ЗМІСТ

Перелік позначень та скорочень 4

Вступ 5

1 Аналітичний огляд 6

1.1 Загальні відомості про некогерентно розсіяні сигнали 6

1.2 Основні відомості про потоки 9

1.3 Засіб швидкої розробки програм C++ Builder 10

1.4 Опис роботи алгоритмів 12

2 Вимоги до процесу розробки програмного забезпечення 18

2.1 Вибір моделі процесу 18

2.2 Вибір методології розробки програмного забезпечення 23

3 Розробка вимог до програмного забезпечення 27

3.1 Вимоги до функціональних характеристик програмного продукту 27

3.2 Вимоги до надійності 27

3.3 Вимоги до інтерфейсу 28

4 Розробка програмного забезпечення 30

5 Тестування програмного забезпечення 38

6 Техніко-економічне обґрунтування 43

6.1 Доцільність розробки продукту 43

6.2 Опис характеристик вибору 43

6.2.1 Найменування розробки 43

6.2.2 Призначення продукту 43

6.2.3 Область використання 43

6.2.4 Характеристика програмного продукту 43

6.3 Дослідження і аналіз ринків збуту 44

6.3.1 Сегментація ринку по споживачам 44

6.3.2 Параметрична сегментація ринку 46

6.4 Оцінка конкурентоздатності 47

6.5 Стратегія маркетингу 49

6.6 Фінансовий план 53

6.6 Висновки 57

7 Цивільний захист 58

7.1 Склад та основні завдання сил цивільного захисту 58

7.2 Завдання і обов’язки суб’єктів господарювання 59

8 Охорона праці 63

8.1 Загальні питання охорони праці та навколишнього середовища 63

8.2 Характеристика умов праці 67

8.3 Охорона навколишнього середовища 81

Висновки 82

Список джерел інформації 84

Додаток А 85

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

|  |  |
| --- | --- |
| ЕОМ | Електронно-обчислювальна машина |
| КПО | коефіцієнтом природної освітленості |
| НДР | науково-дослідницька робота |
| ПЗ | Програмне забезпечення |
| СГП | Спектральна густина потужності |
| СКВ | Середнє квадратичне відхилення |

ВСТУП

В даний час в зв’язку з розвитком радіозв’язку, радіоастрономії, а також освоєння космічного простору виросла кількість досліджень процесів, які проходять в іонізованій області атмосфери Землі – іоносфері.

Для перевірки радіотехнічних систем, оцінки похибок отриманих результатів необхідно використовувати комп'ютерне моделювання. Воно дає змогу виконанням послідовності обчислень з подальшим графічним відображенням їх результатів, відтворювати (імітувати) сигнали, що приймаються радіотехнічними комплексами, зокрема, радарами некогерентного розсіяння [1 – 3].

Метою цього дипломного проекту є розробка комп’ютерної моделі некогерентно розсіяного сигналу для синтезу сигналів із заданими параметрами іоносфери та розв’язання оберненої задачі радіофізики. А саме за допомогою відомої бібліотеки автокореляційних функцій шляхом порівняння знаходження таких важливих параметрів іоносфери, як температура іонів та температура електронів.

Для досягнення поставленої мети сформульовані наступні задачі:

1. Ознайомитися з характеристиками некогерентно розсіяного сигналу (спектр та автокореляційна функція)
2. Розробити алгоритм роботи програми.
3. Написати програмне забезпечення (ПЗ), з урахуванням відомостей про методи прийому та обробки сигналів, а саме: усереднення за висотою, трапецеєподібне додавання, урахування імпульсної характеристики фільтру.
4. Протестувати розроблене ПЗ для генерації сигналу та розв’язання зворотної задачі радіофізики.

1 АНАЛІТИЧНИЙ ОГЛЯД

* 1. **Загальні відомості про некогерентно розсіяні сигнали**

Можливість дослідження стану іоносфери на основі аналізу розсіяння електромагнітної хвилі на вільних електронах була обґрунтована і експериментально перевірена в кінці п’ятдесятих років. Це поклало початок застосування методу некогерентного розсіяння, який дозволяє одночасно отримувати дані про основні параметри іоносфери в широкому діапазоні висот. В 1958 р. У. Гордон, спираючись на явище розсіяння електромагнітної хвилі на вільних електронах, висловив припущення про можливість проведення вимірювань електронної концентрації в іоносфері вище максимуму шару F2 на частотах, більших плазменної частоти цієї області. Експериментальна перевірка цього припущення (К. Боулсом) в 1958 році дала обнадійливі результати і поклала початок розвитку методу некогерентного розсіяння радіохвиль

Метод оснований на відомому явищі томсонівського розсіяння. Некогерентно розсіяне іоносферою випромінювання практично можна виявити за допомогою існуючої радіолокаційної техніки тільки тоді, коли довжина зондуючої хвилі значно більша дебаївської хвилі. Ця вимога задовольняється в денний час для висоти для висоти до 1000 км при довжині хвилі 25 см і більше, а вночі ж при вимірюваннях в області Е необхідна довжина хвилі порядку 1 м.

Переріз розсіяння об’єму іоносфери, що зондується, розміщеного на висоті близько 300 км, еквівалентно по площі 1 см2. Як правило використовуються радіопередавачі, працюючі в дециметровому або в метровому діапазоні хвиль з низьким рівнем шуму, спеціалізовані пристрої обробки інформації.

Некогерентне розсіяння на метрових і дециметрових хвилях обумовлено наявністю флуктацій щільності плазми, викликаних тепловим рухом іонів і електронів. В цьому випадку основною причиною флуктацій щільності електронів є наявність іонів, а в результаті кулонівської взаємодії між ними виникають іонно-звукові хвилі. Іншими словами, кожний іон здійснює дію, що виводить з рівноваги рух всіх електронів всередині сфери дебаївського радіусу і, таким чином, хаотичний рух іонів приводить до відповідних статичних флуктуацій концентрації електронів. Іншою не менш важливою причиною флуктуацій є кулонівське відштовхування самих електронів, що приводить до слабкого резонансу на плазменній частоті. В спектрі розсіяного сигналу виникає компонента, зміщена на величину плазменної частоти для висоти, на якій проходить розсіяння.

Іонно-звукова хвиля подібна звуковим хвилям і поширюється зі швидкістю , близькою до теплової швидкості домінуючих іонів. При наявності такої хвилі в плазмі створюються згустки і розрідження щільності. На цих слабких неоднорідностях розсіюються хвилі, їх результуюча максимальна, якщо хвилі , що розсіюються окремими неоднорідностями сумуються в фазі. При цьому відстань між неоднорідностями повинно бути рівним половині довжини хвилі для зворотного розсіяння. Розсіяний сигнал несе інформацію про розподіл і характер руху не тільки електронної, а й іонної компоненти плазми. Він дозволяє отримати багатий набір параметрів іоносфери: електронну і іонну температури, розподіл по масам, середню швидкість дрейфу часток різних сортів, що в свою чергу дає можливість визначити іоносферне електричне поле, напрям сили струму, швидкість вітру в нейтральній атмосфері та інші важливі параметри.

На сьогодні вісім обсерваторій проводять зондування іоносфери методом некогерентного розсіяння При інституті іоносфери працює радар для дослідження іоносфери методом некогерентного розсіяння. Цей радар являє собою установку, працюючу в імпульсному режимі. Створений в Інституті іоносфери радар працює на частоті близько 150 МГц. Імпульсна потужність пристрою близько 2 МВт. Тривалість імпульсів може варіюватися в широких границях – від 40 мкс до 1 мс. Шумова температура системи не гірше 500 К. Як правило при дослідженнях іоносфери шляхом некогерентного розсіяння вимірюється рівень потужності сигналу, його спектр або автокореляційна функція, так як його коефіцієнт кореляції флуктуацій електронів несе в собі ту ж інформацію, що і спектр потужності. Для вирішення широкого кола задач, виникаючих при дослідженні іоносфери, передбачена можливість роботи комплексу в декількох основних режимах, що відрізняються параметрами зондуючого імпульсу .

Сигнал з виходу блоку кварцових гетеродинів надходить на двохканальний передавальний пристрій, де посилюється, а потім по хвилеводному фідерному тракту передається в збуджуючий рупор двохзеркальної антени. Тут потужний радіоімпульс випромінюється вертикально вверх, а слабкий сигнал відбитого від іоносфери радіоімпульсу, розсіяного на теплових флуктуаціях електронної щільності, приймається тою ж антеною і через антенний комутатор «прийом-передача» надходить на вихідні параметричні посилювачі приймаючого пристрою. Після посилення і перетворення сигнал на проміжній частоті подається на спеціалізований обчислювальний пристрій, де проводиться його первинна обробка – тимчасове накопичення і вирахування його автокореляційної функції. Результати кореляційної обробки надходять на комп’ютер, де по ним визначається значення іоносферних параметрів та видаються дані на друк та на екран відеоконтрольного пристрою.

Напруга, що виникає на виході прийомної системи, являє собою суму напруг шумів системи і напруги власне некогерентно розсіяного сигналу. Тому завдання знаходження автокореляційної функції сигналу зводиться до знаходження різності між кореляційною функцією вихідного сигналу та кореляційною функцією шумів. Обчислення їх проводяться в цифровому корелометрі, що підключений до посилювача проміжної частоти приймача. В такому випадку кореляційна функція сигналу, що входить в склад кореляційної функції суміші сигнал-шум, виявляється помноженою на косинусоїдальний множник проміжної частоти.

Оскільки обробка сигналу ведеться в цифровому вигляді, досліджувана напруга квантується в аналого-цифровому перетворювачі з тактовою частою, кратній проміжній частоті. При накладанні затриманого на час сигналу на прямій отримується значення кореляційної функції, відповідне величні цієї затримки.

Змінюючи затримку в необхідному діапазоні можна отримати в цьому ж діапазоні автокореляційу функцію суміші сигнал-шум. Із інформаційних даних, представлених у вигляді автокореляційної функції, отримують необхідну інформацію. Ця інформація описує висотні залежності концентрації іонів і електронів, а також висотні температурні залежності іонів і електронів. Така операція виконується одночасно для всього діапазону досліджуваних висот.

**1.2 Основні відомості про потоки**

В програмуванні термін потік використовується в кількох значеннях, але у всіх випадках визначається як абстрактна послідовність інструкцій або даних взагалі, прив’язана до відповідного дескриптора (може бути представлений назвою потоку). Потоки є зручним уніфікованим програмним інтерфейсом для зчитування файлів передачі даних між процесами.

В [Unix](http://uk.wikipedia.org/wiki/Unix) і споріднених [системах](http://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), заснованих на [мові програмування C](http://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_C), потік — це джерело або призначення даних, зазвичай індивідуальних байтів або знаків. Потоки — це абстракція, що використовується наприклад при читанні або записі [файлів](http://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), або при зв'язку з вузлами [мережі](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0). Три [стандартні потоки](http://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D1%96_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D0%B8) визначені і доступні для всіх [програм](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). В мові [C++](http://uk.wikipedia.org/wiki/C%2B%2B) концепція потоків реалізована у [бібліотеці](http://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) [iostream](http://uk.wikipedia.org/wiki/Iostream) і низці похідних від неї. [Файлова система](http://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) може підтримувати багато іменнованих незалежних потоків для одного файлу. Є один головний потік, який передає нормальні дані з файлу. Додаткові потоки можуть використовуватися, щоб запам'ятати іконки, короткий звіт і індексацію інформації, зональну інформацію (для файлів, що завантажуються) тощо.

[Конвеєри](http://uk.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D1%94%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)&action=edit&redlink=1) можуть також розумітися, як потоки, також як і будь-яка необмежена (не упакована) інформація, що постачається периферійним пристроєм.

У мові програмування [Scheme](http://uk.wikipedia.org/wiki/Scheme) і деяких інших, потік — ліниво оцінена або затримана послідовність елементів даних. Потік може використовуватися так само як список, але останні елементи обчислюються тільки тоді, коли потрібно. Тому потоки можуть представити нескінченні послідовності.

Поточні обчислення — в [паралельному виконанні](http://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%B5%D0%BB%D1%96%D0%B7%D0%BC_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), особливо в графічній обробці, термін потік застосовується і до [апаратних засобів](http://uk.wikipedia.org/wiki/%D0%90%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), і до (ПЗ). Ним позначають квазі-безперервний потік даних, які обробляються на потоковій мові програмування, щойно програмний стан задовольняє початковій умові потоку.

Багатопоточність – властивість операційної системи або програми, що полягає в тому , що процес породжений в операційній системі може складатися кількох потоків, що виконуються паралельно.

Переваги в багатопоточності наступні:

1. спрощення програм за рахунок використання загального адресного простору;
2. менші відносно процесу часові витрати на створення потоків і взаємодію між ними ;
3. підвищення продуктивності процесу за рахунок розпаралелювання процесорних обчислень і операцій вводу виводу.

В процесі розробки дипломного проекту була обрана мова програмування С++.

**1.3 Засіб швидкої розробки програм C++ Builder**

Кожна програма повинна мати зручний інтерфейс для спілкування з користувачем. Основним елементом інтерфейсу у Windows є вікна. Одним з різновидів вікон є форма, яка може містити кнопки, текстові поля, перемикачі тощо. Тому програми, написані для використання у Windows, зазвичай мають інтерфейс, подібний до вікон та форм. Для швидкого і зручного створювання програм з графічним інтерфейсом використовують спеціальні середовища візуального програмування. Майже кожна сучасна мова програмування має принаймні одне таке середовище: Object Pascal – Borland Delphi, Basic – Visual Basic, C++ – Borland C++ Builder, Microsoft Visual C. Візуальне програмування ще називають Rapid Application Development (RAD) - швидка розробка додатків. Технологія RAD суттєво прискорює створення програм з графічним інтерфейсом.

Інструментальна система Builder, подібно до інших систем візуального програмування (Visual C, Visual Basic, Delphi тощо), насамперед є посередником поміж інтерфейсом прикладного програмування Windows (API – Application Program Interface) та програмістом, надаючи змогу навіть програмістам-початківцям оперативно створювати програмні проекти, які матимуть графічний інтерфейс користувача (GUI – Graphic User Interface) найрізноманітнішої спрямованості, від суто обчислювальних і логічних до графічних і мультимедійних.

C++ Builder – це технологія візуального програмування, де автоматизовано її трудомістку частину − створювання інтерфейсних програм з діалоговими вікнами. Оболонка C++ Builder надає змогу замість повного самостійного написання програми використовувати великий набір готових візуальних об’єктів, так званих компонентів, піктограми яких розміщені на відповідних вкладках палітри компонентів. В С++ Builder існує понад 100 компонентів. Всі компоненти зібрано у бібліотеці візуальних компонентів VCL – Visual Class Library.

С++ Builder призначено для написання програм мовою програмування C++ і поєднує VCL та середовище програмування (IDE – Integrated Development Environment), написані на Delphi з компілятором C++. Цикл створення програм- них проектів у C++ Builder є аналогічний до Delphi, але із суттєвими поліпшеннями. Більшість компонентів, розроблених у Delphi, можна використовувати і в C++ Builder без модифікації, але,на жаль,зворотне твердження не слушне.

С++ Builder дозволяє методом drag-and-drop доволі просто розробляти інтерфейсні програми, що зумовлює підвищення ефективності та простоту програмування, оскільки програмістові не треба кожного разу створювати ті елементи власних програм, котрі може бути реалізовано за допомогою вже існуючих об’єктів.

Основним будівельним об’єктом візуального програмування є компонент. Компонентами в C++ Builder є об'єкти чи класи об'єктів, які є, у певному розумінні, об’єктами реального світу. Їх безпосередньо видно на екрані (за винятком групи невидимих компонентів), їх можна пересувати мишею, вони можуть реагувати на клацання клавіш клавіатури і миші тощо. Своєю чергою, компонентам, на відміну від звичайних об’єктів C++, притаманна наявність властивостей, подій та методів, які дозволяють здійснювати різноманітні операції з цими компонентами.

Властивості (Properties) дозволяють легко встановлювати різні характеристики компонентів, такі як назва, розміри, контекстні підказки чи джерела даних. Методи виконують певні операції над компонентним об’єктом, у тому числі й такі складні, як відтворювання чи перемотування пристрою мультимедіа. Події (Ivents) пов’язують зовнішні впливи, на які реагують компоненти, такі як активізація, натиснення кнопок чи редаговане введення – з кодами реакції на ці впливи. Окрім того, події можуть виникати за таких специфічних змінювань стану компонента, як поновлення даних в інтерфейсних елементах доступу до баз даних.

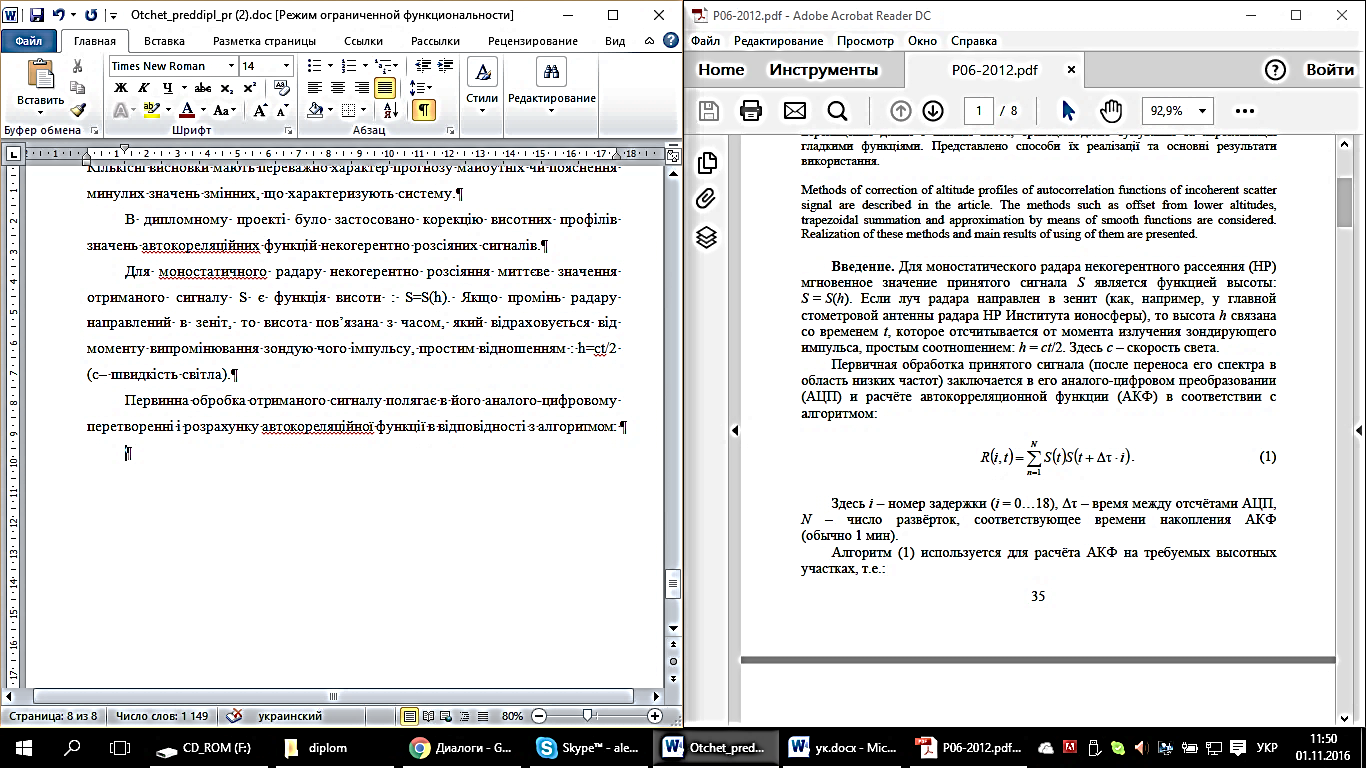
Borland C++ Builder може працювати в середовищі операційних систем від Windows 98 до Windows XP. Особливих вимог, по сучасних мірках, до ресурсів комп'ютера пакет не пред'являє: процесор має бути типу Pentium або Celeron; об'єм оперативної пам'яті повинен складати не менше 128 Мбайт (рекомендується 256 Мбайт) і вільний дисковий простір має бути достатнім (для повної установки версії Enterprise необхідно приблизно 750 Мбайт) [4].

**1.4 Опис алгоритмів роботи.**

В дипломному проекті було застосовано корекцію висотних профілів значень автокореляційних функцій некогерентно розсіяних сигналів.

Для моностатичного радару некогерентно розсіяння миттєве значення отриманого сигналу S є функція висоти: S=S(h). Якщо промінь радару направлений в зеніт, то висота пов’язана з часом, який відраховується від моменту випромінювання зондуючого імпульсу, простим відношенням: h=ct/2 (с– швидкість світла).

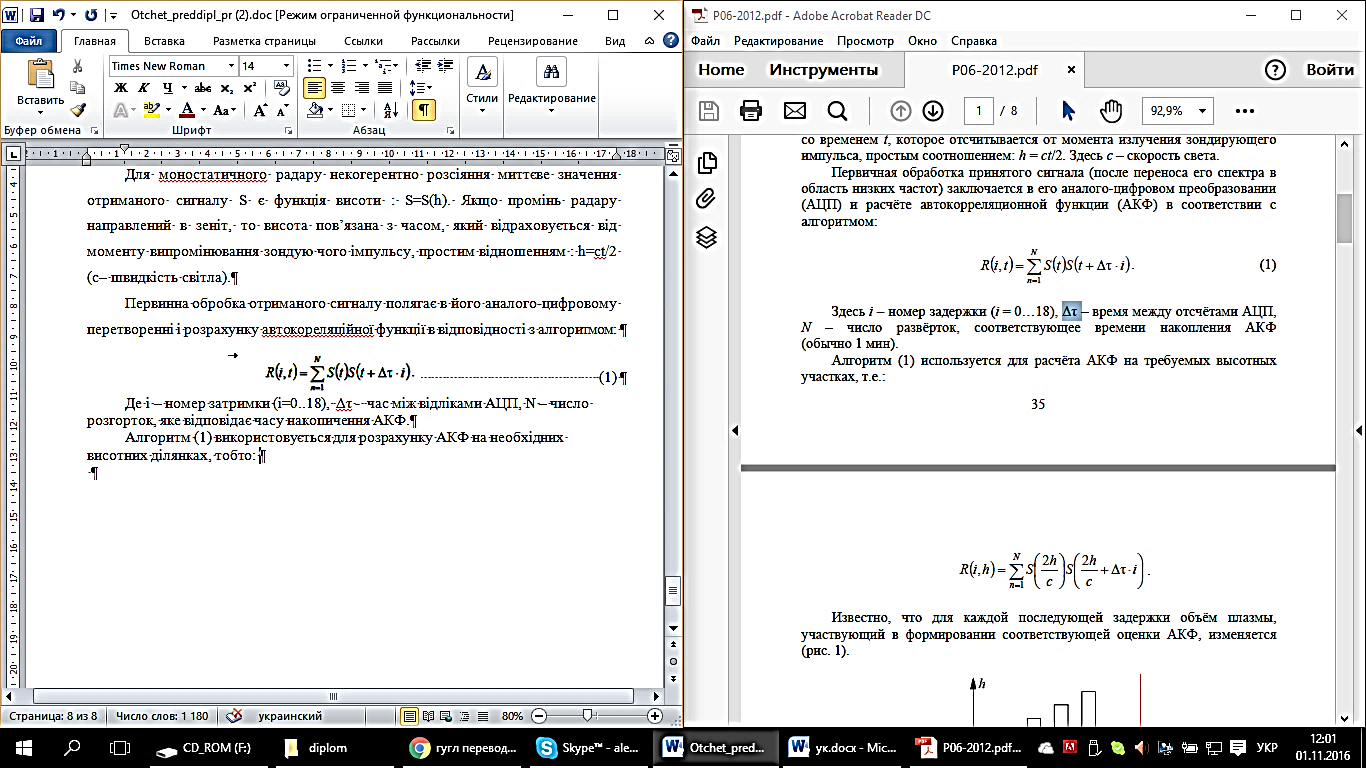
Первинна обробка отриманого сигналу полягає в його аналого-цифровому перетворенні і розрахунку автокореляційної функції в відповідності з алгоритмом:



(1.1)

Де i – номер затримки (і=0..18), Δτ - час між відліками АЦП, N – число розгорток, яке відповідає часу накопичення АКФ.

Алгоритм (1.1) використовується для розрахунку АКФ на необхідних висотних ділянках, тобто:



(1.2)

Відомо, що для кожної наступної затримки об’єм плазми, який бере участь в формуванні відповідної оцінки АКФ, змінюється (рисунок 1.1).



Рисунок 1.1 – Зменшення коррелюючого обєму плазму і зміщення його центру з ростом номеру затримку АКФ при імпульсному зондуванні іоносфери

Цей ефект приводить до зменшення значення АКФ при збільшенні затримки (що легко корректується множенням кожної оцінки АКФ на відповідний коефіцієнт). Другим, більш вагомим наслідком відміченого ефекту є просторове зміщення результуючого розсіючого об’єму.

Перший спосіб коррекції полягає в вирівнюванні даних шляхом зміщення їх з нижніх висот. Якщо дані записані з кроком th , рівним інтервалу Δτ, то просторове вирівнювання можна описати формулою

*.* (1.3)

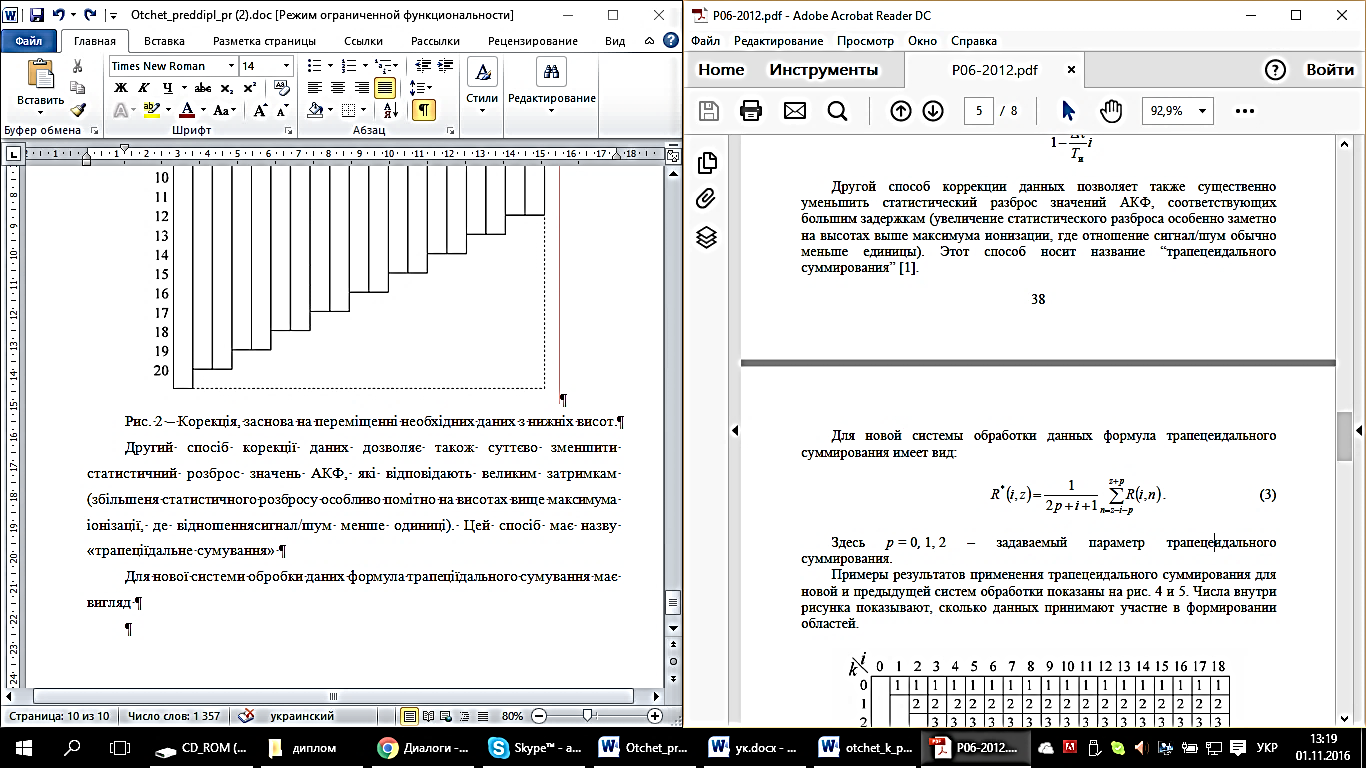
Де z – номер висотної ділянки, mod операція отримання остачі від ділення.

Результати корекції висотних даних, проведених відповідно алгоритму, представлені на рисунку 1.2.



Рисунок 1.2 – Корекція, заснова на переміщенні необхідних даних з нижніх висот.

Другий спосіб корекції даних дозволяє також суттєво зменшити статистичний розсіювання значень АКФ, які відповідають великим затримкам (збільшеня статистичного розсіювання особливо помітно на висотах вище максимума іонізації, де відношення сигнал/шум менше одиниці). Цей спосіб має назву «трапецеїдальне дадавання»

Для нової системи обробки даних формула трапеціїдального додавання має вигляд

(1.4)

Де *p =* 0*,* 1, 2 – параметр, що задається.

Приклади результатів застосування трапецеїдального додавання для нової та попередньої систем обробки показані на рисунку 1.3 та рисунку 1.4. Числа на рисунку показують, скільки даних приймає участь в формуванні областей.



Рисунок 1.3 – Трапецеїдальне додавання з параметром р=0, для нової системи обробки



Рисунок 1.4 – Трапецеїдальне додавання з параметром р=0 для попередньої системи обробки

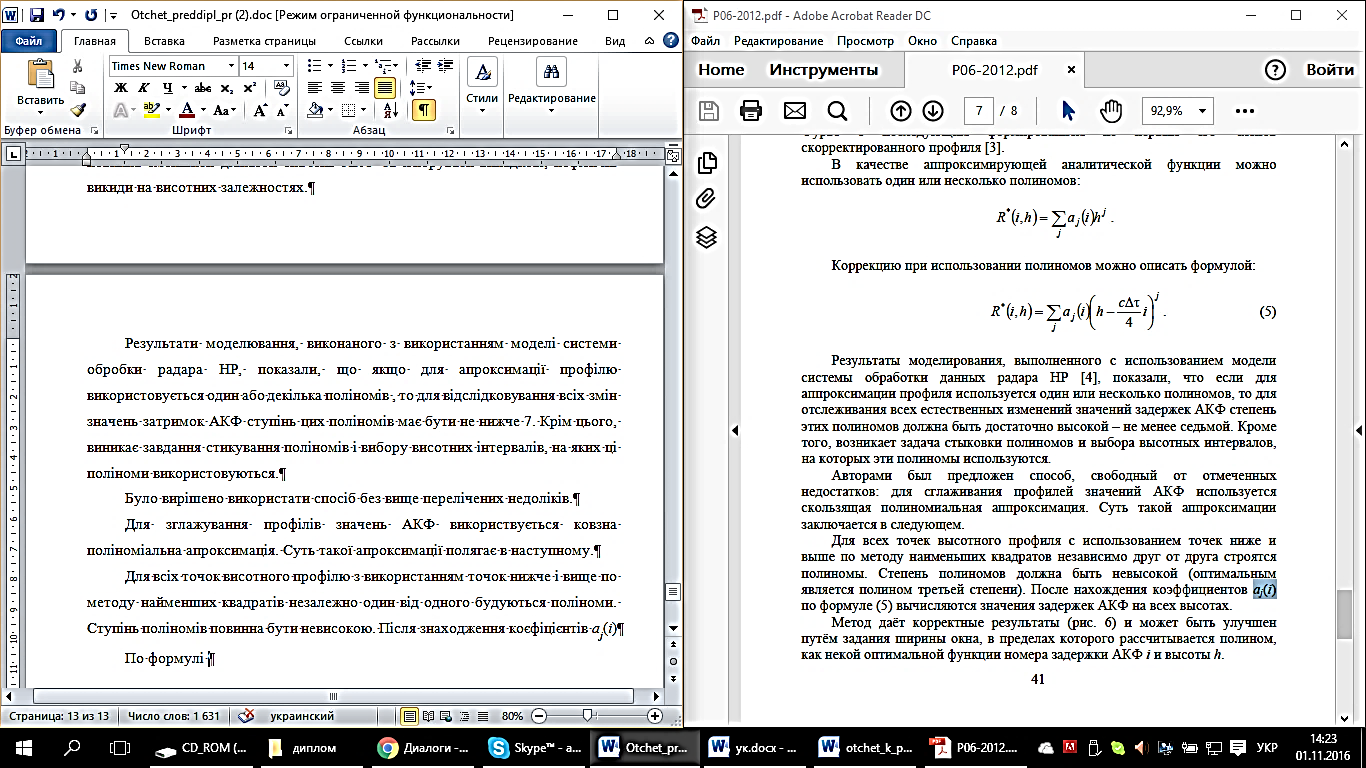
Ще один спосіб корекції даних, отриманих на радарі НР, полягає в апроксимації висотних профілів аналітичними функціями.

Аналіз висотних профілів затримок АКФ для різних сезонів і часу доби показав, що профіль нульової затримки АКФ в більшості випадків має один максимум. Профілі інших затримок можуть мати як максимуми, так і мінімуми, а значення цих затримок можуть бути як додатними так і від’ємними.

До вибору аналітичної функції, якої будуть апроксимуватися висотні профілі затримок АКФ, підлягає ряду вимог. По-перше, апроксимуюча аналітична функція повинна відслідковувати пов’язані зі змінами параметрів іоносфери природні висотні варіації значень АКФ. По-друге, апроксимуюча функція повинна зменшити діапазон значень АКФ та ігнорувати випадкові, нефізичні викиди на висотних залежностях.

Результати моделювання, виконаного з використанням моделі системи обробки радара НР, показали, що якщо для апроксимації профілю використовується один або декілька поліномів , то для відслідковування всіх змін значень затримок АКФ ступінь цих поліномів має бути не нижче 7. Крім цього, виникає завдання стикування поліномів і вибору висотних інтервалів, на яких ці поліноми використовуються.

Для зглажування профілів значень АКФ використвується ковзна поліноміальна апроксимація. Суть такої апроксимації полягає в наступному.

Для всіх точок висотного профілю з використанням точок нижче і вище по методу найменших квадратів незалежно один від одного будуються поліноми. Ступінь поліномів повинна бути невисокою. Після знаходження коєфіцієнтів *aj*(*i*)

По формулі знаходяться значення затримок АКФ на всіх висотах.

Метод дає коректні результати і може бути покращений шляхом зміни ширини вікна, в діапазоні якого розраховується поліном, як деякої оптимальної функції номера затримки АКФ і та висоти h [5].

2 ВИМОГИ ДО ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**2.1 Вибір моделі процесу**

Модель життєвого циклу ПЗ — структура, що визначає послідовність виконання та взаємозв'язку процесів, дій і завдань протягом життєвого циклу. Модель життєвого циклу залежить від специфіки, масштабу і складності проекту і специфіки умов, в яких система створюється і функціонує.

В рамках специфічних моделей життєвого циклу, які приписують правила організації розробки ПЗ в рамках даної галузі або організації, визначаються більш конкретні процеси розробки. Відрізняються вони від стандартів, перш за все, більшою детальністю і чітким описом зв'язків між окремими видами діяльності, визначенням потоків даних (документів і артефактів) в ході життєвого циклу. Таких моделей досить багато, адже фактично кожен раз, коли деяка організація визначає власний процес розробки, в якості основи цього процесу розробляється деяка модель життєвого циклу ПЗ

Найчастіше говорять про наступні моделі життєвого циклу:

1. водоспадна (каскадна) або послідовна;
2. ітераційна й інкрементна;
3. спіральна (spiral) або модель Боема.

Коротко розглянемо дані моделі.

Розглянемо водоспадну (каскадну) модель.

У 1970 році каскадна модель була вперше визначена як альтернативний варіант методу розробки ПЗ за принципом: кодування - усунення помилок, який був широко поширений в той час. Це була перша модель, яка формалізувала структуру етапів розробки ПЗ, надаючи особливого значення вихідним вимогам і проектуванню, а також створенню документації на ранніх етапах процесу розробки. Вона передбачає послідовне виконання всіх етапів проекту в строго фіксованому порядку.

Перехід від однієї фази до іншої здійснюється за допомогою формального огляду. Таким чином, клієнт одержує загальне уявлення про процес розробки, крім того відбувається перевірка якості програмного продукту. Як правило, проходження стадії огляду вказує на домовленість між командою розробників і клієнтом про те, що поточна фаза завершена і можна перейти до виконання наступної фази. Закінчення фази зручно приймати за стадію в процесі виконання проекту. Кожна стадія завершується випуском повного комплекту документації, достатньої для того, щоб розробка могла бути продовжена іншою командою розробників.

У результаті завершення певних фаз формується базова лінія, яка в даній точці "заморожує" продукти розробки. Якщо виникає потреба в їх зміні, тоді для внесення змін використовується формальний процес змін.

У критичних точках каскадної моделі формуються базові лінії, остання з яких є базовою лінією продукту. Після формування заключної базової лінії проводиться огляд приймання.

Спроби оптимізації каскадної моделі призвели до виникнення інших циклів розробки ПЗ.

Відмітною властивістю каскадної моделі можна назвати те, що вона являє собою формальний метод, різновид розробки "зверху вниз", вона складається з незалежних фаз, виконуваних послідовно, і схильна частого огляду.

Переваги:

1. Проста і зручна в застосуванні, так як процес розробки виконується поетапно;
2. Повна і узгоджена документація на кожному етапі;
3. Легко визначити терміни і витрати на проект.

До недоліків відноситься наступне.

У водоспадній моделі перехід від однієї фази проекту до іншого передбачає повну коректність результату (виходу) попередньої фази.

Однак неточність будь-якої вимоги або некоректна її інтерпретація в результаті призводить до того, що доводиться повертатися до ранньої фази проекту і необхідна переробка не просто вибиває проектну команду з графіка, але призводить часто до якісного зростання витрат і, не виключено, до припинення проекту в тій формі, в якій він спочатку замислювався.

На думку сучасних фахівців, основна помилка авторів водоспадної моделі полягає в припущеннях, що проект проходить через весь процес один раз, спроектована архітектура гарна і проста у використанні, проект здійснення розумний, а помилки в реалізації легко усуваються в міру тестування. Ця модель виходить з того, що всі помилки будуть зосереджені в реалізації, а тому їх усунення відбувається рівномірно під час тестування компонентів і системи. Таким чином, водоспадна модель для великих проектів мало реалістична і може бути ефективно використана тільки для створення невеликих систем.

Розглянемо ітеративну або інкрементну модель, що реалізує еволюційний підхід.

Альтернативою послідовної моделі є так звана модель ітеративної і інкрементної розробки, (англ. iterative and incremental development, IID), що отримала також від Т. Гілба в 70-і рр. назву еволюційної моделі. Також цю модель називають ітеративною моделлю та інкрементною моделлю.

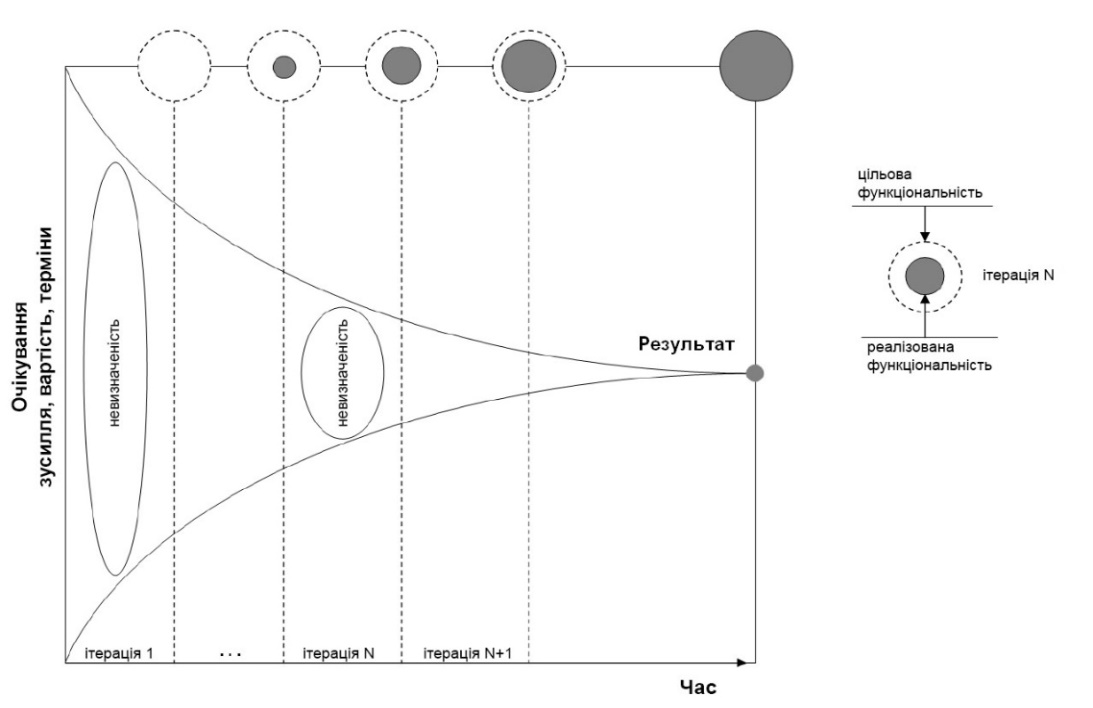


Рисунок 2.1 — Ітеративна модель розробки

Модель IID передбачає розбиття життєвого циклу проекту на послідовність ітерацій, кожна з яких міні-проект, включаючи всі процеси розробки в застосуванні до створення менших фрагментів функціональності, порівняно з проектом в цілому. Мета кожної ітерації — отримання працюючої версії програмної системи, що включає функціональність, визначену інтегрованим змістом всіх попередніх та поточної ітерації. Результат фінальної ітерації містить всю необхідну функціональність продукту. Таким чином, із завершенням кожної ітерації продукт отримує приріст — інкремент — до його можливостей, які, значить, розвиваються еволюційно. Ітеративність, інкрементальність і еволюційність в даному випадку є вислів одного й того ж сенсу різними словами зі злегка різних точок зору. З точки зору структури життєвого циклу таку модель називають ітеративною (iterative).

З точки зору розвитку продукту переваги є у інкрементальної (incremental) моделі.

Досвід індустрії показує, що неможливо розглядати кожен з цих поглядів ізольовано. Найчастіше таку змішану еволюційну модель називають просто ітеративна (кажучи про процес) та/або інкрементна (Кажучи про нарощування функціональності продукту).

Каскадна модель з можливістю повернення на попередній крок при необхідності переглянути його результати, стає ітеративною.

Ітеративний процес припускає, що різні види діяльності не прив'язані намертво до певних етапів розробки, а виконуються в міру необхідності, іноді повторюються, до тих пір, поки не буде отриманий потрібний результат.

Разом з гнучкістю і можливістю швидко реагувати на зміни, ітеративні моделі привносять додаткові складності в управління проектом та відстеження його ходу. При використанні ітеративного підходу значно складніше стає адекватно оцінити поточний стан проекту та спланувати довгостроковий розвиток подій, а також передбачити терміни і ресурси, необхідні для забезпечення певної якості результату. При ітераціях доводиться відкидати частину зробленої раніше роботи.

Різні варіанти ітераційного підходу реалізовані в більшості сучасних методологій розробки (RUP, MSF, XP).

Спіральна модель (англ. spiral model) була розроблена в середині 1980-х років Баррі Боем.

Вона заснована на класичному циклі Демінга PDCA (plan-do-check-act — планування-дія-перевірка-коректування). При використанні цієї моделі ПЗ створюється в кілька ітерацій (витків спіралі) методом прототипування.

Кожна ітерація відповідає створенню фрагмента або версії ПЗ, на ній уточнюються цілі і характеристики проекту, оцінюється якість отриманих результатів і плануються роботи наступної ітерації.

Назву ця модель отримала через зображення ходу робіт в полярних координатах, в яких кут відповідає виконуваному етапу в рамках загальної структури ітерацій, а віддалення від початку координат — витраченим ресурсам.

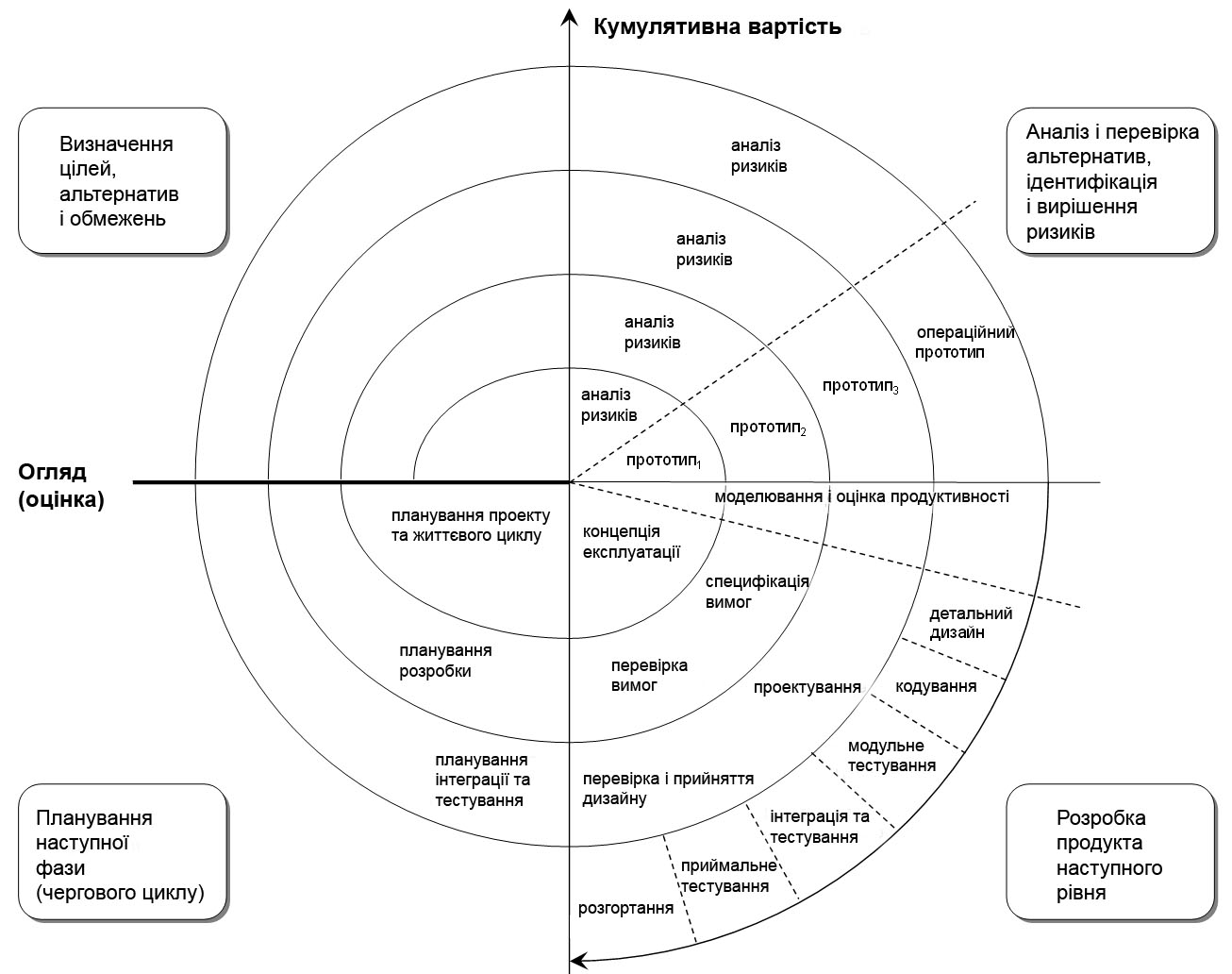


Рисунок 2.2 — Спіральна модель

На кожній ітерації оцінюються:

1. ризик перевищення термінів і вартості проекту;
2. необхідність виконання ще однієї ітерації;
3. ступінь повноти і точності розуміння вимог до системи;
4. доцільність припинення проекту.

Важливо розуміти, що спіральна модель є не альтернативою еволюційної моделі (моделі IID), а спеціально опрацьованим варіантом.

Таким чином, для розробки програмного забезпечення вибираємо спіральну модель розробки програмного забезпечення. Початкові ітерації були реалізовані при написанні бакалаврської роботи. Тоді перед нами стояли завдання створити програмний імітатор некогерентно розсіяного сигналу, зараз вимоги змінились і перед нами постало завдання створити комп’ютерну модель. Методології розробки, які ґрунтуються на даній моделі будуть розглянуті нижче.

**2.2 Вибір методології розробки програмного забезпечення**

Залежно від поставленої замовником задачі, фахівці використовують різні підходи до її реалізації. Таких підходів/методик декілька і всі вони мають свої переваги, і свої недоліки в залежності від того яка саме задача ставиться перед виконавцем робіт.

В ході проектування були вибрані гнучкі методології розробки програмного забезпечення. Гнучка розробка ПЗ — це концептуальний каркас для реалізації проектів програмної інженерії. Вона встановлює першорядною цінністю безпосередньо розробку ПЗ, а всі інші активності (написання проектної документації, підтримка моделей архітектури ПЗ тощо) є другорядними. Метою гнучких методів є розробка ПЗ в строк, в рамках встановленого бюджету і, найважливіше, створення ПЗ саме таким, яким його хоче бачити замовник [6].

Узагальнена модель процесу розробки програмного забезпечення представлена на рисунку 2.3.

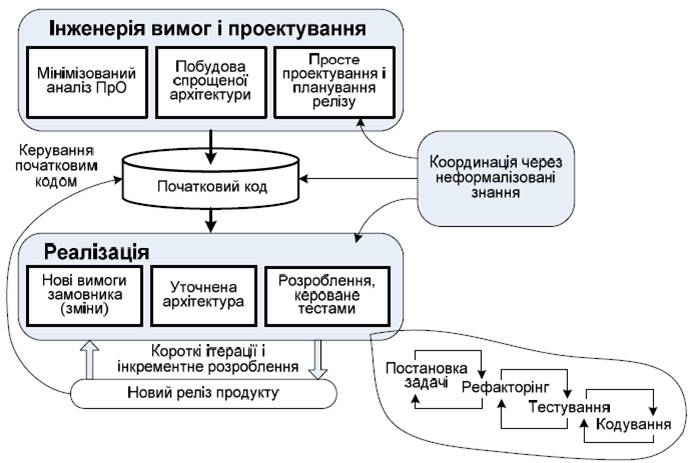


Рисунок 2.3 — Узагальнена модель розробки програмного забезпечення

Головна перевага гнучких технологій — це здатність своєчасно реагувати на часті зміни проектних вимог замовника, а також скорочена дата випуску програмного продукту.

Оскільки проект розроблювався невеликою кількістю програмістів, тому доцільно вибирати методологію, яка передбачає роботу в невеликій команді та часті релізи програмного продукту. Саме для цього підходить гнучка методологія — екстремальне програмування (eXtreme Programming — XP).

Екстремальне програмування — це спрощена методологія організації розробки програм для невеликих і середніх за розміром команд розробників, що займаються створенням програмного продукту в умовах неясних або вимог, що швидко змінюються.

Основними принципами XP є наступні:

1. Ітеративність. Розробка ведеться короткими ітераціями за наявності активного взаємозв'язку із замовником. Ітерації пропонується робити короткими, рекомендована тривалість 2–3. За одну ітерацію група програмістів зобов'язана реалізувати декілька властивостей системи, кожна з яких описується в так званій користувацькій історії (КІ). КІ в даному випадку є початковою інформацією, на підставі якої створюється програмний модуль. Опис КІ є коротким, 1–2 абзаци. КІ пишуться самими користувачами, які в XP є частиною команди. Відсутність формалізації опису вхідних даних проекту в XP прагнуть компенсувати за рахунок активного включення в процес розробки замовника як повноправного члена команди.
2. Простота рішень. Ухвалюється перше просте робоче рішення. Екстремальність методу пов'язана з високим ступенем ризику рішення, обумовленого поверхневістю аналізу і жорстким часовим графіком. Реалізується мінімальний набір головних функцій системи на першій і кожній подальшій ітерації; функціональність розширюється на кожній ітерації.
3. Інтенсивна розробка малими групами (не більше 10 чоловік) і парне програмування (коли два програмісти разом створюють код на одному загальному робочому місці), активне спілкування в групі і між групами. Все це націлено на якомога раніше виявлення проблем (як помилок, так і зриву термінів). Парне програмування направлене на рішення задачі стабілізації проекту. При застосуванні методології XP є високий ризик втрати коду внаслідок відходу програміста, що не витримав інтенсивного графіку роботи. В цьому випадку другий програміст з пари грає роль спадкоємця коду. Важливим є і те, як саме розподілені групи в робочому просторі — в XP використовується відкритий робочий простір, який передбачає швидкий і вільний доступ "всіх до всіх".
4. Зворотний зв'язок із замовником, представник якого фактично залучений в процес розробки.
5. Достатній ступінь сміливості і бажання йти на ризик.

XP розглядає кодування як ключовий чинник впродовж всього проекту. Саме код є сполучною ланкою у спілкуванні між різними командами, розділеними географічно.

Життєвий цикл і поведінка складних об'єктів визначається за допомогою тест-кейсів, тобто знову ж таки через код. Проте процедура розробки має на увазі постачання цього коду відповідно до графіка.

Життєвий цикл проекту в XP складається з послідовності релізів.

Кожний реліз — це повноцінна версія продукту, яку може використовувати замовник, і яка містить додаткову функціональність в порівнянні з попереднім релізом. Реліз з'являється в результаті однієї або декількох ітерацій, що тривають від одного до чотирьох тижнів.

Модель життєвого циклу програмного забезпечення в екстремальному програмуванні показана на рисунку 2.4.

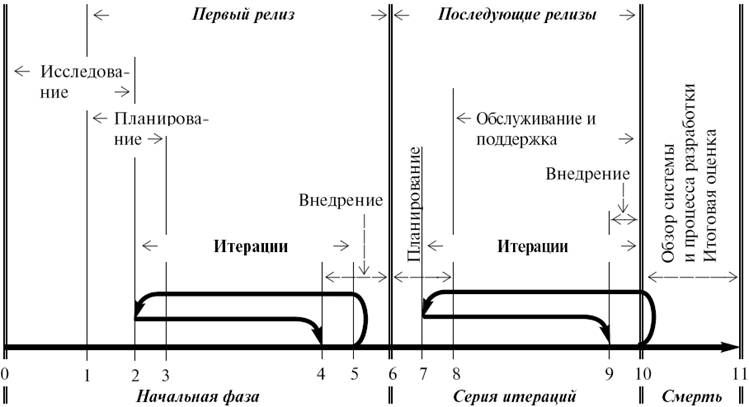


Рисунок 2.4 — Модель життєвого циклу

Таким чином, для розробки та проектування програми вибираємо методологію екстремального програмування, яка ґрунтується на еволюційній моделі розробки програмного забезпечення.

3 РОЗРОБКА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**3.1 Вимоги до функціональних характеристик програмного продукту.**

Функціональність – здатність ПЗ забезпечити функції, які виконують заявлені потреби та потреби, що маються на увазі при використанні ПЗ в заданих умовах, включає наступні функції:

1. функціональна повнота – здатність ПЗ забезпечити відповідний набір функцій для заданих завдань та цілей користувача;
2. правильність – здатність ПЗ забезпечувати вірні або припустимі результати або дії з необхідним ступенем точності;
3. здатність до взаємодії – здатність ПЗ взаємодіяти із зазначеними системами;
4. захищеність – здатність ПЗ забезпечити захист інформації та даних (при їхньому зберіганні і передачі) таким чином, що неавторизовані користувачі та системи не будуть мати можливості до їх читання і модифікації, у той час як авторизованим користувачам і системам не буде відмовлено у доступі до них;
5. узгодженість функціональності – здатність ПЗ до дотримання відповідних стандартів, угод, положень законів або подібних рекомендацій, що стосуються функціональності.

**3.2 Вимоги до надійності**

Надійність – група властивостей, що обумовлює здатність програмного забезпечення зберігати працездатність та перетворювати вихідні дані в очікуваний результат у заданих умовах за встановлений час, включає наступні характеристики:

1. безвідмовність – здатність ПЗ уникати відмов (функціонування без відмов), які є результатом наявності дефектів у ПЗ;
2. стійкість до відхилень – здатність ПЗ підтримувати необхідний рівень працездатності у випадках прояву програмних дефектів або порушення його інтерфейсу;
3. відновлюваність – здатність ПЗ відновлювати заданий рівень працездатності, а також відновлювати дані, які безпосередньо ушкоджені у випадку виникнення відмови, після перезапуску ПЗ (автоматичного або оператором);
4. узгодженість надійності – здатність ПЗ дотримуватись відповідних стандартів, угод, положень законів або подібних рекомендацій, що стосуються надійності.
   1. **Вимоги до інтерфейсу**
5. Інтерфейс повинен бути сумісним з потребами та можливостями користувача.
6. Забезпечувати простоту переходу від виконання однієї операції до іншої.
7. Забезпечувати користувача на високому рівні вказівками стосовно його можливих дій, а також генерувати належний зворотній зв'язок на його запити.
8. Надавати користувачу можливість відчувати себе повноправним керівником ситуації при розв’язанні всіх типів задач, тобто, забезпечувати його всією необхідною інформацією.
9. Забезпечувати користувача різними, взаємно доповнюючими формами представлення результатів в залежності від типу запиту або від характеру отриманого рішення.
10. Допустимі запити користувача повинні бути чіткими і однозначними для користувачів всіх рівнів, а також для прикладних задач всіх типів.
11. Реакція системи на всі типи запитів повинна бути однозначною і зрозумілою і, по можливості, простою.
12. Дружність інтерфейсу - максимальна простота його використання і готовність в повній мірі задовольнити запити користувача при розв’язанні визначеного класу задач.
13. Гнучкість інтерфейсу – можливість його адаптування до розв’язання конкретної задачі. Якщо розв’язувана задача дуже складна, то інтерфейс повинен полегшувати формулювання запитів і видавати результати у формі, яка легко і швидко сприймається користувачем.
14. Інтерфейс не повинен бути перевантажений деталями щодо представлення розв’язку поставленої задачі.
15. Інтерфейс не повинен містити зайвих декоративних деталей, які відволікають від головної задачі.
16. Інтерфейс повинен бути консистентним, тобто, ґрунтуватись на використанні відомих, загальноприйнятих методів і засобів представлення інформації.
17. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В ході написання ПЗ було розроблено алгоритм роботи програми. Алгоритм – послідовність дій, що приводить до рішення поставленої задачі за скінчену кількість операцій. Можна виділити три основних види обчислювальних алгоритмів:

1. лінійний, в якому всі команди виконуються в порядку їх запису;
2. з розгалуженням, в якому в залежності від деякої умови виконуються ті чи інші команди;
3. циклічний, в якому деякі команди повторюються;

Найчастіше алгоритми виконуються у вигляді блок-схем

Блок-схема алгоритму – це графічне представлення логічної структури алгоритму, де кожний етап обробки інформації зображується у вигляді геометричних символів.

Загальний алгоритм має вигляд (рисунок 4.1)

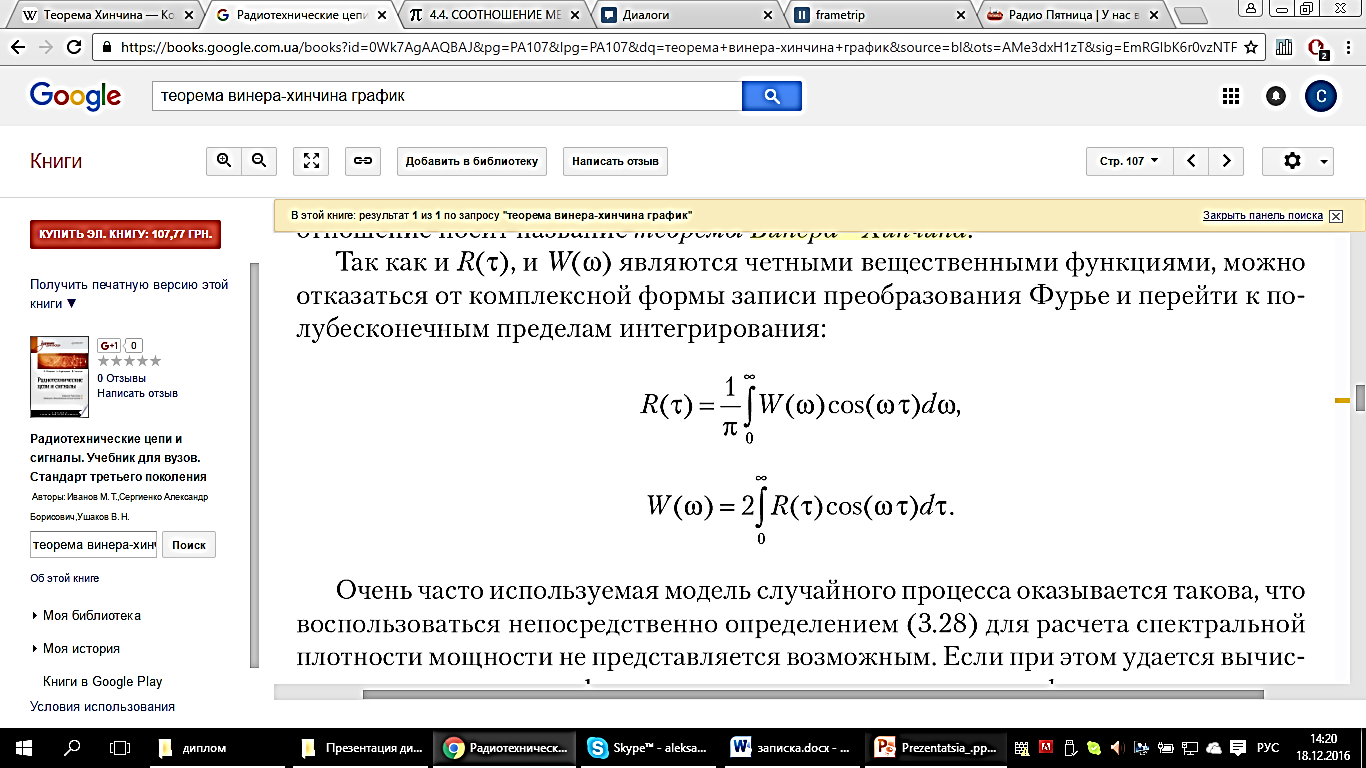


Рисунок 4.1 – Алгоритм роботи програми

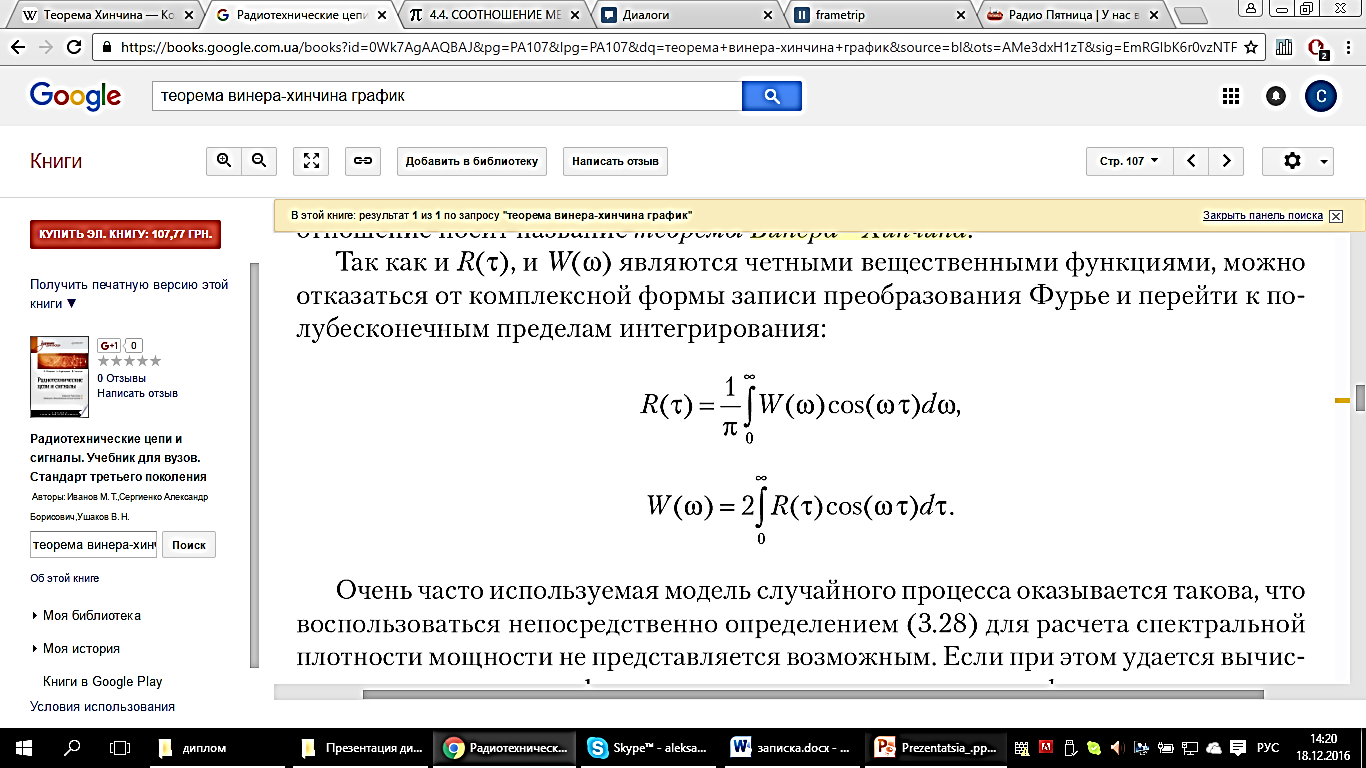
З метою зручності, було прийнято рішення розділити роботу програми на декілька кроків. Програма складається з 5 етапів – кроків. Це дає змогу користувачеві відслідковувати виконання ключових операцій та вносити необхідні параметри іоносфери, які будуть розглянуті нижче.

На першому кроці програма генерує сигнал для заданих параметрів іоносфери. В ході виконання першого кроку формується спектр з кроком в 10 Гц в діапазоні частот 0-15 кГц для кожної із заданих висот. Далі розраховується АКФ для 19 часових затримок, з використанням теореми Віннера-Хінчена.

Теорема Хінчина-Колмогорова (також відома як теорема Вінера-Хінчина) стверджує що спектральна густина потужності стаціонарного в широкому значенні випадкового процесу є перетворенням Фур’є відповідної автокореляційної функції (формули 4.1 та 4.2)



(4.1)



(4.2)

Теорема зручна для аналізу лінійних стаціонарних систем, де вхідні і вихідні значення не інтегруємі в квадратурах, через що перетворення Фур’є не існує. Як наслідок, перетворення Фур’є автокореляційної функції вхідного сигналу системи дорівнює добутку перетворення Фур’є автокореляційної функції вхідного сигналу на квадрат модуля перетворення Фур’є її імпульсної характеристики. Це виконується навіть тоді, коли перетворення Фур’є вхідних і вихідних сигналів не існує, через те що вони не інтегруємі. Через вхідні і вихідні параметри не можуть бути прямо пов’язанні перетворення Фур’є імпульсної характеристики передаточної функції.

Через те, що перетворення Фур’є автокореляційної функції сигналу є спектр потужності сигналу, випливає, що спектр потужності вихідного сигналу рівний добутку спектру потужності вхідного і передаточної функції системи. Цей наслідок використовується в знаходженні спектру параметричним методом.

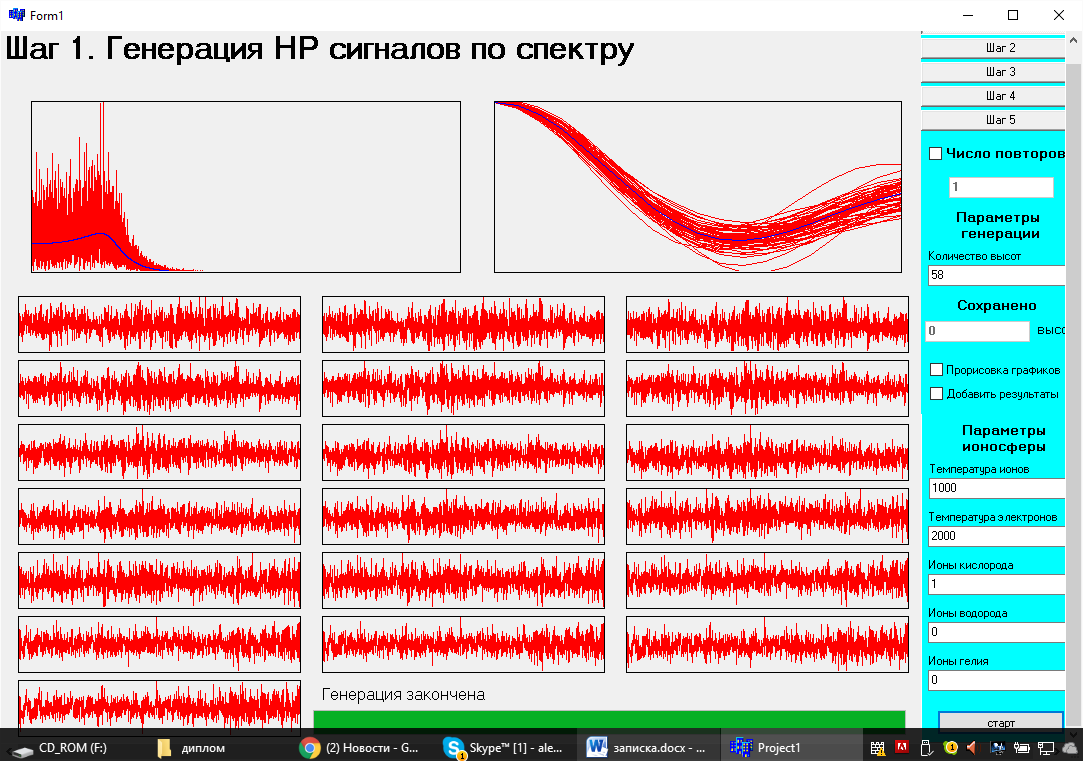


Рисунок 4.2 – Вікно програми, крок 1

На рисунку 4.2 показано робоче вікно програми, а саме крок 1. Справа знаходиться панель управління, на якій знаходять кнопки переходу між етапами роботи програми. Та додаткові параметри, які задає користувач, а саме введення кількості повторів, введення кількості висот, введення таких параметрів іоносфери як: температура іонів та електронів, концентрація іонів кисню та водню та кнопка старт.

На лівому верхньому графіку показано спектр частот з кроком 10 Гц в діапазоні 0 – 15 кГц. На правому верхньому графіку показано розраховані АКФ для 58 висот по 19 часових затримок на кожну. Нижче цих графіків показані 19 сигналів часових затримок. Для зручності користувач може збільшити кожен графік двічі клікнувши по потрібному графіку.

На другому етапі роботи реалізовано додавання сигналів для 22 окремих висот з метою урахування протяжності імпульсу, після чого знову розраховується автокореляційні функції сигналів.

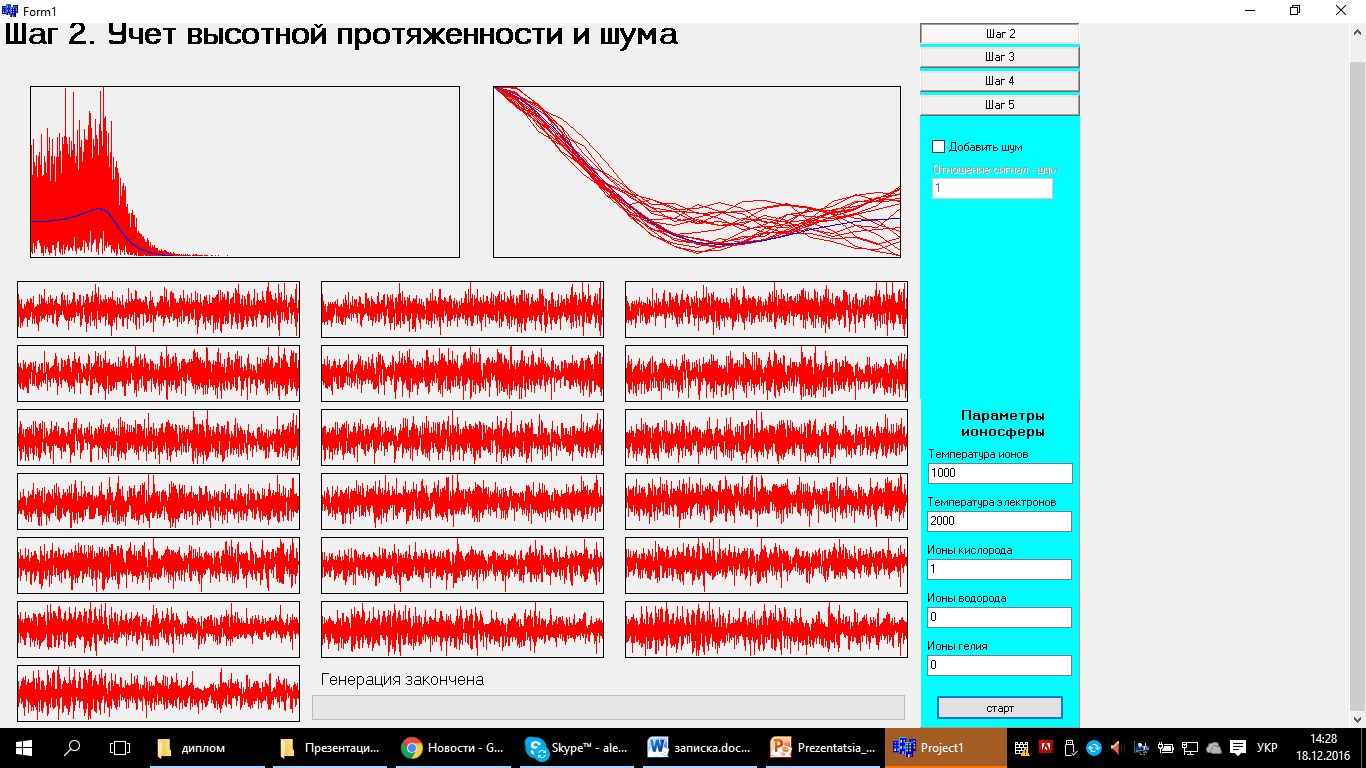


Рисунок 4.3 – Вікно програми, крок 2

На рисунку 4.3 показано інтерфейс другого етапу роботи програми. На даному етапі користувач може задати відношення сигнал шум. По аналогії з першим кроком користувачу показано спектр сигналу, автокореляційні функції та 19 сигналів часових затримок.

На третьому кроці реалізовано трапецеїдальне додавання, яке дозволяє суттєво зменшити статистичний рівень розсіювання значень АКФ, які відповідають великим затримкам (збільшеня статистичного розсіювання особливо помітно на висотах вище максимума іонізації, де відношення сигнал/шум менше одиниці). Це робиться з метою урівняти всі корелючі об’ми.

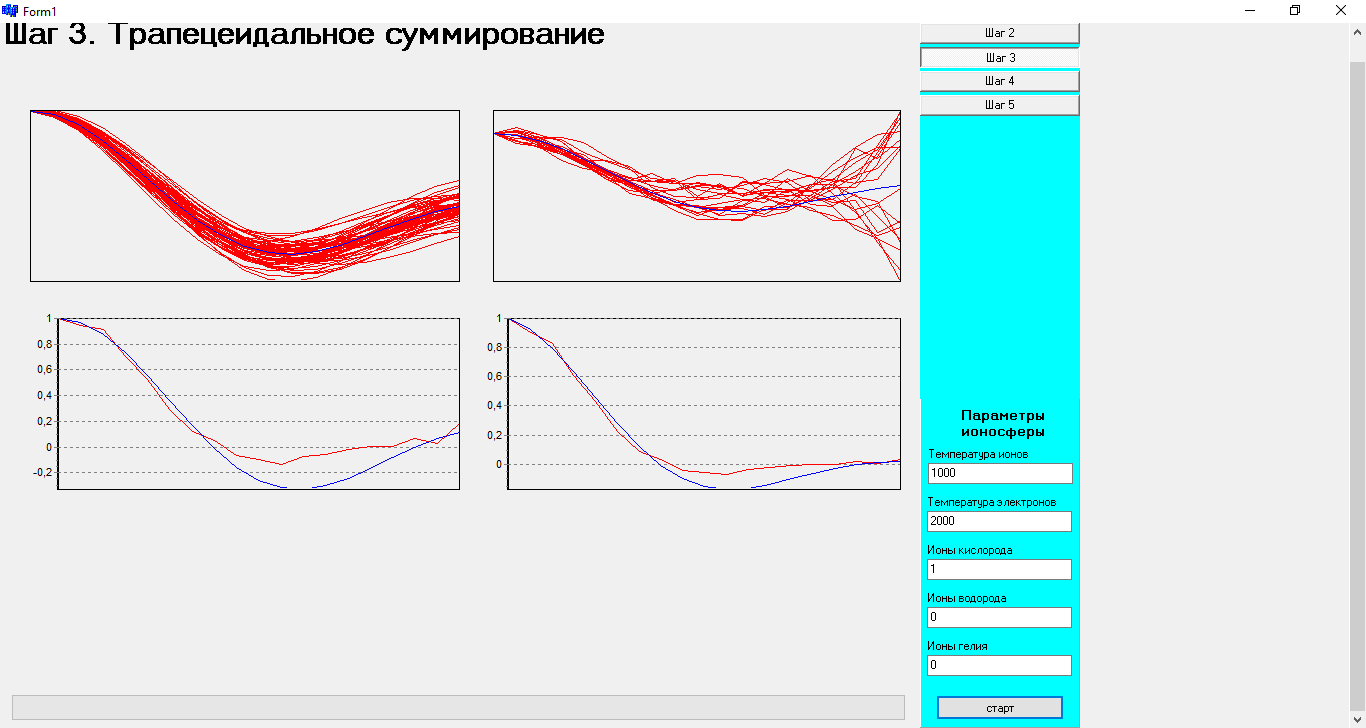


Рисунок 4.4 – Вікно програми, крок 3

На даному кроці ми отримуємо 4 графіки рисунок 4.4. В лівому верхньому кутку ми бачимо АКФ отримані на першому кроці, в правому кутку АКФ отримані на другому кроці. В низу лівий графік показує трапецеїдальне додавання, на правому графіку показано трапецеїдальне додавання помножене на коефіцієнт з метою очистити графік від кутового згладжування.

На 4 кроці програма проводить врахування імпульсних характеристик фільтра в правому верхньому кутку рисунок 4.5. Далі програма робить згортку сигналу між кореляційною функцією з третього кроку і кореляційною імпульсною характеристикою фільтра. Задля наближення до теоретичної функції програма множить згортку на трикутну форму. Це необхідно для врахування зменшення корелюю чого об’єму зі збільшенням затримки.

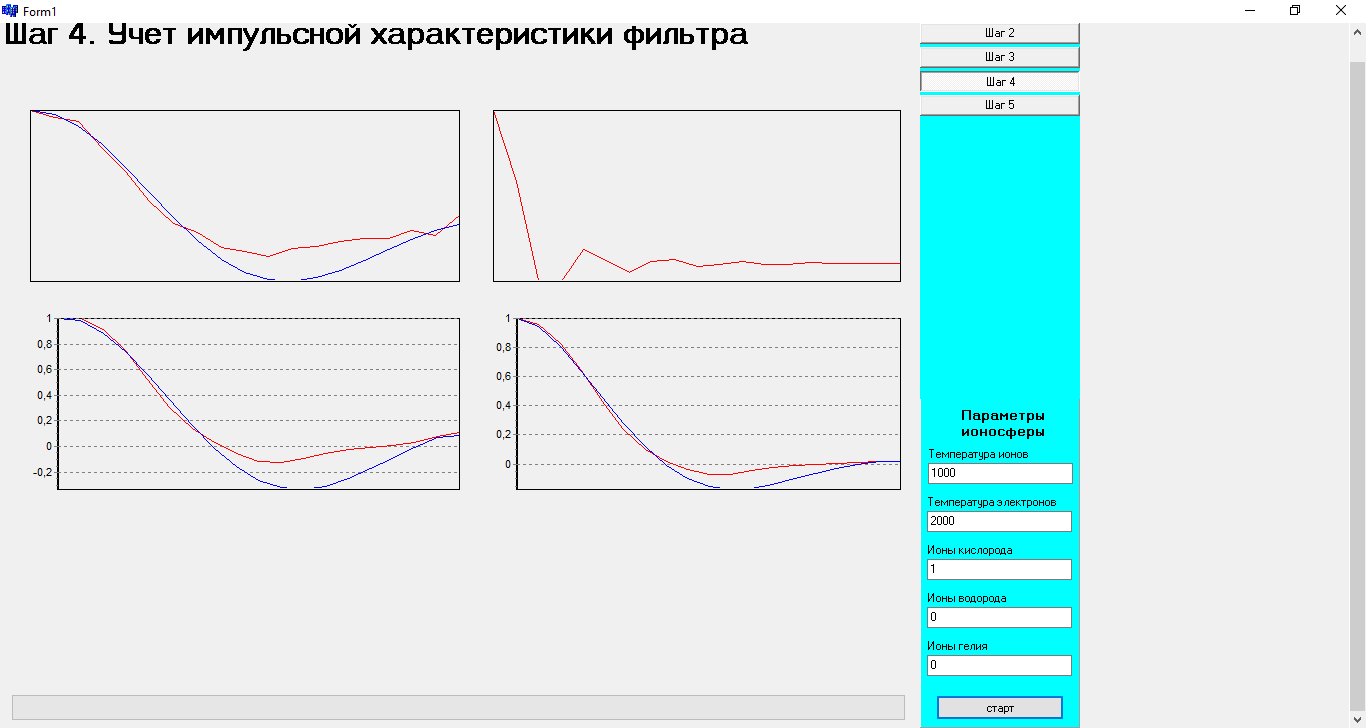


Рисунок 4.5 – Вікно програми, крок 4

На 5 останньому кроці реалізована візуалізація результатів. На даному етапі програма, за допомогою методу найменших квадратів, відновлює температуру іонів та електронів в кисневому наближенні з використанням бібліотеки теоретичних кореляційних функцій.

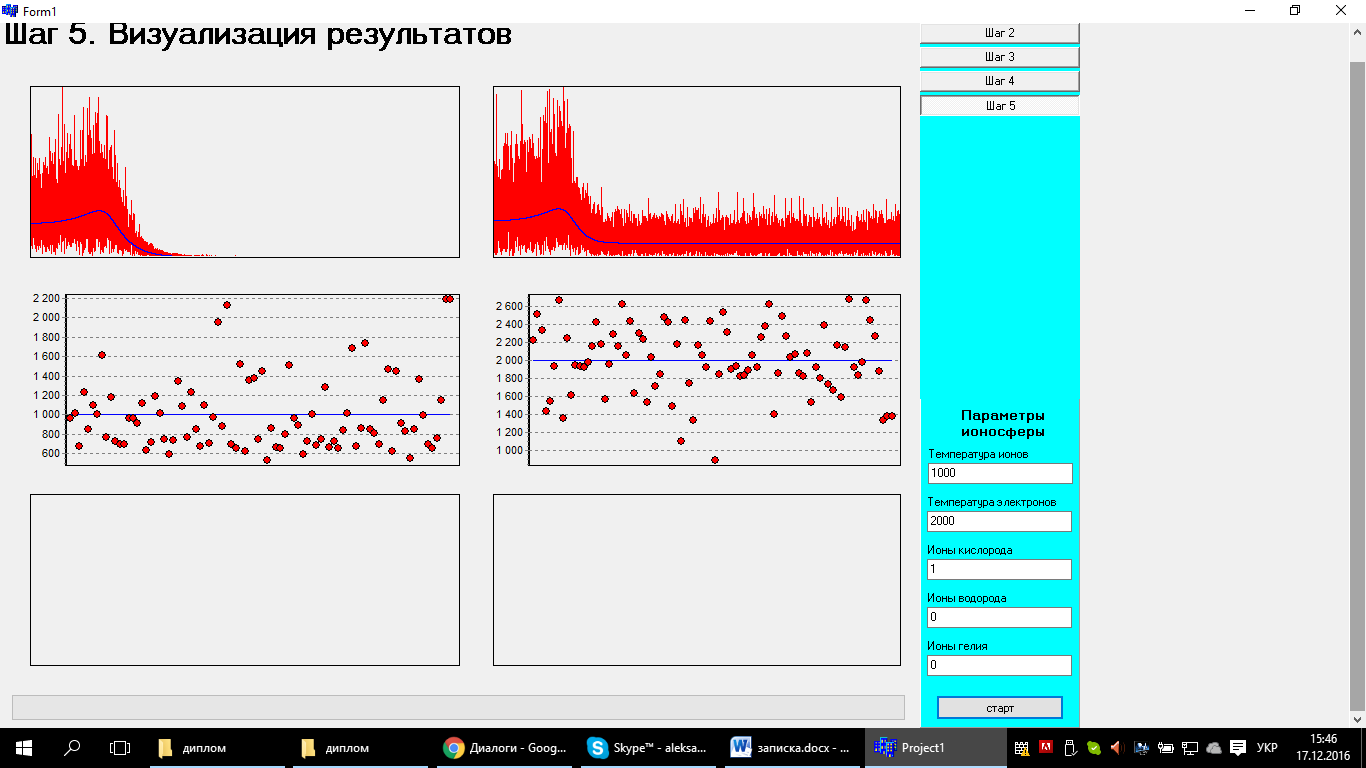


Рисунок 4.6 – Вікно програми, крок 5

Метод найменших квадратів, завдяки широкій сфері застосування, посідає виняткове місце серед методів математичної статистики.

Суть методу найменших квадратів полягає у знаходженні наближеного розв'язку надлишково-визначеної системи. Часто застосовується в регресійному аналізі. На практиці найчастіше використовується лінійний метод найменших квадратів, що використовується у випадку системи лінійних рівнянь.

Вихідний параметр процесу *y*, лінійно залежить від вхідного параметра *x* (суцільна пряма лінія на рисунку. 4.7).

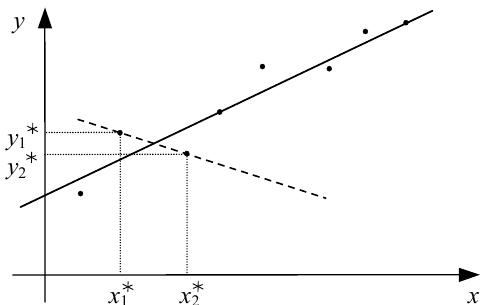


Рисунок 4.7 — Метод найменших квадратів

Тобто припустимо, що статична характеристика цього процесу може бути подана у вигляді

*.* (4.3)

де *a* і *b* — коефіцієнти, для визначення числових значень яких необхідно, як мінімум, задати два значення *x*1, *x*2 вхідній величині x і заміряти відповідні їм значення *y*1 *y*2 вихідної величини *y*, оскільки лише під час виконання цих умов для моделі (4.3) можна скласти систему двох алгебраїчних рівнянь із двома невідомими *a* і *b*

|  |  |  |
| --- | --- | --- |
|  | http://posibnyky.vntu.edu.ua/mk/img/280.gif | (4.4) |

Але результати будь-яких експериментальних вимірювань несуть у собі похибки, обумовлені класом точності вимірювальних засобів, дією різноманітних завад, неточністю зчитування показів приладів, округленням під час приведення даних до однакових умов обробки інформації — список умов виникнення похибок можна продовжити.

Тож через наявність цих похибок в експериментальних значеннях *x*1, *x*2, *y*1, *y*2 безпосередній розв’язок системи рівнянь (4.4) відносно *a* та *b* може нести в собі похибку в 10, 100, 1000 і більше відсотків.

Наприклад, якщо використати лише значення *x*1\*, *x*2\*, *y*1\*, *y*2\* для розв’язання системи рівнянь (4.4), то похибка буде вже не у відсотках, а у характері функціональної залежності (пунктирна лінія на рисунок 4.7).

У свій час Гаусс запропонував інший спосіб визначення коефіцієнтів *a*, *b* моделі (4.3). Він запропонував сформувати суму квадратів різниць ∑N між теоретично заданими за допомогою рівняння (4.3) значеннями вихідної координати y при значеннях аргументу *xi*, *i* = 1,N та її експериментальними значеннями *yi*:

|  |  |  |
| --- | --- | --- |
|  | http://posibnyky.vntu.edu.ua/mk/img/282.gif | (4.5) |

а потім знайти такі значення коефіцієнтів a , b рівняння (4.3), котрі мінімізують вираз (4.5).

Від цієї процедури і назва методу — метод найменших квадратів [7].

Лістинг програми наведено в Додатку А.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Наведемо приклади використання розробленого програмного забезпечення.

При запуску програми ми бачимо вікно, в якому розміщені елементи управління, графіки на які будуть виводиться результати роботи програми та індикатор роботи програми (рисунок 5.1)

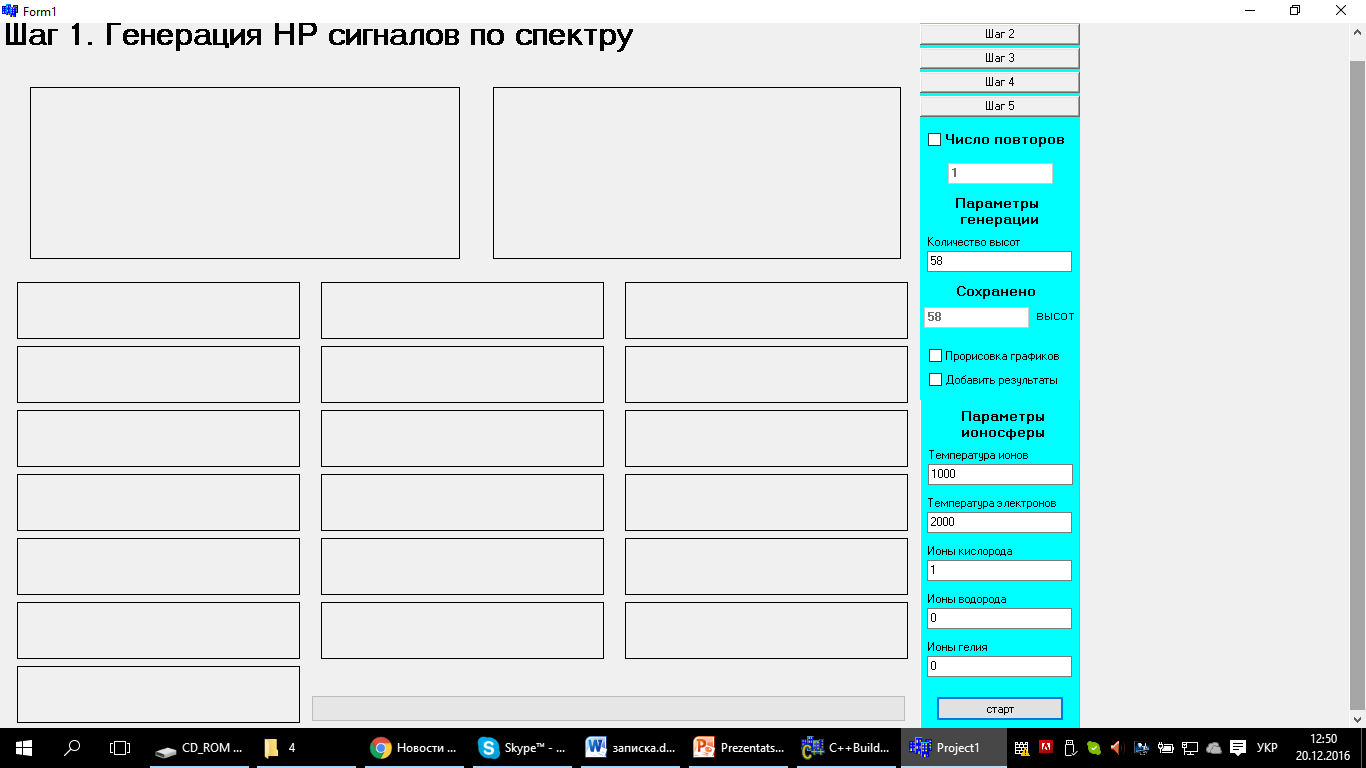


Рисунок 5.1 – Робоча область програми

За замовчування кількість висот рівна 58, температура іонів 1000, температура електронів 2000 і враховуючи те, що обчислення будуть вестися в кисневому приближені значення іонів кисню рівне 1 решта 0. Одразу після натискання кнопки старт програма інформує користувача про етапи роботи. (рисунок 5.2).

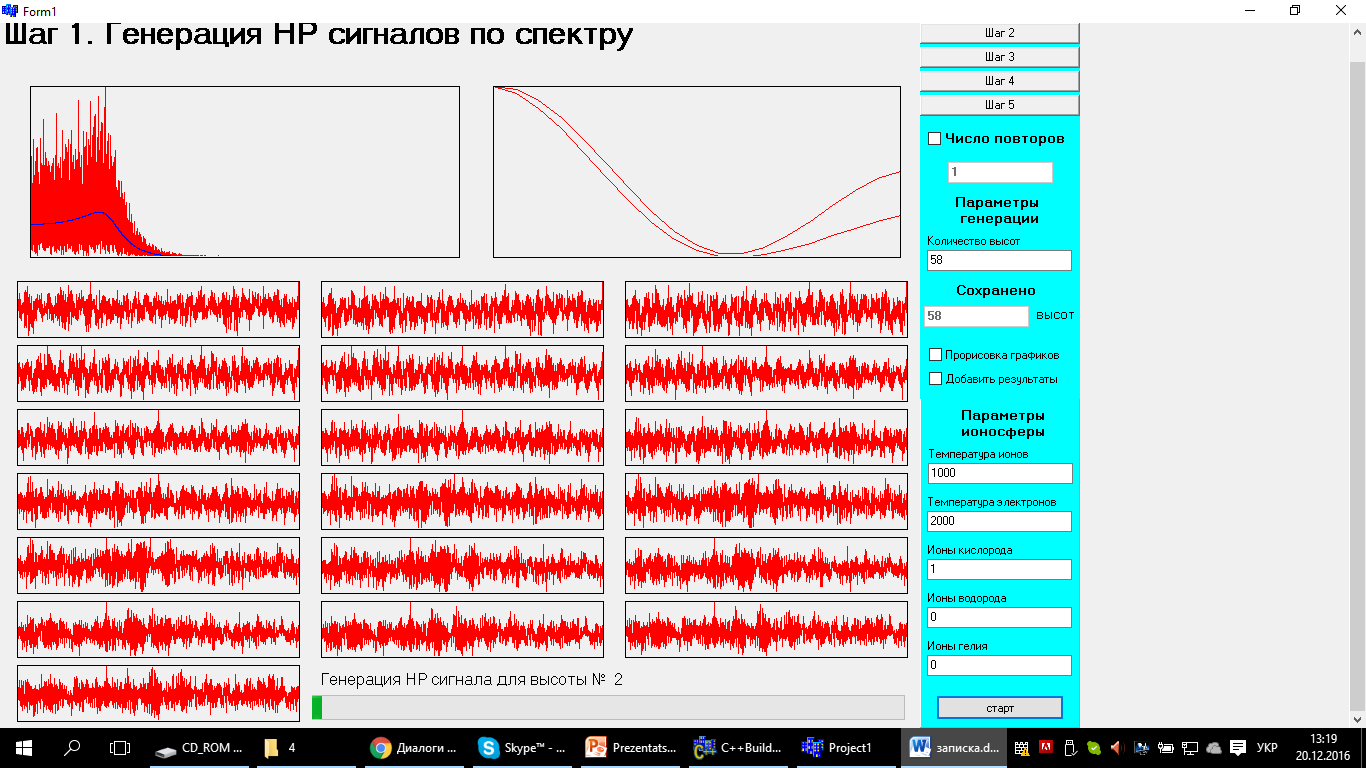


Рисунок 5.2 – Виконання першого етапу

Після кожного етапу програма інформує користувача про завершення операцій

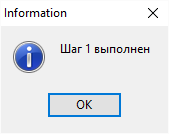


Рисунок 5.3 – Завершення етапу роботи

В програмі також реалізовані такі опції, як прорисковка графіків , додавання результатів.

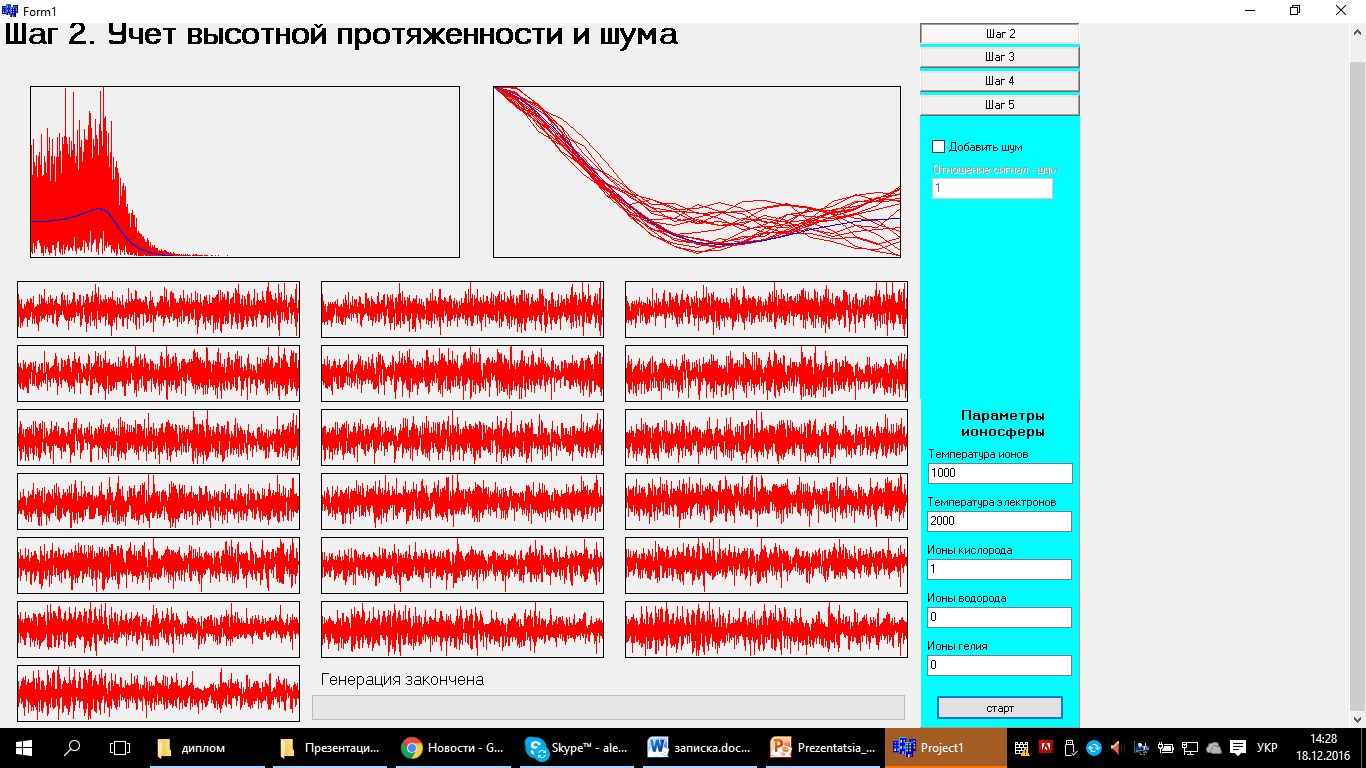


Рисунок 5.4 – Другий етап роботи

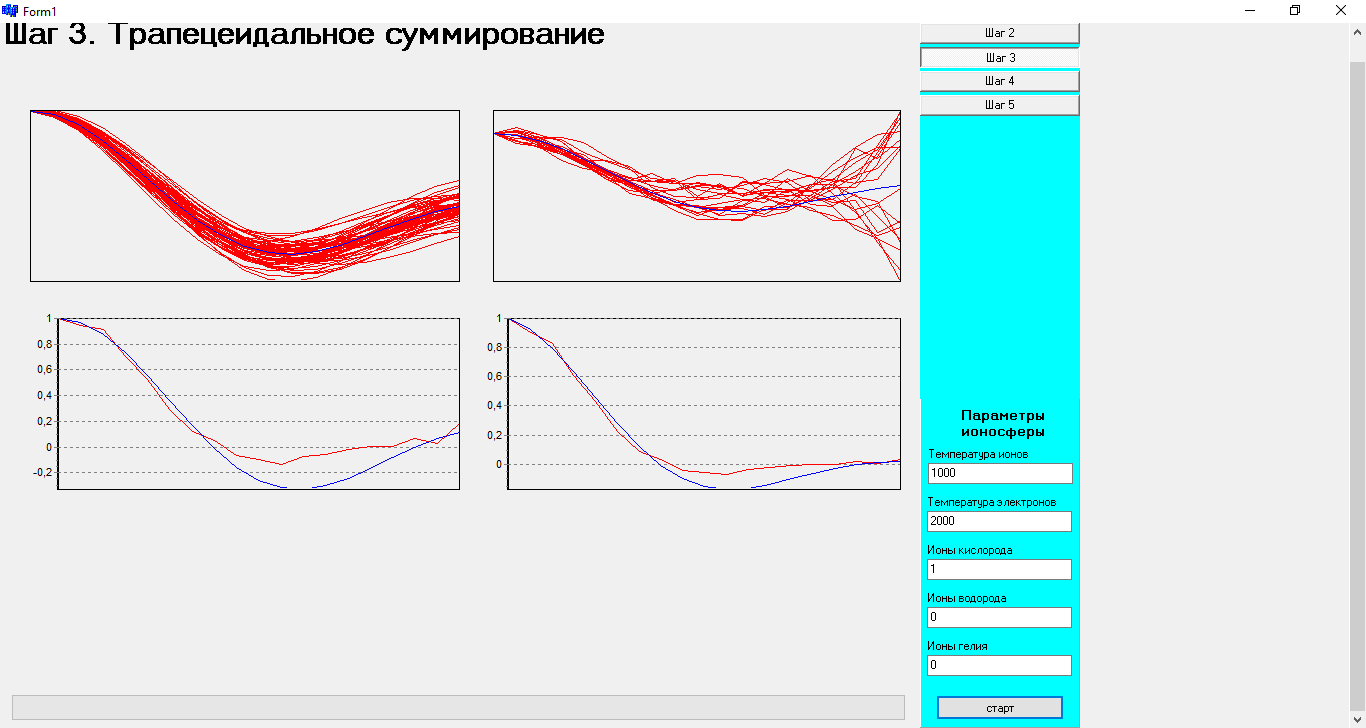


Рисунок 5.5 – Третій етап роботи

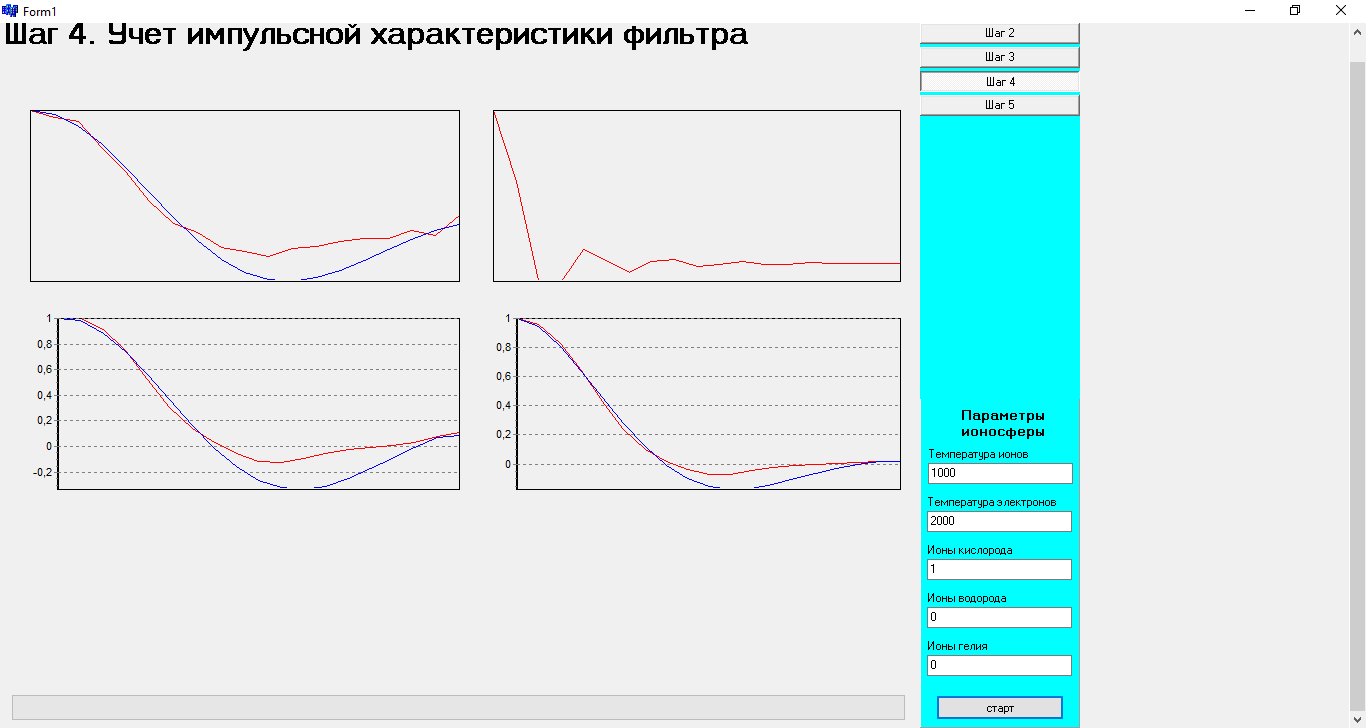


Рисунок 5.6 – Четвертий етап роботи

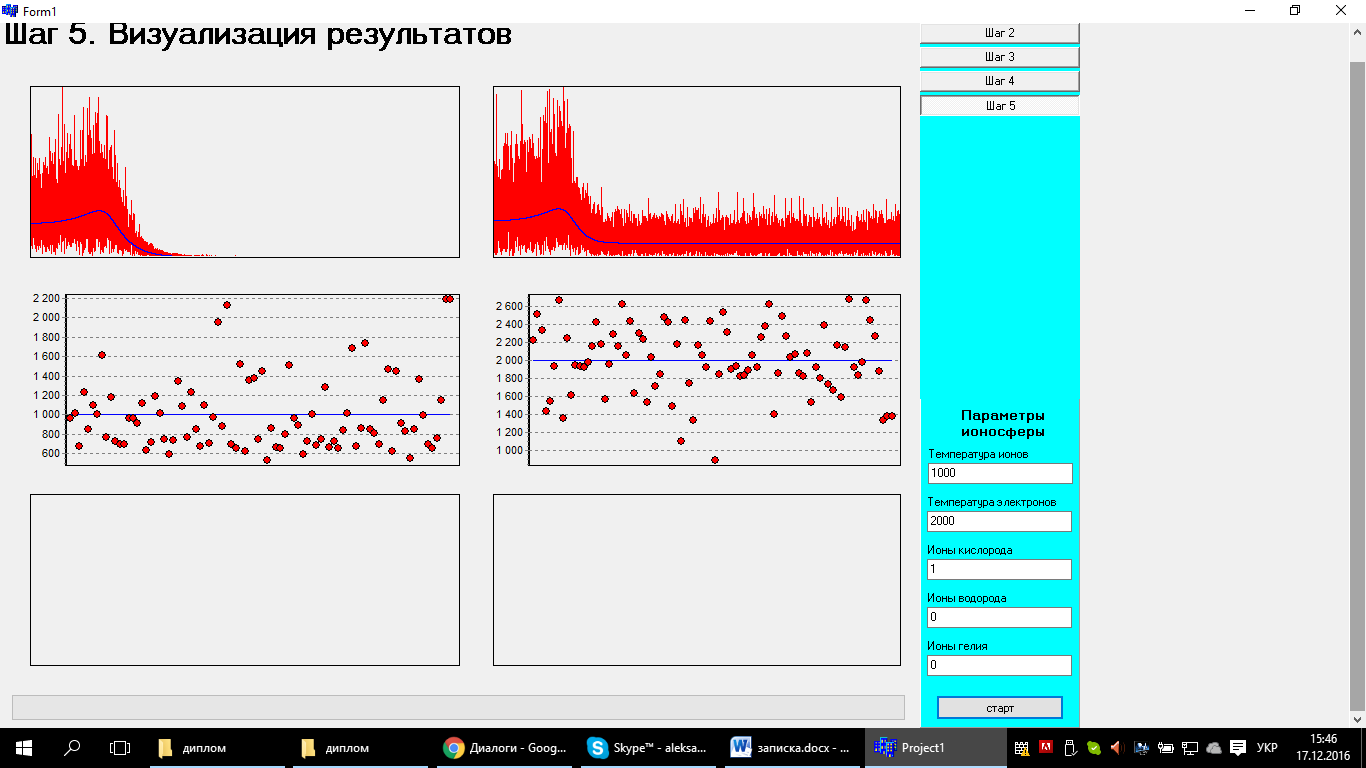


Рисунок 5.7 – П’ятий етап роботи

Програму також було протестовано за інших значеннях температур іонів та електронів

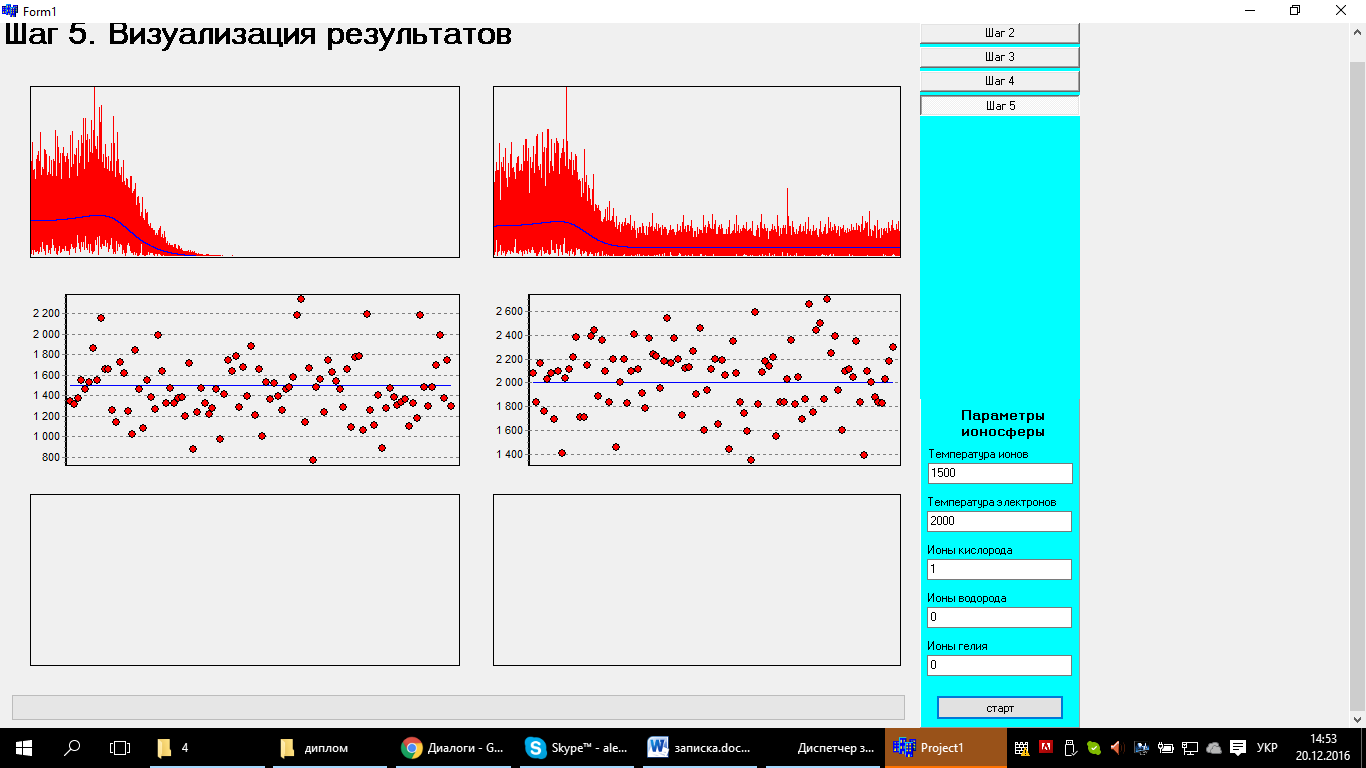


Рисунок 5.8 – Температура іонів 1500, температура електронів 2000

В програмі реалізований захист від помилок, так для коректної роботи необхідно вказати не менше 40 висот. Якщо їх буде менше програма видасть повідомлення про помилку.

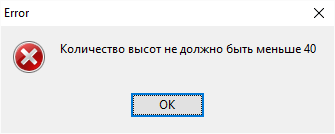


Рисунок 5.9 – Повідомлення про помилку

Виходячи з вище наведеного можна зробити висновок про коректність роботи програми та повноту виконання поставлених задач.

6 Техніко-економічна частина

**6.1 Доцільність розробки продукту**

Ціль розробки програмного продукту, під назвою «Розробка комп’ютерної моделі сигналу, що реєструється радаром некогерентного розсіяння», полягає в написанні програми, для розробки комп’ютерної моделі некогерентно розсіяного сигналу із використанням корекції висотних профілів значень автокореляційних функцій.

Аналіз питань про ринок збуту, конкуренції, стратегії маркетингу і так далі дозволяє робити висновок про доцільність застосування розроблювального програмного забезпечення.

**6.2 Опис програмного продукту**

6.2.1 Найменування розробки

Програмний продукт називається «Розробка комп’ютерної моделі сигналу, що реєструється радаром некогерентного розсіяння».

6.2.2 Призначення продукту

Програмний продукт, що розглядається призначений для побудови комп’ютерної моделі некогерентно розсіяного сигналу із використанням корекції висотних профілів автокореляційних функцій.

6.2.3 Область використання

Програмний продукт орієнтований на використання викладачами університетів, науковими співробітниками і інженерами, у якості навчального інструментарію для студентів ВНЗ.

6.2.4 Характеристика програмного продукту

Характеристики програмного продукту (ПП) приведені в таблиці 6.1.

Таблиця 6.1 — Характеристики ПП

|  |  |
| --- | --- |
| Найменування | Значення параметра |
| Операційна система | Windows 7 та вище, Linux |
| Оперативна пам’ять | 1024 Мб мінімум |
| Процесор/Архітектура | x86 мінімум |
| Мова програмування | C++ |

**6.3 Дослідження й аналіз ринку збуту**

6.3.1 Сегментація ринку по споживачах

Одна з цілей дослідження ринку - сегментування ринку та визначення ємкості сегментів. Ємкість товарного ринку – показник, який характеризує принципово можливий об’єм збуту товару. Ємкість товару визначається об’ємом (у фізичних величинах), що реалізуються на ньому товарів протягом року і складається з ємкості його сегментів. Джерела відомостей про ємкість ринку – статичні, галузеві та економічна інформація. Знаючи ємкість ринку і тенденції його зміни, можливо оцінити ринок його збуту. Сегмент ринку – частина ринку, що виділена особливим чином, тобто група споживачів, які мають певні спільні ознаки. Основними вимогами споживачів є простота використання, мінімальна вартість, надійність. Сегментація ринку по категоріям споживачів представлена в таблиці 6.2.

Таблиця 6.2 – Сегментація ранку по основних споживачах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Галузь використання (сегменти) | Код сегменту | Споживачі | | |
| I | II | III |
| Вищі навчальні заклади | А |  | х | х |
| Науково дослідницькі інститути | Б |  |  | х |
| Регіональні ОЦ | В | х |  | х |

Ймовірний продукт орієнтований на наступні категорії споживачів:

I – інженери, технологи, конструктори, експлуатаційники;

II – викладачі ВНЗів, технікумів, шкіл, лаборанти;

III – наукові співробітники.

Як видно з таблиці 6.2, дана розробка призначена для використання викладачами університетів, науковими співробітниками і інженерами, а також, у якості навчального інструментарію для студентів ВНЗ. Уточнимо ємність сегментів ринку (дані в таблиці 6.3).

Таблиця 6.3 – Аналіз ємності сегментів ринку

|  |  |  |  |
| --- | --- | --- | --- |
| Галузі використання | Кількість об'єктів, шт | Передбачуване число продажів одному об'єктові, шт | Передбачувана ємність сегмента, шт |
| Вищі навчальні заклади | 4 | 6 | 24 |
| Науково дослідницькі інститути | 2 | 1 | 2 |
| Разом місткість ринку | 6 | 7 | 26 |

Як видно з таблиці 6.3, найбільше число передбачуваних продажів одному об'єктові приходиться на науково дослідницькі інститути. Це обумовлено специфікою роботи.

Основними вимогами споживачів є простота використання, мінімальна вартість, надійність.

Продаж розробленого програмного продукту охоплює м. Харків та Харківську область, але не виключний продаж за її межами. Максимальна кількість потенційних споживачів у розглянутому регіоні ­­– 26.

Прогноз обсягів продажів програмного комплексу приведений у таблиці 6.4.

Таблиця 6.4 – Прогноз обсягів продаж

|  |  |
| --- | --- |
| Періоди | Кількість |
| Для першого року реалізації | |
| Січень | 0 |
| Лютий | 2 |
| Березень | 1 |
| Квітень | 1 |
| Травень | 2 |
| Червень | 0 |
| Липень | 0 |
| Серпень | 1 |
| Вересень | 2 |
| Жовтень | 2 |
| Листопад | 1 |
| Грудень | 1 |
| Для другого року реалізації | |
| I квартал | 2 |
| II квартал | 2 |
| III квартал | 3 |
| IV квартал | 1 |
| Для третього року реалізації | |
| За третій рік | 5 |
| Усього | 26 |

6.3.2 Параметрична сегментація ринку

Для проведення багатофакторної сегментації розробки оцінимо її характеристики, що відповідають обраним нами параметрам (по п'ятибальній шкалі). Багатофакторна сегментація приведена в таблиці 6.5.

Таблиця 6.5 – Параметрична сегментація ринку

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактори, що характеризують товар | Категорія  споживачів | | Підсумкова оцінка, бали | Відсоток до загаль-  ного підсумку, % |
|  | А | Б |  |  |
| Ціна | 4 | 5 | 9 | 22 |
| Вимоги до ПЕОМ | 5 | 5 | 10 | 24, 4 |
| Ергометричні характеристики | 3 | 3 | 6 | 14, 6 |
| Надійність роботи | 5 | 5 | 10 | 24, 4 |
| Швидкість обробки | 3 | 3 | 6 | 14, 6 |
| Разом | 20 | 21 | 41 | 100 |

Виходячи з даних таблиці 5.5, можна зробити висновок про те, що такі фактори як ціна і надійність роботи є найбільш важливими, а сегмент ринку Б пред'являє найбільше високі вимоги до сукупності якісних параметрів розроблювального виробу.

**6.4 Оцінка конкурентоздатності**

На ринку існує ряд аналогів, які дозволяють вирішувати подібні задачі, але ці продукти не являються спеціалізованими і розроблені передовими фірмами світу, тому їх ціна дуже велика.

Розроблений програмний продукт поєднує в собі прикладне значення зі зручністю в звертанні і наочністю відображуваної інформації.

Розрахунок узагальненого показника якості будемо робити в порівнянні даного ПП із аналогічним варіантом даної програми.

Вихідні дані для розрахунку приведені в таблиці 6.6.

Таблиця 6.6 – Розрахунок узагальненого показника якості

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Параметри | Одиниці вимі-рювання | Вагомість Bi | Абсолютне значення  параметрів | | Узагальнене значення показників | | | |
| Новий ПП | | Аналог. ПП | |
|  |  |  | Новий ПП  Pi | Аналог. ПП  Piг | Відносний одиничний показник  Miн | Bi∙Mi | Відносний одиничний показник  Мiг | Bi∙Miг |
| 1 Мінім. обсяг ОЗУ | Кб | 0,2 | 60 | 512 | 8,53 | 1,7 | 0 | 0 |
| 2. Час розрахунку тестового приклада | сек. | 0,2 | 30 | 40 | 1,3 | 0,26 | 1,0 | 0,2 |
| 3. Ціна продажу | грн. | 0,2 | 4 500 | 10 000 | 2 | 0,4 | 1,0 | 0,2 |
| 4. Витрати на навчання | грн. | 0,2 | 100 | 1 000 | 10 | 2 | 1,0 | 0,2 |
| 5. Витрати на адаптацію до даних споживача | грн. | 0,2 | 400 | 1 500 | 3,75 | 0,75 | 1,0 | 0,2 |
| Усього |  | 1 |  |  |  | 5,11 |  | 1 |

Величина відносного показника якості обчислюється по формулі:

 або  (6.1)

але повинна виконуватись умова > 1,0.

**6.5 Стратегія маркетингу**

Поширення товару буде вироблятися шляхом прямих продажів.

Розраховуємо основну заробітну плату розроблювача —. Розрахунок виконуємо по формулі:

, (6.2)

де  – час розробки ПП;

– середня заробітна плата розробника:

, (6.3)

де  – заробітна плата i-го розробника;

n – кількість розробників.

Таблиця 6.7 – Основна заробітна плата розроблювачів

|  |  |  |
| --- | --- | --- |
| Посада | Заробітна плата, грн | Кількість розроблювачів |
| Програміст | 3 000 | 1 |

Використовуючи дані, приведені в таблиці 6.7, і виходячи з того, що розробка ПП велася 4 місяців, одержимо:

= 3000 ∙ 4 = 12 000

Експлуатаційні витрати:

, (6.4)

де  – вартість машинного часу роботи ЕОМ (= 8 грн);

 – час налагодження програми на ЕОМ:

 , (6.5)

де *Ф* – кількість робочих днів у місяці (22 дня);

 – кількість годин, пророблених на ЕОМ у день (3 год).

= 4 ∙ 22 ∙ 3 = 264.,

 = 264 ∙ 8 = 2 112.

Потреби в матеріальних ресурсах і устаткуванні для виробництва програмного продукту приведені в таблиці 6.8 і таблиці 6.9.

Таблиця 6.8 – Витрати на обладнання

|  |  |  |  |
| --- | --- | --- | --- |
| Обладнання | Призначення | Кількість | Вартість, грн. |
| Ноутбук Acer Aspire ES1-520-392H | Для написання програми і її налагодження, а також для підготовки документів | 1 | 7 000 |
| Усього Зоб | | | 7 000 |

Вартість основних виробничих фондів визначається по формулі:

 , (6.6)

де – витрати на обладнання (дані в таблиці 6.8).

 2 333,3.

Таблиця 6.9 – Витрати на матеріали

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Матеріали | Призначення | Вартість одиниці в грн. | Кількість, шт | Сума,  грн. |
| Папір | Друк вихідних текстів | 60 | 1 пачка  (250 аркушів) | 60 |
| Разом | | | | 60 |

Інші статті витрат на розробку програмного продукту приведені в таблиці 6.10.

Прибуток П обчислюється як 30% від витрат на розробку ПП (дані в таблиці 6.10).

 = 30% ∙ , (6.7)

 = 0,3 ∙ 9 555,3 = 2 866,59.

Таблиця 6.10 – Розрахунок витрат на розробку програмного продукту.

|  |  |  |
| --- | --- | --- |
| № п/п | Найменування статей витрат | Значення, грн. |
| 1 | Вартість матеріалів | 60 |
| 2 | Основна заробітна плата розроблювачів | 3 000 |
| 3 | Додаткова заробітна плата розроблювачів (10% від п2) | 300 |
| 4 | Відрахування на соціальні нестатки | 1 563 |
| 5 | Експлуатаційні витрати | 2 112 |
| 6 | Вартість основних фондів | 2 333, 30 |
| 7 | Накладні витрати (до 70% від п1) | 42 |
| 8 | Комунальний податок (10% від мін. зар. плати) | 145 |
| Разом | | 9 555, 30 |

Максимальна ціна розроблювального ПП буде:

 (6.8)

= 1,2 ∙ (9 555,3 + 1,3 ∙ 2 866, 59) = 15 938, 24.

Отримана ціна є максимальною. Однак ця ціна може бути зменшена і складатися з витрат на тиражування () і адаптацію () даного продукту споживачам.

Витрати на тиражування складаються з вартості диска, машинного часу, необхідного для розробки і налагодження програми, а також оплати праці виконавця.

Мінімальну ціну  визначаємо по формулі:

, (6.9)

де  – витрати на адаптацію (приймаємо 5% від Зр );

 – прибуток з одного продажу, грн:

, (6.10)

де  – витрати на тиражування ПП:

, (6.11)

де  – час копіювання системи, година (приймаємо 0,08 години);

 – вартість диску, грн (приймаємо 10 грн);

 – зарплата виконавця, грн/година:



 8 ∙ 0,08 + 10 + 68,18 = 78, 82

 0,05 ∙ 9 555,30 = 477, 77

0,3 ∙ (78,82 + 477,77) = 166, 98 .

Мінімальна ціна буде:

1,2 ∙ (78,82 + 477,77 + 1,3 ∙ 166,98) = 928, 4.

Виходячи з отриманих результатів  і  установимо продажну ціну без ПДВ. Ціна програмного продукту знаходиться в межах:

928, 4 ≤  ≤ 15 938, 24.

Приймаємо ціну продажу програмного продукту без ПДВ рівною 4 500 грн.

Реклама продукту буде побудована в такий спосіб. З огляду на специфіку розроблювального програмного продукту реклама повинна здійснюватися серед потенційних споживачів товару, якими є викладачі і студенти ВНЗ. Їм пропонують безкоштовно установити демонстраційну версію, вона буде надійною і простою в експлуатації, досить ефективною і буде встановлена в найкоротший термін. Для одержання системи з такими характеристиками пропонується купити програмний комплекс. Показуються можливості цього продукту на прикладі уже встановленої демонстраційної версії системи. Крім цього, потенційним споживачам направляються письмово-комерційні пропозиції від розроблювачів, у яких приводяться її характеристики й узагальнений досвід використання проданих копій. При продажі копії організується її постпродажне обслуговування, що буде полягати в наступному:

1 особа, що купила продукт, стає зареєстрованим користувачем продукту;

2 роз'яснення зареєстрованому користувачеві неясних йому питань по експлуатації продукту;

3 виправлення виявлених помилок у програмному продукті і безкоштовний обмін версії з помилками на виправлену версію для зареєстрованих користувачів;

4 оповіщення зареєстрованих користувачів про нові версії.

Реклама програмного продукту буде виробляється у виді рекламних проспектів, що будуть розсилатися директорам організацій і підприємств різних форм власності.

Ціна одного рекламного проспекту дорівнює 3 грн. Передбачається одночасно з виходом програмного продукту випустити і розіслати 200 рекламних проспектів.

Таким чином, ціна реклами  буде складати 600 грн.

Крім того, можлива реклама в засобах масової інформації (зокрема, у журналах з науково дослідницькою тематикою). Таку рекламу можна давати 4 рази на рік у продовж перших двох років. Така реклама буде коштувати = 80 грн.

Таким чином, за три роки поширення ПП рекламна компанія  буде коштувати:

 (6.12)

= 600 + 80 ∙ 8 = 1 240.

**6.6 Фінансовий план**

Цей розділ бізнес-плану узагальнює матеріали попередніх розділів і представляє їх у вартісному вираженні, містить таблицю доходів і витрат та знаходження точки беззбитковості.

Цьому передує складання карт прогнозів руху готівки для 1, 2 і 3-го років реалізації ПП. Для першого року реалізації карта прогнозів складена на кожен місяць, для другого –  по кварталах, для третього – у цілому по році. У карти прогнозу вноситься графа "Готівка", тобто різниця між доходами і сумою витрат. За допомогою цих карт побудуємо таблицю доходів і витрат, а також графік беззбитковості.

Карти прогнозів готівки для 1, 2 і 3-го років реалізації ПП представлені відповідно в табл. .6.11, 6.12, 6.13.

Таблиця 6.11 – Карта руху готівки за 1-й рік реалізації

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Доходи, витрати | Сума, грн. | | | | | | | | | | | | | | | | | | | | | Усього |
|  | I | II | | III | | IV | | V | | VI | | VII | | | VIII | IX | X | | XI | | XII |  |
| Обсяг продажів | 0 | 2 | | 1 | | 1 | | 2 | | 0 | | 0 | | | 1 | 2 | 2 | | 1 | | 1 | 13 |
| Доходи від реа-лізації | 0 | 9  000 | | 4  500 | | 4  500 | | 9 000 | | 0 | | 0 | | | 4  500 | 9 000 | 9  000 | | 4  500 | | 4  500 | 58  500 |
| Постійні  витрати |  | | | | | | | | | | | | | | | | | | | | | |
| Витрати Реалізації | 40 | 40 | | 40 | | 40 | | 40 | | 40 | | 40 | | | 40 | 40 | 40 | | 40 | | 40 | 480 |
| Накладні витрати | 42 | 42 | | 42 | | 42 | | 42 | | 42 | | 42 | | | 42 | 42 | 42 | | 42 | | 42 | 504 |
| Витрати на рекламу | 45 | 80 | | 45 | | 45 | | 80 | | 45 | | 45 | | | 45 | 80 | 80 | | 45 | | 45 | 680 |
| Усього | 127 | 162 | | 127 | | 127 | | 162 | | 127 | | 127 | | | 127 | 162 | 162 | | 127 | | 127 | 1 664 |
| Зміні  витрати |  | | | | | | | | | | | | | | | | | | | | | |
| Витрати на тиражування | 50 | | 50 | | 50 | | 50 | | 50 | | 50 | | 50 | 50 | | 50 | | 50 | | 50 | 50 | 600 |
| Податки | 0 | | 1800 | | 900 | | 900 | | 1800 | | 0 | | 0 | 900 | | 1800 | | 1800 | | 900 | 900 | 15300 |
| Витрати на адаптацію | 0 | | 400 | | 400 | | 400 | | 400 | | 0 | | 0 | 400 | | 400 | | 400 | | 400 | 400 | 3600 |
| Усього | 50 | | 2  250 | | 1  350 | | 1  350 | | 2  250 | | 50 | | 50 | 1  350 | | 2  250 | | 2  250 | | 1  350 | 1  350 | 15  900 |
| Готівка | -177 | | 6  588 | | 3  023 | | 3  023 | | 6  588 | | -177 | | -177 | 3  023 | | 6  588 | | 6  588 | | 3  023 | 3  023 | 40  936 |

Таблиця 6.12 – Карта руху готівки на 2-й рік реалізації

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Доходи і  витрати | | Сума, грн | | | | Усього |
| I кв. | II кв. | III кв. | IV кв. |  |
| Обсяг продажів | | 2 | 2 | 3 | 1 | 8 |
| Доходи від реалізації | | 9 000 | 9 000 | 13 500 | 4 500 | 36 000 |
| Постійні  витрати | Зарплата і податки | 120 | 120 | 120 | 120 | 480 |
| Накладні витрати | 126 | 126 | 126 | 126 | 504 |
| Витрати на рекламу | 80 | 80 | 100 | 45 | 305 |
| Усього | 326 | 326 | 346 | 291 | 1 289 |
| Змінні  витрати | Витрати на тиражування | 85 | 85 | 85 | 85 | 340 |
| Податки | 1 800 | 1800 | 2 700 | 900 | 7 200 |
| Витрати на адаптацію | 400 | 400 | 400 | 400 | 1 600 |
| Усього | 2 285 | 2 285 | 3 185 | 1 385 | 9 140 |
| Готівка | | 6 389 | 6 389 | 9 969 | 2 824 | 25 571 |

Таблиця 6.13 – Карта руху готівки на 3-й рік реалізації

|  |  |  |
| --- | --- | --- |
| Доходи і витрати | | Сума, грн |
| Обсяг продажів | | 5 |
| Доходи від реалізації | | 22 500 |
| Постійні  витрати | Зарплата і податки | 480 |
| Накладні витрати | 504 |
| Витрати на рекламу | 200 |
|  | Усього | 1 184 |
| Змінні  витрати | Витрати на тиражування | 200 |
| Податки | 4 500 |
| Витрати на адаптацію | 400 |
| Усього | 5 100 |
| Готівка | | 16 216 |

За результатами розрахунків складаємо таблицю доходів і витрат, результати якої відображені в таблиці 6.14.

Таблиця 6.14 – Таблиця доходів і витрат

| Найменування | | | Сума, грн. | | | | Разом |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поч. реаліз. | 1-й рік | 2-й рік | 3-й рік |
| КІлькість продажів | | |  | 13 | 8 | 5 | 26 |
| Дохід від продажів | | | 0 | 58500 | 36000 | 22500 | 117000 |
| Постійні витрати | Витрати на розробку | Зарплата розроблювача | 3000 |  |  |  | 3000 |
| Вартість матеріалів | 60 |  |  |  | 60 |
| Вартість маш-го часу | 8 |  |  |  | 8 |
| Відрахування на соцстрах | 132 |  |  |  | 132 |
| Відрахування у фонд Чорноб. | 330 |  |  |  | 330 |
| Страхування на випал. безроб. | 45 |  |  |  | 45 |
| Накладні витрати | 42 |  |  |  | 42 |
| З/п реалізації та податки | |  | 480 | 480 | 480 | 1440 |
| Накладні витрати при реалізації | |  | 504 | 504 | 504 | 1512 |
| Усього | | 3 617 | 984 | 984 | 984 | 6 569 |
| Змінні витрати | Витрати на тиражування | |  | 600 | 340 | 200 | 1140 |
| Податки | |  | 15300 | 7200 | 4500 | 27000 |
| Витрати на адаптацію | |  | 3600 | 1600 | 400 | 5600 |
| Усього | | 3617 | 19500 | 9140 | 5100 | 37357 |
| Чистий прибуток | | | -3617 | 39000 | 26860 | 17400 | 79643 |

Визначаємо точку беззбитковості – такий обсяг продажів, при якому окупаються усі витрати.

Аналітично її можна визначити по формулі:

 (6.13)

де Зпост – постійні витрати на термін реалізації ПП, грн.

Зр – витрати на розробку ПП, грн.

Цод – ціна одиниці продукції, грн.

Зпер – перемінні витрати на одиницю продукції, грн.

Зпост і Зр визначаються по таблиці доходів і витрат.

 (6.14)

.

Виходячи з отриманих даних визначаємо точку беззбитковості:



З приведених розрахунків видно, що після продажу 6 екземплярів витрати на розробку ПП цілком окупляться.

**6.7 Висновки**

Проведений аналіз дозволяє зробити висновок про доцільність розробки і продажу продукту «Розробка комп’ютерної моделі сигналу, що реєструється радаром некогерентного розсіяння» на ринку.

Розрахована ємкість сегментів ринку, а отже, і загальний передбачуваний обсяг продаж носить умовний характер і може бути розширений за рахунок модернізації і просування товару на нові сегменти ринку. Модернізація програмного продукту «Розробка комп’ютерної моделі сигналу, що реєструється радаром некогерентного розсіяння» може бути проведена без значних витрат часу і матеріальних ресурсів.

7 ЦИВІЛЬНИЙ ЗАХИСТ

Цивільний захист – це функція держави, спрямована на захист населення, територій, навколишнього природного середовища та майна від надзвичайних ситуацій шляхом запобігання таким ситуаціям, ліквідації їх наслідків і надання допомоги постраждалим у мирний час та в особливий період [1].

У данному розділі диплому розглядається тема: «Склад та основні завдання сил цивільного захисту. Завдання і обов’язки суб’єктів господарювання».

**7.1 Склад та основні завдання сил цивільного захисту [1]**

1 До сил цивільного захисту належать:

1. Оперативно-рятувальна служба цивільного захисту;
2. аварійно-рятувальні служби;
3. формування цивільного захисту;
4. спеціалізовані служби цивільного захисту;
5. пожежно-рятувальні підрозділи (частини);
6. добровільні формування цивільного захисту.

2 Основними завданнями сил цивільного захисту є:

1. проведення робіт та вжиття заходів щодо запобігання надзвичайним ситуаціям, захисту населення і територій від них;
2. проведення аварійно-рятувальних та інших невідкладних робіт;
3. гасіння пожеж;
4. ліквідація наслідків надзвичайних ситуацій в умовах екстремальних температур, задимленості, загазованості, загрози вибухів, обвалів, зсувів, затоплень, радіоактивного, хімічного забруднення та біологічного зараження, інших небезпечних проявів;
5. проведення піротехнічних робіт, пов’язаних із знешкодженням вибухонебезпечних предметів, що залишилися на території України після воєн, сучасних боєприпасів та підривних засобів (крім вибухових пристроїв, що використовуються у терористичних цілях), крім територій, які надані для розміщення і постійної діяльності військових частин, військових навчальних закладів, підприємств та організацій Збройних Сил України, інших військових формувань;
6. проведення вибухових робіт для запобігання виникненню надзвичайних ситуацій та ліквідації їх наслідків;
7. проведення робіт щодо життєзабезпечення постраждалих;
8. надання екстреної медичної допомоги постраждалим у районі надзвичайної ситуації і транспортування їх до закладів охорони здоров’я;
9. здійснення перевезень матеріально-технічних засобів, призначених для проведення аварійно-рятувальних та інших невідкладних робіт, ліквідації наслідків надзвичайних ситуацій та надання гуманітарної допомоги постраждалим внаслідок таких ситуацій;
10. надання допомоги іноземним державам щодо проведення аварійно-рятувальних та інших невідкладних робіт, ліквідації наслідків надзвичайних ситуацій;
11. проведення аварійно-рятувального обслуговування суб’єктів господарювання та окремих територій, на яких існує небезпека виникнення надзвичайних ситуацій.

3 Сили цивільного захисту можуть залучатися до проведення відновлювальних робіт.

**7.2 Завдання і обов’язки суб’єктів господарювання [8]**

1. До завдань і обов’язків суб’єктів господарювання у сфері цивільного захисту належить:

1. забезпечення виконання заходів у сфері цивільного захисту на об’єктах суб’єкта господарювання;
2. забезпечення відповідно до законодавства своїх працівників засобами колективного та індивідуального захисту;
3. розміщення інформації про заходи безпеки та відповідну поведінку населення у разі виникнення аварії;
4. організація та здійснення під час виникнення надзвичайних ситуацій евакуаційних заходів щодо працівників та майна суб’єкта господарювання;
5. створення об’єктових формувань цивільного захисту відповідно до цього Кодексу та інших законодавчих актів, необхідної для їх функціонування матеріально-технічної бази і забезпечення готовності таких формувань до дій за призначенням;
6. створення диспетчерських служб відповідно до цього Кодексу та інших законів, необхідних для забезпечення безпеки об’єктів підвищеної небезпеки;
7. проведення оцінки ризиків виникнення надзвичайних ситуацій на об’єктах суб’єкта господарювання, здійснення заходів щодо неперевищення прийнятних рівнів таких ризиків;
8. здійснення навчання працівників з питань цивільного захисту, у тому числі правилам техногенної та пожежної безпеки;
9. декларування безпеки об’єктів підвищеної небезпеки;
10. розроблення планів локалізації та ліквідації наслідків аварій на об’єктах підвищеної небезпеки;
11. проведення об’єктових тренувань і навчань з питань цивільного захисту;
12. забезпечення аварійно-рятувального обслуговування суб’єктів господарювання відповідно до вимог статті 133 цього Кодексу;
13. здійснення за власні кошти заходів цивільного захисту, що зменшують рівень ризику виникнення надзвичайних ситуацій;
14. забезпечення безперешкодного доступу посадових осіб органів державного нагляду, працівників аварійно-рятувальних служб, з якими укладені угоди про аварійно-рятувальне обслуговування суб’єктів господарювання, для проведення обстежень на відповідність протиаварійних заходів планам локалізації і ліквідації наслідків аварій на об’єктах підвищеної небезпеки та потенційно небезпечних об’єктах, сил цивільного захисту – для проведення аварійно-рятувальних та інших невідкладних робіт у разі виникнення надзвичайних ситуацій;
15. забезпечення дотримання вимог законодавства щодо створення, зберігання, утримання, використання та реконструкції захисних споруд цивільного захисту;
16. здійснення обліку захисних споруд цивільного захисту, які перебувають на балансі (утриманні);
17. дотримання протиепідемічного, протиепізоотичного та протиепіфітотичного режиму;
18. створення і використання матеріальних резервів для запобігання та ліквідації наслідків надзвичайних ситуацій;
19. розроблення заходів щодо забезпечення пожежної безпеки, впровадження досягнень науки і техніки, позитивного досвіду із зазначеного питання;
20. розроблення і затвердження інструкцій та видання наказів з питань пожежної безпеки, здійснення постійного контролю за їх виконанням;
21. забезпечення виконання вимог законодавства у сфері техногенної та пожежної безпеки, а також виконання вимог приписів, постанов та розпоряджень центрального органу виконавчої влади, який здійснює державний нагляд у сферах техногенної та пожежної безпеки;
22. утримання у справному стані засобів цивільного та протипожежного захисту, недопущення їх використання не за призначенням;
23. здійснення заходів щодо впровадження автоматичних засобів виявлення та гасіння пожеж і використання для цієї мети виробничої автоматики;
24. своєчасне інформування відповідних органів та підрозділів цивільного захисту про несправність протипожежної техніки, систем протипожежного захисту, водопостачання, а також про закриття доріг і проїздів на відповідній території;
25. виконання інших завдань і заходів у сфері цивільного захисту, передбачених цим Кодексом та іншими законодавчими актами.

2 Організація заходів цивільного захисту суб’єкта господарювання здійснюється підрозділами (посадовими особами) з питань цивільного захисту, які створюються (призначаються) керівниками зазначених суб’єктів господарювання з урахуванням таких вимог:

1. у суб’єктах господарювання, віднесених до відповідних категорій цивільного захисту, з чисельністю працюючих понад 3 тисячі осіб створюються підрозділи з питань цивільного захисту;
2. у суб’єктах господарювання, а також закладах охорони здоров’я із загальною чисельністю працюючих та осіб, які перебувають на лікуванні, від 200 до 3 тисяч осіб та у суб’єктах господарювання, віднесених до другої категорії цивільного захисту, призначаються посадові особи з питань цивільного захисту;
3. у навчальних закладах з денною формою навчання з чисельністю 500 і більше осіб, які навчаються, призначаються посадові особи з питань цивільного захисту;
4. у суб’єктах господарювання з чисельністю працюючих до 200 осіб призначаються особи з питань цивільного захисту за рахунок штатної чисельності суб’єкта господарювання.

3 Громадяни України, іноземці та особи без громадянства, які здійснюють господарську діяльність та зареєстровані відповідно до Закону як підприємці, виконують заходи цивільного захисту особисто.

8 ОХОРОНА ПРАЦІ І НАВКОЛИШНЬОГО СЕРЕДОВИЩА

**8.1 Загальні питання охорони праці та навколишнього середовища**

Охорона праці є системою законодавчих актів, соціально-економічних, організаційних, технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, що забезпечують безпеку, збереження здоров'я і працездатності людини в процесі праці.

Задача охорони праці полягає в тому, щоб звести до мінімуму ймовірність ураження працюючого під дією шкідливого виробничого чинника чи захворювання під дією шкідливого виробничого чинника з одночасним забезпеченням комфортних умов при максимальній продуктивності праці.

Дотримання умов та правил охорони праці і виробничої санітарії забезпечують безпеку праці. Покращення умов праці, впливає на результати роботи, приводить до зниження виробничого травматизму, професійних захворювань, зберігає здоров’я робітників і водночас приводить до зменшення затрат на оплату компенсацій і пільг на лікування а також за роботу в несприятливих умовах праці.

Закон України «Про охорону навколишнього природного середовища » був прийнятий 25 червня 1991 року і визначає правові, економічні, соціальні основи охорони навколишнього середовища.

Завданням законодавства про охорону навколишнього природного середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності на навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною.

Інструкції, що діють у ВНЗ належать до нормативних актів про охорону праці, чинних у межах навчального закладу. Такі інструкції розробляються на основі чинних державних міжгалузевих і галузевих нормативних актів про охорону праці, примірних інструкцій та технологічної документації ВНЗ з урахуванням конкретних умов виробництва та вимог безпеки, викладених в експлуатаційній та ремонтній документації підприємств — виготовлювачів обладнання, що використовується у навчальному закладі. Вони затверджуються ректором і є обов’язковими для дотримання працівниками відповідних професій або при виконанні відповідних робіт у ВНЗ.

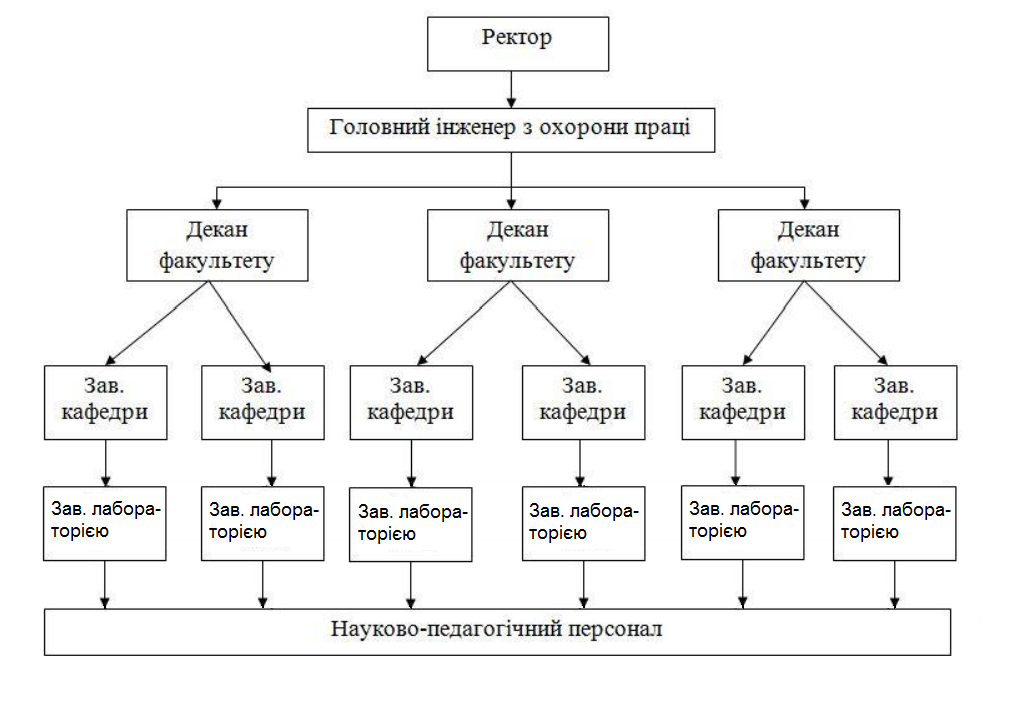


Рисунок 8.1 — Організаційна структура розробки інструкції з охорони праці, яка існує в навчальному закладі

Організаційна структура — це сукупність способів, за допомогою яких процес праці спочатку поділяється на окремі робочі завдання, а потім досягається координація дій з вирішення завдань. По суті справи, організаційна структура визначає розподіл відповідальності і повноважень всередині будь-якої організації, установи, навчального закладу. Організаційна структура розробки інструкцій з охорони праці, що приведена на рисунку 1 , класифікується як дивізіональна схема. Вона дозволяє розробляти одну інструкцію протягом 6-7 днів та має необґрунтовано велику кількість задіяних кадрів. Це є значним недоліком механізму розробки інструкцій. Крім того, така структура має певні недоліки:

1 існуюча структура управління призвела до зростання ієрархічності, тобто вертикалі управління;

2 можливість виникнення конфліктів серед підрозділів; невисока координація діяльності відділень (структурних підрозділів), служби роз'єднані, горизонтальні зв'язки ослаблені;

3 збільшення часу на складання інструкцій внаслідок дублювання одних і тих самих функцій у підрозділах і відповідного збільшення чисельності співробітників, задіяних на виконанні завдання;

Закон України «Про охорону праці» визначає основні положення по охороні праці і регулює взаємини між працівниками і адміністрацією. В Україні законодавство по охороні праці складається із Закона України «Про охорону праці», Кодексу законів про працю і інших нормативних актів. У таблиці 8.1 приведені закони і нормативні документи, які використані в розділі.

Таблиця 8.1 — Закони і нормативні документи, які використані в розділі

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п/п | Позначення закону чи нормативного документа | Назва закону чи нормативного документу | Дата затвердження та введення у дію | Орган влади, що затвердив документ |
| 1 | 2 | 3 | 4 | 5 |
| 1  2 | Закон України  Закон України | Про охорону праці    Про охорону навколишнього середовища | Нова редакція 25.11. 2002 р.  25.06. 1991 р. | Верховна рада України  Рада Міністрів СРСР |

Продовження таблиці 8.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п/п | Позначення закону чи нормативного документа | Назва закону чи нормативного документу | Дата затвердження та введення у дію | Орган влади, що затвердив документ |
| 1 | 2 | 3 | 4 | 5 |
| 3  4  5  6  7  8  9  10  11  12  13 | ГОСТ 12.0.003 — 74  ГОСТ 12.1.005 — 88  ГОСТ 12.1.003 – 83\*  НПАОП 0.03-3.14-85  ГОСТ 12.1.012-90  НПАОП 0.00-1.28-10  ГОСТ 12.1.045 – 84  ГОСТ 12.2.007.0-75  ГОСТ 14255-69  ДБН В.1.1-7-2002  ГОСТ 12.1.004-91 | Опасные и вредные производственные факторы. Классификация  Общие санитарно-гигиенические требования к воздуху рабочей зоны  Шум. Общие требования безопасности  Санітарні норми допустимих рівнів шуму на робочих місцях № 3223–85  Вибрационная безопасность. Общие требования  Правила охорони праці під час експлуатації електронно-обчислювальних машин  Электростатические поля. Допустимые уровни на рабочих местах и требования к проведению контроля  Изделия электрические. Общие требования  Аппараты электрические напряжением до 1000В. Оболочки. Степени защиты.  Захист від пожежі. Пожежна безпека об’єктів будівництва  Пожарная безопасность. Общие требования | 01.01.1976 р.  01.01.89 р.  01.07.89 р.  1985 р.  01.07.91 р.  23.06.2010 р.  01.07.85 р.  01.01.76 р  1970 р.  2002 р.  01.07.91 р. | Рада Міністрів СРСР  Рада Міністрів СРСР  Рада Міністрів СРСР  Рада Міністрів СРСР  Рада Міністрів СРСР  Кабінет Міністрів  Рада Міністрів СРСР  Рада Міністрів СРСР  Рада Міністрів СРСР  Кабінет Міністрів  Рада Міністрів СРСР |

Закінчення таблиці 8.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п/п | Позначення закону чи нормативного документа | Назва закону чи нормативного документу | Дата затвердження та введення у дію | Орган влади, що затвердив документ |
| 1 | 2 | 3 | 4 | 5 |
| 14  15  16  17  18  19  20  21  22 | ISO 14001-97  ГН 3.3.5-8-6.6.1 2002 р.  СНиП 2.04.05-91  ДБН В.2.5-28-2006  НРБУ – 97  ДНАОП 0.03 – 3.06 – 80  ПУЕ – 2011  НАПБ Б.03.002-2007  ДСТУ Б В.2.5-38:2008 | Система керування навколишнім середовищем  Гігієнічна класифікація праці. Гігієнічні нормативи  Отопление, вентиляция и кондиционирование воздуха  Природне і штучне освітленн  Норми радіаційної безпеки України  Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень  Правила улаштування електроустановок  Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною, та пожежною небезпекою  Улаштування блискавкозахисту будівель і споруд (ІЕС 62305:2006,NEQ) | 1998 р.  2001 р.  1992  2008  1997  Діє з 01.01.81  2011  2007  2008 | Кабінет Міністрів  Наказ Міністерства охорони  здоров'я України  Кабінет Міністрів  Мінбуд України  Кабінет Міністрів  Рада Міністрів СРСР  Кабінет Міністрів  Наказ МНС від 03.12.2007р  Нацстандарт України |

**8.2 Характеристика умов праці**

Найменування факторів та їх значення для оцінки умов праці за ступенем шкідливості та небезпечності приведені в таблиці 8.2.

Таблиця 8.2 — Найменування факторів та їх значення

| Фактор виробничого середовища  (одиниці вимірювання) | Величина |
| --- | --- |
| Хімічний (мг/м.куб.) | Азоту оксиди  3 |
| Біологічний  Мікроорганізми-продуценти (мг/м.куб.) | ≤ГДК |
| Фізичні: |  |
| -шум  еквівалентний рівень звуку (дБАекв) | 5 |
| -вібрація  еквівалентний    коректований рівень віброшвидкості (дБекв) | 2 |
| -інфразвук  еквівалентний загальний рівень звукового тиску (дБ Лінекв) | ≤ГДР |
| -ультразвук  повітряний: рівні звукового тиску в октавних (1/3 октавних) смугах частот(дБ) | ≤ГДР |
| -неіонізуючі електромагнітні випромінювання (Гц) | ≤ГДР |
| -мікроклімат  Температура повітря, більше (оС)  Швидкість руху повітря (м/с)  Відносна вологість повітря (%)  Теплове випромінювання (Вт/м2) | 1а  22-24  0,1  50  - |
| -освітленість  Коефіцієнт природного освітлення (КПО, %)  Освітленість робочої поверхні (Елк) для розрядів зорових робіт I-IV (лк) | 1,5  Ен |

Закінчення таблиці 8.2

| Фактор виробничого середовища  (одиниці вимірювання) | Величина |
| --- | --- |
| Важкість праці  Сумарна маса вантажів, що переміщуються протягом кожної години зміни:  з робочої поверхні:  - для чоловіків  - для жінок | 2  1 |
| Напруженість праці  Сенсорні навантаження  Тривалість зосередженого спостереження (в % від часу зміни)  Кількість виробничих об'єктів одночасного спостереження | 75  1 |

Оцінка умов праці за ступенем шкідливості та небезпечності відповідно до варіанту проведена згідно з гігієнічними нормативами ГН 3.3.5-8-6.6.1 2002 та представлена у вигляді таблиці 8.3 [12].

Таблиця 8.3 — Оцінка умов праці за ступенем шкідливості та небезпечності

| Фактор виробничого середовища  (одиниці вимірювання) | КЛАС УМОВ ПРАЦІ | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Оптимальний 1 | Допустимий 2 | Шкідливий 3 | | | | Небезпечний |
| 1 ступінь | 2 ступінь | 3 ступінь | 4 ступінь |
| Хімічний |  |  |  | Азоту оксиди 3 |  |  |  |
| Біологічний  Мікроорганізми-продуценти | ≤ГДК |  |  |  |  |  |  |
| Фізичні: |  |  |  |  |  |  |  |
| -шум  еквівалентний рівень звуку (дБАекв) |  | 5 |  |  |  |  |  |
| -вібрація  еквівалентний    коректований рівень віброшвидкості (дБекв) |  | 2 |  |  |  |  |  |
| -інфразвук  еквівалентний загальний рівень звукового тиску (дБ Лінекв) |  | ≤ГДР |  |  |  |  |  |
| -ультразвук  повітряний: рівні звукового тиску в октавних (1/3 октавних) смугах частот (дБ) | 2 | ≤ГДР |  |  |  |  |  |
| -неіонізуючі електромагнітні випромінювання (Гц) |  | ≤ГДК |  |  |  |  |  |
| -мікроклімат  Температура повітря, більше (оС)  Швидкість руху повітря (м/с) | 22–24  0,1 |  |  |  |  |  |  |

Закінчення таблиці 8.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Фактор виробничого середовища  (одиниці вимірювання) | КЛАС УМОВ ПРАЦІ | | | | | | |
| Оптимальний 1 | Допустимий 2 | Шкідливий 3 | | | | Небезпечний |
| 1 ступінь | 2 ступінь | 3 ступінь | 4 ступінь |
| Відносна вологість повітря (%)  Теплове випромінювання (Вт/м2) | 50  ≤ГДК | 60 |  |  |  |  |  |
| -освітленість  Коефіцієнт природного освітлення (КПО, %)  Освітленість робочої поверхні (Елк) для розрядів зорових робіт I-IV(лк) | 1,5 | Ен |  |  |  |  |  |
| Важкість праці  Сумарна маса вантажів, що переміщуються протягом кожної години зміни:  З робочої поверхні:  - для чоловіків  - для жінок | 2  1 |  |  |  |  |  |  |
| Напруженість праці  Сенсорні навантаження  Тривалість зосередженого спостереження (в % від часу зміни)  Кількість виробничих об'єктів одночасного спостереження | 1 |  | 75 |  |  |  |  |
| Загальна оцінка умов праці |  |  | + |  |  |  |  |

Умови праці за ступенем шкідливості та небезпечності у розглянутому варіанті відповідають шкідливому класу першого ступеню.

Робота на ПЕОМ не вимагає фізичної напруги, але вимагає максималь-ної концентрації уваги, і, отже, тримає її в постійній напрузі. Тому ця робота, відповідно до ГОСТ 12.1.005-88 та ГН 3.3.5-8-6.6.1 2002 р. віднесена до лег-кої категорії 1а - легка фізична, виходячи з таких чинників: категорії важкості робіт, періоду року. Енерговитрати складають до 139 Вт.

Згідно з НПАОП 0.00-1.28-10 умови праці за ПЕОМ повинні бути оп-тимальними. Оптимальні норми температури параметрів метеорологічних умов відповідно до категорії робіт і залежно від періоду року приведені в таблиці 8.4.

Таблиця 8.4. — Оптимальні значення параметрів метеорологічних умов, для постійних робочих місць.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категорія робіт по важкості | Період року | Температура, 0С | Відносна вологість % | Швидкість руху повітря в приміщенні, м\с |
| 1а | Холодний | 22 – 24 | 60 – 40 | 0,1 |
| 1а | Теплий | 23 – 25 | 60 – 40 | 0,1 |

Приміщення обладнане системами централізованого опалення (загальне парове), кондиціювання повітря та штучною припливно-витяжною вентиляцією відповідно до СНиП 2.04.05-92 [16] . Нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин мають відповідати вимогам ГОСТ 12.1.005–88 та ГН 3.3.5-8-6.6.1 2002.

Стан освітлення виробничих, службових і допоміжних приміщень регламентується ДБН В.2.5-28-2006 [2]. Мінімальний розмір об'єкта розрізнення на екрані дисплея – 0,3−0,5 мм. Це відповідає ІІІ розряду зорової роботи, тобто це робота високої точності. Контраст об'єкта з фоном − ∙середній, фон − середній, отже, підрозряд − "г".

Для визначення нормованого значення КПО скористаємося формулою:

*еN = еН ∙ mN* , (8.1)

де *еН* — нормована величина КПО для будинків 1,5%;

*mN* — коефіцієнт світлового клімату, *m* = 0,9.

*N* — номер групи забезпеченості природного світла

еN = 1,5 0,9;

еN = 1,35%

У таблиці 8.5. наведені нормативні показники освітлення робочої зони (природного та штучного).

Таблиця 8.5 — Нормативні показники освітлення у виробничому приміщенні.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Найменування робіт | Плоша підлоги (кв. м.) | Розряд зорової роботи | Освітлення | | |
| Суміщене | | Штучне |
| вид  освітлення | КПО (%) | нормована освітленість Е (лк) |
| Робота оператора за ПЕОМ | 35 | IIIг | бокове | 1,5 | 300-500 |

Штучне освітлення створюється електричними джерелами світла. Мінімальна освітленість при загальному штучному освітленні складає Emin = 300 лк. Штучне освітлення реалізується шляхом встановлення визначеної кількості світильників і потужності ламп.

Для розрахунку загального рівномірного освітлення при горизонтальній робочій поверхні основним являється метод коефіцієнта використання світлового потоку, враховуючи світловий потік, відображений від стелі і стін. Світловий потік розраховується за формулою:

(8.1)

де Emin — нормована мінімальна освітленість, лк;

k — коефіцієнт запасу;

S — освітлювальна площа, м2;

Z — коефіцієнт мінімальної освітленості;

N — число світильників;

n — число ламп в світильнику;

η — коефіцієнт використання світлового потоку в долях одиниці.

З урахуванням розряду зорових робіт (IV) Emin = 300 лк.

Освітлювальна площа S = 35 м2.

Коефіцієнт запасу k враховує запиленість приміщення, зниження світлового потоку ламп в процесі експлуатації. Значення k = 1,2.

Коефіцієнт мінімальної освітленості Z характеризує нерівномірність освітлення. Для люмінесцентних ламп Z = 1,1.

Для визначення коефіцієнту використання світлового потоку η знаходять індекс приміщення і передбачувані коефіцієнти відображення поверхонь приміщення: стелі ρн, стін ρс, підлоги ρр (для світильних адміністративно-конторських приміщень: ρн = 70%, ρс = 50%, ρр = 30%), вибирають тип світильника.

Для освітлення приміщення використовуються люмінесцентні лампи як с гігієнічної так і з економічної точок зору. Для зниження пульсації люмінесцентних ламп до нормальних величин варто використовувати двухлампові світильники з лампами типу ЛБ, ЛТВ, ЛХБ, ЛТПЦ, ЛН.

Індекс приміщення визначається за формулою:

(8.2)

де A, B, h — довжина, ширина і розрахункова висота (висота підвісу світильника над робочою поверхнею) приміщення, м;

(8.3)

де Н — геометрична висота приміщення;

hсв - підвіс світильника. hсв = 0,2 ~ 0,8 м;

hр — висота робочої поверхні. hр = 0,8 ~ 1,0 м.

А = 5 м, В = 4 м; h = 4 - 0,2 - 0,8 = 3 м;

1,02.

На основі отриманого значення і коефіцієнту використання світлового потоку η = 56,2%.

Світловий потік, згідно з формулою (8.1), рівний:



В якості джерел світла використовуються люмінесцентні лампи потужністю 40 Вт або енергозберігаючі потужністю 36 Вт типу ЛБ, ЛХБ, ЛДЦ як найбільш ефективні з точки зору спектрального складу, кольорова температура випромінювання котрих знаходиться в діапазоні 3500 - 4200 К.

На основі розрахованого значення світлового потоку підбирається стандартна лампа, в даному випадку це люмінесцентна лампа TLD 36W/29 зі значенням світлового потоку Фл ~ 1200 лм.

Після цього розраховується необхідна кількість цих ламп для розглянутого приміщення:



В приміщенні потрібно встановити 9 світильників.

В робочому приміщенні штучне освітлення виконується системою загального рівномірного освітлення. Загальне освітлення виконано у вигляді ліній світильників. Лінії світильників розташовані зліва від робочих місць. Кількість світильників в лінії дорівнює 3. В кожному світильнику дві лампи типу TLD 36W/29. Коефіцієнт запасу світильників дорівнює 1,2, коефіцієнт пульсації ламп - 5 %, для чого розташовані поряд світильники підключені до різних фаз трьохфазної мережі. Таким чином, загальна кількість світильників дорівнює 9, що відповідає розрахунковому значенню кількості світильників даного типу. Можна зробити висновок, що система штучного освітлення даного приміщення відповідає вимогам.

Шум є одним з найбільш поширених у виробництві шкідливих чинників. Відповідно до ГОСТ 12.1.003-83\* і НПАОП 0.03-3.14-85 в приміщеннях програмістів обчислювальних машин рівні звуку і еквівалентні рівні звуку не повинні перевищувати 50 дБА. Згідно з ГОСТ 12.1.012-90 рівень вібрації для категорії 3, тип В, в умовах “комфорту” не повинна перевищувати 75 дБ. Для зменшення рівня звуку і вібрації застосовуються демпфуючі матеріали (гумова прокладка під принтер).

Основним джерелом електромагнітного випромінювання, зокрема рентгенівського, в приміщенні є ЕПТ моніторів. Згідно з НРБУ-97 [18] потужність експозиційної дози рентгенівського випромінювання трубки в будь-якій крапці перед екраном на відстані 5 см від його поверхні не повинна перевищувати 100 мкР/г. Захист користувачів ЕОМ від ЕМB і рентгенівського випромінювання забезпечується за допомогою екранів із спеціального затемненого скла.

Відповідно до ГОСТ 12.1.045-84 допустимий рівень напруженості електростатичних полів має бути не більше 20 кВ/м. У приміщеннях для запобігання утворенню статичної електрики і захисту від нього мають бути нейтралізатори і зволожувачі повітря, підлога повинна мати антистатичне покриття, а також необхідно робити заземлення екрану дисплея.

ЕМB і статична електрика приводять до іонізації повітря, в результаті якої відбувається утворення позитивних іонів, що вважаються несприятливими для здоров'я людини ( іони потрапляють разом з повітрям в дихальні шляхи, викликаючи ускладнення). Відповідно до ДНАОП 0.03 – 3.06 – 80 [17] норма змісту легких аеронів обох знаків повинна складати від 1500 до 5000 в 1 см3 повітря. Заходами щодо зниження кількості іонів в повітрі є зволоження повітря і провітрювання приміщення.

Згідно існуючим рекомендаціям час безперервної роботи з екраном не повинен перевищувати 4 години, тривалість перерви для відпочинку повинна складати від 5 до 15 хвилин. Сумарний час роботи — до 50% тривалості зміни. Перерви мають бути 10-15 хвилин кожну годину роботи.

Тривалі перерви ведуть до порушення робочої установки, розладу динамічного стереотипу. Загальна перерва через 4 години. Додаткова перерва через 3 години і за 2 години до закінчення роботи.

Основну небезпеку, для працюючих в офісі, представляє підвищена напруга електричної мережі і електроустаткування, зокрема — ПЕОМ.

ПЕОМ є однофазним споживачем електроенергії, що живеться від змінного струму напругою 220В і частотою 50Гц, від мережі із заземленою нейтраллю. За способом захисту людини від поразки електричним струмом ЕОМ повинна відповідати першому класу захисту згідно з ГОСТ 12.2.007.0-75. Захист від випадкового дотику до струмоведучих частин забезпечують конструктивні, схемно-конструктивні і експлуатаційні заходи захисту.

По ступеню небезпеки поразки електричним струмом приміщення відноситься до приміщень без підвищеної небезпеки, відповідно до ПУЕ – 2011 [20].

ПЕОМ відносять до електроустановок закритого типу виконання (токопровідними частини знаходяться в кожухах) з діючою напругою до 1000 В. Згідно з ГОСТ 14255-69 і ПУЕ-2011[20] ступінь захисту персоналу від зіткнення з токопровідними частинами усередині захисного корпусу і від попадання води у всередину корпусу відповідає IP-44 .

Схемно-конструктивні заходи електробезпеки забезпечують безпеку дотику людини до металевих не токопровідних частин електричних апаратів при випадковому пробої з ізоляції і виникнення електричного потенціалу на них. Як схемно-конструктивний захід безпеки передбачається занулення. Експлуатаційними заходами електробезпеки є дотримання правил безпеки при роботі з високою напругою і наступних заходів безпеки:

1 не підключати і не відключати роз’єми кабелів при включеній напрузі мережі;

2 технічне устаткування і ремонтні роботи проводити тільки при вимкненому живленні мережі.

Працівник, що поступає на роботу, обов'язково проходить ввідний і первинний інструктаж з охорони праці в цілях профілактики нещасних випадків, а також інструктаж по дотриманню безпеки при роботі з ПЕОМ на робочому місці.

У приміщенні розташовано 5 робочих місця з комп'ютерами. Відповідно до норм НПАОП 0.00-1.28-10 площа, що відводиться для робочого місця з комп'ютером повинна бути не менше 6 м2, об'єм не менше 20 м3. Розміри даного приміщень складають: довжина - 7 м, ширина - 5 м, висота - 3,7 м, тобто загальна фактична площа складає 35 м2. Необхідна площа на 5 робочих місць із установленими ПК складає 30 м2, що не перевищує фактичну. Обсяг кабінету на одного працюючого складає 25,9 м3, отже відповідає нормі - не менше 20 м3.

Забезпечення організації робочого місця оператора за дисплеєм передбачає організацію робочого місця відповідно до антропометричних характеристик; виконання ергономічних вимог до розміщення технічних засобів на робочому місці; до світло- та кольоро-технічних характеристик дисплею, до літерно-цифрової інформації, клавіатури; необхідні санітарно-гігієнічні умови праці згідно з НПАОП 0.00-1.28-10.

Існують певні вимоги до розмірів робочого місця:

1 висота робочої поверхні 655 мм;

2 висота сидіння 420 мм (бажано такого, що регулюється);

3 відстань від сидіння до нижнього кінця робочої поверхні 150мм;

4 розміри простору для ніг 650x500x600.

Ергономічні вимоги до розміщення технічних засобів на робітничому місці такі:

1. природний нахил корпуса вперед на 5-100, кут згинання між стегном і гомілкою 95-1350;
2. Кут спостереження екрану, а також інших засобів відображення інформації в горизонтальній площині не перевищує 600;
3. Основний дисплей знаходиться внизу, а інформаційні поля екранів - в куту спостереження в вертикальній площині +100 вверх і 300 вниз від горизонтальної лінії зору.
4. ступні на підлозі;
5. стегно – горизонтально;
6. оператор максимальних розмірів не повинен упиратися ступнею в стійку стола або підставки;
7. оператор має можливість спиратися ліктем на робочу поверхню;
8. можливість роботи з документами;

Можливими причинами пожеж в приміщенні є несправність електропроводки і електроустаткування, коротке замикання в мережі, зберігання горючих матеріалів (папери), блискавка і так далі. Як профілактичні міри по попередженню причин пожеж використовується постійний контроль за станом електричної проводки і сполучних проводів, зберігання паперу в сейфах і шафах, що не згорають.

По категорії вибухо- та пожежонебезпеки згідно з НАПБ Б.03.002-2007 [21] дане приміщення відноситься до категорії В - пожежонебезпечне із-за твердих матеріалів, що згорають (робочі столи, папір, ізоляція і ін.). Виходячи з категорії пожежонебезпеки і поверховості будівлі, ступінь вогнестійкості будівлі II згідно з ДБН В 1.1 - 2002.

Згідно з ГОСТ 12.1.004-91 пожежна безпека забезпечується системами запобігання пожежі, пожежного захисту і організаційно-технічними заходами.

У системі запобігання пожежі передбачені наступні заходи:

1 Контроль і профілактика ізоляції.

2 Наявність плавких запобіжників в устаткуванні.

3 Вибір ступеня захисту оболонок комп'ютера відповідно до класу пожежонебезпечної зони приміщення П-II [19] - не нижче IP-44 для електроустановок і IP-23 для світильників.

4 Блискавкозахисту будівлі. Для даного класу встановлюється III категорія блискавкозахисту [22].

Система пожежного захисту передбачені наступні заходи:

1 Система автоматичної пожежної сигналізації оснащена димовими сигналізаторами.

2 Приміщення оснащене вуглекислотними вогнегасниками - ВВК-1,4.

3 Для успішної евакуації персоналу двері приміщення не повинні перевищувати наступних розмірів:

1 ширина не менше 1,5 м;

2 висота не менше 2,0 м.

3 ширина коридору 1,8 м.

Перелік первинних засобів пожежогасіння приведений у таблиці 8.6.

Таблиця 8.6 — Перелік первинних засобів пожежогасіння

|  |  |  |  |
| --- | --- | --- | --- |
| Площа приміщення, м2 | Первинні засоби пожежогасінні (тип, найменування) | Кількість,  шт. | Вогнегасний ефект |
| 35 | вуглекислотні вогнегасники ручні ВВК-1,4  Біля приміщення (5 м) розташовано пожежний щит та пожежний кран. | 2 | Розбавлення повітря вуглекислим газом і зниження в нім змісту кисню до концентрації, при якій припиняється горіння. Вогнегасний ефект вказаним газом обумовлений втратами теплоти і зниженням теплового ефекту реакції, припиненням доступу кисню до елементів, що горять. |

Організаційними заходами пожежної профілактики є навчання виробничого персоналу протипожежним правилам, видання необхідних інструкцій і плакатів, засобів наочної агітації. Обов'язковою є наявність плану евакуації.

**8.3 Охорона навколишнього середовища**

Закон України про ” Охорону навколишнього середовища” був прийнятий 25 червня 1991 року (редакції Закону 1993,1996 р.р.) [17] та визначає правові, економічні, соціальні основи охорони навколишнього середовища. Завдання Закону полягає в регулюванні стосунків в області охорони природи, використанні і відтворенні природних ресурсів, забезпеченні і ліквідації наслідків негативної дії на навколишнє середовище господарської і іншої діяльності людини, збереження природних ресурсів, генетичного фонду нації, ландшафтів і інших природних об'єктів.

Робота на ЕОМ типу IBM PC/AT не впливає негативно на навколишнє середовище. Вся апаратура, тара та документація, що використовується на ІОЦ допускає нетоксичну переробку після використання. Необхідно виконувати вимоги стандарту ISO – 14000, котрий визначає вимоги до організації виробничого процесу з мінімальною шкодою для навколишнього середовища.

ВИСНОВКИ

Метою роботи була розробка комп’ютерної моделі некогерентно розсіяного сигналу для синтезу сигналів із заданими параметрами та розв’язання оберненої задачі радіофізики.Для рішення поставленої задачі було розроблене програмне забезпечення. В процесі роботи було пройдено низку наступних етапів.

1. Проведено аналітичний огляд, в якому було оглянуто основні відомості про метод некогерентного розсіяння сигналу, методи корекцій висотних профілів значень автокореляційних функцій некогерентно розсіяних сигналів. Визначено методології та технології розробки програмного забезпечення. Обрано середовище розробки та мова програмування.
2. Розроблено функціональні вимоги, вимоги з надійності та вимоги до інтерфейсу програми.
3. Розроблено програмне забезпечення у вигляді набору модулів, що реалізує комп’ютерну модель некогерентно розсіяного сигналу, що реєструється радаром некогерентного розсіяння. В відповідності з вимогами розроблено інтерфейс програмного забезпечення.
4. Проведено тестування розробленої комп’ютерної моделі некогерентно розсіяного сигналу, яке показало правильність прийнятих рішень та повноту виконання поставлених завдань та вимог.
5. Програмне забезпечення впроваджене в Інститут іоносфери для оцінки похибки температур іонів та електронів . Також програмне забезпечення може бути використане в якості лабораторних робіт для студентів НТУ «ХПІ» з дисципліни «Цифрова обробка даних».
6. У розділі цивільного захисту була розглянута тема про укриття населення у захисних спорудах цивільного захисту.
7. Проведений економічний аналіз показує доцільність розробки даного продукту. Ціна розробки програмного забезпечення для фільтрації некогерентно розсіяного сигналу на основі нечіткої логіки становить 4 500,00 грн.
8. У розділі охорони праці розглянуті і сформульовані вимоги до забезпечення робочого місця оптимальними умовами праці роботи за комп'ютером

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

* 1. Рогожкин Е.В., Пуляев В.А. Использование манипуляции при некогерентном рассеянии. Сообщение 1. Обработка ионосферных сигналов при амплитудной манипуляции несущей. Вестник Харьк. политехн. ин-та. 1986, 248: Исслед. ионосферы методом некогерентного рассеяния. Вып.5, с.27-30.
  2. Обчислювальні процедури при аналізі некогерентного розсіяння в іоносферній плазмі : монографія / В. О. Пуляєв, Є.В, Рогожкін, О.В. Богомаз. –Х : Вид-во «Підручник НТУ «ХПІ»», 2014. – 264 с.
  3. Богомаз О.В. Моделювання розсіяного сигналу і висотного розподілу його параметрів// Вісник Національного технічного університету «Харківський політехнічний інститут». Збірник наукових праць. Тематичний випуск: Радіофізика та іоносфера.- Харків : НТУ «ХПІ». 2010. - №48 с.72-76.
  4. Архангельский А. Я., Програмирование в С++ Builder. 7-е изд. – М.: «Бином-Пресс», 2010 г. с 527-553 .
  5. Богомаз О.В., Котов Д.В., Сюсюк М.Н. Корекція висотних профілів значень автокореляційних функцій некогерентно розсіяних сигналів// Вісник Національного технічного університету «Харківський політехнічний інститут». Збірник наукових праць. Тематичний випуск: Радіофізика та іоносфера.- Харків : НТУ «ХПІ». 2012. - №57 с.35-42.
  6. Моделі життєвого циклу в сучасних методологіях розробки ПЗ – Режим доступу: <http://kpi.km.ua/teachers/radelchukgi.html?id=130>.
  7. **Екскурс у метод найменших квадратів -** <http://posibnyky.vntu.edu.ua/mk/3-4.html>
  8. Кодекс цивільного захисту України – ВРУ № 5403 – VІ, від 2.10.2012 р

ДОДАТОК А

Лістинг основного файлу проекту

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include <math.h>

#include <vector>

#include <TeeStore.hpp>

#include "Unit1.h"

#include "Unit2.h"

#include "Unit3.h"

#include "direct.cpp"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

extern int NumberOfThreads;

extern int cycle;

extern TLineSeries\* Lll[19];

extern TChart\* Graph[19];

extern AnsiString fname;

int NumberOfCycles, NumberOfIterations;

long int HeightsStep1, HeightsStep2;

AnsiString capt="Генерация НР сигнала для высоты № ";

double Acf\_th[LAG];

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

long int fsize(ifstream& file)

{

long int size;

file.seekg(0,ios::end);

size=file.tellg();

return size;

}

void TForm1::init()

{

ProgressBar1->Min=0;

ProgressBar1->Step=1;

long int size;

char\* fname="1\\1corr.dat";

HeightsStep1=0;

ifstream fcorr;

fcorr.open(fname,ios::binary);

if (fcorr)

{

size=fsize(fcorr);

HeightsStep1=size/152;

fcorr.close();

}

Edit1->Text=HeightsStep1;

}

void TForm1::readBinary(const char \*filename)

{

ifstream fIn(filename, ios::binary | ios::in);

for (int c = 0; c < C; c++)

fIn.read((char\*) PHASE[c], R\*sizeof(double));

fIn.close();

}

void TForm1::readBinaryS(const char \*filename)

{

ifstream fIn(filename, ios::binary | ios::in);

for (int c = 0; c < C; c++)

fIn.read((char\*) m, C\*sizeof(double));

fIn.close();

}

void TForm1::ThreadStart()

{

SignalThread\* Thr[19];

for (int i=0; i<LAG; i++)

{

Thr[i]=new SignalThread(false,i);

}

}

double TForm1::phase()

{

double res;

unsigned int val;

\_asm

{

db 0x0f

db 0x31

mov val, eax

}

srand(val/41);

res=fmod(rand(),2\*M\_PI);

return res;

}

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

Series2->Clear();

Series3->Clear();

Series4->Clear();

Series5->Clear();

Series6->Clear();

Series7->Clear();

Series8->Clear();

Series9->Clear();

Series10->Clear();

Series11->Clear();

Series12->Clear();

Series13->Clear();

Series14->Clear();

Series15->Clear();

Series16->Clear();

Series17->Clear();

Series18->Clear();

Series19->Clear();

Series20->Clear();

if (PageControl1->ActivePageIndex)

Chart1->RemoveAllSeries();

Chart21->RemoveAllSeries();

Chart22->RemoveAllSeries();

Chart23->RemoveAllSeries();

Chart24->RemoveAllSeries();

Chart25->RemoveAllSeries();

Refresh();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

NumberOfCycles=StrToInt(Form1->LabeledEdit6->Text);

NumberOfIterations=StrToInt(Form1->Edit2->Text);

ProgressBar1->Max=NumberOfCycles;

PageControl1Change(PageControl1);

double Ti=StrToFloat(Form1->LabeledEdit1->Text);

double Te=StrToFloat(Form1->LabeledEdit2->Text);

double g1=StrToFloat(Form1->LabeledEdit3->Text);

double g2=StrToFloat(Form1->LabeledEdit4->Text);

m = new double[C];

for (int i=0; i<C; i++)

{

m[i]=sqrt(spectrum\_3\_full(1, 4, 16, g1, g2, Ti, Te, 1, 0, i));

}

acf\_3\_full(1, 4, 16, g1, g2, Ti, Te, 1, 0, Acf\_th, LAG, tauC);

switch(PageControl1->ActivePageIndex)

{

case 0:

{

PHASE = new double\* [C];

for (int c = 0; c < C; c++)

{

PHASE[c] = new double[R];

}

AMPL= new double\* [C];

for (int c = 0; c < C; c++)

{

AMPL[c] = new double[R];

}

SIGNAL = new double\* [R];

for (int r = 0; r < R; r++)

SIGNAL[r] = new double[LAG];

CORR = new double[LAG];

Chart21->BottomAxis->Minimum=0;

Chart21->BottomAxis->Maximum=C-1;

ProgressBar1->Position=cycle+1;

Label3->Caption=capt+IntToStr(cycle+1);

Form1->Refresh();

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

PHASE[i][j]=phase();

}

}

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

AMPL[i][j]=m[i]\*sqrt(-2.0\*log(1E-10+phase()/(2\*M\_PI)));

}

}

TLineSeries \*LineSeries = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

for (int i=0; i<C; i++)

{

LineSeries->AddXY(i,AMPL[i][0],"",clRed);

}

LineSeries = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

for (int i=0; i<C; i++)

{

LineSeries->AddXY(i,m[i],"",clBlue);

}

Form1->ThreadStart();

break;

};

case 1:

{

ofstream fcorr, fcorr1("2\\corr.txt");

fcorr.open("2\\2corr.dat",ios::binary | ios::trunc);

Chart21->BottomAxis->Minimum=0;

Chart21->BottomAxis->Maximum=C-1;

if(HeightsStep1<HeightLimit)

{

AnsiString msg;

msg="Количество высот не должно быть меньше "+IntToStr(HeightLimit);

MessageDlg(msg,mtError,TMsgDlgButtons()<<mbOK,0);

fcorr.close();

}

else

{

double sigmaN=0;

double sig, phi, mm, sum;

CORR = new double[LAG];

SIGNAL2 = new double\*\* [LAG];

for (int l = 0; l < LAG; l++)

{

SIGNAL2[l] = new double\*[HeightsStep1];

for (int h = 0; h < HeightsStep1; h++)

SIGNAL2[l][h]= new double[R];

}

PHASESIG = new double\* [C];

for (int c = 0; c < C; c++)

{

PHASESIG[c] = new double[R];

}

AMPLSIG= new double\* [C];

for (int c = 0; c < C; c++)

{

AMPLSIG[c] = new double[R];

}

ifstream f1;

for (int l=0; l<LAG; l++)

{

fname="1\\1sig"+IntToStr(l+1)+".dat";

f1.open(fname.c\_str(), ios::binary);

for (int h =0; h < HeightsStep1; h++)

{

f1.read((char\*)SIGNAL2[l][h], R\*sizeof(double));

}

f1.close();

}

HeightsStep2=HeightsStep1-HeightLimit+1;

SIGNALH = new double\* [R];

for (int r = 0; r < R; r++)

SIGNALH[r] = new double[LAG];

int start, sum1;

Chart1->BottomAxis->Minimum=0;

Chart1->BottomAxis->Maximum=18;

if(CheckBox4->Checked)

{

PHASE = new double\* [C];

for (int c = 0; c < C; c++)

{

PHASE[c] = new double[R];

}

AMPL= new double\* [C];

for (int c = 0; c < C; c++)

{

AMPL[c] = new double[R];

}

CORRN = new double[LAG];

double q=StrToFloat(LabeledEdit7->Text);

sigmaN=sqrt((HeightNumber\*DispS)/(C\*q));

NOISE = new double\* [R];

for (int r = 0; r < R; r++)

NOISE[r] = new double[LAG+HeightsStep2-1];

ProgressBar1->Min=0;

ProgressBar1->Max=C;

Label3->Caption="Идет генерация случайных фаз гармоник шума...";

Label3->Refresh();

for (int i=0; i<C; i++)

{

ProgressBar1->Position=i+1;

for (int j=0; j<R; j++)

{

PHASE[i][j]=phase();

}

}

Label3->Caption="";

Label3->Refresh();

Label3->Caption="Идет генерация случайных амплитуд гармоник шума...";

Label3->Refresh();

for (int i=0; i<C; i++)

{

ProgressBar1->Position=i+1;

for (int j=0; j<R; j++)

{

AMPL[i][j]=sigmaN\*sqrt(-2.0\*log(1E-10+phase()/(2\*M\_PI)));

}

}

Label3->Caption="";

Refresh();

Label3->Caption="Идет генерация шума...";

Label3->Refresh();

ProgressBar1->Max=LAG+HeightsStep2-1;

for (int l=0; l<LAG+HeightsStep2-1; l++)

{

ProgressBar1->Position=l;

for(int j=0; j<R; j++)

{

sig=0;

for(int i=0; i<C; i++)

{

phi = PHASE[i][j];

mm=AMPL[i][j];

sig=sig+mm\*cos(2\*M\_PI\*tauC\*l\*i+phi);

};

NOISE[j][l]=sig;

}

}

}

for (int c=0; c<C; c++)

{

m[c]=sqrt(m[c]\*m[c]\*HeightNumber+sigmaN\*sigmaN);

}

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

PHASESIG[i][j]=phase();

}

}

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

AMPLSIG[i][j]= m[i]\*sqrt(-2.0\*log(1E-10+phase()/(2\*M\_PI)));

if (CheckBox4->Checked)

{ AMPLSIG[i][j]=sqrt(AMPLSIG[i][j]\*AMPLSIG[i][j]+AMPL[i][j]\*AMPL[i][j]+2\*AMPLSIG[i][j]\*AMPL[i][j]\*cos(PHASE[i][j]-PHASESIG[i][j]));

}

}

}

TLineSeries \*LineSeries2 = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries2);

for (int i=0; i<C; i++)

{

LineSeries2->AddXY(i,AMPLSIG[i][0],"",clRed);

}

LineSeries2 = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries2);

for (int i=0; i<C; i++)

{

LineSeries2->AddXY(i,m[i],"",clBlue);

}

ProgressBar1->Min=0;

ProgressBar1->Max=HeightsStep2;

for (int h=0; h< HeightsStep2; h++)

{

if(CheckBox4->Checked)

{

for (int i=0; i<LAG; i++)

{

start=floor(9-i/2);

sum=0;

for (int j=0; j<R; j++)

{

sum=sum+NOISE[j][start+h-1]\*NOISE[j][start+h-1+i];

}

CORRN[i]=sum;

}

}

TLineSeries \*LineSeries = new TLineSeries(Chart1);

Chart1->AddSeries(LineSeries);

ProgressBar1->Position=h+1;

Label3->Caption=capt+IntToStr(h+1);

for (int l=0; l<LAG; l++)

{

Lll[l]->Clear();

}

Form1->Refresh();

for (int l=0; l< LAG; l++)

{

for (int r=0; r < R; r++)

{

sum=0;

for (int i=0; i<HeightNumber; i++)

{

sum=sum+SIGNAL2[l][h+i+l][r];

}

if(CheckBox4->Checked)

{

SIGNALH[r][l]=sum+NOISE[r][l+h];

}

else

{

SIGNALH[r][l]=sum;

}

Lll[l]->AddXY(r,SIGNALH[r][l],"",clRed);

}

}

for (int i=0; i<LAG; i++)

{

start=floor(9-i/2);

sum=0;

for (int j=0; j<R; j++)

{

sum=sum+SIGNALH[j][start]\*SIGNALH[j][start+i];

}

CORR[i]=sum;

if(CheckBox4->Checked)

{

CORR[i]=CORR[i]-CORRN[i];

}

}

sum1=CORR[0]+1E-10;

for (int i=0; i<LAG; i++)

{

CORR[i]=CORR[i]/sum1;

fcorr1<<CORR[i]<<' ';

LineSeries->AddXY(i,CORR[i],"",clRed);

}

fcorr1<<'\n';

Refresh();

fcorr.write((char\*)Form1->CORR,LAG\*sizeof(double));

Sleep(2000);

}

TLineSeries \*LineSeries1 = new TLineSeries(Form1->Chart1);

Form1->Chart1->AddSeries(LineSeries1);

for (int i=0; i<LAG; i++)

{

LineSeries1->AddXY(i,Acf\_th[i]\*(22-i)/22.,"",clBlue);

LineSeries1->LinePen->Width=1.75;

}

for (int l = 0; l< LAG; l++)

{

for (int h = 0; h < HeightsStep1; h++)

{

delete [] SIGNAL2[l][h];

}

delete [] SIGNAL2[l];

}

delete [] SIGNAL2;

delete [] CORR;

delete [] SIGNALH;

for (int c = 0; c < C; c++)

{

delete [] PHASESIG[c];

}

delete [] PHASESIG;

for (int c = 0; c < C; c++)

{

delete [] AMPLSIG[c];

}

delete [] AMPLSIG;

if (CheckBox4->Checked)

{

delete [] CORRN;

for (int c = 0; c < C; c++)

{

delete [] PHASE[c];

}

delete [] PHASE;

for (int c = 0; c < C; c++)

{

delete [] AMPL[c];

}

delete [] AMPL;

for (int r = 0; r < R; r++)

delete [] NOISE[r];

delete [] NOISE;

}

Label3->Caption="Генерация закончена";

ProgressBar1->Position=0;

fcorr.flush();

fcorr.close();

if(!Form1->CheckBox2->Checked)

{

AnsiString msg;

msg="Шаг "+IntToStr(PageControl1->ActivePageIndex+1)+" выполнен";

MessageDlg(msg,mtInformation,TMsgDlgButtons()<<mbOK,0);

}

else

{

Form1->PageControl1->SelectNextPage(true);

Form1->Button2Click(Form1->Button2);

}

}

break;

};

case 2:

{

TLineSeries \*LineSeries;

ifstream f1, f2;

ofstream fcorr, fcorr1("3\\corr.txt");

fcorr.open("3\\3corr.dat",ios::binary | ios::trunc);

int Number1=HeightsStep1-HeightLimit+1;

int Number2=HeightsStep1-HeightLimit-LAG+2;

CORR = new double[LAG];

CORR2 = new double\* [Number1];

for (int i=0; i<Number1; i++)

{

CORR2[i] = new double[LAG];

}

f1.open("1\\1corr.dat", ios::binary);

f2.open("2\\2corr.dat", ios::binary);

Chart21->BottomAxis->Minimum=0;

Chart21->BottomAxis->Maximum=18;

Chart22->BottomAxis->Minimum=0;

Chart22->BottomAxis->Maximum=18;

Chart24->BottomAxis->Minimum=0;

Chart24->BottomAxis->Maximum=18;

Chart1->BottomAxis->Minimum=0;

Chart1->BottomAxis->Maximum=18;

for (int i=0; i<HeightsStep1; i++)

{

LineSeries = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

f1.read((char\*)CORR, LAG\*sizeof(double));

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,CORR[k],"",clRed);

LineSeries->LinePen->Width=1.75;

}

};

LineSeries = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,Acf\_th[k],"",clBlue);

LineSeries->LinePen->Width=1.75;

}

for (int i=0; i<Number1; i++)

{

LineSeries = new TLineSeries(Chart1);

Chart1->AddSeries(LineSeries);

f2.read((char\*)CORR2[i], LAG\*sizeof(double));

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,CORR2[i][k]\*22./(22-k),"",clRed);

LineSeries->LinePen->Width=1.75;

}

};

LineSeries = new TLineSeries(Chart1);

Chart1->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,Acf\_th[k],"",clBlue);

LineSeries->LinePen->Width=1.75;

}

if (Number1<LAG)

{

AnsiString msg;

msg="Количество высот не должно быть меньше "+IntToStr(LAG);

MessageDlg(msg,mtError,TMsgDlgButtons()<<mbOK,0);

}

else

{

int start;

for(int z=0; z<Number2; z++)

{

double sum;

for (int i=0; i<LAG; i++)

{

sum=0;

start=floor(9-i/2);

for (int j=0; j<i+1; j++)

{

sum=sum+CORR2[z+start+j][i];

}

CORR[i]=sum/(i+1)\*22./(22-i);

fcorr1<<CORR[i]<<' ';

}

fcorr1<<'\n';

fcorr.write((char\*)CORR,LAG\*sizeof(double));

TLineSeries \*LineSeries = new TLineSeries(Chart22);

Chart22->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,CORR[k],"",clRed);

LineSeries->LinePen->Width=1.75;

}

LineSeries = new TLineSeries(Chart24);

Chart24->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,CORR[k]\*(22-k)/22.,"",clRed);

LineSeries->LinePen->Width=1.75;

}

}

LineSeries = new TLineSeries(Chart22);

Chart22->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,Acf\_th[k],"",clBlue);

LineSeries->LinePen->Width=1.75;

}

LineSeries = new TLineSeries(Chart24);

Chart24->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,Acf\_th[k]\*(22-k)/22.,"",clBlue);

LineSeries->LinePen->Width=1.75;

}

Form1->Refresh();

Sleep(2000);

fcorr.flush();

fcorr.close();

delete [] CORR;

for (int i=0; i<Number1; i++)

{

delete [] CORR2;

}

if(!Form1->CheckBox2->Checked)

{

AnsiString msg;

msg="Шаг "+IntToStr(PageControl1->ActivePageIndex+1)+" выполнен"; MessageDlg(msg,mtInformation,TMsgDlgButtons()<<mbOK,0);

}

else

{

Form1->PageControl1->SelectNextPage(true);

Form1->Button2Click(Form1->Button2);

}

}

break;

};

case 3:

{

ifstream fcorr, filter("filter.dat");

ofstream fcorr1, fcorr2, fconv("4\\acf\_th\_conv.txt");

fcorr1.open("4\\4corr.dat",ios::binary | ios::app);

fcorr2.open("4\\corr4.txt", ios::app);

CORR = new double[LAG];

fcorr.open("3\\3corr.dat",ios::binary);

double\* corr\_filter = new double [2\*LAG-1];

double\* corr\_func = new double [2\*LAG-1];

double\* corr\_conv = new double [LAG];

for (int i=0; i<LAG; i++)

{

filter>>corr\_filter[LAG+i-1];

corr\_filter[LAG-i-1]=corr\_filter[LAG+i-1];

}

Chart1->BottomAxis->Minimum=0;

Chart1->BottomAxis->Maximum=18;

Chart21->BottomAxis->Minimum=0;

Chart21->BottomAxis->Maximum=18;

Chart22->BottomAxis->Minimum=0;

Chart22->BottomAxis->Maximum=18;

Chart24->BottomAxis->Minimum=0;

Chart24->BottomAxis->Maximum=18;

TLineSeries\* LineSeries= new TLineSeries(Chart1);

Chart1->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,corr\_filter[LAG+k-1],"",clRed);

LineSeries->LinePen->Width=1.75;

}

for (int i=0; i<HeightsStep1-HeightLimit-LAG+2; i++)

{

fcorr.read((char\*)CORR, LAG\*sizeof(double));

LineSeries= new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,CORR[k],"",clRed);

LineSeries->LinePen->Width=1.75;

}

for (int ii=0; ii<LAG; ii++)

{

corr\_func[LAG-ii-1]=CORR[ii]\*(tauPulse-ii\*tauC)/tauPulse;

corr\_func[LAG+ii-1]=CORR[ii]\*(tauPulse-ii\*tauC)/tauPulse;

}

double sum;

for (int ii=0; ii<LAG; ii++)

{

sum=0;

for (int k=ii; k<2\*LAG-1; k++)

{

sum=sum+corr\_func[k]\*corr\_filter[k-ii];

}

corr\_conv[ii]=sum;

}

for (int ii=LAG-1; ii>-1; ii--)

{

double temp=corr\_conv[ii]/corr\_conv[0];

corr\_conv[ii]=temp;

};

LineSeries= new TLineSeries(Chart22);

Chart22->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,corr\_conv[k]/((tauPulse-k\*tauC)/tauPulse),"",clRed);

LineSeries->LinePen->Width=1.75;

}

LineSeries= new TLineSeries(Chart24);

Chart24->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,corr\_conv[k],"",clRed);

LineSeries->LinePen->Width=1.75;

}

fcorr1.write((char\*)corr\_conv,LAG\*sizeof(double));

for (int kk=0; kk<LAG; kk++)

{

fcorr2<<corr\_conv[kk]<<' ';

}

fcorr2<<'\n';

}

LineSeries= new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,Acf\_th[k],"",clBlue);

LineSeries->LinePen->Width=1.75;

}

double\* Acf\_th\_conv = new double [LAG];

double\* Acf\_th\_corr = new double [2\*LAG-1];

for (int i=0; i<LAG; i++)

{

Acf\_th\_corr[LAG-i-1]=Acf\_th[i]\*(tauPulse-i\*tauC)/tauPulse;

Acf\_th\_corr[LAG+i-1]=Acf\_th[i]\*(tauPulse-i\*tauC)/tauPulse;

}

double sum;

for (int i=0; i<LAG; i++)

{

sum=0;

for (int k=i; k<2\*LAG-1; k++)

{

sum=sum+Acf\_th\_corr[k]\*corr\_filter[k-i];

}

Acf\_th\_conv[i]=sum;

}

for (int ii=LAG-1; ii>-1; ii--)

{

double temp=Acf\_th\_conv[ii]/Acf\_th\_conv[0];

Acf\_th\_conv[ii]=temp;};

LineSeries= new TLineSeries(Chart22);

Chart22->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,Acf\_th\_conv[k]/((tauPulse-k\*tauC)/tauPulse),"",clBlue);

LineSeries->LinePen->Width=1.75;

}

LineSeries= new TLineSeries(Chart24);

Chart24->AddSeries(LineSeries);

for (int k=0; k<LAG; k++)

{

LineSeries->AddXY(k,Acf\_th\_conv[k],"",clBlue);

LineSeries->LinePen->Width=1.75;

fconv<<Acf\_th\_conv[k]<<' ';

}

fconv<<'\n';

fcorr1.close();

fcorr2.close();

fconv.close();

delete [] CORR;

delete [] corr\_filter;

delete [] corr\_func;

delete [] corr\_conv;

delete [] Acf\_th\_conv;

delete [] Acf\_th\_corr;

Form1->Refresh();

Sleep(2000);

if(!Form1->CheckBox2->Checked)

{

AnsiString msg;

msg="Шаг "+IntToStr(PageControl1->ActivePageIndex+1)+" выполнен"; MessageDlg(msg,mtInformation,TMsgDlgButtons()<<mbOK,0);

}

else

{

NumberOfIterations--;

Form1->Edit2->Text=NumberOfIterations;

if(NumberOfIterations)

{

Form1->PageControl1->ActivePage=TabSheet1;

Form1->Button2Click(Form1->Button2);

}

else

{

AnsiString msg;

msg="Все операции выполнены";

MessageDlg(msg,mtInformation,TMsgDlgButtons()<<mbOK,0);

}

}

break;

}

case 4:

{

Label3->Caption="Идет генерация случайных фаз гармоник шума...";

Label3->Refresh();

AMPL= new double\* [C];

for (int c = 0; c < C; c++)

{

AMPL[c] = new double[R];

}

Chart21->BottomAxis->Minimum=0;

Chart21->BottomAxis->Maximum=C-1;

Label3->Caption="";

Label3->Refresh();

Label3->Caption="Идет генерация случайных амплитуд гармоник шума...";

Label3->Refresh();

for (int i=0; i<C; i++)

{

AMPL[i][0]=m[i]\*sqrt(-2.0\*log(1E-10+phase()/(2\*M\_PI)));

}

TLineSeries \*LineSeries = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

for (int i=0; i<C; i++)

{

LineSeries->AddXY(i,AMPL[i][0],"",clRed);

}

LineSeries = new TLineSeries(Chart21);

Chart21->AddSeries(LineSeries);

for (int i=0; i<C; i++)

{

LineSeries->AddXY(i,m[i],"",clBlue);

}

Chart1->BottomAxis->Minimum=0;

Chart1->BottomAxis->Maximum=C-1;

Label3->Caption="";

Refresh();

Label3->Caption="Идет генерация шума...";

Label3->Refresh();

double sigmaN=0;

double sig, phi, mm, sum;

PHASESIG = new double\* [C];

for (int c = 0; c < C; c++)

{

PHASESIG[c] = new double[R];

}

AMPLSIG= new double\* [C];

for (int c = 0; c < C; c++)

{

AMPLSIG[c] = new double[R];

}

HeightsStep2=HeightsStep1-HeightLimit+1;

int start, sum1;

PHASE = new double\* [C];

for (int c = 0; c < C; c++)

{

PHASE[c] = new double[R];

}

double q=StrToFloat(LabeledEdit7->Text);

sigmaN=sqrt((HeightNumber\*DispS)/(C\*q));

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

PHASE[i][j]=phase();

}

}

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

AMPL[i][j]=sigmaN\*sqrt(-2.0\*log(1E-10+phase()/(2\*M\_PI)));

}

}

for (int c=0; c<C; c++)

{

m[c]=sqrt(m[c]\*m[c]\*HeightNumber+sigmaN\*sigmaN);

}

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

PHASESIG[i][j]=phase();

}

}

for (int i=0; i<C; i++)

{

for (int j=0; j<R; j++)

{

AMPLSIG[i][j]= m[i]\*sqrt(-2.0\*log(1E-10+phase()/(2\*M\_PI)));

}

}

Label3->Caption="";

Refresh();

Label3->Caption="Генерация завершина...";

Label3->Refresh();

TLineSeries \*LineSeries2 = new TLineSeries(Chart1);

Chart1->AddSeries(LineSeries2);

for (int i=0; i<C; i++)

{

LineSeries2->AddXY(i,AMPLSIG[i][0],"",clRed);

}

LineSeries2 = new TLineSeries(Chart1);

Chart1->AddSeries(LineSeries2);

for (int i=0; i<C; i++)

{

LineSeries2->AddXY(i,m[i],"",clBlue);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int m=0, n=19, m\_M=0, temp1,temp2;

double delta, dif;

double ti\_r=0, te\_r=0;

temp1= StrToInt(LabeledEdit1->Text);

temp2= StrToInt(LabeledEdit2->Text);

string s,s2;

vector <string> v;

vector <string> w;

ifstream fin,fin2;

fin.open("4\\corr4.txt");

getline(fin,s);

while (fin)

{

v.push\_back(s);

getline(fin,s);

}

fin2.open("acf\_o\_10K.txt");

getline(fin2,s2);

while (fin2)

{

w.push\_back(s2);

getline(fin2,s2);

}

m=v.size();

m\_M=w.size();

fin.close();

fin2.close();

Chart22->BottomAxis->Minimum=-1;

Chart22->BottomAxis->Maximum=m+1;

Chart24->BottomAxis->Minimum=-1;

Chart24->BottomAxis->Maximum=m+1;

vector<vector<double> > acf4(m,vector<double>(n));

vector<vector<double> > acf(m\_M,vector<double>(n));

vector<int>ti;

vector<int>te;

int x,y;

ifstream f("acf\_o\_10K.txt");

ofstream F("4\\ti\_te.txt");

ifstream f2("4\\corr4.txt");

Label3->Caption="";

Refresh();

Label3->Caption="Идет считывание данных...";

Label3->Refresh();

for (int i=0; i<m\_M; i++)

{

f >>x>> y;

ti.push\_back(x);

te.push\_back(y);

for(int j=0; j<n; j++)

{

f>>acf[i][j];

}

}

for (int i=0; i<m; i++)

{

for(int j=0; j<n; j++)

{

f2>>acf4[i][j];

}

}

LineSeries = new TLineSeries(Chart22);

Chart22->AddSeries(LineSeries);

Series22=new TPointSeries(Chart22);

Chart22->AddSeries(Series22);

Series22->Pointer->Style=psCircle;

LineSeries2 = new TLineSeries(Chart24);

Chart24->AddSeries(LineSeries2);

Series23=new TPointSeries(Chart24);

Chart24->AddSeries(Series23);

Series23->Pointer->Style=psCircle;

ProgressBar1->Min=0;

ProgressBar1->Max=m;

Label3->Caption="";

Refresh();

Label3->Caption="Идет расчет температур ...";

Label3->Refresh();

for (int i=0; i<m; i++)

{

delta=1000;

for (int t=0; t<m\_M; t++)

{

dif=0;

for(int j=0; j<n; j++)

{

dif+=(acf4[i][j]-acf[t][j])\*(acf4[i][j]-acf[t][j]);

}

if (dif<delta)

{

delta=dif;

ti\_r=ti[t];

te\_r=te[t];

}

}

F<<ti\_r<<" "<<te\_r<<endl;

Series23->AddXY(i,te\_r,"",clRed);

Series22->AddXY(i,ti\_r,"",clRed);

LineSeries2->AddXY(i,temp2,"",clBlue);

LineSeries->AddXY(i,temp1,"",clBlue);

ProgressBar1->Position=i+1;

}

Label3->Caption="";

Refresh();

Label3->Caption="Значения температур построены...";

Label3->Refresh();

f.close();

F.close();

f2.close();

Sleep(2000);

ProgressBar1->Position=0;

Label3->Caption="";

Refresh();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

break; }; };

if((PageControl1->ActivePageIndex)&&(!Form1->CheckBox2->Checked))

{

bool Flash;

FlashWindow(Form1->Handle,Flash);

FlashWindow(Application->Handle,Flash);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

Chart1->Width=1000;

Chart1->Height=700;

Chart1->SaveToBitmapFile("1.bmp") ;

Chart1->Width=300;

Chart1->Height=60;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button5Click(TObject \*Sender)

{

if (MessageDlg("Закрыть приложение?", mtConfirmation, TMsgDlgButtons() << mbYes << mbNo,0) == mrYes)

{

Close();

}

}

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

init();

Lll[0]=Form1->Series2;

Lll[1]=Form1->Series3;

Lll[2]=Form1->Series4;

Lll[3]=Form1->Series5;

Lll[4]=Form1->Series6;

Lll[5]=Form1->Series7;

Lll[6]=Form1->Series8;

Lll[7]=Form1->Series9;

Lll[8]=Form1->Series10;

Lll[9]=Form1->Series11;

Lll[10]=Form1->Series12;

Lll[11]=Form1->Series13;

Lll[12]=Form1->Series14;

Lll[13]=Form1->Series15;

Lll[14]=Form1->Series16;

Lll[15]=Form1->Series17;

Lll[16]=Form1->Series18;

Lll[17]=Form1->Series19;

Lll[18]=Form1->Series20;

Graph[0]=Form1->Chart2;

Graph[1]=Form1->Chart3;

Graph[2]=Form1->Chart4;

Graph[3]=Form1->Chart5;

Graph[4]=Form1->Chart6;

Graph[5]=Form1->Chart7;

Graph[6]=Form1->Chart8;

Graph[7]=Form1->Chart9;

Graph[8]=Form1->Chart10;

Graph[9]=Form1->Chart11;

Graph[10]=Form1->Chart12;

Graph[11]=Form1->Chart13;

Graph[12]=Form1->Chart14;

Graph[13]=Form1->Chart15;

Graph[14]=Form1->Chart16;

Graph[15]=Form1->Chart17;

Graph[16]=Form1->Chart18;

Graph[17]=Form1->Chart19;

Graph[18]=Form1->Chart20;

for (int i=0; i<19; i++)

{

Graph[i]->BottomAxis->Minimum=0;

Graph[i]->BottomAxis->Maximum=1462;

};

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CheckBox4Click(TObject \*Sender)

{

if (CheckBox4->Checked)

{

LabeledEdit7->Enabled=true;

}

else

{

LabeledEdit7->Enabled=false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::PageControl1Change(TObject \*Sender)

{

Button1Click(Button2);

Label3->Caption="";

switch(PageControl1->ActivePageIndex)

{

case 0:

{

Label2->Caption="Шаг 1. Генерация НР сигналов по спектру";

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart22->Visible=false;

Chart23->Visible=false;

Chart24->Visible=false;

Chart25->Visible=false;

ProgressBar1->Left=312;

ProgressBar1->Width=593;

break;

}

case 1:

{

Label2->Caption="Шаг 2. Учет высотной протяженности и шума";

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart22->Visible=false;

Chart23->Visible=false;

Chart24->Visible=false;

Chart25->Visible=false;

ProgressBar1->Left=312;

ProgressBar1->Width=593;

break;

}

case 2:

{

Label2->Caption="Шаг 3. Трапецeидальное суммирование";

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart22->Visible=true;

Chart23->Visible=false;

Chart24->Visible=true;

Chart25->Visible=false;

ProgressBar1->Left=12;

ProgressBar1->Width=893;

break;

}

case 3:

{

Label2->Caption="Шаг 4. Учет импульсной характеристики фильтра";

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart23->Visible=false;

Chart25->Visible=false;

Chart22->Visible=true;

Chart24->Visible=true;

ProgressBar1->Left=12;

ProgressBar1->Width=893;

break;

}

case 4:

{

Label2->Caption="Шаг 5. Визуализация результатов";

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart22->Visible=true;

Chart23->Visible=true;

Chart24->Visible=true;

Chart25->Visible=true;

ProgressBar1->Left=12;

ProgressBar1->Width=893;

break;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CheckBox2Click(TObject \*Sender)

{

if(CheckBox2->Checked)

{

Form1->Edit2->Enabled=true;

}

if(!CheckBox2->Checked)

{

Form1->Edit2->Enabled=false;

Form1->Edit2->Text=1;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart2Click(TObject \*Sender)

{

Chart2->Top=200;

Chart2->Left=60;

Chart2->Width=800;

Chart2->Height=400;

Chart1->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart2DblClick(TObject \*Sender)

{

Chart2->Top=280;

Chart2->Left=8;

Chart2->Width=300;

Chart2->Height=60;

Chart1->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart3Click(TObject \*Sender)

{

Chart3->Top=200;

Chart3->Left=60;

Chart3->Width=800;

Chart3->Height=400;

Chart2->Visible=false;

Chart1->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart3DblClick(TObject \*Sender)

{

Chart3->Top=280;

Chart3->Left=312;

Chart3->Width=300;

Chart3->Height=60;

Chart2->Visible=true;

Chart1->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart4Click(TObject \*Sender)

{

Chart4->Top=200;

Chart4->Left=60;

Chart4->Width=800;

Chart4->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart1->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart4DblClick(TObject \*Sender)

{

Chart4->Top=280;

Chart4->Left=616;

Chart4->Width=300;

Chart4->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart1->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart5Click(TObject \*Sender)

{

Chart5->Top=200;

Chart5->Left=60;

Chart5->Width=800;

Chart5->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart1->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart5DblClick(TObject \*Sender)

{

Chart5->Top=344;

Chart5->Left=8;

Chart5->Width=300;

Chart5->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart1->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart6Click(TObject \*Sender)

{

Chart6->Top=200;

Chart6->Left=60;

Chart6->Width=800;

Chart6->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart1->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart6DblClick(TObject \*Sender)

{

Chart6->Top=344;

Chart6->Left=312;

Chart6->Width=300;

Chart6->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart1->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart7Click(TObject \*Sender)

{

Chart7->Top=200;

Chart7->Left=60;

Chart7->Width=800;

Chart7->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart1->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart7DblClick(TObject \*Sender)

{

Chart7->Top=344;

Chart7->Left=616;

Chart7->Width=300;

Chart7->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart1->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart8Click(TObject \*Sender)

{

Chart8->Top=200;

Chart8->Left=60;

Chart8->Width=800;

Chart8->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart1->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart8DblClick(TObject \*Sender)

{

Chart8->Top=408;

Chart8->Left=8;

Chart8->Width=300;

Chart8->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart1->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart9Click(TObject \*Sender)

{

Chart9->Top=200;

Chart9->Left=60;

Chart9->Width=800;

Chart9->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart1->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart9DblClick(TObject \*Sender)

{

Chart9->Top=408;

Chart9->Left=312;

Chart9->Width=300;

Chart9->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart1->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart10Click(TObject \*Sender)

{

Chart10->Top=200;

Chart10->Left=60;

Chart10->Width=800;

Chart10->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart1->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart10DblClick(TObject \*Sender)

{

Chart10->Top=408;

Chart10->Left=616;

Chart10->Width=300;

Chart10->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart1->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart11Click(TObject \*Sender)

{

Chart11->Top=200;

Chart11->Left=60;

Chart11->Width=800;

Chart11->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart1->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart11DblClick(TObject \*Sender)

{

Chart11->Top=472;

Chart11->Left=8;

Chart11->Width=300;

Chart11->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart1->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart12Click(TObject \*Sender)

{

Chart12->Top=200;

Chart12->Left=60;

Chart12->Width=800;

Chart12->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart1->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart12DblClick(TObject \*Sender)

{

Chart12->Top=472;

Chart12->Left=312;

Chart12->Width=300;

Chart12->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart1->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart13Click(TObject \*Sender)

{

Chart13->Top=200;

Chart13->Left=60;

Chart13->Width=800;

Chart13->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart1->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart13DblClick(TObject \*Sender)

{

Chart13->Top=472;

Chart13->Left=616;

Chart13->Width=300;

Chart13->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart1->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart14Click(TObject \*Sender)

{

Chart14->Top=200;

Chart14->Left=60;

Chart14->Width=800;

Chart14->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart1->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart14DblClick(TObject \*Sender)

{

Chart14->Top=536;

Chart14->Left=8;

Chart14->Width=300;

Chart14->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart1->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart15Click(TObject \*Sender)

{

Chart15->Top=200;

Chart15->Left=60;

Chart15->Width=800;

Chart15->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart1->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart15DblClick(TObject \*Sender)

{

Chart15->Top=536;

Chart15->Left=312;

Chart15->Width=300;

Chart15->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart1->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart16Click(TObject \*Sender)

{

Chart16->Top=200;

Chart16->Left=60;

Chart16->Width=800;

Chart16->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart1->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart16DblClick(TObject \*Sender)

{

Chart16->Top=536;

Chart16->Left=616;

Chart16->Width=300;

Chart16->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart1->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart17Click(TObject \*Sender)

{

Chart17->Top=200;

Chart17->Left=60;

Chart17->Width=800;

Chart17->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart1->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart17DblClick(TObject \*Sender)

{

Chart17->Top=600;

Chart17->Left=8;

Chart17->Width=300;

Chart17->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart1->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart18Click(TObject \*Sender)

{

Chart18->Top=200;

Chart18->Left=60;

Chart18->Width=800;

Chart18->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart1->Visible=false;

Chart19->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart18DblClick(TObject \*Sender)

{

Chart18->Top=600;

Chart18->Left=312;

Chart18->Width=300;

Chart18->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart1->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart19DblClick(TObject \*Sender)

{

Chart19->Top=600;

Chart19->Left=616;

Chart19->Width=300;

Chart19->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart1->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart19Click(TObject \*Sender)

{

Chart19->Top=200;

Chart19->Left=60;

Chart19->Width=800;

Chart19->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart1->Visible=false;

Chart20->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart20Click(TObject \*Sender)

{

Chart20->Top=200;

Chart20->Left=60;

Chart20->Width=800;

Chart20->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart1->Visible=false;

Chart21->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart20DblClick(TObject \*Sender)

{

Chart20->Top=664;

Chart20->Left=8;

Chart20->Width=300;

Chart20->Height=60;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart1->Visible=true;

Chart21->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormShortCut(TWMKey &Msg, bool &Handled)

{

if ((PageControl1->ActivePageIndex==0)||(PageControl1->ActivePageIndex==1))

{

if (Msg.CharCode == 0x1B) // Клавіша ESC Esc 27 0x1B VK\_ESCAPE

{

// MessageDlg("Нажата клавиша ESC.", mtInformation,TMsgDlgButtons()<<mbOK ,0);

Chart2->Top=280;

Chart2->Left=8;

Chart2->Width=300;

Chart2->Height=60;

Chart3->Top=280;

Chart3->Left=312;

Chart3->Width=300;

Chart3->Height=60;

Chart4->Top=280;

Chart4->Left=616;

Chart4->Width=300;

Chart4->Height=60;

Chart5->Top=344;

Chart5->Left=8;

Chart5->Width=300;

Chart5->Height=60;

Chart6->Top=344;

Chart6->Left=312;

Chart6->Width=300;

Chart6->Height=60;

Chart7->Top=344;

Chart7->Left=616;

Chart7->Width=300;

Chart7->Height=60;

Chart8->Top=408;

Chart8->Left=8;

Chart8->Width=300;

Chart8->Height=60;

Chart9->Top=408;

Chart9->Left=312;

Chart9->Width=300;

Chart9->Height=60;

Chart10->Top=408;

Chart10->Left=616;

Chart10->Width=300;

Chart10->Height=60;

Chart11->Top=472;

Chart11->Left=8;

Chart11->Width=300;

Chart11->Height=60;

Chart12->Top=472;

Chart12->Left=312;

Chart12->Width=300;

Chart12->Height=60;

Chart13->Top=472;

Chart13->Left=616;

Chart13->Width=300;

Chart13->Height=60;

Chart14->Top=536;

Chart14->Left=8;

Chart14->Width=300;

Chart14->Height=60;

Chart15->Top=536;

Chart15->Left=312;

Chart15->Width=300;

Chart15->Height=60;

Chart16->Top=536;

Chart16->Left=616;

Chart16->Width=300;

Chart16->Height=60;

Chart17->Top=600;

Chart17->Left=8;

Chart17->Width=300;

Chart17->Height=60;

Chart18->Top=600;

Chart18->Left=312;

Chart18->Width=300;

Chart18->Height=60;

Chart19->Top=600;

Chart19->Left=616;

Chart19->Width=300;

Chart19->Height=60;

Chart20->Top=664;

Chart20->Left=8;

Chart20->Width=300;

Chart20->Height=60;

Chart21->Top=80;

Chart21->Left=16;

Chart21->Width=457;

Chart21->Height=185;

Chart1->Top=80;

Chart1->Left=480;

Chart1->Width=433;

Chart1->Height=185;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart1->Visible=true;

Chart21->Visible=true;

Chart20->Visible=true;

}

}

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart21->Top=80;

Chart21->Left=16;

Chart21->Width=457;

Chart21->Height=185;

Chart1->Top=80;

Chart1->Left=480;

Chart1->Width=433;

Chart1->Height=185;

Chart22->Top=288;

Chart22->Left=16;

Chart22->Width=457;

Chart22->Height=185;

Chart24->Top=288;

Chart24->Left=480;

Chart24->Width=433;

Chart24->Height=185;

Chart1->Visible=true;

Chart22->Visible=true;

Chart21->Visible=true;

Chart24->Visible=true;

// Chart25->Visible=true;

if (PageControl1->ActivePageIndex==4)

{

Chart23->Visible=true;

Chart25->Visible=true;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart21Click(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==0)||(PageControl1->ActivePageIndex==1))

{

Chart21->Top=200;

Chart21->Left=60;

Chart21->Width=800;

Chart21->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart1->Visible=false;

Chart20->Visible=false;

}

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart21->Top=200;

Chart21->Left=60;

Chart21->Width=800;

Chart21->Height=400;

Chart1->Visible=false;

Chart22->Visible=false;

Chart23->Visible=false;

Chart24->Visible=false;

Chart25->Visible=false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart21DblClick(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==0)||(PageControl1->ActivePageIndex==1))

{

Chart21->Top=80;

Chart21->Left=16;

Chart21->Width=457;

Chart21->Height=185;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart1->Visible=true;

Chart20->Visible=true;

}

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart21->Top=80;

Chart21->Left=16;

Chart21->Width=457;

Chart21->Height=185;

Chart1->Visible=true;

Chart22->Visible=true;

// Chart23->Visible=true;

Chart24->Visible=true;

// Chart25->Visible=true;

if (PageControl1->ActivePageIndex==4)

{

Chart23->Visible=true;

Chart25->Visible=true;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart1Click(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==0)||(PageControl1->ActivePageIndex==1))

{

Chart1->Top=200;

Chart1->Left=60;

Chart1->Width=800;

Chart1->Height=400;

Chart2->Visible=false;

Chart3->Visible=false;

Chart4->Visible=false;

Chart5->Visible=false;

Chart6->Visible=false;

Chart7->Visible=false;

Chart8->Visible=false;

Chart9->Visible=false;

Chart10->Visible=false;

Chart11->Visible=false;

Chart12->Visible=false;

Chart13->Visible=false;

Chart14->Visible=false;

Chart15->Visible=false;

Chart16->Visible=false;

Chart17->Visible=false;

Chart18->Visible=false;

Chart19->Visible=false;

Chart21->Visible=false;

Chart20->Visible=false;

}

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart1->Top=200;

Chart1->Left=60;

Chart1->Width=800;

Chart1->Height=400;

Chart21->Visible=false;

Chart22->Visible=false;

Chart23->Visible=false;

Chart24->Visible=false;

Chart25->Visible=false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart1DblClick(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==0)||(PageControl1->ActivePageIndex==1))

{

Chart1->Top=80;

Chart1->Left=480;

Chart1->Width=433;

Chart1->Height=185;

Chart2->Visible=true;

Chart3->Visible=true;

Chart4->Visible=true;

Chart5->Visible=true;

Chart6->Visible=true;

Chart7->Visible=true;

Chart8->Visible=true;

Chart9->Visible=true;

Chart10->Visible=true;

Chart11->Visible=true;

Chart12->Visible=true;

Chart13->Visible=true;

Chart14->Visible=true;

Chart15->Visible=true;

Chart16->Visible=true;

Chart17->Visible=true;

Chart18->Visible=true;

Chart19->Visible=true;

Chart20->Visible=true;

Chart21->Visible=true;

}

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart1->Top=80;

Chart1->Left=480;

Chart1->Width=433;

Chart1->Height=185;

Chart21->Visible=true;

Chart22->Visible=true;

//Chart23->Visible=true;

Chart24->Visible=true;

//Chart25->Visible=true;

if (PageControl1->ActivePageIndex==4)

{

Chart23->Visible=true;

Chart25->Visible=true;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart22Click(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart22->Top=200;

Chart22->Left=60;

Chart22->Width=800;

Chart22->Height=400;

Chart21->Visible=false;

Chart1->Visible=false;

Chart23->Visible=false;

Chart24->Visible=false;

Chart25->Visible=false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart22DblClick(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart22->Top=288;

Chart22->Left=16;

Chart22->Width=457;

Chart22->Height=185;

Chart21->Visible=true;

Chart1->Visible=true;

//Chart23->Visible=true;

Chart24->Visible=true;

//Chart25->Visible=true;

if (PageControl1->ActivePageIndex==4)

{

Chart23->Visible=true;

Chart25->Visible=true;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart24Click(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart24->Top=200;

Chart24->Left=60;

Chart24->Width=800;

Chart24->Height=400;

Chart21->Visible=false;

Chart1->Visible=false;

Chart23->Visible=false;

Chart22->Visible=false;

Chart25->Visible=false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Chart24DblClick(TObject \*Sender)

{

if ((PageControl1->ActivePageIndex==2)||(PageControl1->ActivePageIndex==3)||(PageControl1->ActivePageIndex==4))

{

Chart24->Top=288;

Chart24->Left=480;

Chart24->Width=433;

Chart24->Height=185;

Chart21->Visible=true;

Chart1->Visible=true;

//Chart23->Visible=true;

Chart22->Visible=true;

//Chart25->Visible=true;

if (PageControl1->ActivePageIndex==4)

{

Chart23->Visible=true;

Chart25->Visible=true;

}

}

}

//---------------------------------------------------------------------------