l$, $b_l$, $c_l$, отримаємо кінцеві значення шуканих коефіцієнтів:

$$\begin{aligned}
a &= a_l + \delta a,\\
b &= b_l + \delta b,\\
c &= c_l + \delta c.
\end{aligned} \qquad (1.3.8)$$

Контрольна формула буде

$$[\delta y^2] - \delta a[x^2\delta y] - \delta b[x\delta y] - \delta c[\delta y] = [\varepsilon\varepsilon]. \qquad (1.3.9)$$

## 1.4. Обробка матеріалів при рівновідстоячих значеннях аргументів

Позначивши інтервал аргументу через x і число визначень n за допомогою формул

$$X = x_i - x_1,$$

$$[X] = \chi \frac{n[n-1]}{2},$$

$$[X^2] = \chi 2 \frac{n(n-1)(2n-1)}{6},$$

$$[X^3] = \chi^3 \frac{n^2(n-1)^2}{4},$$

.....

$$[X^i] = \chi i \sum_0^n [k-1]^i,$$

$$[Xy] = \chi[(k-1)y_k],$$

$$[Xy]^2 = \chi[(k-1)y_k{}^2],$$

$$[Xy^i] = \chi[(k-1)y_k{}^i],$$

$$[X^2 y] = \chi^2[(k-1)^2 y_k],$$

$$[X^3 y] = \chi^3[(k-1)^3 y_k],$$

$$[X^i y^i] = \chi^i[(k-1)^i y^i{}_k]. \qquad (1.4.1)$$

формули (1.2.3) приводять до виду

$$a' = \frac{180\left\{(n-1)[(k-1)^2 y_k] - [(k-1)y_k] - \frac{(n-1)(n-2)}{6}[y]\right\}}{\chi^2 n(n-1)(n+1)(n-2)(n+2)},$$

$$b' = \frac{180\left\{(n-1)[(k-1)^2 y_k] - \frac{(2n)-1(n+2)}{15}[(k-1)y_k] + \frac{(n-1)(2n-1)(n-2)}{10}[y]\right\}}{\chi n(n-1)(n+1)(n-2)(n+2)}, \quad (1.4.2)$$

$$c' = \frac{3\left\{6(2n-1)[(k-1)y_k] - 10[(k-1)^2 y_k] - (3n^2 - 3n + 2)[y]\right\}}{n(n+1)(n+2)}$$

Підставляючи у вихідне рівняння (1.1.7) отримані значення коефіцієнтів, визначених при умові, що $x_i` = x_i - x_1$, знайдемо

$$\varphi(x) = a'x'^2 + b'x' + c' = a'(x - x1) + b'(x - x1) + c' = \\ = a'x^2 + (b' - 2a'x)x + (a'x2 - b'x + c') \quad (1.4.3)$$

Звідки

$$a = a', \\ b = b' - 2a'x_1, \quad (1.4.4) \\ c = a'x^2 - b'x_1 + c'.$$

Контрольна формула буде

$$[y^2] - a(x^2[y] + 2x_1\chi[(k-1)y_k] + \chi^2[(k-1)^2 y_k]) - b(x_1[y] + \chi[(k-1)y_k]) - c[y] = [\varepsilon\varepsilon]. \quad (1.4.5)$$

Можна добитися і подальших спрощень, виразивши визначені значення функції через кінцеві різниці першого порядку.

Якщо прийняти $y_1 = 0$, то формули спрощуються

$$[y] = [(k-1)\Delta y_k],$$

$$[xy] = \chi\left\{\frac{n(n-1)}{2}[\Delta y] - \frac{[k(k-1)(2k-1)\Delta y_k]}{6}\right\}. \quad (1.4.6)$$

Аналогічно

$$[x^2 y] = \chi^2\left\{\frac{n(n-1)(2n-1)}{6}[\Delta y] - \frac{[k(k-1)(2k-1)\Delta y_k]}{6}\right\}. \quad (1.4.7)$$

Підстановка значень цих різниць, а також сум [x], $[x^2]$, $[x^3]$ при рівновідстоячих значеннях аргументу дає вираз шуканих коефіцієнтів через кінцеві різниці першого порядку

$$a' = \frac{30[k(n-k)(n-2k)\Delta y_k]}{\chi^2 n(n-1)(n+1)(n-2)(n+2)},$$

$$b' = \frac{6(n-1)(n+2)[(n-k)\Delta y_k] - 5(n-1)[k(n-k)(n-2k)\Delta y_k]}{\chi n(n-1)(n+1)(n-2)(n+2)},$$

$$c' = \frac{(n+1)(n+2)[(n-k)\Delta y_k] - 3(n+2)[k(n-k)(n-2k)\Delta y_k]}{n(n+1)(n+2)} - \quad (1.4.8)$$

$$- \frac{5[k(n-k)(n-2k)\Delta y_k]}{n(n+1)(n+2)}.$$

Після обчислення a`, b`, c` будемо мати рівняння

$$\varphi(x') = a'x^2 + b'x + c', \quad (1.4.9)$$

або, повертаючись до початкового рівняння,

$$\varphi(x) = a'(x - x_1)^2 + b'(x - x_1) + c' + y_1. \quad (1.4.10)$$

Звідси

$$a = a',$$
$$b = b' - 2a'x_1, \quad . \quad (1.4.11)$$
$$c = c' + a'x^2 - b'x_1 + y_1.$$

Правильність обчислень контролюється за допомогою формули

$$[y^2] - [y]\left\{ax_1^2 + bx_1 + c + a\chi(n-1)\left(x_1 + \chi\frac{2n-1}{6} + 6\chi\frac{n-1}{2}\right)\right\} - \quad (1.4.12)$$

$$- \chi[k(n-k)\Delta y](ax_1 + \frac{b}{2}) - a\chi\left(\frac{n-1}{2}[k(n-k)\Delta y_k] - \frac{[kn-k)(n-2k)\Delta y_k]}{}\right) = [\varepsilon\varepsilon].$$

## 1.5. Побудова математичних моделей в окремих випадках

### 1.5.1. Побудова математичної моделі квадратичною залежністю, у випадку, якщо c=0

При цьому

$$y = ax^2 + bx. \quad (1.5.1)$$

Початкові рівняння для цього випадку будуть

$$
\begin{aligned}
&ax_1^{\,2} + bx_1 - y_1 = \varepsilon_1,\\
&ax_2^{\,2} + bx_2 - y_2 = \varepsilon_2,\\
&\text{.....}\\
&ax_n^{\,2} + bx_n - y_n = \varepsilon_n.
\end{aligned}
\qquad (1.5.2)
$$

Невідомі коефіцієнти a і b визначаються із рішення двох нормальних рівнянь

$$
\begin{aligned}
&a[x^4] + b[x^3] - [x^2 y] = 0,\\
&a[x^3] + b[x^2] - [xy] = 0.
\end{aligned}
\qquad (1.5.3)
$$

із яких

$$
\begin{aligned}
a &= \frac{[yx^2][x^2] - [xy][x^3]}{[x^4][x^2] - [x^3][x^3]},\\
b &= \frac{[yx][x^4] - [x^2 y][x^3]}{[x^4][x^2] - [x^3][x^3]}.
\end{aligned}
\qquad (1.5.4)
$$

Коефіцієнт b можна підрахувати і за формулою

$$
b = \frac{[xy]}{[x^2]} - a\frac{[x^3]}{[x^2]}. \qquad (1.5.5)
$$

Контрольна формула має вигляд

$$
[y^2] - a[x^2 y] - b[xy] = [\varepsilon\varepsilon]. \qquad (1.5.6)
$$

Шукане рівняння запишеться у вигляді

$$
\varphi(x) = ax^2 - a\frac{[x^3]}{[x^2]}x + \frac{[xy]}{[x^2]}x. \qquad (1.5.7)
$$

Зрівноважена крива проходить через точки з координатами

$$
(0;\,0),\ \left(\frac{[x^3]}{[x^2]};\frac{[xy][x^3]}{[x^2][x^2]}\right),\left(\frac{[x^4]}{[x^3]};\frac{[x^2 y][x^4]}{[x^3][x^3]}\right). \qquad (1.5.8)
$$

На практиці знайшов застосування прийом, який полягає в тому, що праву і ліву частини рівняння (1.5.1) ділять на $x$, в результаті чого отримують рівняння першого степеня відносно x

$$
q = ax + b, \qquad (1.5.9)
$$

де

$$q = \frac{y}{x}. \qquad (1.5.10)$$

Рішення цього рівняння ведеться за формулами прямолінійної залежності.

Початкові рівняння мають вигляд

$$\begin{aligned} &ax_1 + b - q_1 = \eta_1, \\ &ax_2 + b - q_2 = \eta_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &ax_n + b - q_n = \eta_n. \end{aligned} \qquad (1.5.11)$$

При цьому

$$q_1 = \frac{y_1}{x_1}, q_2 = \frac{y_2}{x_2}, \ldots, q_n = \frac{y_n}{x_n}. \qquad (1.5.12)$$

Система нормальних рівнянь буде

$$\begin{aligned} &a[x^2] + b[x] - [qx] = 0, \\ &a[x] + bn - [q] = 0. \end{aligned} \qquad (1.5.13)$$

Звідки

$$a = \frac{n[qx] - [x][q]}{n[x^2] - [x][x]}, b = \frac{[x^2][q] - [qx][x]}{n[x^2] - [x][x]}. \qquad (1.5.14)$$

Коефіцієнт b можна обчислити і за допомогою формули

$$b = \frac{[q]}{n} - a\frac{[s]}{n}. \qquad (1.5.15)$$

Зрівноважене значення функції запишеться

$$q(x) = ax^2 + bx. \qquad (1.5.16)$$

Попередній контроль виконується за формулою

$$[q^2] - a[qx] - b[q] = [\eta\eta]. \qquad (1.5.17)$$

Кінцева контрольна формула має вигляд

$$a(a[x^4] + b[x^3] - 2[qx^3]) + b(a[x^3] - 2[qx^2]) + [q^2x^2] = [\eta\eta xx]. \qquad (1.5.18)$$

При рівновідстоячих значеннях аргументу величини, які входять у формули для визначення коефіцієнтів і в контрольні формули, підраховуються за формулами як і в попередньому випадку.

Підстановка (1.5.15) в (1.5.16)

$$\varphi(x) = a\left(x^2 - \frac{[x]}{n}x\right) + \frac{[q]}{n}x. \qquad (1.5.17)$$

дає можливість визначити координати точки, розташованої на експериментальній кривій, які просто виражаються через результати визначень, а саме

$$x = \frac{[x]}{n}; y = \frac{[q][x]}{n^2}. \qquad (1.5.18)$$

Зроблені раніше зауваження відносно небажаності такого перетворення залишаються в силі для цього і для всіх аналогічних випадків.

### 1.5.2. Побудова математичної моделі квадратичною залежністю, за умови, що b=0, c=0

При цьому $y = ax^2$.

В даному випадку складається одне нормальне рівняння $a[x^4] = [x^2 y]$.

Із якого слідує $a = \frac{[x^2 y]}{[x^4]}x^2$.

Ймовірніше значення функції буде $\varphi(x) = \frac{[x^2 y]}{[x^4]}x^2$.

Експериментальна крива проходить через точки

$$(0; 0), \left(1; \frac{[x^2 y]}{[x^4]}\right), \left(\frac{[x^4]}{[x^2 y]}; 1\right). \qquad (1.6.1)$$

Контрольною являється формула $[y^2] - a[x^2 y] = [\varepsilon\varepsilon]$.

Якщо праву і ліву частини рівняння поділити на x, то отримаємо прямолінійну залежність $q = ax$.

Коефіцієнт a знаходиться за формулою $a = \frac{[qx]}{[x^2]}$.

Попередній контроль виконують за формулою $[q^2]-a[qx]=[\eta\eta]$.

Кінцева контрольна формула має вигляд

$$a(a[x^4]-2[qx^2])+[q^2x^2]=[\eta\eta xx]. \qquad (1.6.2)$$

## 1.6. Оцінка точності результатів при побудові математичної моделі квадратичним поліномом

Перейдемо до формули середньої квадратичної похибки коефіцієнтів a, b, c при параболічній залежності другого степеня.

По аналогії із прямолінійною залежністю, після взяття частинних похідних виразів (1.2.7) по $y_i$, підведення до квадрату і додавання, отримаємо

$$\left[\left(\frac{\partial a}{\partial y}\right)^2\right]=\frac{n[x^2]-[x][x]}{S}, \qquad (1.7.1)$$

$$\left[\left(\frac{\partial b}{\partial y}\right)^2\right]=\frac{n[x^4]-[x^2][x^2]}{S},\ldots\ldots\ldots(1.7.2)$$

$$\left[\left(\frac{\partial c}{\partial Q}\right)^2\right]=\frac{[x^2][x^4]-[x^3][x^3]}{S}, \qquad (1.7.3)$$

де

$$S=n[x^2][x^4]-[x^3][x^3]+[x]([x^2][x^3]-[x][x^4])+ \\ +[x^2]([x][x^3]-[x^2][x^2]). \qquad (1.7.4)$$

Після підстановки отриманих значень сум частинних похідних у виразах середніх квадратичних похибок будемо мати

$$m_a=\sqrt{\frac{[\varepsilon\varepsilon]}{n-3}\cdot\frac{n[x^2]-[x][x]}{S}}, \\ m_b=\sqrt{\frac{[\varepsilon\varepsilon]}{n-3}\cdot\frac{n[x^4]-[x^2][x^2]}{S}}, \qquad (1.7.5) \\ m_c=\sqrt{\frac{[\varepsilon\varepsilon]}{n-3}\cdot\frac{[x^2][x^4]-[x^3][x^3]}{S}}.$$

Ці ж формули застосовуються і для випадку, коли коефіцієнти вичислені за допомогою (1.3.5) шляхом зрівноваження поправок до наближених значень невідомих.

Представляючи середню квадратичну похибку одиниці ваги

$$m = \sqrt{\frac{[\varepsilon\varepsilon]}{n-3}}, \qquad (1.7.6)$$

а обернені ваги зрівноважених коефіцієнтів

$$\sqrt{\frac{1}{p_a}} = \sqrt{\frac{n[x^2]-[x][x]}{S}}, \qquad (1.7.7)$$

$$\sqrt{\frac{1}{p_b}} = \sqrt{\frac{n[x^4]-[x^2][x^2]}{S}}, \qquad (1.7.8)$$

$$\sqrt{\frac{1}{p_c}} = \sqrt{\frac{[x^2][x^4]-[x^3][x^3]}{S}}. \qquad (1.7.9)$$

де ваги коефіцієнтів будуть

$$P_a = \frac{S}{n[x^2]-[x][x]}, \qquad (1.7.10)$$

$$P_b = \frac{S}{n[x^4]-[x^2][x^2]}, \qquad (1.7.11)$$

$$P_c = \frac{S}{[x^2][x^4]-[x^3][x^3]}. \qquad (1.7.12)$$

Середні квадратичні похибки коефіцієнтів, обчислені з врахуванням (1.4.8) визначаються формулами

$$m_a = \sqrt{\frac{[\varepsilon\varepsilon]}{n-3}\cdot\frac{n[x'^2]}{n([x'^2][x'^4]-[x'^3][x'^3])-[x'^2][x'^2][x'^2]}}, \qquad (1.7.13)$$

$$m_b = \sqrt{\frac{[\varepsilon\varepsilon]}{n-3}\cdot\frac{n[x'^4]-[x'^2][x'^2]+4\frac{[x]}{n}(n[x'^2]-[x][x'^2])}{n([x'^2][x'^4]-[x'^3][x'^3])-[x'^2][x'^2][x'^2]}}, \qquad (1.7.14)$$

$$m_c = \sqrt{\frac{[\varepsilon\varepsilon]}{n-3} \cdot \frac{Q}{n([x'^2][x'^4]-[x'^3][x'^3])-[x'^2][x'^2][x'^2]}}, \qquad 1.7.15)$$

де

$$Q = [x'^2][x'^4]-[x'^3][x'^3]+\left(\frac{[x]}{n}\right)^2\left(n[x'^4]-[x'^2][x'^2]\right)+$$
$$+2\left(\frac{[x]}{n}\right)^2([x][x'^3]-[x'^2][x'^2])+[x'^2]\frac{[x]}{n}\left(\frac{[x]^3}{n^2}-2[x]\right). \qquad .(1.7.16)$$

При рівновідстоячих значеннях аргументу формули середніх квадратичних похибок набувають виду

$$m_a = \sqrt{\frac{[\varepsilon\varepsilon]}{n-3} \cdot \frac{180}{\chi^4 n(n-1)(n+1)(n-2)(n+2)}}, \qquad (1.7.17)$$

$$m_b = \sqrt{\frac{[\varepsilon\varepsilon]}{n-3} \cdot \frac{12\{60x_1^{\,2}+60x_1\chi(n-1)+\chi^2(2n-1)(8n-11)}{\chi^4 n(n-1)(n+1)(n-2)(n+2)}}, \qquad (1.7.18)$$

$$m_c = \sqrt{\frac{[\varepsilon\varepsilon]}{n-3} \cdot \frac{T}{\chi^4 n(n-1)(n+1)(n-2)(n+2)}}, \qquad (1.7.19)$$

де

$$T = 3\{60x_1^{\,4}+120x_1\chi(n-1)+12x_1^{\,2}\chi^2(7n^2-15n+7)+$$
$$+12x_1\chi^3(n-1)(n-2)(2n-1)+ \qquad (1.7.20)$$
$$+\chi^4(n-1)(n-2)(3n^2-3n+2)+12x_1\chi 3(n-1).$$

Ці ж формули застосовують для підрахунку середніх квадратичних похибок коефіцієнтів, виражених через кінцеві різниці (1.4.8).

При $y = ax^2 + bx$, формули середніх квадратичних похибок

$$m_a = \sqrt{\frac{[\varepsilon\varepsilon]}{n-2} \cdot \frac{[x^2]}{[x^4][x^2]-[x^3][x^3]}},$$
$$m_b = \sqrt{\frac{[\varepsilon\varepsilon]}{n-2} \cdot \frac{[x^4]}{[x^4][x^2]-[x^3][x^3]}}. \qquad (1.7.21)$$

В тому випадку, коли обчислення коефіцієнтів ведеться за формулами (1.5.14), середні квадратичні похибки визначаються за формулами

$$m_a = \sqrt{\frac{[\eta\eta]}{n-2} \cdot \frac{n}{n[x^2]-[x][x]}}, \qquad (1.7.22)$$

$$m_b = \sqrt{\frac{[\eta\eta]}{n-2} \cdot \frac{[x^2]}{n[x^2]-[x][x]}}. \qquad (1.7.23)$$

Величина η підраховуються за формулами (1.5.11).

Для квадратичної залежності $y = ax^2$ середня квадратична похибка коефіцієнта дається формулою

$$m_a = \sqrt{\frac{[\varepsilon\varepsilon]}{n-1} \cdot \frac{1}{[x^4]}}. \qquad (1.7.24)$$

Якщо значення коефіцієнта $a$ підраховувалось із формули (1.6.8) $a = \frac{[qx]}{[x^2]}$, то середня квадратична похибка вказаного коефіцієнта

$$m_a = \sqrt{\frac{[\eta\eta]}{n-1} \cdot \frac{1}{[x^2]}}. \qquad (1.7.25)$$

## 1.7. Середня квадратична похибка зрівноваженої функції квадратичного поліному

Середня квадратична похибка зрівноваженої функції підраховується за формулою

$$m_\varphi = \sqrt{m_4^2 \left[ \left( \frac{\partial \varphi}{\partial y} \right)^2 \right]}, \qquad (1.8.1)$$

де

$$m_y^2 = \frac{[\varepsilon\varepsilon]}{n-3}, \qquad (1.8.2)$$

$$\left(\frac{\partial\varphi}{\partial y_i}\right)^2 = \left(\frac{\partial}{\partial y_i}x^2 + \frac{\partial b}{\partial y_i}x + \frac{\partial c}{\partial y_i}\right)^2 =$$

$$= \left(\frac{\partial a}{\partial y_i}\right)^2 x^4 + \left(\frac{\partial b}{\partial y_i}\right)^2 x^2 + \left(\frac{\partial c}{\partial y_i}\right)^2 + 2\left(\frac{\partial a}{\partial y_i}\cdot\frac{\partial b}{\partial y_i}\right)x^3 + \qquad (1.8.3)$$

$$+ 2\left(\frac{\partial b}{\partial y_i}\cdot\frac{\partial c}{\partial y_i}\right)x^2 + 2\left(\frac{\partial b}{\partial y_i}\cdot\frac{\partial c}{\partial y_i}\right)x.$$

Після знаходження сум цієї рівності отримаємо

$$\left[\left(\frac{\partial\varphi}{\partial y}\right)^2\right] = \left[\left(\frac{\partial a}{\partial y}\right)^2\right]x4 + \left[\left(\frac{\partial b}{\partial y}\right)^2\right]x2 + \left[\left(\frac{\partial c}{\partial y}\right)^2\right] +$$

$$+ 2\left[\frac{\partial a}{\partial y}\cdot\frac{\partial b}{\partial y}\right]x^3 + 2\left[\frac{\partial a}{\partial y}\cdot\frac{\partial c}{\partial y}\right]x^2 + 2\left[\frac{\partial b}{\partial y}\cdot\frac{\partial c}{\partial y}\right]x. \qquad (1.8.4)$$

Підставляючи (1.8.2), (1.8.3), (1.8.4) в (1.8.1), будемо мати

$$m\varphi = \sqrt{m_a^2x^4 + m_b^2x^2 + m_c^2 + 2\frac{[EE]}{n-3}\left\{\left[\frac{\partial a}{\partial y}\cdot\frac{\partial b}{\partial y}\right]x^3 + \left[\frac{\partial a}{\partial y}\cdot\frac{\partial c}{\partial y}\right]x^2 + \left[\frac{\partial b}{\partial y}\cdot\frac{\partial c}{\partial y}\right]x\right\}} \qquad (1.8.5)$$

Розкриваючи вирази, які стоять у фігурних дужках, після виконання всіх дій, отримаємо

$$\left[\frac{\partial a}{\partial y}\cdot\frac{\partial b}{\partial y}\right]x^3 = \frac{1}{S}\left([x][x^2] - n[x^3]\right)x^3,$$

$$\left[\frac{\partial a}{\partial y}\cdot\frac{\partial c}{\partial y}\right]x^2 = \frac{1}{S}\left([x][x^3] - [x^2][x^2]\right)x^2, \qquad (1.8.6)$$

$$\left[\frac{\partial b}{\partial y}\cdot\frac{\partial c}{\partial y}\right]x = \frac{1}{S}\left([x^3][x^2] - [x][x^4]\right)x.$$

Таким чином, кінцевий результат розрахунку середньої квадратичної похибки зрівноваженої функції при поліномі другої степені буде мати вигляд

$$m_\varphi = \sqrt{m_a^2 x^4 + m_b^2 x^2 + m_c^2 + 2\frac{[\varepsilon\varepsilon]}{n-3}\frac{E}{S}}, \qquad (1.8.7)$$

де

$$E = ([x][x^2] - n[x^3])x^3 + ([x][x^3] - \\ -[x^2][x^2])x^2 + ([x^2][x^3] - [x][x^4])x, \qquad (1.8.8)$$

$$\Delta = S = n([x^2][x^4] - [x^3][x^3]) + [x]([x^2][x^3] - \\ -[x][x^4]) + [x^2]([x][x^3] - [x^2][x^2]). \qquad (1.8.9)$$

Для випадку перетворень за допомогою переносу початку координат в точку $x' = x - \frac{[x]}{n}$ формула (1.8.7) перетвориться у

$$m_\varphi = \sqrt{m_a^2 x^4 + m_b^2 x^2 + m_c^2 + 2\frac{[\varepsilon\varepsilon]}{n-3}\frac{E'}{S'}}, \qquad (1.8.10)$$

де

$$E' = (-n[x'^3] - 2[x][x'^2])x^3 + ([x][x'^2] + \\ + [x'^2]\frac{[x]^3}{n^2} - [x'^2])x^2 + ([x'^2][x'^3] - [x][x'^4])x, \qquad (1.8.11)$$

$$S' = n([x'^2][x'^4] - [x'^3][x'^3]) + [x'^2][x'^2][x'^2]. \qquad (1.8.12)$$

При рівновідстоячих значеннях аргументу величини E і S будуть мати вигляд

$$E = -360\{2x_1 + \chi(n-1)\}x^3 + 60\{6x_1^2\chi(n-1) + \\ + \chi^2(n-1)(n-2)\}x^2 - 36\{20x_1^3 + 30x_1^2\chi(n-1) + \\ + 2x_1\chi^2(7n^2 - 15n + 7) + \chi^3(n-1)(n-2)(2n-1)\}x. \qquad (1.8.13)$$

$$S = \chi^4 n(n-1)(n+1)(n-2)(n+2). \qquad (1.8.14)$$

Ці формули також застосовуються при вираженні шуканих коефіцієнтів a, b, c через кінцеві різниці.

Для того, щоб проаналізувати формулу середньої квадратичної похибки (1.8.7), надамо їй дещо інший вигляд.

Для цього виразимо $c$ із (1.1.7) через коефіцієнти $a$ і $b$: $c = \frac{[y]}{n} - b\frac{[x]}{n} - a\frac{[x^2]}{n}$ і підставимо його у (1.1.8)

$$\varphi(x) = a\left(x^2 - \frac{[x^2]}{n}\right) + b\left(x - \frac{[x]}{n}\right) + \frac{[y]}{n}. \qquad (1.8.15)$$

У цій формулі залежними величинами від результатів експерименту, являються $a$, $b$ і $\frac{[y]}{n}$. Знаходячи середню квадратичну похибку цього виразу по наведеним вище правилам і маючи на увазі, що

$$\frac{\partial\frac{[y]}{n}}{\partial y_i} = \frac{1}{n}; \left[\left(\frac{\partial\frac{[y]}{n}}{\partial y}\right)^2\right] = \frac{1}{n}, \qquad (1.8.16)$$

отримаємо після всіх дій і перетворень

$$m_\varphi = \sqrt{m_a^2\left(x^2 - \frac{[x^2]}{n}\right)^2 + m_b^2\left(x - \frac{[x]}{n}\right)^2 + m_{\frac{[y]}{n}}^2 + \frac{+2\frac{[\varepsilon\varepsilon]}{n-3}\frac{([x][x^2]-n[x^3])(x^2-\frac{[x^2]}{n})(x-\frac{[x]}{n})}{S}}{}} \qquad (1.8.17)$$

де

$$S = n([x^2][x^4]-[x^3][x^3]) + [x]([x^2][x^3]-[x][x^4]) + \\ + [x^2]([x][x^3]-[x^2][x^2]), \qquad (1.8.18)$$

$$m_{\frac{[y]}{n}}^2 = \frac{[\varepsilon\varepsilon]}{n(n-3)}. \qquad (1.8.19)$$

Розглядаючи цей вираз, бачимо, що при значеннях x рівному $\sqrt{\frac{[x^2]}{n}}$, похибка коефіцієнта a не впливає на величину середньої квадратичної похибки функції.

В даному випадку

$$m_\varphi = \sqrt{m_b^2\left(x - \frac{[x]}{n}\right)^2 + m^2_{\frac{[Q]}{n}}}\,. \qquad (1.8.20)$$

При значеннях x рівному $\frac{[x]}{n}$, на величину середньої квадратичної похибки функції не впливає похибка коефіцієнта

$$m_\varphi = \sqrt{m_a^2 + \left(x2 - \frac{[x2]}{n}\right) + m^2_{\frac{[y]}{n}}}\,. \qquad (1.8.21)$$

Таким чином, при одному із зазначених в середній частині інтервалу спостережень, середня квадратична похибка функції отримує мінімальне значення, збільшуючись до кінців інтервалу і за межами його.

Зона розсіювання обмежується кривими, які проходять через точки з абсцисами $\sqrt{\frac{[x^2]}{n}}$ і $\frac{[x]}{n}$.

$$\varphi(x) = ax^2 + bx + c \pm m_\varphi\,, \qquad (1.8.22)$$

де $m_\varphi$ виражається формулою (1.8.17) (рис.1)

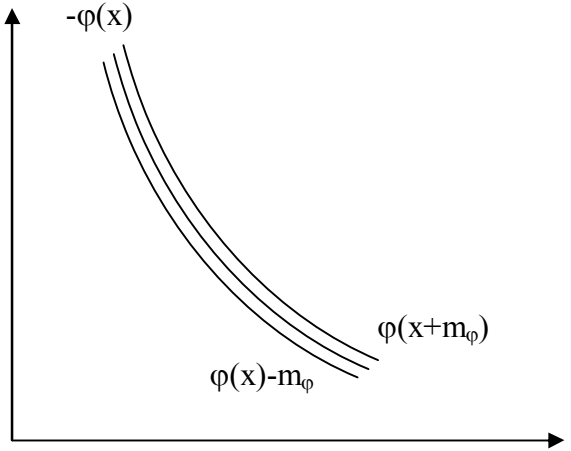


Рис.1. Зона розсіювання функції $\varphi(x) = ax^2 + bx + c$

У випадку квадратичної залежності (1.5.1) $\varphi(x) = ax^2 + bx$ середня квадратична похибка виражається формулою

$$m_\varphi = \sqrt{m_a^2 x^4 + m_b^2 x^2 - 2\frac{[\varepsilon\varepsilon]}{n-3} \cdot \frac{[x^3]}{[x^4][x^2] - [x^3][x^3]} x^3}. \quad (1.8.23)$$

При перетворенні рівняння до прямолінійної залежності шляхом ділення правої і лівої частини (1.5.1) на $x$ $q = ax + b$, формули середньої квадратичної похибки зрівноваженої функції

$$m_q = \sqrt{m_a^2 x^2 + m_b^2 - 2\frac{[\varepsilon\varepsilon]}{n-2} \cdot \frac{[x]}{n[x^2] - [x][x]} x}, \; m_\varphi = m_q x. \; , (1.8.24)$$

Для випадку $y = ax^2$, формула середньої квадратичної похибки зрівноваженої функції

$$m_\varphi = \sqrt{\frac{[\varepsilon\varepsilon]}{n-1} \cdot \frac{x^4}{[x^4]}} \quad . \qquad (1.8.25)$$

Якщо для рішення задачі застосовувалася формула $q = ax$, то

$$m_q = \sqrt{\frac{[\eta\eta]}{n-1} \cdot \frac{x^2}{[x^2]}}, \qquad (1.8.26)$$

$$m_\varphi = m_q x. \qquad (1.8.26)$$

# РОЗДІЛ 2

# ОСНОВИ С++. СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ MICROSOFT VISUAL STUDIO 2010

## 2.1. Основні характеристики мови С++

С++ — це статична типізована мова програмування загального призначення що компілюється.

Вона підтримує такі парадигми програмування, як процедурне програмування, модульність, роздільна компіляція, обробка виключень, абстракція даних, типи (об'єкти), віртуальні функції, об'єктно-орієнтовне програмування, узагальнене програмування, контейнери і алгоритми, поєднує властивості як мов високого рівня, так і низького. В порівнянні з її попередницею – мовою С, – найбільше уваги приділено підтримці об'єктно-орієнтовного і узагальненого програмування.[3] Назва «С++» походить від назви мови С, в якій унарний оператор ++ означає інкремент змінної.

Являючись однією з найпопулярніших мов програмування, С++ широко використовується для розробки програмного забезпечення. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високоефективних серверів, а також розважальних програм (наприклад, відеоігри). Існує декілька реалізацій мови С++ – як безкоштовних, так і комерційних. Найбільш популярні проект GNU, Microsoft, Intel і Embarcadero (Borland). С++ дуже сильно вплинув на інші мові програмування, в першу чергу на Java і C#.

При створенні С++ Бьйорн Страуструп прагнув зберегти сумісність с мовою С. Кількість програм, корті можуть з однаковим успіхом транслюватися як компілятором С, так і компіляторами С++, досить велика – почасти завдяки тому, що синтаксис С++ був оснований на синтаксисі С.

## 2.2. Стандартизація мови

В 1998 році був опублікований стандарт мови ISO/IEC 14882:1998 (відомий як C++98),[4] розроблений комітетом по стандартизації (C++ (ISO/IEC JTC1/SC22/WG21 working group).

В 2003 році був опублікований стандарт мови ISO/IEC 14882:2003, де були виправлені помилки що виявились і недоліки попередньої версії стандарту.

В 2005 році був виданий звіт «Library Technical Report 1» (що має коротку назву TR1). Не являючись офіційною частиною стандарту, звіт описує розширення стандартної бібліотеки, котрі, як очікувалось авторами, повинні бути включені в наступну версію мови C++. Рівень підтримки TR1 покращується майже у всіх компіляторах мови C++.

З 2009 року велась робота по оновленню попереднього стандарту, попередньою версією нового стандарту спочатку був C++99, а через рік C++0x, сьогодні – C++11, куди було включено доповнення в ядро мови і розширення стандартної бібліотеки, в тому числі більшу частину TR1.

## 2.3. Огляд мови

### 2.3.1. Необ'єктно-орієнтовані можливості

В цьому пункті описуються можливості, що безпосередньо не пов'язані з об'єктно-орієнтованим програмуванням (ООП), але більшість з них, однак, дуже важливі в поєднання з ООП.

В C++ доступні наступні вбудовані типи даних:

- Символьні: char, wchar_t.
- Цілочисельні із знаком: signed char, short int, int, long int (і long long int, в стандарті C++11).
- Цілочисельні без знаку: unsigned char, unsigned short int, unsigned int, unsigned long int(і unsigned long long int, в стандарті C++11).
- З плаваючою крапкою: float, double, long double.
- Логічні: bool, що мають значення true і false.

Операції порівняння повертають тип bool. Вирази в дужках після if, while зводяться до типу bool.[5]

Функції можуть приймати аргументи по посиланню. Наприклад, функція void f(int &x) {x=3;} присвоює своєму аргументу значення 3. Функції також можуть повертати результат по посиланню, і посилання можуть бути поза всяким зв'язком з функціями. Наприклад, {double &b=a[3]; b=sin(b);}, еквівалентно a[3]=sin(a[3]);. При програмуванні посилання в деякій степені схожі на вказівники, з наступними особливостями: перед використанням посилання повинне бути ініціалізоване; посилання увесь час вказує на одну і ту ж адресу; у виразі посилання позначає безпосередньо той об'єкт чи ту функцію, на котру вказує, звертання ж до об'єкта чи функції через вказівник потребує розіменування вказівника. Існують і інші відмінності у використанні вказівників та посилань. Концептуально посилання – друге ім'я змінної або функції, інша назва одної і тої ж адреси, існує лише тільки в тексті програми і замінюється адресою при компіляції. А вказівник – зберігає адресу змінної до якої звертаються.

Один або декілька останніх аргументів функції можуть задаватись по замовчуванню. Наприклад, якщо функція описана як void f(int x, int y=5, int z=10), то виклики f(1), f(1,5) и f(1,5,10) є еквівалентними.

При описі функцій відсутність аргументів в дужках означає, на відміну від C, що аргументів немає, а не те, що вони невідомі. Якщо аргументи невідомі, потрібно використовувати три крапки, наприклад int printf(const char* fmt, ...).

Специфікатор inline дозволяє оголошувати inline-функції. Функція, оголошена в середині тіла класу є inline по замовчуванню. Спочатку такі функції задумувались як функції, що є хорошим кандидатом на оптимізацію, при якій в місцях звернення до функції компілятор вставляє тіло цієї функції, а не код виклику. В дійсності компілятор не зобов'язаний реалізовувати підстановку тіла для inline-функції, але може, виходячи із заданих критеріїв оптимізації, виконувати підстановку тіла для функції, а не код виклику. Найбільш значимою особливістю inline-функцій є те, що вона може багаторазово визначатись в декількох одиницях трансляції, в той час

як функція, що не є inline, може визначатись в програмі не більше одного разу.

Специфікатор volatile використовується в описі змінних і інформує компілятор, що значення даної змінної може бути змінене способом, котрий компілятор не в змозі відстежити. Для змінних, що оголошені як volatile, компілятор не повинен застосовувати засоби оптимізації, що змінюють положення змінної в пам'яті або операції, що покладаються на незмінність значення змінної в проміжку між двома присвоєннями їй значення.

В програмі написаній на C++ може бути декілька функцій з одним і тим же ім'ям, але різними типами аргументів або їх кількістю. Це називається перевантаженням функції; при цьому тип значення, що повертається на перевантаження не впливає. Наприклад, перевантажені функції можуть виглядати наступним чином:

```
void Print(int x);
void Print(double x);
void Print(int x, int y);
```

Операторні функції здебільшого схожі на звичайні (не операторні функції) функції. За виключенням операторів new, new[], delete і delete[], не можна змінювати поведінку операторів для вбудованих типів (скажімо змінити операцію множення для значень типу int); не можна додавати нові оператори яких немає в C++; не можна змінювати існуючі пріоритети і асоціативність операторів

В доповнення до функцій malloc і free в C++ введено операторні функції operator new, operator new[], operator delete і operator delete[], а також оператори new, new[], delete и delete[]. Якщо T – довільний об'єктний тип, що не є типом масиву, X – довільний об'єктний тип і A – тип масиву з деякою кількістю n елементів, щомають тип X, то:

- new T – виділяє пам'ять, достатню для розміщення одного об'єкта типа T, можливо, об'єкт в цій пам'яті ініціалізується, і повертає вказівник T*.
- new X[n] і new A – виділяють пам'ять, достатню для розміщення n об'єктів типа X, можливо, ініціалізують

кожен об’єкт в цій пам’яті, і повертають вказівник типа X*.

- delete p – знищує об’єкт (якщо він не є масивом), на котрий посилається вказівник p і вивільняє область пам’яті, що раніше була виділена для нього new-виразом.
- delete [] p – знищує кожен об’єкт в масиві, на котрий посилається вказівник p і вивільняє область пам’яті, що раніше була виділена для цього масиву new-виразом.

Операція delete перевіряє чи її аргумент не рівний NULL, у протилежному випадку вона нічого не виконує. Для ініціалізації об’єкта non-POD класового типу new-вираз викликає конструктор; для знищення об’єкта класового типа delete-вираз викликає деструктор.

### 2.3.2. Об’єктно-орієнтовні особливості мови

C++ додає до C об’єктно-орієнтовні можливості – вводить класи, котрі забезпечують три найважливіші властивості ООП: інкапсуляцію, наслідування і поліморфізм.

В стандарті C++ під класом розуміють користувацький тип, оголошений з використанням одного з ключових слів class, struct або union.

В тілі класу можна вказати лише заголовок функції, а можна описати й всю функцію В останньому випадку функція буде вважатись inline.

Нестатичні функції-члени класу (і лише вони) можуть мати специфікатор const. Такі функції не мають права змінювати поля класу (крім полів, зазначених як mutable). Якщо вони намагаються це зробити, компілятор повинен видати повідомлення про помилку.

В C++ при *наслідуванні* одного класу іншим, відбувається наслідування також реалізації батьківського класу, також клас-нащадок може додавати свої поля і функції, або ж перевизначити функції базового класу.

Конструктор нащадка викликає конструктори базових класів, а потім вже конструктори нестатичних членів-даних, що є екземплярами класу. Деструктори ж працюють у зворотному порядку.

Наслідування буває публічним, захищеним і закритим. Нащадок – це більше ніж базовий клас, тому, якщо наслідування відкрите, то він може використовуватись всюди, де використовується базовий клас, але не навпаки.

*Поліморфізмом* в програмуванні називається пере визначення нащадком функцій-членів базового класу, наприклад:

```
class Figure
{
   ...
   void Draw() const;
   ...
};
class Square : public Figure
{
   ...
   void Draw() const;
   ...
};
class Circle : public Figure
{
   ...
   void Draw() const;
   ...
};
```

Яка саме з функцій буде викликана – Figure::Draw(), Square::Draw() або Circle::Draw() – визначається під час компіляції. Наприклад:

```
Circle *c = new Circle(0,0,5);
Figure *f = c; c->Draw();
f->Draw();
```

Не дивлячись не те, що обидва вказівника вказують на один і той же об’єкт класу Circle, для с буде викликана Circle::Draw(), а для f — Figure::Draw(), оскільки f – вказівник на об’єкт класу Figure. Такий поліморфізм називається статичним.

Але в C++ є і динамічний поліморфізм, коли функція, що викликається визначається під час виконання. Для цього функції-члени базового класу повинні бути оголошені віртуальними.

Но в C++ есть и динамический полиморфизм, когда вызываемая функция определяется во время выполнения. Для этого функции-члены базового класса должны быть объявлены виртуальными.

Чисто віртуальною функцією називається віртуальна функція-член, котра оголошена зі специфікатором =0:

```
class Figure
{
   ...
   virtual void Draw() const = 0;
);
```

Чисто віртуальна функція може бути залишена без оголошення, крім випадку, коли потрібно виконати її виклик.

Абстрактним класом називається такий, в котрого є хоча б одна чисто віртуальна функція-член. Об’єкти таких класів створювати заборонено. Абстрактні класи часто використовуються як інтерфейси. На відміну від чистих інтерфейсів інших мов, абстрактні класи C++ можуть мати невіртуальні функції-члени і члени-дані.

Клас в C++ може складатись з полів, вкладених типів і функцій-членів. *Інкапсуляція* в C++ реалізується через зазначення рівня доступу до членів класу: вони бувають відкритими, захищеними і закритими. В C++ структури формально відрізняються лише тим, що по замовчування члени і базові класи в них публічні (відкриті), а у класу закриті.

Функції-друзі – це функції, що не є функціями-членами і тим не менше мають доступ до захищених і закритих членів класу. Вони повинні бути оголошені в тілі класу як friend.

Існують також класи-друзі. Якщо клас А – друг класу В, то всі його власні (не унаслідувані) функції-члени можуть звертатись до будь-яких членів класу В. Однак, не навпаки.

В класах завжди є спеціальні функції – конструктори і деструктори, котрі можуть бут оголошені явно і неявно. Конструктор викликається для ініціалізації об'єкта при його створенні, а деструктор – для знищення. Зокрема, конструктор може бути викликаним для виконання перетворення до класового типу.

## 2.4. Переваги мови

Перш за все, слід підкреслити, що оцінювати переваги і, особливо, недоліки С++ необхідно в контексті тих принципів, на яких будувалась мова, і вимогах, що до неї спочатку висувались.

С++ – надзвичайно потужна мова, що містить засоби створення ефективних програм практично будь-якого призначення, від низькорівненвих утиліт і драйверів до складних програмних комплексів найрізноманітнішого призначення, зокрема:

- Підтримуються різноманітні стилі і технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, мета програмування (шаблони, макроси).
- Передбачуване виконання програм є важливою перевагою для побудови систем реального часу. Весь код, що неявно генерується компілятором для реалізації мовних можливостей оголошено в стандарті. Також строго визначені місця програми, в котрих цей код виконується. Це дає можливість замірювати або розраховувати час реакції програми на зовнішню подію.
- Автоматичний виклик деструкторів об'єктів при їх знищені, при чому в порядку, зворотному виклику конструкторів. Це спрощує і дає більш надійний спосіб вивільнення ресурсів, а також дозволяє гарантовано виконувати переходи станів програми, не обов'язково пов'язаних з вивільненням ресурсів.

- Користувацькі функції-оператори дозволяють коротко і ємко записувати вирази над користувацькими типами в алгебраїчній формі.
- Мова підтримує поняття фізичної і логічної константності. Це робить програму надійнішою.
- Використовуючи шаблони, можна створювати узагальнені контейнери і алгоритми для різних типів даних, а також спеціалізувати і визначати на етапі компіляції.
- Можливість імітації розширення мови для підтримки парадигм, котрі не підтримуються компіляторами напряму.
- Можливість створення вбудованих предметно-орієнтованих мов програмування.
- Використовуючи шаблони і множинне наслідування можна імітувати класи-домішки і комбінаторну параметризацію бібліотек.
- Кросплатформеність: стандарт мови накладає мінімальні вимоги на ЕОМ для запуску скомпільованих програм.
- Ефективність. Мова спроектована так, щоб надати програмісту максимальний контроль над усіма аспектами структури і порядку виконання програми. Жодна з можливостей мови, що призводить до додаткових накладних затрат, не є обов'язковою до використання.
- Є можливість роботи на низькому рівні з пам'яттю та адресами.
- Висока сумісність з мовою С, що дозволяє використовувати увесь існуючий код на С.

## 2.5. Microsoft Visual Studio 2010

Середовище розробки Microsoft Visual Studio 2010 підтримує два окремих, але тісно пов'язаних між собою різновидів мови С++, а саме: вихідну версію С++, що являє собою стандарт ISO/IEC, а також нову версію С++, що має назву С++/CLI, котра булла розроблена корпорацією Microsoft і тепер затверджена в стандарті ECMA. Ці дві версії мови С++ доповнюють одна одну і грають зовсім різні ролі. Версія ISO/ANSI С++ призначена для написання

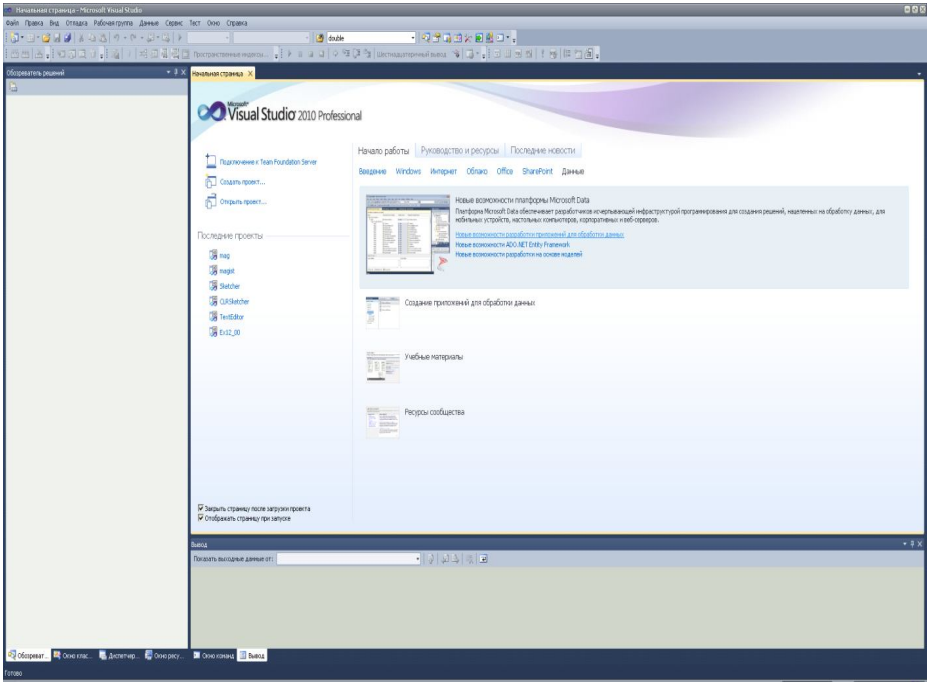
високоефективних програм, що здатні функціонувати на комп'ютері «рідним» чином, в той час як версія C++/CLI була спеціально розроблена для написання програм, орієнтованих на Windows Forms.

Середовище розробки Microsoft Visual Studio 2010 повністю підтримує оригінальну стандартну мову C++ ISO/IEC, а також володіє перевагами нових базових засобів мови C++, що надаються новітніми стандартами ISO/IEC для мови C++.

Середовище розробки Microsoft Visual Studio 2010 підтримує також стандарт C++/CLI, що є діалектом мови C++. Ідея створення C++/CLI заклечалась в доданні нових можливостей базовій мові C++, котрі дозволять розробляти програми під віртуальну машину, що підтримується середовищем .NET Framework. Це зближує мову C++ з іншими мовами, такими як BASIC і C#, що здатні використовувати середовище .NET Framework. Нині мова C++/CLI є стандартом ECMA разом із стандартом інфраструктури CLI, котрий визначає середовище віртуальної машини для платформи .NET.[6]

## 2.6. Основні елементи інтерфейсу Microsoft Visual Studio 2010

При запуску середовища розробки Microsoft Visual Studio 2010 відкривається вікно зображене на рис.2

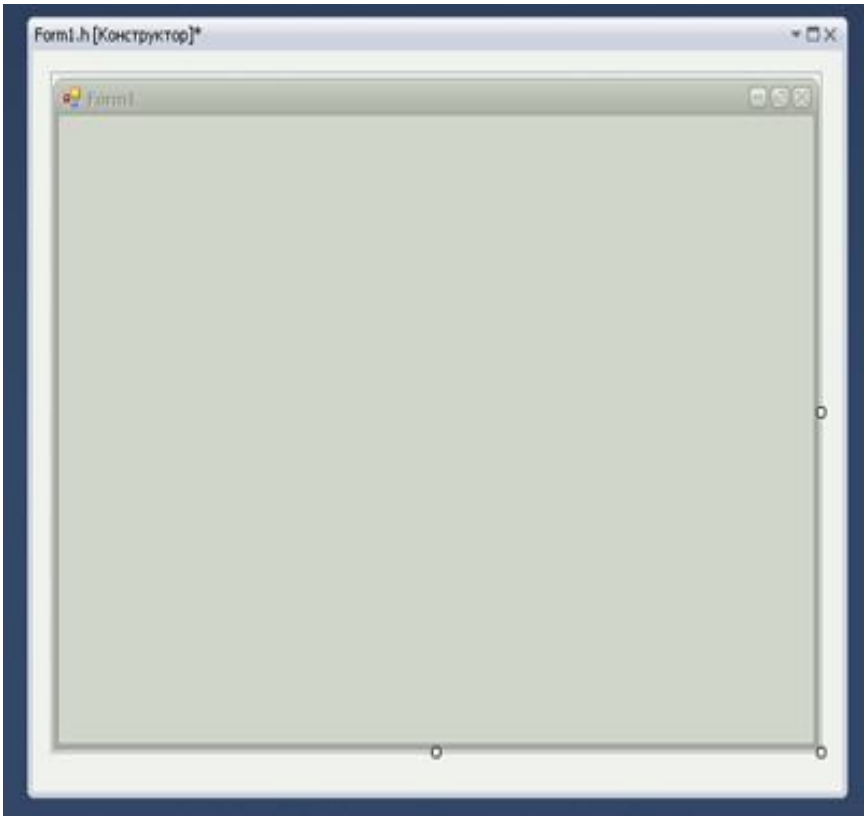



(Рис. 2)

Ліва частина вікна називається «Провідником рішень», праве верхнє вікно, що містить стартову сторінку – це вікно редактора, а вкладка, котру видно знизу вікна – це вікно виводу. Вікно провідника рішень дозволяє виконувати навігацію по програмним файлам, відображаючи їх вміст у вікні редактора, а також додавати нові файли до вашої програми. Вікно провідника рішень містить одну додаткову вкладку, котра відображає «Представлення ресурсів». Вікно редактора – це місце, де відбувається ввід і модифікація вихідного коду і інших компонентів програми. Вікно

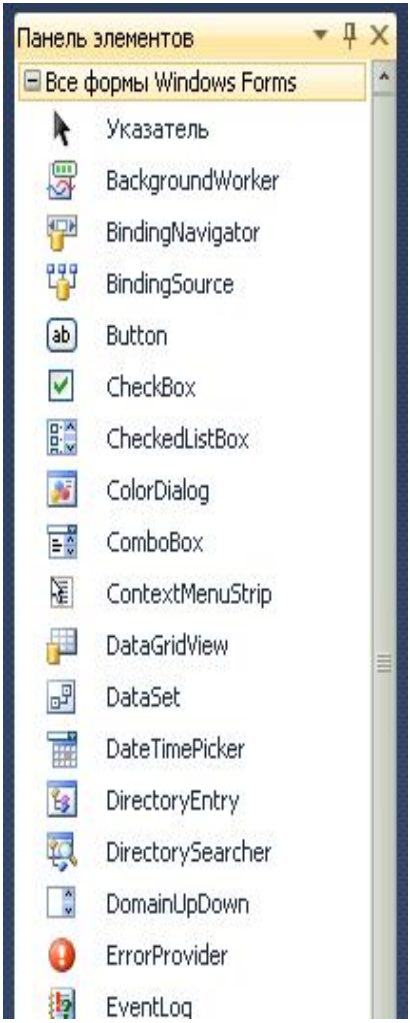
виводу відображає повідомлення отримані при компонуванні програми.

При створенні програми за допомогою Windows Forms вікно редактора буде мати наступний вигляд (рис. 3):




(Рис. 3)

Тепер вікно редактора демонструє образ вікна програми замість її коду. Причина в тому що розробка GUI для Windows Forms в основному орієнтована на підхід графічного проектування, а не на підхід написання коду. Вікно Toolbox (рис.4) пропонує список стандартних компонентів, котрі можна додати в програму Windows Forms .




(Рис. 4)

Microsoft Visual Studio 2010 надає можливість графічно редагувати будь-які компоненти GUI в будь-який час і код буде відповідним чином відкоректований.[6]

# РОЗДІЛ 3. ЕТАПИ РОЗРОБКИ ПРОГРАМИ ТА ДЕМОНСТРАЦІЯ ЇЇ РОБОТИ

## 3.1. Розробка допоміжних бібліотек

Перед тим як приступити безпосередньо до розробки програми, підготовлено ряд бібліотек, що описують основні об'єкти та функції які будуть застосовуватись у програмі. А саме бібліотеку CMatrix.h, для роботи з матрицями і Modeling.h для побудови, власне, моделі та обчислення її основних математичних і економічних характеристик.

За допомогою бібліотеки CMatrix.h можна створювати матриці будь-якої розмірності і будь-якого типу даних. Також в бібліотеці передбачені наступні методи роботи з матрицями:

- int getRows(); – повертає кількість рядків матриці;
- int getColumns(); – повертає кількість стовпців матриці;
- int getNumOfElem(); – повертає кількість елементів матриці;
- T& operator () (unsigned int i, unsigned int j) – дозволязвертатись до елемента матриці $x_{ij;}$
- CMatrix<T>& CMatrix<T>::operator = (const CMatrix<T> &rhs) – перевантажений оператор =, дозволяє присвоювати значення одніє матриці іншій;
- CMatrix<T>& CMatrix<T>::operator += (const CMatrix<T> &rhs);, CMatrix<T>& CMatrix<T>::operator -= (const CMatrix<T> &rhs);, CMatrix<T>& CMatrix<T>::operator *= (const CMatrix<T> &rhs);, CMatrix<T>& CMatrix<T>::operator *= (const T &rhs);, CMatrix<T>& operator + (const CMatrix<T> &lhs, const CMatrix<T> &rhs);, CMatrix<T>& operator - (const CMatrix<T> &lhs, const CMatrix<T> &rhs);, CMatrix<T>& operator * (const CMatrix<T> &lhs, const CMatrix<T> &rhs);, CMatrix<T>& operator * (const CMatrix<T> &lhs, const T &rhs);, CMatrix<T>& operator * (const T &lhs, const CMatrix<T> &rhs); – перевантажені оператори, що дозволяють виконувати алгебраїчні дії над матрицями;

- T Determinant (const CMatrix<T> &M); – функція що повертає визначник матриці;
- CMatrix<T>& Inverse (const CMatrix<T> &M); – функція повертає матрицю обернену до тої, що передається в параметрі;
- CMatrix<T>& Minor (const CMatrix<T> &M, int fix_row, int fix_column); – повертає мінор матриці що передається у першому параметрі по рядку і стовпцю що передаються у другому і третьому параметрах відповідно;
- CMatrix<T>& Transpos (const CMatrix<T> &M) – повертає транспоновану матрицю що передається у параметрі.

Лістинг коду бібліотеки приведено в додатку А.

В бібліотеці Modeling.h описаний об'єкт, що містить усі основні характеристики моделі. Ці характеристики обчислюються в 4 етапи під час виконання програми. Слід зазначити, що для побудови моделей застосовано метод найменших квадратів.

На першому етапі створюються динамічні масиви в яких зберігаються введені користувачем дані, а саме: кількість спостережень, ціна товару та відповідний цій ціні обсяг продажу за досліджуваний проміжок часу, а також витрати що пов'язані із кількістю виготовленої та реалізованої продукції і витрати що від неї не залежать.

На другому етапі обраховуються коефіцієнти a, b, c емпіричного рівняння:

$$y = ax^2 + bx + c,$$

що й буде уособленням математичних моделей у програмі. Для їх обрахунку обчислюється матриця обернена до матриці коефіцієнтів нормальних рівнянь. Після завершення виконання другого етапу вона не знищується, на відміну від інших змінних, що містили допоміжні обрахунки, а навпаки зберігається як один із членів класу. Це робиться тому, що дана матриця неодноразово буде використовуватись при обрахунку інших параметрів побудованих моделей.

На третьому етапі обраховуються основні економічні характеристики моделі, такі як товарообіг, собівартість проданої

продукції, еластичність регресії попиту, прибуток, а також значення зрівноваженої функції. Усі знайдені значення зберігаються у вигляді динамічних масивів у змінних класу.

На завершальному етапі відбувається оцінка точності побудованої моделі. Обраховуються наступні величини:

- Абсолютні похибки зрівноваженої функції – це різниці значень отриманих практично та теоретично.
- Обернені ваги зрівноваженої функції за способом найменших квадратів:

$$\frac{1}{P_\varphi} = \varphi Q \varphi^T,$$

де $\varphi$ – значення коефіцієнтів початкових рівнянь функції;

$\varphi^T$ – транспонована матриця коефіцієнтів[2];

$Q$ – обернена матриця коефіцієнтів нормальних рівнянь.

- Квадратичні похибки зрівноваженої функції, обчислюються як добуток кореня квадратного з обернених ваг на абсолютну похибку зрівноваженої функції.
- Обернені ваги коефіцієнтів рівняння. Ці значення розміщені у діагоналі оберненої матриці коефіцієнтів нормальних рівнянь $Q$ наступним чином: $Q_{11}=1/P_c$, $Q_{22}=1/P_b$ і $Q_{33}=1/P_a$.[2]
- Середня квадратична похибка

$$\mu = \sqrt{\frac{[VV]}{n-m-1}},$$

де $V$ – абсолютні похибки, $n$ – кількість спостережень, $m$ – порядок полінома[3].

- Середні квадратичні похибки коефіцієнтів емпіричного рівняння обчислюються як добуток середньої квадратичної похибки на корінь квадратний оберненої ваги коефіцієнта.
- Значимість коефіцієнтів емпіричного рівняння це відношення самого коефіцієнта до його середньої квадратичної похибки.
- Контроль зрівноваження обчислюється за контрольною формулою

$$[y^2] - a[yx^2] - b[yx] - c[y] = [\varepsilon\varepsilon]$$

Лістинг коду бібліотеки приводиться в додатку Б.

### 3.2. Візуальне оформлення програми

Усі вікна та форми програми було розроблено за допомогою Windows Forms. Допоміжні бібліотеки підключаються до головного вікна програми, адже саме у ньому будуть відображатись усі результати розрахунків.

Після запуску програми у головному вікні відображається лише меню, що містить три пункти: «Нова модель», «Про програму», «Вихід».

При натисканні на пункті «Нова модель» відбувається виклик форми для внесення даних згідно яких проводитиметься моделювання. Функція обробки даної події виглядає наступним чином:

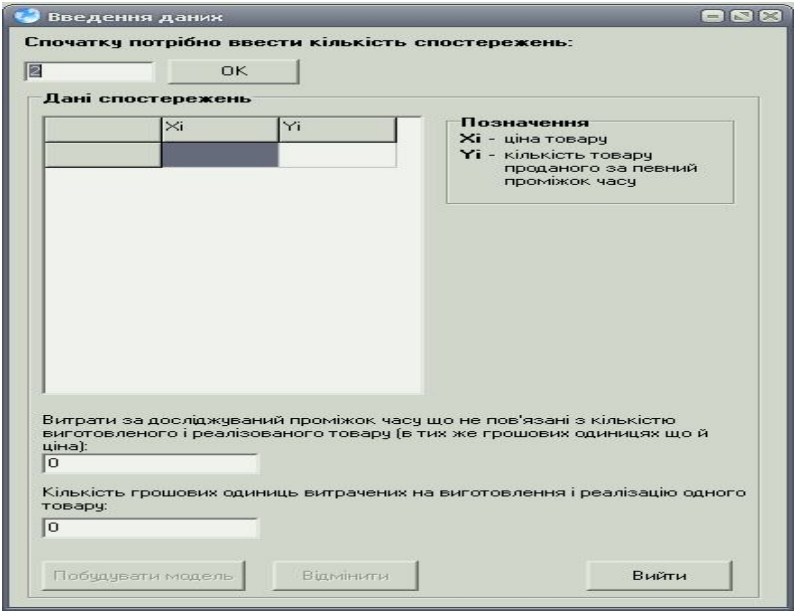
```
void __fastcall TMainForm::N1Click(TObject *Sender)
{
    InputForm->ShowModal() ;
}
```

При обранні пункту «Про програму» викликається невеличке вікно із короткою інформацією про версію програми та її авторів (рис. 5).


(Рис. 5)

Пункт меню «Вихід» закриває програму, після чого відбувається знищення усіх введених даних, та вивільнення пам’яті виділеної під них.

Отже, повернемось до форми введення даних, що, як було зазначено раніше, викликається при натисканні на пункт меню «Нова модель». При відкритті ця форма має наступний вигляд:




(Рис. 6)

Користувачу спочатку пропонується ввести кількість спостережень (тобто відповідну кількість пар «ціна» – «кількість проданого товару»). Після чого в таблицю, яка знаходиться нижче додається зазначена кількість рядків. Далі користувач може або ввести необхідні дані у таблицю, або ж повернутись до попереднього пункту, якщо, наприклад, була вказана невірна кількість спостережень, натиснувши кнопку «Відмінити». При цьому всі дані з таблиці буде видалено, а кількість рядків стане

такою як на початку. Також дану форму можна закрити натиснувши кнопку «Вихід».

Для зручності усі кнопки у формі введення даних забезпечені підказками, тобто якщо затримати курсор над будь-якою з них на декілька секунд, то з'явиться відповідна спливаюча підказка, що більш розширено описує призначення даної кнопки.

При натисканні на кнопку «Побудувати модель» відбувається перевірка введених даних. Якщо якісь із введених користувачем значень не коректні, або ж не заповнені усі потрібні поля, з'являється повідомлення про помилку із роз'ясненням що саме було зроблено не так. Форма заповнена даними демонструється у додатку В.

Коли перевірка відбулась і усі введені дані визнані коректними відбувається, власне, побудова моделі, та обрахунок її математичних та економічних характеристик. Під час цього обрахунку відображається панель прогресу, де описується поточний етап розрахунків. Це дуже зручно при обробці великої кількості спостережень. Користувач може слідкувати за процесом обчислення і у разі неполадки йому буде відомо на якому саме етапі щось пішло не так.

Саме у тілі цієї форми відбуваються усі розрахунки, а обчислені дані передаються на вивід у головне вікно програми. Лістинг коду цих розрахунків:

```
void __fastcall TBuildModel::FormActivate(TObject *Sender)
{
        BuildModelComment->Caption = "Запис даних
спостережень..." ;
        /*Додання необхіної кількості рядків у таблицю*/
        int n = InputForm->TableData->RowCount - 1 ;
        double* x = new double[n] ;     // Масив цін
        double* y = new double[n] ;     // Масив кількостей
проданого товару
        /*Виведення даних у головне вікно програми*/
        for (int i = 1 ; i < InputForm->TableData->RowCount ; i++)
        {
```

```
                /*Перевірка чи не залишилось у таблиці порожніх
комірок*/
                if (InputForm->TableData->Cells[1][i].IsEmpty() ||
                   InputForm->TableData->Cells[2][i].IsEmpty())
                {
                     /*Виведення повідомлення про помилку*/
                     ErrorForm2->ShowModal() ;
                     BuildModelProgressBar->Position = 100 ;
                     progressBarTimer->Enabled = true ;
                     return ;
                }
                *(x + i - 1) = InputForm->TableData-
>Cells[1][i].ToDouble() ;
                *(y + i - 1) = InputForm->TableData-
>Cells[2][i].ToDouble() ;
           }
           /*Перевірка чи не порожні поля із значеннями витрат*/
           if ((InputForm->InputDataIndependCost->Text.IsEmpty() ||
              (InputForm->InputDataIndependCost->Text.ToDouble() <
0)) ||
              (InputForm->InputDataDependCost->Text.IsEmpty() ||
              (InputForm->InputDataDependCost->Text.ToDouble() <
0)))
           {
                /*Виведення повідомлення про помилку*/
                ErrorForm3->ShowModal() ;
                BuildModelProgressBar->Position = 100 ;
                progressBarTimer->Enabled = true ;
                return ;
           }
           /*Перевірка коректності введених значень витрат*/
           if ((InputForm->InputDataDependCost->Text.ToDouble() ==
0) &&
              (InputForm->InputDataIndependCost->Text.ToDouble() ==
0))
           {
```

```
                /*Виведення повідомлення про помилку*/
                ErrorForm4->ShowModal() ;
                BuildModelProgressBar->Position = 100 ;
                progressBarTimer->Enabled = true ;
                return ;
            }
            BuildModelProgressBar->Position += 25;
            BuildModelComment->Caption = "Обчислення
коефіцієнтів..." ;
            /*Створення об'єкту, що зберігатиме дані можелювання*/
            MainForm->pAprObj = new CModeling(n, x, y,
            InputForm->InputDataIndependCost->Text.ToDouble(),
            InputForm->InputDataDependCost->Text.ToDouble()) ;
            /*Обчислення коефіцієнтів емпіричного рівняння*/
            MainForm->pAprObj->Aproximate() ;
            /*Виведення в головнуму вікні коефіцієнтів емпіричного
рівняння*/
            MainForm->outputA->Text = FormatFloat("0.000000",
            MainForm->pAprObj->coef_a) ;
            MainForm->outputB->Text = FormatFloat("0.000000",
            MainForm->pAprObj->coef_b) ;
            MainForm->outputC->Text = FormatFloat("0.000000",
            MainForm->pAprObj->coef_c) ;
            BuildModelProgressBar->Position += 25;
            BuildModelComment->Caption = "Виведення економічних
характеристик..." ;
            MainForm->MainFormObservOutput->Cells[1][0] = "Ціна" ;
            MainForm->MainFormObservOutput->Cells[2][0] = "Обсяг
продажу" ;
            MainForm->MainFormObservOutput->Cells[3][0] = "Теор.
обсяг продажу" ;
            MainForm->MainFormObservOutput->Cells[4][0] = "Коеф.
еластичності" ;
            MainForm->MainFormObservOutput->Cells[5][0] =
"Собівартість" ;
```

```
            MainForm->MainFormObservOutput->Cells[6][0] =
"Товарообіг" ;
            MainForm->MainFormObservOutput->Cells[7][0] =
"Прибуток" ;
            for (int i = 1 ; i < n+1 ; i++)
                  MainForm->MainFormObservOutput->Cells[0][i] = i ;
            MainForm->MainFormObservOutput->RowCount = n + 1 ;
            /*Обчислення економічних характеристик*/
            MainForm->pAprObj->getCalculations() ;
            /*Виведення таблиці з економічними характеристиками
моделей*/
            for (int i = 1 ; i < n+1 ; i++)
            {
                  /*Виведення цін*/
                  MainForm->MainFormObservOutput->Cells[1][i] =
                    MainForm->pAprObj->x[i-1] ;
                  /*Виведення попиту*/
                  MainForm->MainFormObservOutput->Cells[2][i] =
                    MainForm->pAprObj->y[i-1] ;
                  /*Виведення значеннь зрівноваженої функції*/
                  MainForm->MainFormObservOutput->Cells[3][i] =
FormatFloat("0.000000", MainForm->pAprObj->calcY[i-1]) ;
                  /*Виведення коефіцієнту еластичності*/
                  MainForm->MainFormObservOutput->Cells[4][i] =
FormatFloat("0.000000", MainForm->pAprObj->coefEl[i-1]) ;
                  /*Вивеення собівартості*/
                  MainForm->MainFormObservOutput->Cells[5][i] =
FormatFloat("0.000000", MainForm->pAprObj->cost[i-1]) ;
                  /*Виведення товарообігу*/
                  MainForm->MainFormObservOutput->Cells[6][i] =
FormatFloat("0.000000", MainForm->pAprObj->trade[i-1]) ;
                  /*Виведення прибутку*/
                  MainForm->MainFormObservOutput->Cells[7][i] =
FormatFloat("0.000000", MainForm->pAprObj->profit[i-1]) ;
                  /*Відображення економічних характеристик у вигляді
графіків*/
```

```
                MainForm->empericalData->AddXY(x[i-1], y[i-1]) ;
                MainForm->balancedData->AddXY(x[i-1],
                 MainForm->pAprObj->calcY[i-1]) ;
                MainForm->Cost->AddXY(x[i-1],
                 MainForm->pAprObj->cost[i-1]) ;
                MainForm->Trade->AddXY(x[i-1],
                 MainForm->pAprObj->trade[i-1]) ;
                MainForm->Profit->AddXY(x[i-1],
                 MainForm->pAprObj->profit[i-1]) ;
                MainForm->Demand->AddXY(x[i-1],
                 MainForm->pAprObj->calcY[i-1]) ;
                MainForm->StatisticDemand->AddXY(x[i-1], y[i-1]) ;
                MainForm->Trade_2->AddXY(x[i-1],
                 MainForm->pAprObj->trade[i-1]) ;
                MainForm->CoefEl->AddXY(x[i-1],
                 MainForm->pAprObj->coefEl[i-1]) ;

        }
        BuildModelProgressBar->Position += 25;
        BuildModelComment->Caption = "Обчислення критичних
значеннь економічних характеристик..." ;
        double p1, p2 ;
        int index = -1 ;
        double temp = sqrt(pow(MainForm->pAprObj->coef_b,2) -
3*MainForm->pAprObj->coef_a*MainForm->pAprObj->coef_c) ;
        /*Можливі критичні точки товарообігу*/
        p1 = (-MainForm->pAprObj->coef_b + temp)/(3*
              MainForm->pAprObj->coef_a) ;
        p2 = (-MainForm->pAprObj->coef_b - temp)/(3*
    MainForm->pAprObj->coef_a) ;
        /*Перевірка чи потрапляє знайдена критична точка у
          область досліджуваних значень*/
        if (p1 >= x[0] && p1 <= x[n-1])
        {
              for (int i = 0 ; i < n ; i++)
                    if (x[i] >= p1)
```

```
                {
                    index = i ;
                    break ;
                }
            /*Перевірка чи знайдена критична точка є точкою
мінімуму*/
            if ((MainForm->pAprObj->trade[index] <=
                MainForm->pAprObj->trade[0]) &&
            (MainForm->pAprObj->trade[index] <=
              MainForm->pAprObj->trade[n-1]))
            {
                /*Виведення значень економічних показників в
цій точці*/
                MainForm->NoneMinTrade->Visible = false ;
                MainForm->minTradePriceText->Visible = true ;
                MainForm->MinTradePrice->Visible = true ;
                MainForm->MinTradePrice->Text =
FormatFloat("0.000000", p1) ;
                MainForm->minTradeCoefElText->Visible = true ;
                MainForm->minTradeCoefEl->Visible = true ;
                MainForm->minTradeCoefEl->Text =
FormatFloat("0.000000", MainForm->pAprObj->getCoefEl(p1)) ;
                MainForm->minTradeValueText->Visible = true ;
                MainForm->minTradeVal->Visible = true ;
                MainForm->minTradeVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p1)) ;
            }
            /*Перевірка чи знайдена критична точка є точкою
максимуму*/
            if ((MainForm->pAprObj->trade[index] >=
                MainForm->pAprObj->trade[0]) &&
            (MainForm->pAprObj->trade[index] >=
              MainForm->pAprObj->trade[n-1]))
            {
                /*Виведення значень економічних показників в
цій точці*/
```

```
                    MainForm->NoneMaxTrade->Visible = false ;
                    MainForm->maxTradePriceText->Visible = true ;
                    MainForm->maxTradePrice->Visible = true ;
                    MainForm->maxTradePrice->Text =
FormatFloat("0.000000", p1) ;
                    MainForm->maxTradeCoefElText->Visible = true ;
                    MainForm->maxTradeCoefEl->Visible = true ;
                    MainForm->maxTradeCoefEl->Text =
FormatFloat("0.000000", MainForm->pAprObj->getCoefEl(p1)) ;
                    MainForm->maxTradeValueText->Visible = true ;
                    MainForm->maxTradeVal->Visible = true ;
                    MainForm->maxTradeVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p1)) ;
                }
            }
            /*Перевірка чи потрапляє знайдена критична точка у
              область досліджуваних значень*/
            if (p2 >= x[0] && p2 <= x[n-1])
            {
                for (int i = 0 ; i < n ; i++)
                    if (x[i] >= p2)
                    {
                        index = i ;
                        break ;
                    }
                /*Перевірка чи знайдена критична точка є точкою
мінімуму*/
                if ((MainForm->pAprObj->trade[index] <=
                     MainForm->pAprObj->trade[0]) &&
                    (MainForm->pAprObj->trade[index] <=
                     MainForm->pAprObj->trade[n-1]))
                {
                    /*Виведення значень економічних показників в
цій точці*/
                    MainForm->NoneMinTrade->Visible = false ;
                    MainForm->minTradePriceText->Visible = true ;
```

```
                    MainForm->MinTradePrice->Visible = true ;
                    MainForm->MinTradePrice->Text =
FormatFloat("0.000000", p2) ;
                    MainForm->minTradeCoefElText->Visible = true ;
                    MainForm->minTradeCoefEl->Visible = true ;
                    MainForm->minTradeCoefEl->Text =
FormatFloat("0.000000", MainForm->pAprObj->getCoefEl(p2)) ;
                    MainForm->minTradeValueText->Visible = true ;
                    MainForm->minTradeVal->Visible = true ;
                    MainForm->minTradeVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p2)) ;
                }
                /*Перевірка чи знайдена критична точка є точкою
максимуму*/
                if ((MainForm->pAprObj->trade[index] >=
                     MainForm->pAprObj->trade[0]) &&
                    (MainForm->pAprObj->trade[index] >=
                     MainForm->pAprObj->trade[n-1]))
                {
                    /*Виведення значень економічних показників в
цій точці*/
                    MainForm->NoneMaxTrade->Visible = false ;
                    MainForm->maxTradePriceText->Visible = true ;
                    MainForm->maxTradePrice->Visible = true ;
                    MainForm->maxTradePrice->Text =
FormatFloat("0.000000", p2) ;
                    MainForm->maxTradeCoefElText->Visible = true ;
                    MainForm->maxTradeCoefEl->Visible = true ;
                    MainForm->maxTradeCoefEl->Text =
FormatFloat("0.000000", MainForm->pAprObj->getCoefEl(p2)) ;
                    MainForm->maxTradeValueText->Visible = true ;
                    MainForm->maxTradeVal->Visible = true ;
                    MainForm->maxTradeVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p2)) ;
                }
            }
```

```
        double dv = 4 * (pow((MainForm->pAprObj->coef_b -
         MainForm->pAprObj->dependCost*MainForm->pAprObj-
      >coef_a), 2)   +  3 * MainForm->pAprObj->coef_a * (MainForm-
      >pAprObj->
      coef_b*MainForm->pAprObj->dependCost -
      MainForm->pAprObj->coef_c)) ;
            /*Обчислення можливих критичних точок прибутку*/
            p1 = (MainForm->pAprObj->dependCost *
                   MainForm->pAprObj->coef_a -
                   MainForm->pAprObj->coef_b + 0.5*sqrt(dv))/
                  (3*MainForm->pAprObj->coef_a) ;
            p2 = (MainForm->pAprObj->dependCost *
                   MainForm->pAprObj->coef_a -
                   MainForm->pAprObj->coef_b - 0.5*sqrt(dv))/
                 (3*MainForm->pAprObj->coef_a) ;
            /*Перевірка чи потрапляє знайдена критична точка у
            область досліджуваних значень*/
            if (p1 >= x[0] && p1 <=x[n-1])
            {
                  for (int i = 0 ; i < n ; i++)
                        if (x[i] >= p1)
                        {
                              index = i ;
                              break ;
                        }
                  /*Перевірка чи знайдена критична точка є точкою
мінімуму*/
                  if ((MainForm->pAprObj->profit[index] <=
                       MainForm->pAprObj->profit[0]) &&
                     (MainForm->pAprObj->profit[index] <=
                      MainForm->pAprObj->profit[n-1]))
                  {
                        /*Виведення значень економічних показників в
цій точці*/
                        MainForm->NoneMinProfit->Visible = false ;
                        MainForm->minProfitPriceText->Visible = true ;
```

```
                    MainForm->minProfitPrice->Visible = true ;
                    MainForm->minProfitPrice->Text =
FormatFloat("0.000000", p1) ;
                    MainForm->minProfitTradeText->Visible = true ;
                    MainForm->minProfitTrade->Visible = true ;
                    MainForm->minProfitTrade->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p1));
                    MainForm->minProfitValText->Visible = true ;
                    MainForm->minProfitVal->Visible = true ;
                    MainForm->minProfitVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getProfit(p1)) ;
                }
                /*Перевірка чи знайдена критична точка є точкою
максимуму*/
                if ((MainForm->pAprObj->profit[index] >=
                     MainForm->pAprObj->profit[0]) &&
                    (MainForm->pAprObj->profit[index] >=
                   MainForm->pAprObj->profit[n-1]))
                {
                    /*Виведення значень економічних показників в
цій точці*/
                    MainForm->NoneMaxProfit->Visible = false ;
                    MainForm->maxProfitPriceText->Visible = true ;
                    MainForm->maxProfitPrice->Visible = true ;
                    MainForm->maxProfitPrice->Text =
FormatFloat("0.000000", p1) ;
                    MainForm->maxProfitTradeText->Visible = true ;
                    MainForm->maxProfitTrade->Visible = true ;
                    MainForm->maxProfitTrade->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p1));
                    MainForm->maxProfitValText->Visible = true ;
                    MainForm->maxProfitVal->Visible = true ;
                    MainForm->maxProfitVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getProfit(p1)) ;
                }
            }
```

```
            /*Перевірка чи потрапляє знайдена критична точка у
            область досліджуваних значень*/
            if (p2 >= x[0] && p2 <=x[n-1])
            {
                for (int i = 0 ; i < n ; i++)
                    if (x[i] >= p2)
                    {
                        index = i ;
                        break ;
                    }
                /*Перевірка чи знайдена критична точка є точкою
мінімуму*/
                if ((MainForm->pAprObj->profit[index] <=
                     MainForm->pAprObj->profit[0]) &&
                    (MainForm->pAprObj->profit[index] <=
                     MainForm->pAprObj->profit[n-1]))
                {
                    /*Виведення значень економічних показників в
цій точці*/
                    MainForm->NoneMinProfit->Visible = false ;
                    MainForm->minProfitPriceText->Visible = true ;
                    MainForm->minProfitPrice->Visible = true ;
                    MainForm->minProfitPrice->Text =
FormatFloat("0.000000", p2) ;
                    MainForm->minProfitTradeText->Visible = true ;
                    MainForm->minProfitTrade->Visible = true ;
                    MainForm->minProfitTrade->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p2));
                    MainForm->minProfitValText->Visible = true ;
                    MainForm->minProfitVal->Visible = true ;
                    MainForm->minProfitVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getProfit(p2)) ;
                }
                /*Перевірка чи знайдена критична точка є точкою
максимуму*/
                if ((MainForm->pAprObj->profit[index] >=
```

```
                    MainForm->pAprObj->profit[0]) &&
                   (MainForm->pAprObj->profit[index] >=
                    MainForm->pAprObj->profit[n-1]))
                {
                    /*Виведення значень економічних показників в
цій точці*/
                    MainForm->NoneMaxProfit->Visible = false ;
                    MainForm->maxProfitPriceText->Visible = true ;
                    MainForm->maxProfitPrice->Visible = true ;
                    MainForm->maxProfitPrice->Text =
FormatFloat("0.000000", p2) ;
                    MainForm->maxProfitTradeText->Visible = true ;
                    MainForm->maxProfitTrade->Visible = true ;
                    MainForm->maxProfitTrade->Text =
FormatFloat("0.000000", MainForm->pAprObj->getTrade(p2));
                    MainForm->maxProfitValText->Visible = true ;
                    MainForm->maxProfitVal->Visible = true ;
                    MainForm->maxProfitVal->Text =
FormatFloat("0.000000", MainForm->pAprObj->getProfit(p2)) ;
                }
            }
            BuildModelProgressBar->Position += 25;
            BuildModelComment->Caption = "Обчислення
характеристик побудованої моделі..." ;
            MainForm->ModelCharactTable->RowCount = n + 1 ;
            MainForm->ModelCharactTable->Cells[1][0] = "Ціна" ;
            MainForm->ModelCharactTable->Cells[2][0] = "Теор. обсяг
продажу" ;
            MainForm->ModelCharactTable->Cells[3][0] = "Абс.
похибки" ;
            MainForm->ModelCharactTable->Cells[4][0] = "1/P" ;
            MainForm->ModelCharactTable->Cells[5][0] = "m(p)" ;
            MainForm->ModelCharactTable->Cells[6][0] =
"Конструювання" ;
            /*Обчислення математичних характеристик моделі*/
            MainForm->pAprObj->getModelCharacts() ;
```

```
            /*Виведення таблиці з характеристиками моделей*/
            for (int i = 1 ; i < n+1 ; i++)
            {
                MainForm->ModelCharactTable->Cells[0][i] = i ;
                /*Ціна*/
                MainForm->ModelCharactTable->Cells[1][i] = x[i-1] ;
                /*Значення зрівноваженої функції*/
                MainForm->ModelCharactTable->Cells[2][i] =
FormatFloat("0.000000", MainForm->pAprObj->calcY[i-1]) ;
                /*Абсолютні похибки*/
                MainForm->ModelCharactTable->Cells[3][i] =
FormatFloat("0.000000", MainForm->pAprObj->abs_error[i-1]) ;
                /*Обернені ваги*/
                MainForm->ModelCharactTable->Cells[4][i] =
FormatFloat("0.000000", MainForm->pAprObj->abs_error_weight[i-1])
;
                /*Квадратичні похибки*/
                MainForm->ModelCharactTable->Cells[5][i] =
FormatFloat("0.000000", MainForm->pAprObj->square_errors[i-1]) ;
                /*Значення функції з урахуванням похибок*/
                MainForm->ModelCharactTable->Cells[6][i] =
FormatFloat("0.000000", MainForm->pAprObj->refinedY[i-1]) ;
                /*Відображення характеристик моделі у вигляді
графіку*/
                MainForm->InversWeights->AddXY(x[i-1],
                MainForm->pAprObj->abs_error_weight[i-1]) ;
                MainForm->AbsoluteErrors->AddXY(x[i-1],
                MainForm->pAprObj->abs_error[i-1]) ;
                MainForm->MeanSquareErrors->AddXY(x[i-1],
                MainForm->pAprObj->square_errors[i-1]) ;
            }
            /*Таблиця коефіцієнтів емпіричної функції*/
            MainForm->CoefCharactTable->Cells[0][1] = "a" ;
            MainForm->CoefCharactTable->Cells[0][2] = "b" ;
            MainForm->CoefCharactTable->Cells[0][3] = "c" ;
            MainForm->CoefCharactTable->Cells[1][0] = "1/p" ;
```

```
            MainForm->CoefCharactTable->Cells[2][0] = "m(коеф.)" ;
            MainForm->CoefCharactTable->Cells[3][0] = "t(коеф.)" ;
            /*Обернені ваги коефіцієнтів*/
            MainForm->CoefCharactTable->Cells[1][1] =
FormatFloat("0.000000", MainForm->pAprObj->inv_weight_a) ;
            MainForm->CoefCharactTable->Cells[1][2] =
FormatFloat("0.000000", MainForm->pAprObj->inv_weight_b) ;
            MainForm->CoefCharactTable->Cells[1][3] =
FormatFloat("0.000000", MainForm->pAprObj->inv_weight_c) ;
            /*Середні квадратичні похибки коефіцієнтів*/
            MainForm->CoefCharactTable->Cells[2][1] =
FormatFloat("0.000000", MainForm->pAprObj->m_a) ;
            MainForm->CoefCharactTable->Cells[2][2] =
FormatFloat("0.000000", MainForm->pAprObj->m_b) ;
            MainForm->CoefCharactTable->Cells[2][3] =
FormatFloat("0.000000", MainForm->pAprObj->m_c) ;
            /*Значимість коефіцієнтів*/
            MainForm->CoefCharactTable->Cells[3][1] =
FormatFloat("0.000000", MainForm->pAprObj->t_a) ;
            MainForm->CoefCharactTable->Cells[3][2] =
FormatFloat("0.000000", MainForm->pAprObj->t_b) ;
            MainForm->CoefCharactTable->Cells[3][3] =
FormatFloat("0.000000", MainForm->pAprObj->t_c) ;
            /*Контроль зрівноваження*/
            MainForm->controlCompensationLeftSide->Text =
            MainForm->pAprObj->controlCompensLftSd ;
            MainForm->controlCompensationRightSide->Text =
            MainForm->pAprObj->controlCompensRghtSd ;
            MainForm->controlCompensationDelta->Text =
             MainForm->pAprObj->controlCompensDelta ;
            /*Закриття форми введення спостережень*/
            InputForm->Close();
            MainForm->MainFormPageControl->Visible = true ;
            /*Вивільнення динамічної пам'яті*/
            delete []x ;
            delete []y ;
```

```
        x = 0x000000 ;
        y = 0x000000 ;
        BuildModelProgressBar->Position = 100 ;
        progressBarTimer->Enabled = true ;
    }
```

Якщо обробка пройшла успішно, то панель прогресу закривається разом із формою введення даних, а головне вікно знову стає активним.

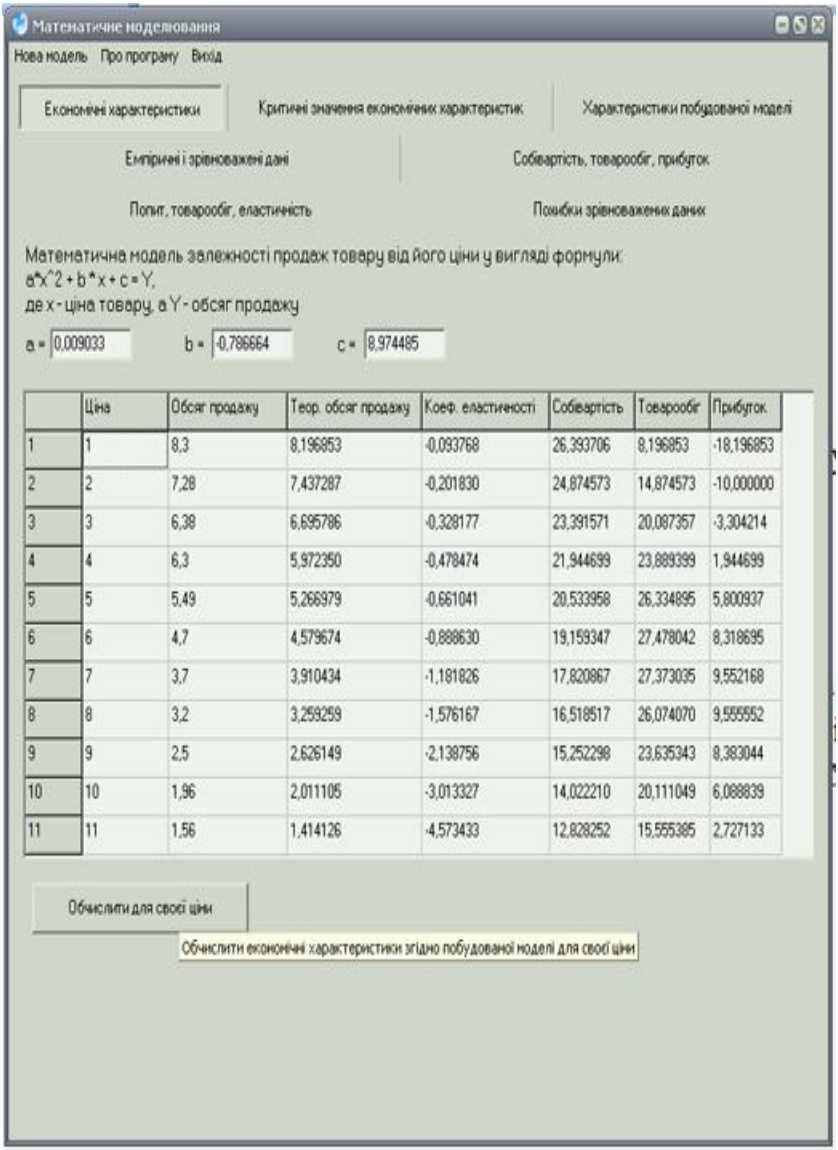
Тепер у головному вікні відображаються усі обчислені дані, для зручності їх розділено на 7 вкладок:

- Економічні характеристики
- Критичні значення економічних характеристик
- Характеристики побудованої моделі
- Емпіричні і зрівноважені дані (Додаток Д)
- Собівартість, товарообіг, прибуток (Додаток Е)
- Попит, товарообіг, еластичність (Додаток Ж)
- Похибки зрівноважених даних (Додаток З)

У перших трьох вкладках дані відображаються в табличному вигляді, а у інших чотирьох у вигляді графіків.

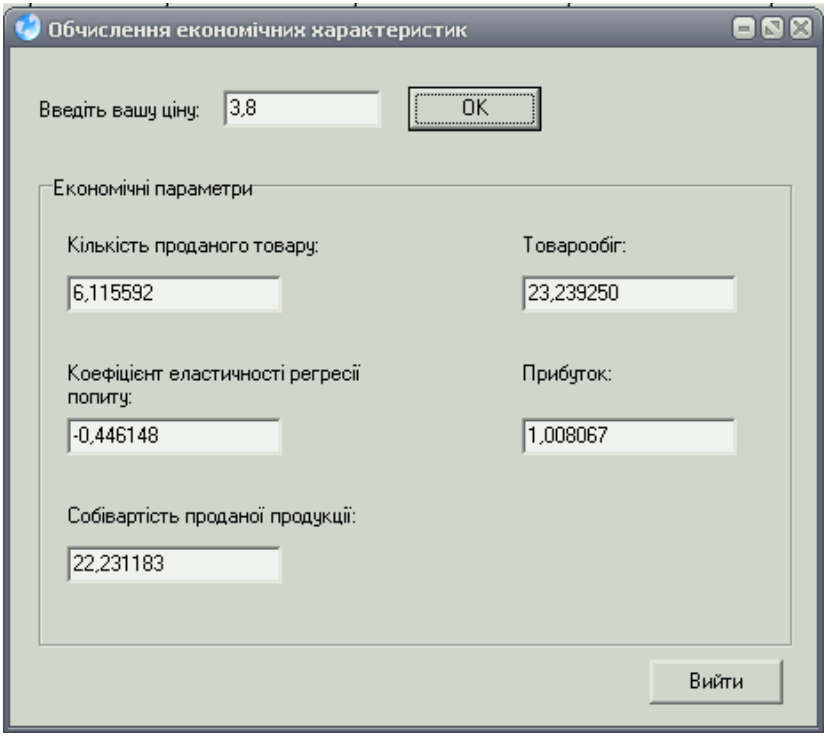
У вкладці «Економічні характеристики» розміщені такі дані: коефіцієнти емпіричної функції, ціна товару, обсяг продажу, теоретичний обсяг продажу, коефіцієнт еластичності попиту, собівартість реалізованої продукції, товарообіг, прибуток (рис. 7).

Також, передбачена можливість обрахувати усі вище зазначені характеристики (крім коефіцієнтів) для довільної ціни. Для цього під таблицею реалізована кнопка «Обчислити для своєї ціни», що викликає відповідну форму(рис. 8). Після вводу ціни і натискання кнопки «ОК» відбувається обчислення і виведення потрібних значень. У формі також реалізовано контроль, якщо користувач введе некоректне значення, наприклад, від'ємне, то форма видасть повідомлення про помилку із роз'ясненням яким повинен бути коректний ввід. Лістинг коду даної форми розміщений у додатку К.



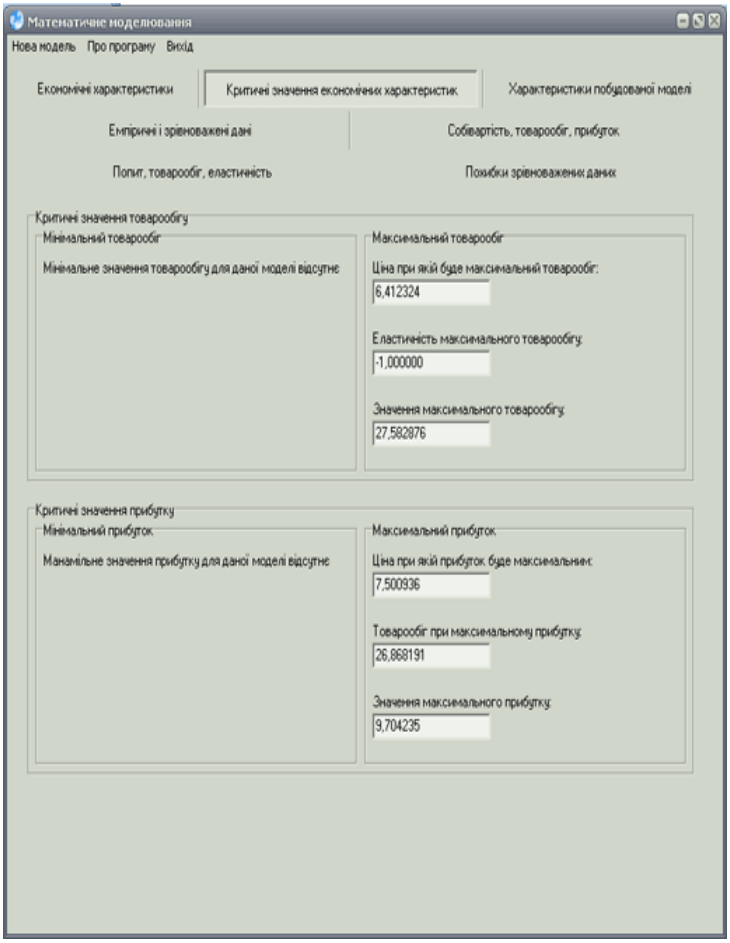

| | Ціна | Обсяг продажу | Теор. обсяг продажу | Коеф. еластичності | Собівартість | Товарообіг | Прибуток |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 8,3 | 8,196853 | -0,093768 | 26,393706 | 8,196853 | -18,196853 |
| 2 | 2 | 7,28 | 7,437287 | -0,201830 | 24,874573 | 14,874573 | -10,000000 |
| 3 | 3 | 6,38 | 6,695786 | -0,328177 | 23,391571 | 20,087357 | -3,304214 |
| 4 | 4 | 6,3 | 5,972350 | -0,478474 | 21,944699 | 23,889399 | 1,944699 |
| 5 | 5 | 5,49 | 5,266979 | -0,661041 | 20,533958 | 26,334895 | 5,800937 |
| 6 | 6 | 4,7 | 4,579674 | -0,888630 | 19,159347 | 27,478042 | 8,318695 |
| 7 | 7 | 3,7 | 3,910434 | -1,181826 | 17,820867 | 27,373035 | 9,552168 |
| 8 | 8 | 3,2 | 3,259259 | -1,576167 | 16,518517 | 26,074070 | 9,555552 |
| 9 | 9 | 2,5 | 2,626149 | -2,138756 | 15,252298 | 23,635343 | 8,383044 |
| 10 | 10 | 1,96 | 2,011105 | -3,013327 | 14,022210 | 20,111049 | 6,088839 |
| 11 | 11 | 1,56 | 1,414126 | -4,573433 | 12,828252 | 15,555385 | 2,727133 |

(Рис. 7)



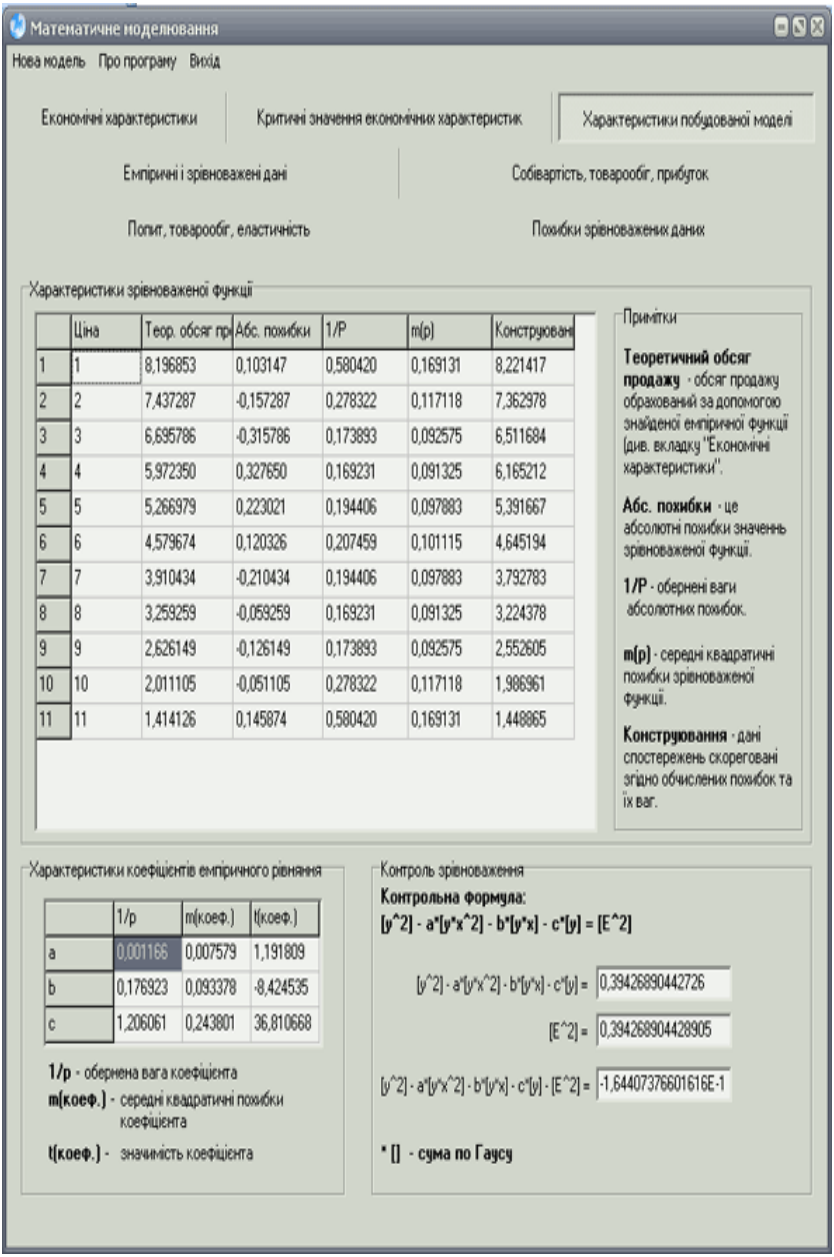

(Рис. 8)

У вкладці «Критичні значення економічних характеристик» виводяться мінімальні і максимальні значення прибутку і товарообігу, якщо такі існують. Якщо ж їх немає, то у формі виводиться відповідне повідомлення (рис. 9)




(Рис.9)

Вкладка «Характеристики побудованої моделі» розділена на три основних блока: «Характеристики зрівноваженої функції», «Характеристики коефіцієнтів емпіричного рівняння» і «Контроль зрівноваження». У зв'язку з тим, що в назвах колонок таблиці є багато скорочень, поряд із табличними даними знаходяться примітки, що коротко описують їх значення. (рис. 10)




| | Ціна | Теор. обсяг пр | Абс. похибки | 1/P | m(p) | Конструювані |
|---|---|---|---|---|---|---|
| 1 | 1 | 8,196853 | 0,103147 | 0,580420 | 0,169131 | 8,221417 |
| 2 | 2 | 7,437287 | -0,157287 | 0,278322 | 0,117118 | 7,362978 |
| 3 | 3 | 6,695786 | -0,315786 | 0,173893 | 0,092575 | 6,511684 |
| 4 | 4 | 5,972350 | 0,327650 | 0,169231 | 0,091325 | 6,165212 |
| 5 | 5 | 5,266979 | 0,223021 | 0,194406 | 0,097883 | 5,391667 |
| 6 | 6 | 4,579674 | 0,120326 | 0,207459 | 0,101115 | 4,645194 |
| 7 | 7 | 3,910434 | -0,210434 | 0,194406 | 0,097883 | 3,792783 |
| 8 | 8 | 3,259259 | -0,059259 | 0,169231 | 0,091325 | 3,224378 |
| 9 | 9 | 2,626149 | -0,126149 | 0,173893 | 0,092575 | 2,552605 |
| 10 | 10 | 2,011105 | -0,051105 | 0,278322 | 0,117118 | 1,986961 |
| 11 | 11 | 1,414126 | 0,145874 | 0,580420 | 0,169131 | 1,448865 |

| | 1/p | m(коеф.) | t(коеф.) |
|---|---|---|---|
| a | 0,001166 | 0,007579 | 1,191809 |
| b | 0,176923 | 0,093378 | -8,424535 |
| c | 1,206061 | 0,243801 | 36,810668 |

(Рис. 10)

# ВИСНОВКИ

**В ході виконання даної роботи було проаналізовано засоби мови С++, які можуть використовуватись у математичному моделюванні і на основі отриманої інформації розроблено повноцінний програмний продукт, що:**

- дозволяє моделювати та досліджувати складні економічні явища на основі введених користувачем даних;
- встановлює залежність між ціною на товар та попитом на основі формули:

$$y = ax^2 + bx + c,$$

де $y$ – попит на товар, а $x$ – ціна одиниці товару;

- обчислює ряд економічних характеристик змодельованого явища, таких як еластичність регресії попиту, собівартість реалізованої продукції, товарообіг, прибуток від проданого товару;
- визначає критичні значення прибутку і товарообігу, якщо такі є;
- обчислює характеристики значень зрівноваженої функції: обернені ваги, абсолютні похибки та середні квадратичні похибки;
- обчислює характеристики коефіцієнтів функції залежності попиту від ціни: їх обернені ваги, середні квадратичні похибки та значимість;
- проводить контроль зрівноваження;
- наочно демонструє залежність економічних показників один від одного за допомогою графіків;
- здійснює контроль над коректністю введених користувачем даних.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. URL http://mat.1september.ru/2003/14/no14_1.htm (12 лютого 2012)
2. Літнарович Р.М. Основи наукових досліджень. Аналіз індивідуального ринку. МЕГУ, Рівне, 2010.-4с.
3. Б. Страуструп. 2.1. Что такое C++? // Язык программирования C++. Указ. соч. — С. 57
4. Stroustrup, Bjarne URL http://www2.research.att.com/~bs/glossary.html (12 лютого 2012)
5. ISO/IEC 14882:1998, розділ 6.4, пункт 4: «The value of a condition that is an initialized declaration in a statement other than a switch statement is the value of the declared variable implicitly converted to bool … The value of a condition that is an expression is the value of the expression, implicitly converted to bool for statements other than switch; if that conversion is ill-formed, the program is ill-formed».
6. Айвор Хортон «Visual C++ 2010 полный курс», Москва, 2010
7. Бугір М.К. Математика для економістів.Посібник.-К.:Видавничий центр «Академія»,2003,-520 с.
8. Літнарович Р.М. Побудова і дослідження економіко-математичної моделі поліномом m-го порядку.Вісник МЕГУ.Збірник наукових праць.Серія: Системні науки та кібернетика. Випуск 1.МЕГУ, Рівне,2009.- с.41-51.
9. Літнарович Р.М. Конструювання і дослідження математичних моделей. Множинний аналіз. Частина 1. МЕГУ, Рівне, 2009.-127с.
10. Ник Рендольф, Дэвид Гарднер, Майкл Минутилло, Крис Андерсон Visual Studio 2010 для профессионалов = Professional Visual Studio 2010. — М.: «Диалектика», 2011. — С. 1184.
11. Герберт Шилдт Полный справочник по C++, 4-е издание = C++: The Complete Reference, 4th Edition. — М.: «Вильямс», 2011. — С. 800
12. URL http://ru.wikipedia.org/wiki/%D0%A1%2B%2B
13. URL http://ru.wikipedia.org/wiki/Microsoft_Visual_Studio

# ДОДАТКИ

## Додаток А

(лістинг коду бібліотеки CMatrix.h)

```
#if !defined ( __matrix_h )
#define __matrix_h

#include <istream>
#include <ostream>
#include <fstream>
#include "mathematics.h"
/*прототип шаблонного класу CMatrix*/
template <class T> class CMatrix;
/*перевантажений оператор +*/
template <class T> const CMatrix<T>& operator +( const CMatrix<T>
&, const CMatrix<T> & );
/*перевантажений оператор - */
template <class T> const CMatrix<T>& operator -( const CMatrix<T>
&, const CMatrix<T> & );
/*перевантажені оператори * */
template <class T> const CMatrix<T>& operator *( const CMatrix<T>
&, const CMatrix<T> & );
template <class T> const CMatrix<T>& operator *( const CMatrix<T>
&, const T & );
template <class T> const CMatrix<T>& operator *( const T &, const
CMatrix<T> & );
/*перевантажений оператор / */
template <class T> const CMatrix<T>& operator /( const CMatrix<T>
&, const CMatrix<T> & );
/*функція, що повертає визначник матриці*/
template <class T> T Determinant( const CMatrix<T> & );
/*функція, що повертая обернену матрицю*/
template <class T> CMatrix<T>& Inverse( const CMatrix<T> & );
/*функція, що повертає мінор матриці*/
template <class T> CMatrix<T>& Minor( const CMatrix<T> &,
unsigned int = 0u, unsigned int = 0u);
```

```
/*функція, що повертає транспоновану матрицю*/
template <class T> CMatrix<T>& Transpose( const CMatrix<T> & );

template <class T>
   class CMatrix
   {
      private:

         /*допоміжний деструктор*/
         void Destruction( void );
         /*перевантажений оператор >>*/
         friend std::istream& operator >>( std::istream &input,
CMatrix<T> &InM )
         {
            unsigned int limit_i = InM.getRows();
            unsigned int limit_j = InM.getColumns();

            for ( unsigned int i = 0u; i < limit_i; i++ )
               for ( unsigned int j = 0u; j < limit_j; j++ )
               {
                  input >> InM(i, j);
               }

            return( input );
         }
         /*перевантажений оператор <<*/
         friend std::ostream& operator <<( std::ostream &output, const
CMatrix<T> &OutM )
         {
            unsigned int limit_i = OutM.getRows();
            unsigned int limit_j = OutM.getColumns();

            for ( unsigned int i = 0u; i < limit_i; i++ )
            {
               for ( unsigned int j = 0u; j < limit_j; j++ )
                  output << OutM(i, j) << ' ';
```

```
            output << '\n';
        }

        return( output );
    }

public:
        /*допоміжний конструктор*/
        void Construction( unsigned int, unsigned int );
    /*конструктор по замовчуванню*/
    CMatrix( unsigned int = 1u, unsigned int = 1u );
    /*конструктор копіювання*/
    CMatrix( const CMatrix<T> & );
    /*конструктор*/
    CMatrix( const char * );
    /*перевантажений оператор =*/
    const CMatrix<T>& operator  =( const CMatrix<T> & );
    /*перевантажений оператор +=*/
    const CMatrix<T>& operator +=( const CMatrix<T> & );
    /*перевантажений оператор -=*/
    const CMatrix<T>& operator -=( const CMatrix<T> & );
    /*перевантажений оператор *= */
    const CMatrix<T>& operator *=( const CMatrix<T> & );
    const CMatrix<T>& operator *=( const T & );
    /*перевантажений оператор /=*/
    const CMatrix<T>& operator /=( const CMatrix<T> & );
    const CMatrix<T>& operator /=( const T & );
    /*деструктор*/
    ~CMatrix( void );
    /*функція, що повертає кількість рядків в матриці*/
    unsigned int getRows( void )    const { return( this->rows ); }
    /*функція що повертає кількість стовпців в матриці*/
    unsigned int getColumns( void ) const { return( this->columns );
}
    /*функція що повертає кількість елементів матриці*/
```

```
        unsigned int getNumOfElem( void ) const { return(( *this
).getRows() * ( *this ).getColumns()); }
        /*перевантажений оператор () */
        const T& operator ()( unsigned int i, unsigned int j ) const {
return( *( this->p + ( this->columns * i + j ))); }
              T& operator ()( unsigned int i, unsigned int j )       { return( *(
this->p + ( this->columns * i + j ))); }

    private:
        /* кількість рядків і стовпців*/
        unsigned int rows, columns;
        T* p;       //вказівник на матрицю
  };
/*реалізація вищезазначених функцій*/
template <class T>
  void CMatrix<T>::Construction( unsigned int rows, unsigned int
columns )
  {
    this->Destruction();

    this->rows      = (rows < 1u) ? 1u : rows;
    this->columns   = ( columns < 1u ) ? 1u : columns;

    unsigned int elements = this->rows * this->columns;

    this->p = new T[ elements ];

    for ( unsigned int i = 0u; i < elements; i++ )
      *( this->p + i ) = T( );

    return;
  }

template <class T>
  inline void CMatrix<T>::Destruction( void )
  {
```

```
    if ( this->p )
       delete this->p;

    this->rows    = 0u;
    this->columns = 0u;
    this->p       = 0x00000000;

    return;
  }

template <class T>
  inline CMatrix<T>::CMatrix( unsigned int rows, unsigned int columns
)
  {
    this->rows    = 0u;
    this->columns = 0u;
    this->p       = 0x00000000;

    this->Construction( rows, columns );
  }

template <class T>
  CMatrix<T>::CMatrix( const CMatrix<T> &other )
  {
    if (( this != &other ) && ( &other ))
    {
      this->rows    = 0u;
      this->columns = 0u;
      this->p       = 0x00000000;

      this->Construction( other.getRows(), other.getColumns() );

      unsigned int limit_i = ( *this ).getRows();
      unsigned int limit_j = ( *this ).getColumns();

      for ( unsigned int i = 0u; i < limit_i; i++ )
```

```
            for ( unsigned int j = 0u; j < limit_j; j++ )

                ( *this ).operator()(i, j) = other.operator()(i, j);
        }

        this->rows      = 0u;
        this->columns   = 0u;
        this->p         = 0x00000000;
    }

template <class T>
    CMatrix<T>::CMatrix( const char *filename )
    {
        this->rows      = 0u;
        this->columns   = 0u;
        this->p         = 0x00000000;

        unsigned int number_of_rows     = 0u;
        unsigned int number_of_columns  = 0u;
        unsigned int number_of_elements = 0u;

        std::ifstream inFile( filename );

        if ( inFile )
        {
            char ch;
            while ( inFile.get( ch ) )
            {
                if ( ch == '\t' ) number_of_elements ++;
                if ( ch == '\n' ) number_of_rows++;
            }

            number_of_rows++;
            number_of_elements += number_of_rows;

            number_of_columns = number_of_elements/number_of_rows;
```

```
        this->Construction( number_of_rows, number_of_columns );

        inFile.close();

        std::ifstream inFile( filename );

        if ( inFile )
           for ( unsigned int i = 0u; i < number_of_rows; i++ )
              for ( unsigned int j = 0u; j < number_of_columns; j++ )
                 inFile >> ( *this )(i, j);

        inFile.close();
     }

  }

template <class T>
  const CMatrix<T>& CMatrix<T>::operator =( const CMatrix<T>
&rhs )
  {
     if (( this != &rhs ) && ( &rhs ))
     {
        if ( ( *this ).getRows() != rhs.getRows() || ( *this ).getColumns()
!= rhs.getColumns() )

           this->Construction( rhs.getRows(), rhs.getColumns() );

        unsigned int limit_i = ( *this ).getRows();
        unsigned int limit_j = ( *this ).getColumns();

        for ( unsigned int i = 0u; i < limit_i; i++ )
           for ( unsigned int j = 0u; j < limit_j; j++ )

              ( *this ).operator()(i, j) = rhs.operator()(i, j);
     }
```

```
      return( *this );
   }

template <class T>
   const CMatrix<T>& CMatrix<T>::operator +=( const CMatrix<T>
&rhs )
   {
      if ( this->rows == rhs.getRows() && this->columns ==
rhs.getColumns() )
      {
         unsigned int limit_i = this->rows;
         unsigned int limit_j = this->columns;

         for ( unsigned int i = 0u; i < limit_i; i++ )
            for ( unsigned int j = 0u; j < limit_j; j++ )

               ( *this )(i, j) += rhs(i, j);
      }

      return( *this );
   }

template <class T>
   const CMatrix<T>& CMatrix<T>::operator -=( const CMatrix<T>
&rhs )
   {
      if ( this->rows == rhs.getRows() && this->columns ==
rhs.getColumns() )
      {
         unsigned int limit_i = this->rows;
         unsigned int limit_j = this->columns;

         for ( unsigned int i = 0u; i < limit_i; i++ )
            for ( unsigned int j = 0u; j < limit_j; j++ )
```

```
                ( *this )(i, j) -= rhs(i, j);
        }

        return( *this );
    }

template <class T>
    const CMatrix<T>& CMatrix<T>::operator *=( const CMatrix<T>
&rhs )
    {
        if ( this->columns == rhs.getRows() )
        {
            CMatrix<T> *pM = new CMatrix<T>( this->rows,
rhs.getColumns() );

            unsigned int limit_i = pM->getRows();
            unsigned int limit_j = pM->getColumns();
            unsigned int limit_k = this->columns;   // rhs.getRows();

            for ( unsigned int i = 0u; i < limit_i; i++ )
                for ( unsigned int j = 0u; j < limit_j; j++ )
                    for ( unsigned int k = 0u; k < limit_k; k++ )

                        ( *pM )(i, j) += ( *this )(i, k) * rhs(k, j);

            this->Destruction();
            ( *this ) = ( *pM );
        }

        return( *this );
    }

template <class T>
    const CMatrix<T>& CMatrix<T>::operator *=( const T &rhs )
    {
        CMatrix<T> *pM = new CMatrix<T>( *this );
```

```
      unsigned int limit_i = this->rows;
      unsigned int limit_j = this->columns;

      for ( unsigned int i = 0u; i < limit_i; i++ )
         for ( unsigned int j = 0u; j < limit_j; j++ )

            ( *this )(i, j) += ( *pM )(i, j) * rhs;

      return( *this );
   }

template <class T>
   inline CMatrix<T>::~CMatrix( void )
   {
      this->Destruction();
   }

template <class T>
   const CMatrix<T>& operator +( const CMatrix<T> &lhs, const
CMatrix<T> &rhs )
   {
      CMatrix<T> *pM = 0x00000000;

      if ( lhs.getRows() == rhs.getRows() && lhs.getColumns() ==
rhs.getColumns() )
      {
         pM = new CMatrix<T>( lhs.getRows(), rhs.getColumns() );

         unsigned int limit_i = pM->getRows();
         unsigned int limit_j = pM->getColumns();

         for ( unsigned int i = 0u; i < limit_i; i++ )
            for ( unsigned int j = 0u; j < limit_j; j++ )

               ( *pM )(i, j) = lhs(i, j) + rhs(i, j);
```

```
    }

    return( *pM );
  }

template <class T>
  const CMatrix<T>& operator -( const CMatrix<T> &lhs, const
CMatrix<T> &rhs )
  {
    CMatrix<T> *pM = 0x00000000;

    if ( lhs.getRows() == rhs.getRows() && lhs.getColumns() ==
rhs.getColumns() )
    {
      pM = new CMatrix<T>( lhs.getRows(), rhs.getColumns() );

      unsigned int limit_i = pM->getRows();
      unsigned int limit_j = pM->getColumns();

      for ( unsigned int i = 0u; i < limit_i; i++ )
        for ( unsigned int j = 0u; j < limit_j; j++ )

          ( *pM )(i, j) = lhs(i, j) - rhs(i, j);
    }

    return( *pM );
  }

template <class T>
  const CMatrix<T>& operator *( const CMatrix<T> &lhs, const
CMatrix<T> &rhs )
  {
    CMatrix<T> *pM = 0x00000000;

    if ( lhs.getColumns() == rhs.getRows() )
    {
```

```
        pM = new CMatrix<T>( lhs.getRows(), rhs.getColumns() );

        unsigned int limit_i = pM->getRows();
        unsigned int limit_j = pM->getColumns();
        unsigned int limit_k = lhs.getColumns();    // rhs.getRows();

        for ( unsigned int i = 0u; i < limit_i; i++ )
           for ( unsigned int j = 0u; j < limit_j; j++ )
              for ( unsigned int k = 0u; k < limit_k; k++ )

                 ( *pM )(i, j) += lhs(i, k) * rhs(k, j);
     }

     return( *pM );
  }

template <class T>
  const CMatrix<T>& operator *( const CMatrix<T> &lhs, const T
&rhs )
  {
     CMatrix<T> *pM = new CMatrix<T>( lhs.getRows(),
lhs.getColumns() );

     unsigned int limit_i = pM->getRows();
     unsigned int limit_j = pM->getColumns();

     for ( unsigned int i = 0u; i < limit_i; i++ )
        for ( unsigned int j = 0u; j < limit_j; j++ )

           ( *pM )(i, j) += lhs(i, j) * rhs;

     return( *pM );
  }

template <class T>
```

```
   const CMatrix<T>& operator *( const T &lhs, const CMatrix<T>
&rhs )
   {
      CMatrix<T> *pM = new CMatrix<T>( rhs.getRows(),
rhs.getColumns() );

      unsigned int limit_i = pM->getRows();
      unsigned int limit_j = pM->getColumns();

      for ( unsigned int i = 0u; i < limit_i; i++ )
         for ( unsigned int j = 0u; j < limit_j; j++ )

            ( *pM )(i, j) += lhs * rhs(i, j);

      return( *pM );
   }

template <class T>
   T Determinant( const CMatrix<T> &M )
   {
      T result = T();

      unsigned int elements = M.getRows() * M.getColumns();

      if ( M.getRows() == M.getColumns() )
      {
         switch ( elements )
         {
            case 1u: result = M(0u, 0u); break;
            case 4u: result = M(0u, 0u) * M(1u, 1u) - M(0u, 1u) * M(1u,
0u); break;

            default:
            {
               unsigned int limit_j = M.getColumns();
```

```
                for ( unsigned int j = 0u; j < limit_j; j++ )
                    result += M( 0u, j ) * power(( -1.0 ), ( signed int )( 2u + j
)) * Determinant( Minor( M, 0u, j ) );
            }
            break;

        }
    }

    return( result );
}

template <class T>
CMatrix<T>& Inverse( const CMatrix<T> &M )
{
    CMatrix<T> ( *pM ) = 0x00000000;

    T DetM = Determinant( M );

    if ( M.getRows() == M.getColumns() )
    {
        pM = new CMatrix<T>( M.getRows(), M.getColumns() );

        unsigned int limit_i = pM->getRows();
        unsigned int limit_j = pM->getColumns();

        for ( unsigned int i = 0u; i < limit_i; i++ )
            for ( unsigned int j = 0u; j < limit_j; j++ )
                ( *pM )(i, j) = power(( -1.0 ), ( signed int )( 2u + i + j )) *
Determinant( Minor( M, i, j ) );

        ( *pM ) = (( 1/DetM ) * Transpose( *pM  ));
    }

    return( *pM );
}
```

```
template <class T>
   CMatrix<T>& Minor( const CMatrix<T> &M, unsigned int fix_row,
unsigned int fix_column )
   {
      CMatrix<T> *pM = new CMatrix<T>( M.getRows()-1,
M.getColumns()-1 );

      unsigned int limit_i = pM->getRows();
      unsigned int limit_j = pM->getColumns();

      for ( unsigned int i = 0u, a = 0u; i < limit_i; i++, a++ )
         for ( unsigned int j = 0u, b = 0u; j < limit_j; j++, b++ )
         {
            if ( a == fix_row ) a++;
            if ( b == fix_column ) b++;

            ( *pM )(i, j) = M(a, b);
         }

      return( *pM );
   }

template <class T>
   CMatrix<T>& Transpose( const CMatrix<T> &M )
   {
      CMatrix<T> ( *pM ) = 0x00000000;

      if ( &M )
      {
         ( pM ) = new CMatrix<T>( M.getColumns(), M.getRows() );

         unsigned int limit_i = ( pM )->getRows();
         unsigned int limit_j = ( pM )->getColumns();

         for ( unsigned int i = 0u; i < limit_i; i++ )
```

```
            for ( unsigned int j = 0u; j < limit_j; j++ )
               ( pM )->operator ()(i, j) = M(j, i);
        }

        return( *pM );
    }

#endif
```

**Додаток Б**

(лістинг коду бібліотеки Modeling.h)

```
#include <math.h>

#if !defined ( modeling_h )
#define modeling_h

class CModeling
{
        public:
            int numOfObserv ; // кількість спостережень
                  double *x ;     // масив цін
                  double *y ;     //масив попитів
            double *calcY ; //масив значень зрівноваженої функції
            double *coefEl ;        // коефіцієнт еластичності
            double *cost ;  //собівартість
            double *trade ; //товарообіг
            double *profit ;        //прибуток
            double *abs_error ;     //абсолютні похибки зрівноваженої ф-
ції
            double *abs_error_weight ;      // обернені ваги
            double *refinedY ;      // конструювання
            double *square_errors ; //середні квадратичні похибки
            double independCost, dependCost ;       // витрати
            double coef_a, coef_b, coef_c ; // коефіцієнти емпіричного
рівняння
```

```
        CMatrix<double> InversMatr ;    //обернена матриця
        double meanSquareError ;        //середня квадратична
похибка
        double m_a, m_b, m_c ;  //середні квадратичні похибки
коеф.
        /*обернені ваги коефіцієнтів*/
        double inv_weight_a, inv_weight_b, inv_weight_c ;
        double t_a, t_b, t_c ;  // значимість коефіцієнтів
        /*контроль зрівноваження*/
        double controlCompensLftSd, controlCompensRghtSd,
controlCompensDelta ;

        /*конструктори*/
        CModeling ()
        {}

            CModeling (int n)
            {
                numOfObserv = n ;
                x = new double[n] ;
                y = new double[n] ;
            independCost = 0 ;
            dependCost = 0 ;
            }

            CModeling (int n, const double xn[], const double yn[],
double indepC, double depC)
            {
                numOfObserv = n ;
                x = new double[n] ;
                y = new double[n] ;
                for (int i = 0 ; i < numOfObserv ; i++)
                {
                    x[i] = xn[i] ;
                    y[i] = yn[i] ;
                }
```

```
        independCost = indepC ;
        dependCost = depC ;
        }

        CModeling (const CModeling& Obj)
        {
                numOfObserv = Obj.numOfObserv ;
                x = new double[numOfObserv] ;
                y = new double[numOfObserv] ;
                for (int i = 0 ; i < numOfObserv ; i++)
                {
                        x[i] = Obj.x[i] ;
                        y[i] = Obj.y[i] ;
                }
        independCost = Obj.independCost ;
        dependCost = Obj.dependCost ;
        }

/*деструктор*/
        ~CModeling ()
        {
                delete[] x;
                delete[] y;
        delete[] calcY;
        delete[] coefEl;
        delete[] cost;
        delete[] trade;
        delete[] profit;
        delete[] abs_error ;
        delete[] abs_error_weight ;
        delete[] refinedY ;
        delete[] square_errors ;
                x = 0x000000 ;
                y = 0x000000 ;
        calcY = 0x000000 ;
        coefEl = 0x000000 ;
```

```
        cost = 0x000000 ;
        trade = 0x000000 ;
        profit = 0x000000 ;
        abs_error = 0x000000 ;
        abs_error_weight = 0x000000 ;
        refinedY = 0x000000 ;
        square_errors = 0x000000 ;
        }
/*сума попиту*/
        float SumY ()
        {
                double sum = 0 ;
                for (int i = 0 ; i < numOfObserv ; i++)
                        sum += y[i] ;
                return sum ;
        }
/*сума квадратів попиту*/
        float SumYSquar ()
        {
                double sum = 0 ;
                for (int i = 0 ; i < numOfObserv ; i++)
                        sum += (y[i] * y[i]) ;
                return sum ;
        }
/*сума цін*/
        float SumX ()
        {
                double sum = 0 ;
                for (int i = 0 ; i < numOfObserv ; i++)
                        sum += x[i] ;
                return sum ;
        }
/*сума квадратів цін*/
        float SumXSquar ()
        {
                double sum = 0 ;
```

```
            for (int i = 0 ; i < numOfObserv ; i++)
                sum += (x[i] * x[i]) ;
            return sum ;
        }
    /*сума кубів цін*/
        float SumXCube ()
        {
            double sum = 0 ;
            for (int i = 0 ; i < numOfObserv ; i++)
                sum += (x[i] * x[i] * x[i]) ;
            return sum ;
        }
    /*сума цін у 4 степені*/
        float SumXToForthDeg ()
        {
            double sum = 0 ;
            for (int i = 0 ; i < numOfObserv ; i++)
                sum += (x[i] * x[i] * x[i] * x[i]) ;
            return sum ;
        }
    /* сума добутків ціни на попит*/
        float SumXY ()
        {
            double sum = 0 ;
            for (int i = 0 ; i < numOfObserv ; i++)
                sum += (x[i] * y[i]) ;
            return sum ;
        }
     /*сума добутків квадратів цін на попит*/
        float SumXSquarY ()
        {
            double sum = 0 ;
            for (int i = 0 ; i < numOfObserv ; i++)
                sum += (x[i] * x[i] * y[i]) ;
            return sum ;
        }
```

```
/*сума квадратів абс. похибок*/
double SumESquar ()
{
	double sum = 0 ;
	for (int i = 0 ; i < numOfObserv ; i++)
			sum += pow((y[i] - calcY[i]), 2) ;
		return sum ;
}
/*1й етап обрахунків*/
	void Aproximate ()
	{
		CMatrix<double> Coef(3u, 3u) ;
		CMatrix<double> FreeTermsVect(3u, 1u) ;
		Coef(0, 0) = numOfObserv ;
		Coef(0, 1) = SumX() ;
		Coef(1, 0) = SumX() ;
		Coef(0, 2) = SumXSquar() ;
		Coef(1, 1) = SumXSquar() ;
		Coef(2, 0) = SumXSquar() ;
		Coef(1, 2) = SumXCube() ;
		Coef(2, 1) = SumXCube() ;
		Coef(2, 2) = SumXToForthDeg () ;
		FreeTermsVect(0, 0) = SumY() ;
		FreeTermsVect(1, 0) = SumXY() ;
		FreeTermsVect(2, 0) = SumXSquarY() ;
		CMatrix<double> AprCoef(3u, 1u) ;
	InversMatr.Construction(3u, 3u) ;
	InversMatr = Inverse(Coef) ;
		AprCoef = InversMatr * FreeTermsVect ;
		coef_a = AprCoef(2, 0) ;
		coef_b = AprCoef(1, 0) ;
		coef_c = AprCoef(0, 0) ;
	}
/*2й етап обрахунків*/
void getCalculations ()
{
```

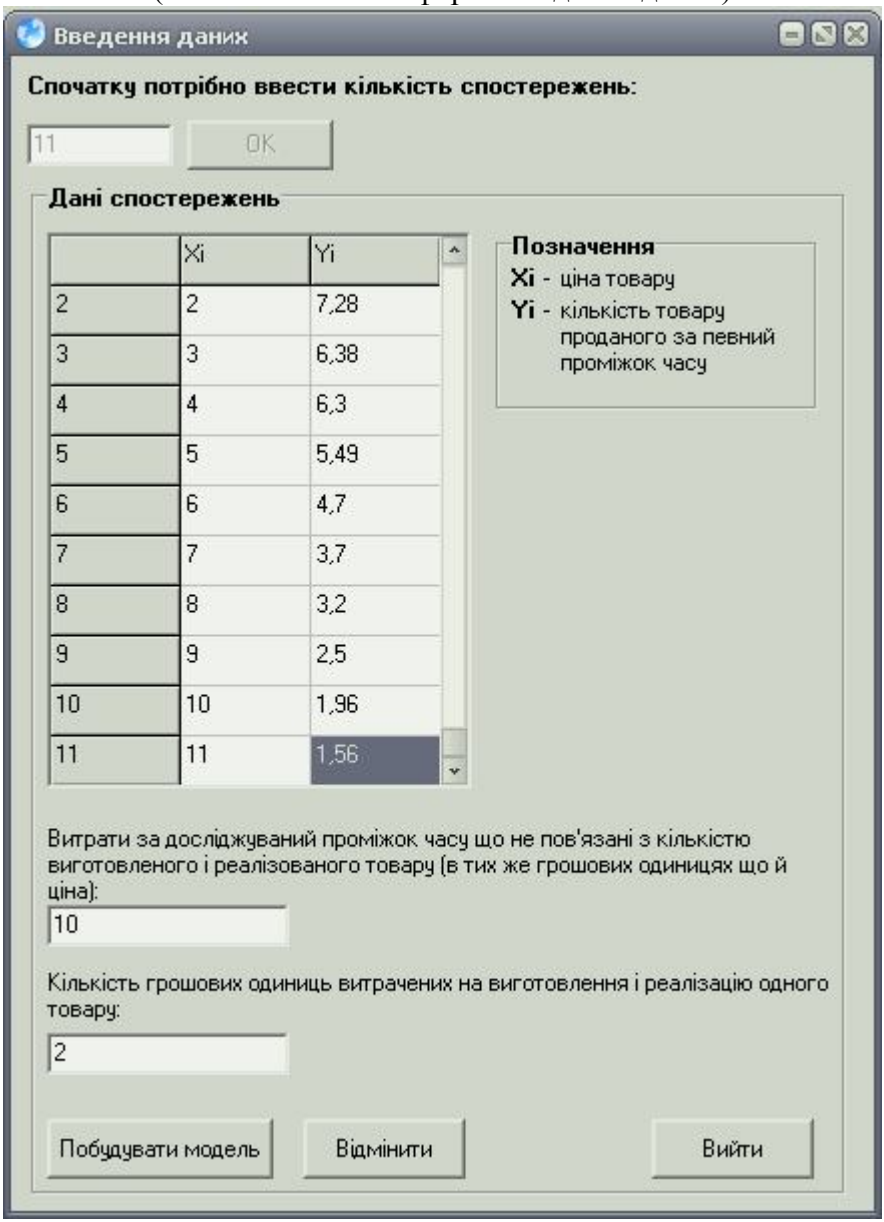
```
                calcY = new double[numOfObserv] ;
                coefEl = new double[numOfObserv] ;
                cost = new double[numOfObserv] ;
                trade = new double[numOfObserv] ;
                profit = new double[numOfObserv] ;
                for (int i = 0 ; i < numOfObserv ; i++)
                {
                        calcY[i] = x[i]*x[i]*coef_a + x[i]*coef_b + coef_c ;
                        coefEl[i] = (coef_b + 2*coef_a*x[i])*x[i]/calcY[i] ;
                        cost[i] = independCost + dependCost*calcY[i] ;
                        trade[i] = x[i]*calcY[i] ;
                        profit[i] = trade[i] - cost[i] ;
                }
        }
        /*значення зрівноваженої ф-ці для заданої ціни*/
        double getCalcY (double xn) {return xn*xn*coef_a +
xn*coef_b + coef_c ;}
        /*значення коефіцієнту еластичності для заданої ціни*/
        double getCoefEl (double xn) {return (coef_b +
2*coef_a*xn)*xn/getCalcY(xn);}
        /*значення собівартості продукції для заданої ціни*/
        double getCost (double xn) {return independCost +
dependCost*getCalcY(xn) ;}
        /*значення товарообігу для заданої ціни*/
        double getTrade (double xn) {return xn*getCalcY(xn) ;}
        /*значення прибутку для заданої ціни*/
        double getProfit (double xn) {return getTrade(xn) - getCost(xn)
; }
        /*3й етап обрахунків*/
        void getModelCharacts ()
        {
                abs_error = new double[numOfObserv] ;
                abs_error_weight = new double[numOfObserv] ;
                refinedY = new double[numOfObserv] ;
                square_errors = new double[numOfObserv] ;
                CMatrix<double> equatCoefMatr(numOfObserv, 3u) ;
```

```
                for (int i = 0 ; i < numOfObserv ; i++)
                {
                        equatCoefMatr(i, 0) = 1 ;
                        equatCoefMatr(i, 1) = x[i] ;
                        equatCoefMatr(i, 2) = x[i]*x[i] ;
                }
                CMatrix<double> auxiliaryMatr(numOfObserv, 3u) ;
                auxiliaryMatr = equatCoefMatr*InversMatr ;
                CMatrix<double> temp1(1u, 3u);
                CMatrix<double> temp2(3u, 1u) ;
                CMatrix<double> temp3(1u, 1u) ;
                meanSquareError = sqrt(SumESquar()/8) ;
                for (int i = 0 ; i < numOfObserv ; i++)
                {
                        abs_error[i] = y[i] - calcY[i] ;
                        temp1(0,0) = equatCoefMatr(i,0) ;
                        temp1(0,1) = equatCoefMatr(i,1) ;
                        temp1(0,2) = equatCoefMatr(i,2) ;
                        temp2(0,0) = auxiliaryMatr(i,0) ;
                        temp2(1,0) = auxiliaryMatr(i,1) ;
                        temp2(2,0) = auxiliaryMatr(i,2) ;
                        temp3 = temp1 * temp2 ;
                        abs_error_weight[i] = temp3(0,0) ;
                        refinedY[i] = y[i] -
abs_error[i]*sqrt(abs_error_weight[i]) ;
                        square_errors[i] =
sqrt(abs_error_weight[i])*meanSquareError ;
                }
                inv_weight_a = InversMatr(2,2) ;
                inv_weight_b = InversMatr(1,1) ;
                inv_weight_c = InversMatr(0,0) ;
                m_a = meanSquareError*sqrt(inv_weight_a) ;
                m_b = meanSquareError*sqrt(inv_weight_b) ;
                m_c = meanSquareError*sqrt(inv_weight_c) ;
                t_a = coef_a/m_a ;
                t_b = coef_b/m_b ;
```

```
                t_c = coef_c/m_c ;
                controlCompensLftSd = SumYSquar() -
coef_a*SumXSquarY() - coef_b*SumXY() - coef_c*SumY() ;
                controlCompensRghtSd = SumESquar() ;
                controlCompensDelta = controlCompensLftSd -
controlCompensRghtSd ;
            }
} ;
#endif
```

**Додаток В**

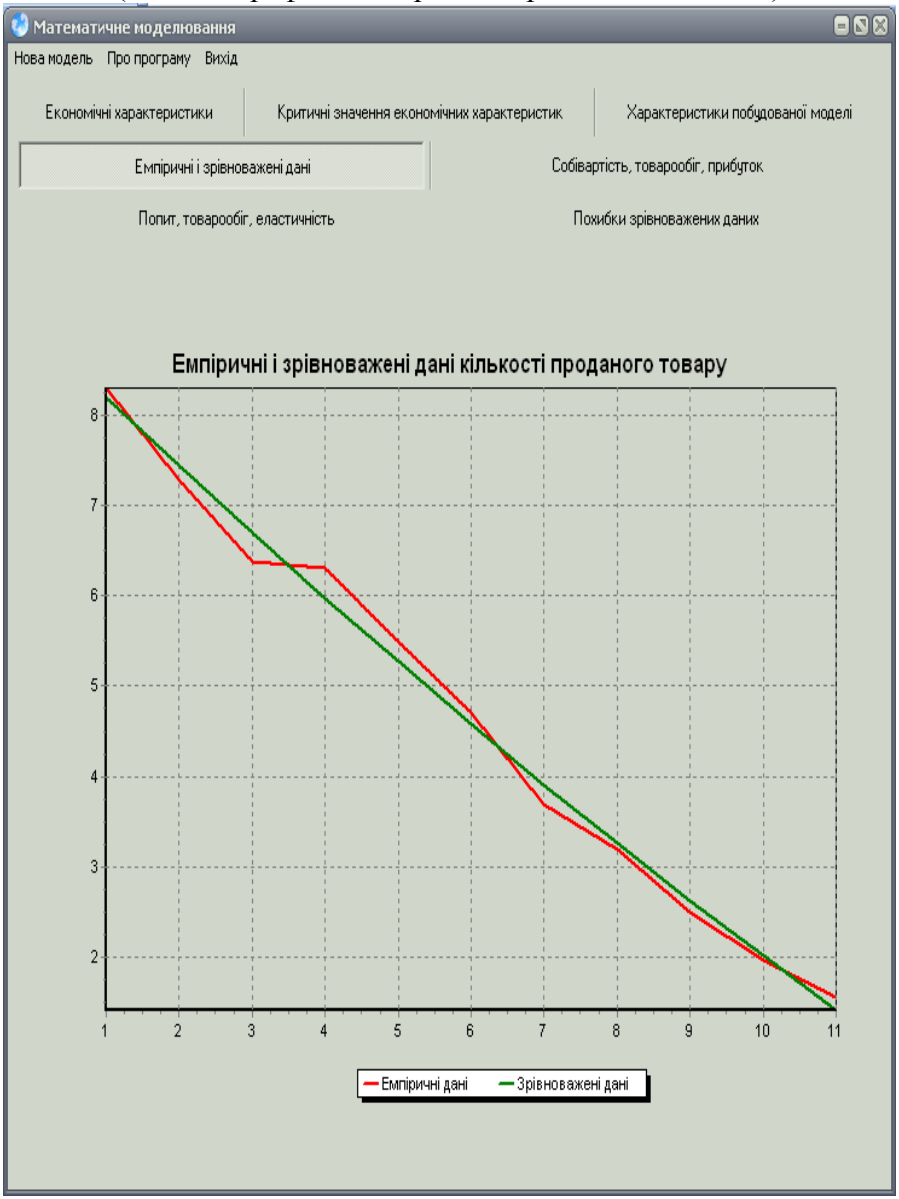
(Рис.11. заповнена форма введення даних)



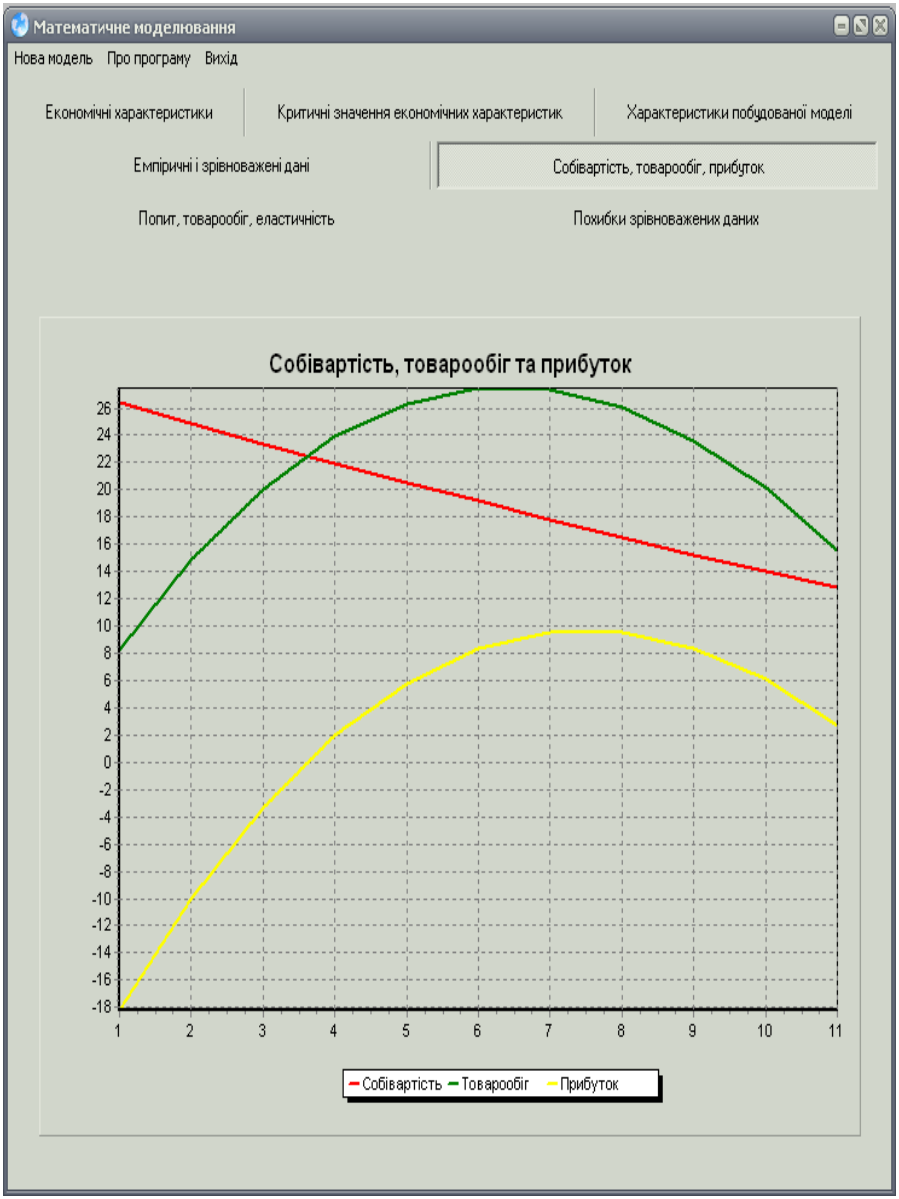

| | Xi | Yi |
|---|---|---|
| 2 | 2 | 7,28 |
| 3 | 3 | 6,38 |
| 4 | 4 | 6,3 |
| 5 | 5 | 5,49 |
| 6 | 6 | 4,7 |
| 7 | 7 | 3,7 |
| 8 | 8 | 3,2 |
| 9 | 9 | 2,5 |
| 10 | 10 | 1,96 |
| 11 | 11 | 1,56 |

**Додаток Г**

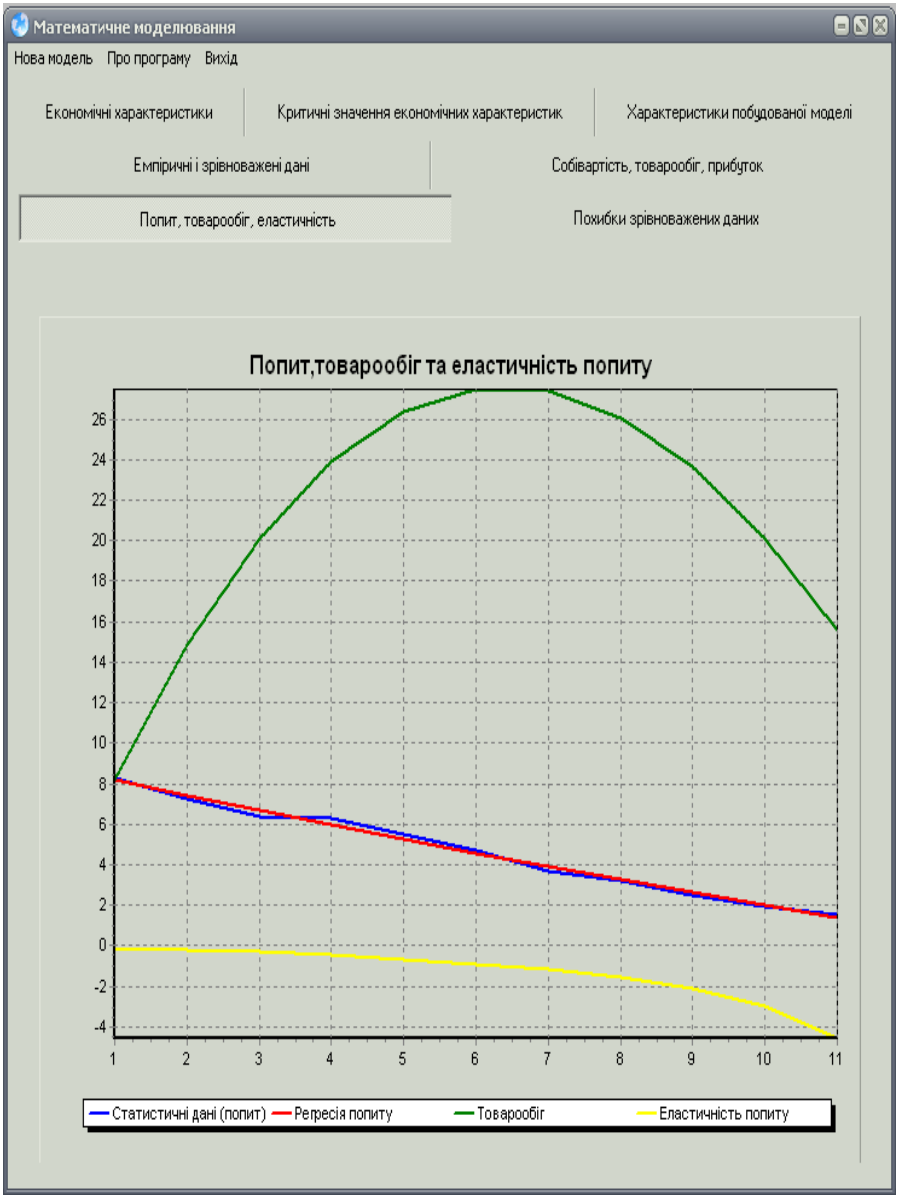
(Рис 12. Графік «Емпіричні і зрівноважені дані»)



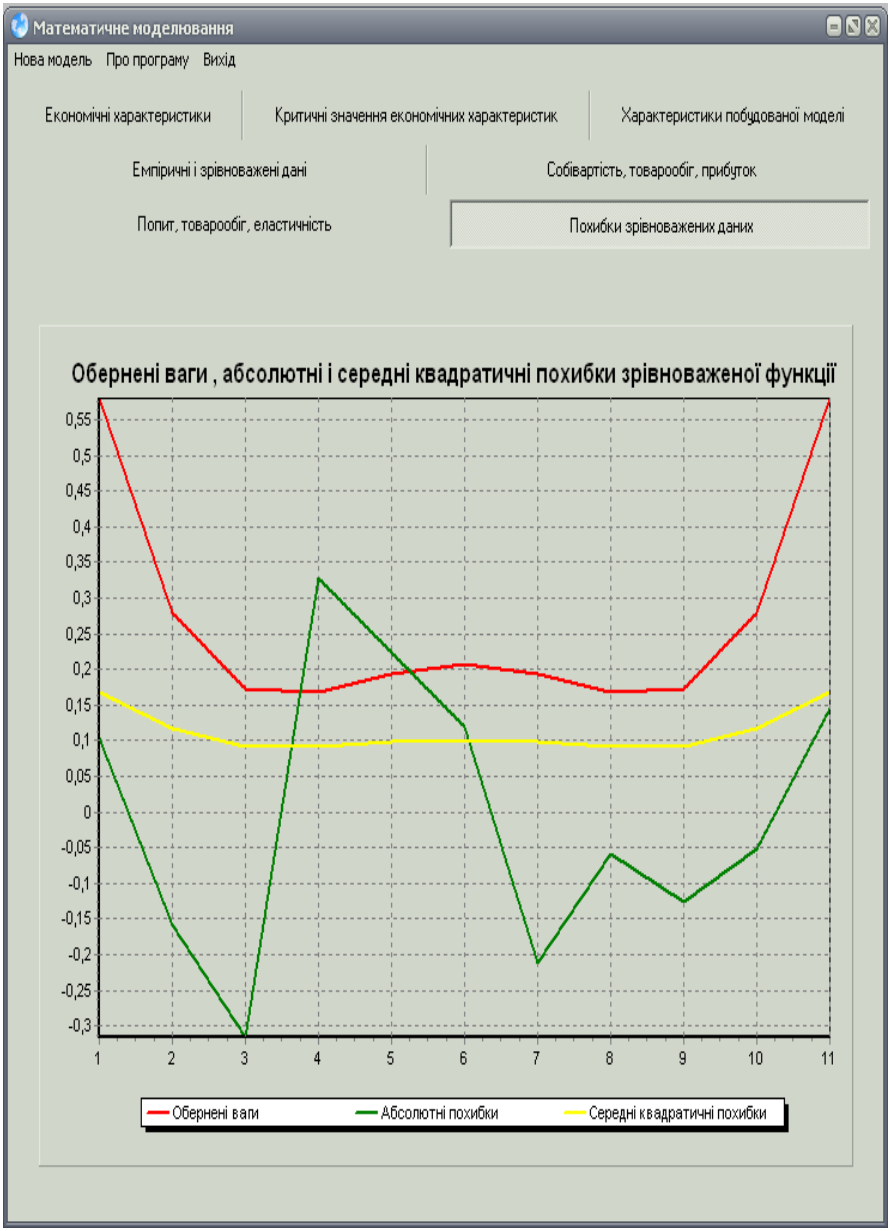
**Додаток Д.** (Рис 13. Графік «Собівартість, товарообіг, прибуток»)



**Додаток Е**. (Рис 14. Графік «Попит, товарообіг та еластичність попиту»)



**Додаток Є**. (Рис 15. Графік «Похибки зрівноважених даних»)



**Додаток Ж**

(лістинг коду форми «Обчислення економічних характеристик»)

```
#include <vcl.h>
#pragma hdrstop

#include "calcYourEconParamWindow.h"
#include "errorWindow5.h"
#include "mainWindow.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormCalcEconomicParam *FormCalcEconomicParam;
//---------------------------------------------------------------------------
__fastcall
TFormCalcEconomicParam::TFormCalcEconomicParam(TComponent*
Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFormCalcEconomicParam::FormActivate(TObject
*Sender)
{
      /*очищення форми від попередніх записів*/
       YourPrice->Text = 0 ;
       calcYOut->Clear() ;
       calcCoefElOut->Clear() ;
       calcCostOut->Clear() ;
       calcTradeOut->Clear() ;
       calcProfitOut->Clear() ;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TFormCalcEconomicParam::enterPriceButClick(TObject
*Sender)
{
        /*перевірка коректності введеної ціни*/
      if (YourPrice->Text.IsEmpty() || (YourPrice->Text.ToDouble() <=
0))
      {
            /*виведення повідомлення про помилку*/
           ErrorForm5->ShowModal() ;
           return ;
      }
      /*обрахунок економічних характеристик по заданій ціні*/
      double price = YourPrice->Text.ToDouble() ;
      calcYOut->Text = FormatFloat("0.000000",
      MainForm->pAprObj->getCalcY(price)) ;
      calcCoefElOut->Text = FormatFloat("0.000000",
      MainForm->pAprObj->getCoefEl(price)) ;
      calcCostOut->Text = FormatFloat("0.000000",
      MainForm->pAprObj->getCost(price)) ;
      calcTradeOut->Text = FormatFloat("0.000000",
      price * calcYOut->Text.ToDouble()) ;
      calcProfitOut->Text = FormatFloat("0.000000",
      calcTradeOut->Text.ToDouble() - calcCostOut->Text.ToDouble()) ;
}
//---------------------------------------------------------------------------
void __fastcall
TFormCalcEconomicParam::calcEkonomParamButExitClick(
    TObject *Sender)
{
        /*закриття форми*/
     FormCalcEconomicParam->Close() ;
}
//---------------------------------------------------------------------------
```

**Ціхоцька Катерина Валентинівна**
**спеціаліст системотехнік, магістрант інформаційних технологій**

**ДИСЕРТАЦІЯ**

**на здобуття кваліфікації магістр з інформатики**

**на тему:**

# ЗАСОБИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ НА МОВІ ПРОГРАМУВАННЯ С++ ПРИ ВИВЧЕННІ СКЛАДНИХ ЕКОНОМІЧНИХ ЯВИЩ

**8.080201 – „Інформатика"**

**Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука**
**Факультет кібернетики**
**Кафедра математичного моделювання**

**Комп'ютерний набір в редакторі Microsoft®Office® Word 2007**
**К.В Ціхоцька**
**Редагування, верстка, макетування та дизайн**
**Р.М.Літнарович**
**Науковий керівник Р. М. Літнарович, доцент, кандидат технічних наук**

**33027,м.Рівне,Україна**
**Вул.акад. С.Дем'янчука,4, корпус 1**
**Телефон:(+00380) 362 23-73-09**
**Факс:(+00380) 362 23-01-86**
**E-mail:mail@regi.rovno.ua**
**tkvpay@gmail.com**