МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Костюченка Владислава Олеговича
(прізвище, ім'я, по батькові здобувача)

на тему «Розробка програмного забезпечення аналізу криптовалют»
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

науковий керівник к.е.н., доцент _____ Лисенко В.С.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 р. № 12
Завідувач кафедри _____
(підпис)

д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет Інформаційних технологій
Кафедра Інформатики і прикладного програмного забезпечення
Спеціальність Інженерія програмного забезпечення
Форма навчання Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
« 11 » червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи « Розробка програмного забезпечення аналізу криптовалют»

Керівник роботи к.е.н., доцент Лисенко В.С.
затверджені наказом закладу вищої освіти від «04» березня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Розробка алгоритму розв'язання задачі

Розділ 3. Організація інформаційного забезпечення

Розділ 4. Розробка програмного забезпечення задачі

Об'єкт дослідження: процес розрахунку доходності від майнінгу криптовалюти bitcoin

Предмет дослідження: інформаційна підтримка процесу розрахунку доходності криптовалют та аналіз курсів криптовалют в часі

Мета кваліфікаційної роботи: *розробка програмного забезпечення аналізу криптовалют*

5. Дата видачі завдання «04» березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ ___Лисенко В.С.___
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___Костюченко В.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення аналізу криптовалют»

Костюченка Владислава Олеговича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

У дипломній роботі розроблено програмне забезпечення аналізу криптовалют. Програмне забезпечення складається з двох частин: калькулятор криптовалют розроблено на мові C# з використанням Windows Forms, технології ADO .NET для обробки баз даних, а також модуль зміни динаміки курсів валют, розроблений на мові Visual C++ з використанням бібліотеки OpenGL для роботи з 3D-графікою та технології ADO для роботи з базами даних.

При розробці програмного забезпечення передбачено динамічне оновлення курсів з ресурсу apirone.com за допомогою класу WebClient, при розрахунку доходності враховуються витрати на електроенергію, а також всі види комісій.

Результати роботи рекомендуються до застосування користувачами для добутку монет криптовалют, а також їх обміну.

Ключові слова: Криптовалюта, Bitcoin, Litecoin, програмне забезпечення, інтернет, калькулятор, OpenGL, ADO, ADO .NET.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

.NET FrameWork	Платформа фірми Microsoft, призначена для розробки нативних та web-додатків
ПЗ	Програмне забезпечення
БД	База даних
СУБД	Система управління базами даних
WPF	Windows Presentation Foundation – тип проекту для побудови клієнт-серверних додатків для Windows
ADO .NET	ActiveX Data Object для .NET – технологія доступу і управління базами даних для платформи .NET

ЗМІСТ

ВСТУП

В час стрімкого розвитку комп’ютерних технологій значне місце займає криптовалюта. На відміну від "паперових" грошей, які випускають уряди різних країн, криптовалюта непідконтрольна жодному регулятору. Валюта не має адміністратора і повністю автономна, тому й заборонити їй користуватися ніхто не може. Можна сказати, що криптовалюта – це набір чисел, який може передаватися від одного комп'ютера до іншого, причому інформація про транзакції зберігається в зашифрованому вигляді у всіх учасників системи. Першим видом критовалюти були Bitcoin, потім з’явилися такі його різновиди як Litecoin, Namecoin тощо.

Все більше користувачів займаються видобутком монет криптовалюти – майнінгом. З цією метою постає необхідність в розробці програмного забезпечення калькулятора криптовалют, яке дозволить розрахувати доходність видобутку монет в залежності від багатьох факторів.

Отже, метою роботи є розробка програмного забезпечення аналізу криптовалют.

Для досягнення поставленої мети, були поставлені і вирішені наступні **задачі:**

1. Детально ознайомитись криптовалютами Bitcoin та Litecoin. Вивчити принципи та характер роботи онлайн-калькуляторів.
2. Розробити алгоритм задачі, а також інформаційне забезпечення, визначитись з вхідною та вихідною інформацією.
3. Реалізувати програмне забезпечення калькулятору криптовалют:
   3.1. Передбачити динамічне оновлення курсів криптовалют з мережі Інтернет – ресурс **apirone.com**.
   3.2. Розрахувати доходність криптовалют для алгоритму SHA-256.
   3.3. Реалізувати 3D-графіку виведення статистичної інформацію, використовуючи OpenGL.

Об’єктом нашого дослідження є процес розрахунку доходності від майнінгу криптовалюти bitcoin.

Предмет дослідження – інформаційна підтримка процесу розрахунку доходності криптовалют та аналіз курсів криптовалют в часі.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика та види криптовалют

Криптовалюта (англ. Cryptocurrency) – вид цифрової валюти, емісія та облік якої засновані на криптографічних методах, наприклад методі захисту Proof-of-work [1] і асиметричного шифрування, а функціонування системи відбувається децентралізовано в розподіленої комп'ютерної мережі. Станом на червень 2013 програмне забезпечення всіх кріптовалют базувалося на відкритому вихідному коді системи Bitcoin. У криптовалют за замовчуванням не передбачено примусового повернення платежів, однак є можливості угод за участю посередника, коли для завершення або скасування угоди потрібна згода всіх трьох або довільних двох сторін, кошти не можуть бути примусово заморожені або вилучені без доступу до приватного ключу власника, проте учасники угоди можуть добровільно тимчасово взаємно блокувати свої кошти в якості застави. Як правило, є верхня межа загального обсягу емісії. Однак у деяких криптовалют, таких як PPCoin, Novacoin, Sifcoin та інших відсутня фіксована верхня межа загального обсягу емісії і можлива як емісія за рахунок наявних накопичень, так і демісія шляхом обов'язкового знищення невеликої фіксованої суми в кожній транзакції. Всі існуючі на даний момент кріптовалюти використовуються псевдонимність – всі транзакції публічні, але прив'язки до конкретної людини за замовчуванням немає, однак особа користувача може бути встановлена, якщо відома необхідна додаткова інформація. Ведеться розробка Zerocoin, де планується замінити псевдонимність на анонімність. Криптографія з метою конфіденційних платежів почала використовуватися з 1990 року в системі DigiCash Девіда Чома, компанія якого збанкрутувала в 1998 році. Однак, його платіжна система була централізованою, а вперше термін «криптовалюта» почав використовуватися після появи піринговой платіжної системи Bitcoin, яка була

розроблена в 2009 році людиною або групою осіб під псевдонімом Сатосі Накамото і використовує хешування SHA-256 і систему proof-of-work. Пізніше з'явилися інші незалежні від Bitcoin криптовалюти, використовувані різновидом Bitcoin: Namecoin (децентралізована DNS, яка використовує однойменну криптовалюту для реєстрації внутрішніх доменів. Bit), Litecoin (використовує хешування Scrypt, збільшена верхня межа загальної емісії, зменшено час підтвердження транзакцій), PPCoin (використовує гібридний механізм proof-of-work/proof-of-stake, не має верхньої межі на загальний обсяг емісії), Novacoin (аналогічна PPCoin, але використовує scrypt і зменшені коефіцієнти, пов'язані з емісією). Також було створено множину інших різновидів, але більшість з них не несуть у собі нічого нового (або є точною копією Bitcoin, або відмінності обмежуються тільки значеннями межі і швидкості емісії або алгоритмом хеш-функції) і не отримали широкого розповсюдження. Більшість таких різновидів валют з'явилося на тлі двох великих, що супроводжуються підвищеною увагою з боку засобів масової інформації, бульбашок на ринку Bitcoin в 2011 і 2013 роках.

Наведемо характеристику найбільш розповсюджених криптовалют (табл. 1.1).

Таблиця 1.1

Характеристика криптовалют

Валюта	**Код**	**Рік появи**	**Сайт**	**Хеш**
Bitcoin	BTC	2009	bitcoin.org	SHA-256
Litecoin	LTC	2011	litecoin.org	Scrypt
Namecoin	NMC	2011	dot-bit.org	SHA-256
PPCoin	PPC	2012	ppcoin.org	SHA-256
Quark	QRK	2013	qrc.cc	(blake, Bmw, Keccak, Skein)

Найпопулярнішими крипто валютами є Bitcoin та Litecoin. Розглянемо їх більш детально.

Bitcoin.

Bitcoin (від англ. Bit – біт і coin – монета) – пірінгова система електронної готівки, що використовує однойменну цифрову валюту, яку часто називають криптовалютой або віртуальною валютою [2]. Мережа повністю децентралізована, не має центрального адміністратора або якого-небудь його аналога. Bitcoin можуть використовувати для оплати товарів або послуг у продавців, готових їх приймати. Є можливість обміну на звичайні гроші через спеціалізовані майданчики для торгів або обмінники. Одна з особливостей – емісія нових bitcoin. Вона децентралізована, лімітована за обсягом і часом, що розподіляється випадково серед бажаючих, які використовують обчислювальні потужності свого обладнання для захисту платіжної системи методом proof-of-work від повторного витрачання коштів. Діяльність з обслуговування системи з можливістю отримати винагороду у формі емітованих bitcoin і комісійних зборів отримала назву майнінг (від англ. Mining – видобуток корисних копалин). Базовим елементом цієї платіжної системи є програма-клієнт з відкритим вихідним кодом. За допомогою мережевого протоколу прикладного рівня запущені на множині комп'ютерів клієнти з'єднуються між собою в тимчасову мережу. Для забезпечення функціонування та захисту системи використовуються криптографічні методи.

Платіжний засіб, що використовується в системі Bitcoin, являє собою цифрові монети - криптографічний сутність, що відповідає певним вимогам [2]. Котирування bitcoin заснована на довірі до нього, формується виключно балансом попиту та пропозиції, не прив'язана до якої-небудь валюті або іншому активу. На відміну від фіатних грошей, bitcoin не має органу (центробанку або держави), який би прагнув забезпечити ліквідність на заданому рівні, зобов'язався сам і / або зобов'язував інших приймати оплату в bitcoin-монетах або міг би штучно знизити його купівельну спроможність шляхом додаткової емісії. Bitcoin є електронною готівкою, а не борговим зобов'язанням емітента, що відрізняє його від традиційних електронних грошей і безготівкових розрахунків. Часто стверджується, що обмеження емісії є захистом від інфляції.

Ряд авторів вважають, що обмежена кількість bitcoin не є достатньою умовою для гарантування тенденції зростання курсу, так як ще однією необхідною умовою для цього є збільшення обсягу пропозиції товарів і послуг за bitcoin і сервісів, пов'язаних з ним. Тобто неспекулятивна цінність bitcoin безпосередньо залежить від обсягу тільки тих товарів і послуг, які можна буде за них придбати, а не загальносвітової товарної маси. Емісія та обіг bitcoin повністю децентралізовані, не залежать від будь-якого регулюючого органу, обсяг емісії відомий заздалегідь. Дані про переміщення та емісії bitcoin зберігаються в розподіленій базі даних. Bitcoin – монети можуть бути відправлені будь-якому іншому користувачеві системи. При цьому можна використовувати будь-які дробові суми з точністю до восьмого знака після десяткової коми. Усі транзакції знаходяться у відкритому доступі, але без розкриття інформації про реального власника. Кожен користувач може створити собі необмежену кількість адрес. Секретні ключі асиметричних пар ключів зберігаються у файлі гаманця wallet.dat, а відповідні їм публічні ключі використовуються для формування bitcoin-адрес. Гіпотетично є ймовірність того, що ланцюжок блоків буде анульовано і в системі головною буде визнана інший ланцюжок блоків. Імовірність такої події різко знижується з ростом довжини ланцюжка. Але якщо контролювати більше половини обчислювальної потужності всієї мережі, то така підміна можлива для будь ланцюжка, що гіпотетично дозволяє реалізувати подвійну трату одних і тих же bitcoin. Принцип тимчасової мережі і відсутність адміністративного центру унеможливлює державне чи приватне регулювання системи, а також маніпуляції із зміною сумарної кількості bitcoin. Емісія здійснюється автоматично: нові bitcoin – монети в якості винагороди отримують відносно випадковим чином ті, хто використовує обчислювальні потужності свого обладнання для підтримки роботи системи Bitcoin (для створення нових блоків бази). Обсяг емісії алгоритмічно обмежений так, щоб загальна кількість емітованих bitcoin не перевищило 21 мільйони. Спочатку розмір винагороди за кожен створений блок становив 50 bitcoin-монет. Після формування кожних 210000 блоків (приблизно раз на 4 роки) розмір

винагороди буде зменшуватися вдвічі. 28 листопада 2012 відбулося перше зменшення емісійної складовою нагороди з 50 до 25 bitcoin. На 6930000 блоці (приблизно в 2131) емісія буде зупинена зовсім (розмір винагороди 50 → 25 → 12.5 → ... → 0). Формування блоків продовжиться і далі, але за них вже не буде нараховуватися винагороду емітованими bitcoin. Система передбачає можливість стягувати комісію за обробку транзакцій з учасників. На сьогодні сплата такої комісії можлива в добровільному порядку, але не є обов'язковою. Передбачається, що коли винагорода за новий блок у формі емісії істотно скоротиться, основним джерелом стимулювання стануть комісійні збори. Діяльність по створенню нових блоків з можливістю отримати винагороду у формі емітованих bitcoin і комісійних зборів отримала назву «Майнінг» (від англ. Mining – видобуток корисних копалин). Продукція, обчислення потрібні для забезпечення захисту від повторного витрачання одних і тих же bitcoin, а зв'язок майнінгу з емісією стимулює людей надавати обчислювальні потужності і підтримувати роботу мережі. Майнінгом можна займатися як поодинці, так і спільно, скориставшись послугами спеціалізованих веб-служб, які називають «пулами (від англ. Pool – загальний фонд). Користувачі надають пулу свої обчислювальні потужності. Особливість задачі дозволяє застосувати максимальне розпаралелювання обчислень, коли кожен учасник шукає свій варіант вирішення без ув'язки його результатів з рішеннями інших. У свою чергу, пул, розподіляє отримані ним bitcoin між користувачами, відповідно до встановлених власником пулу правилами. Основна причина об'єднання в пули – зменшення ризику тривалого неотримання нагороди. Імовірність отримання нагороди в довільний десятихвилинний період приблизно дорівнює співвідношенню його обчислювальної потужності до обчислювальної потужності всієї мережі. І якщо це співвідношення дуже маленьке, то ймовірність отримання нагороди навіть за тривалий проміжок часу також буде низькою. У перших версіях клієнта була кнопка «згенерувати нові біткойни», але після створення програмного забезпечення для майнінгу за допомогою центрального процесора виявився нерентабельним через занадто малу

ймовірність отримати винагороду, і кнопку прибрали. Передача bitcoin-монет здійснюється безпосередньо, без посередництва будь-яких фінансових організацій. Скасування стандартних транзакцій неможлива, однак можливе використання мультіпідписів, в тому числі для угод за участю арбітра. Немає обов'язкової комісії, однак комісія може бути сплачена добровільно для прискорення обробки операцій.

З Bitcoin стикаються як IT-фахівці, так і менш кваліфіковані користувачі. У зв'язку з цим, з одного боку, сформувалася система побутових термінів, використовувана звичайними користувачами, а з іншого боку, сформувалася система термінів для фахівців, яка в основному виходить від розробників Bitcoin-qt і Bitcoind. Розбіжності стосуються найбільш часто використовуваних термінів. Перелічимо їх:

- BTC – скорочена назва одиниць обліку. Використовується замість слова «Bitcoin» для однозначного зазначення, що мається на увазі сама цифрова валюта, а не мережа, набір алгоритмів або яка-небудь інша сутність, що відноситься до даної тематики.
- Bitcoind – програма (демон), в якій реалізований протокол Bitcoin, використовувана допомогою командного рядка або віддаленого виклику процедур (JSON - RPC).
- Bitcoin-qt – перша програма з графічним інтерфейсом Qt, сумісна з bitcoind. Bitcoin можна також використовувати і через інші програми.
- Пара ключів – публічний і приватний ключ. Використовується для генерації адреси та підписування транзакції на переклад BTC.
- Адреса – є ідентифікатором, що містить близько 33 алфавітно-цифрових символів. Використовується як для отримання, так і для відправки BTC. Являє собою 160-бітний хеш від відкритого ключа ECDSA ключової пари.
- Гаманець – використовується для особистого сховища BTC (account) або всі акаунти всередині wallet.dat. Account – поняття в протоколі Bitcoin для спрощення створення онлайн-сервісу за допомогою bitcoind. Містить деяку

кількість ключових пар і службову інформацію. Дані про акаунти і адресах зберігаються у файлі « wallet.dat». У Bitcoin-qt акаунти відображаються як мітки. Не слід плутати гаманець як весь wallet.dat, акаунт і адресу.

- Транзакція – запис про переведення BTC з однієї групи адрес (0 і більше) на іншу групу адрес (1 і більше). Містить підписаний відправником хеш транзакції за допомогою якої відправник раніше отримав BTC та адреси одержувачів BTC.
- Блок-запис у ланцюжку блоків (базі даних), яка містить в собі множину очікуваних підтверджень транзакцій і підтверджує їх.

База даних публічно зберігає в незашифрованому вигляді інформацію про всі транзакції, що підписуються за допомогою асиметричного шифрування. Для запобігання багаторазової витрати однієї і тієї ж суми використовуються мітки часу, реалізовані шляхом розбиття бази даних на ланцюжок спеціальних блоків, кожен з яких, в числі іншого, містить в собі хеш попереднього блоку і свій порядковий номер. Кожен новий блок здійснює підтвердження транзакцій, інформацію про яких містить і додаткове підтвердження транзакцій у всіх попередніх блоках ланцюжка. Для зменшення розміру БД використовується деревоподібне хешування. Для більш наочного пояснення механізму роботи платіжної системи Сатосі Накамото ввів поняття «цифрова монета», визначивши його як ланцюжок цифрових підписів. Виходячи з даного визначення, кожна монета має свій власний номінал. Кожній bitcoin-адресі може зіставлятися будь-яка кількість монет. За допомогою транзакцій монети можна ділити і об'єднувати, при цьому їх сумарний номінал за вирахуванням комісії зберігається.

На рис. 1.1. наведено схему роботи з транзакціями при добутку Bitcoin.

Коли один користувач передає якусь суму іншому користувачеві, він створює нову транзакцію, яка містить хеш попередньої транзакції, підписаний ним, і публічний ключ наступного власника. Потім ця інформація широкомовною запитом відправляється в мережу.

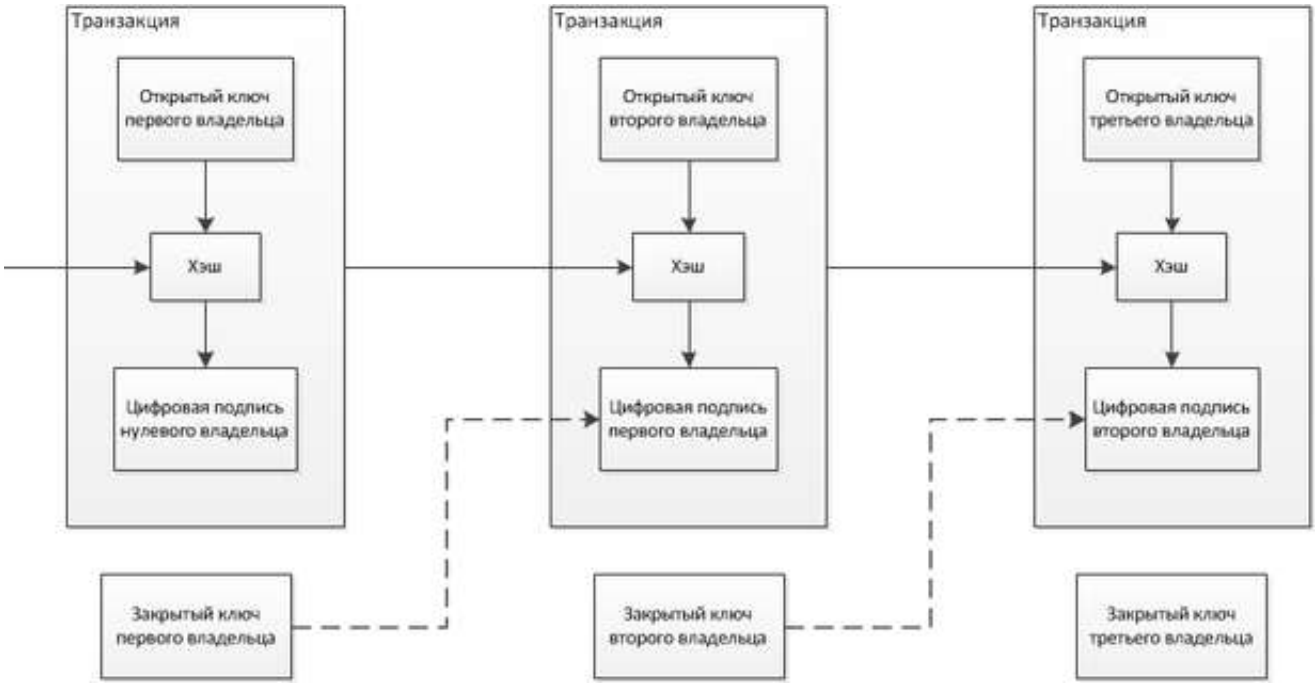


Рис. 1.1. Схема роботи з транзакціями

Інші вузли мережі перевіряють підписи, перш ніж прийняти транзакцію до обробки. Транзакції підтримують множинні входи (результати попередніх транзакцій) і виходи (вказівки про одержувачів). У загальному випадку транзакція може містити довільну кількість виходів (можливі випадки, коли необхідно передати кошти кільком одержувачам за допомогою однієї транзакції, що дозволить заощадити на комісійних зборах). Транзакція також може містити множину входів, які можуть бути навіть співпадати з bitcoin-адресами. Таке може мати місце, коли було кілька вхідних транзакцій на одну адресу. Кожна перша і тільки перша транзакція в блоці не має входів і зараховує винагороду за створення даного блоку. Така транзакція повинна отримати 120 підтверджень, щоб отримані за допомогою неї bitcoin могли бути використані. Значення з усіх входів підсумовуються, і сума розподіляється по виходах. Різниця між сумою на вході і сумою на виході вважається комісією за здійснення транзакції. Розмір винагороди, що зараховується першою транзакцією, є сумою всіх комісій у транзакції, включених до блоку, і фіксованого значення, спочатку рівного 50 і зменшується вдвічі кожні 210000 блоків. Транзакції обов'язково містять вказівки про одержувачів, наприклад, bitcoin-адреси чи інші умови . Більшість транзакцій, що мають входи, мають

мінімум два виходи: з покажчиком одержувача монети з номіналом, рівним відправленій сумі, і покажчика на відправника для «здачі» – монети з номіналом, який залишився від сумарного номіналу на вході за вирахуванням комісії. «Bitcoin-qt » відправляє кожну здачу на нову bitcoin -адресу з резерву заздалегідь створених і прихованих від користувача адрес. Інформація про те, яка саме монета є здачею, відсутня в базі даних.

Litecoin.

Litecoin (від англ. Lite – «легкий», англ. Coin – «монета») – різновид Bitcoin, пірінгова електронна платіжна система, що використовує однойменну криптовалюту. Створення та передача Litecoin ґрунтується на протоколі без централізованого адміністрування, заснованому на технології Bitcoin. Програма має відкритий вихідний код. Litecoin замислювався розробниками, як еволюція Bitcoin і має ряд відмінностей від нього. Станом на 16 лютого 2014 1 LTC коштує приблизно 16 USD на біржі BTC-E. Litecoin - друга за величиною криптовалюта в світі з ринковою капіталізацією близько 410 мільйонів доларів США. Litecoin можуть використовуватися для обміну на bitcoin або звичайні гроші в обмінниках, а також для електронної оплати товарів/послуг у продавців, готових їх приймати. Для забезпечення функціонування та захисту системи використовуються криптографічні методи. У мережах Bitcoin і Litecoin дані записуються ідентичним чином - у вигляді пов'язаного ланцюжка блоків, кожен з яких містить хеш попереднього. Відмінності лише в хеш-функціях і заданому середньому часом знаходження блоку мережею.

У мережах Bitcoin і Litecoin транзакції здійснюються за адресами. Адреси Bitcoin складаються з 27-34 символів і починаються з 1 або 3. Адреси Litecoin складаються з 33 символів і завжди починаються з літери L.

Для підтримки працездатності мережі, а також для забезпечення необхідного рівня захищеності (зокрема для запобігання можливості атаки «Double Spending») використовується механізм циклічного хешування. У разі, якщо числове значення хешу заголовка блоку дорівнює або нижче певної мети, створюється новий блок. В іншому випадку, змінюється блок службової

інформації в заголовку і хеш перераховується. Коли варіант знайдений, вузол розсилає отриманий блок іншим підключеним вузлам. Інші вузли перевіряють блок. Якщо помилок немає, то блок вважається доданим в ланцюжок і наступний блок повинен включити в себе його хеш. Результат хешування практично непередбачуваний. Таким чином, ймовірність створити новий блок для кожного окремо взятого користувача дорівнює відношенню кількості хешів в секунду (виражається звичайно в KH/s) обчислюваного на його обладнанні до кількості обчислюваних хешів в секунду у всій мережі. Той, хто створив новий блок отримує винагороду з деякої кількості нових монет. Процес використання ресурсів свого обладнання для виконання цих обчислень (створення нових блоків бази) називається майнінг. За знаходження нового блоку в мережі встановлена винагорода, спочатку рівна 50 LTC і зменшена вдвічі за кожні 840 000 блоків. Для доказу виконання роботи Bitcoin використовує хеш- функцію SHA256, що робить майнінг Bitcoin надзвичайно розпаралелювальним завданням. Litecoin використовує scrypt як доказ виконання роботи. Хеш-функція scrypt використовує SHA256 як підпрограму, покладаючись на велику кількість арифметичних обчислень, але також вимагаючи наявності швидкого доступу до великих обсягів пам'яті. Це робить запуск декількох екземплярів scrypt на пристрої сучасної відеокарти трохи більш складним завданням. Це також означає, що вартість виробництва спеціалізованого обладнання для майнінгу на інтегральних схемах спеціального призначення буде значно вище, ніж вартість виробництва подібних пристроїв для SHA256. Оскільки сучасні GPU володіють великими обсягами пам'яті, вони більшою мірою придатні для майнінгу Litecoin, проте їх перевага в порівнянні з CPU є менш значною, ніж чим у випадку з Bitcoin (перевага в 10 разів проти 20 для Bitcoin) [3]. Параметри функції scrypt використовувані Litecoin (N = 1024, p = 1, r = 1) дозволяють користувачам Litecoin запускати клієнт в багатозадачному режимі, не зачіпаючи продуктивність системи. Ці параметри, за твердженням Коліна Персиваля, творця scrypt, також зменшують ефективність використання ASIC приблизно в 10 разів. Так як вірогідність

створення нового блоку і отримання нагороди залежить від обчислювальної потужності обладнання користувача, то із зростанням кількості майнер і їх сумарної продуктивності для звичайного користувача шанс дуже невисокий. Щоб підвищити ймовірність отримання нагороди, майнери об'єднують свої обчислювальні потужності в пули. У разі успіху нагорода розподіляється між учасниками.

Складність обчислення Litecoin підбирається таким чином, щоб в середньому один блок генерувався 2,5 хвилини, що в чотири рази швидше, ніж Bitcoin, що дозволяє швидше отримувати підтвердження транзакцій. Транзакція, як правило, вважається завершеною після 6 блоків, або 15 хвилин.

Емісія Litecoin алгоритмічно обмежена. Максимальна кількість litecoin, яке увійде в обіг перевищує максимальне число bitcoin в 4 рази (84 мільйони проти 21). Первісна нагорода за кожен блок дорівнює 50 litecoin. Швидкість генерації litecoin зменшується вдвічі за кожні 840 000 блоків, що в 4 рази більше блоків, ніж з Bitcoin. Оскільки блоки litecoin формуються в 4 рази швидше, ніж блоки bitcoin, це означає, у них темпи емісії та винагороди будуть подібні. Наприклад, до 2020 року близько 3/4 всіх litecoin будуть згенеровані.

Базовим елементом платіжної системи Litecoin є програма-клієнт з відкритим вихідним кодом. За допомогою мережевого протоколу прикладного рівня запущені на множині комп'ютерів клієнти з'єднуються між собою в однорангову пірингову мережу. Емісія litecoin алгоритмічно обмежена. Litecoin передбачає анонімне володіння і переказ грошових коштів. Пірінгова мережа Litecoin регулює транзакції, баланси і емісію за допомогою scrypt, докази виконання роботи схеми. При виявленні користувачем досить маленького хеш-значення створюється блок. Подібно Bitcoin, Litecoin використовує принцип тимчасової мережі і відсутність адміністративного центру, що унеможливлює державне регулювання і маніпуляції курсом шляхом зміни кількості Litecoin у зверненні. Процеси емісії і передачі організовані аналогічно з Bitcoin, але відрізняються параметрами і деякими деталями. В даний час Litecoin обмінюється як на звичайні гроші, так і на Bitcoin, в основному на онлайн-майданчиках.

За вимогою до хешів блоків в мережі Litecoin відповідає параметр, який називається «складність». Так як обчислювальні потужності мережі непостійні, цей параметр перераховується клієнтами мережі таким чином, щоб один блок генерувався приблизно 2,5 хвилини. Атака "Time Warp" заснована на недолікі, притаманному Bitcoin і всім його різновидам (у тому числі Litecoin). Недолік полягає в тому, що при перерахунку складності неправильно обробляється останній блок. Зловмисник може неодноразово спробувати вирішити останній блок перед перерахунком, приписавши йому тимчасову позначку, на дві години перевищує поточний час, тим самим зменшуючи складність приблизно на 0,5 %. Через цей недолік, ці додаткові дві години не враховуються при наступному перерахунку. Як тільки складність досить сильно впаде, можна приступати до майнінгу "швидких" litecoin. Таким чином, зловмисник, що володіє 51 % обчислювальної потужності мережі, може знизити складність до одиниці, і почати добувати монети. Для мережі Bitcoin дана атака практично нездійсненна, тому що ймовірність вирахувати останній блок перед перерахунком кожні два тижні при поточній потужності мережі та складності майже відсутня.

1.2. Вхідна та вихідна інформація

Вхідна інформація та її коректність є найважливішими параметрами для отримання точних результатів розрахунку. Вона являє собою сукупність всіх даних, які використовуються для розрахунку вихідної інформації. Вхідна інформація повинна бути точною та вичерпною.

Для правильного проектування програмного забезпечення необхідно визначитись, які вхідні дані будуть у нашому розпорядженні та необхідні для вичерпної інформації про оцінку. Вхідна інформація представляє собою сукупність показників, характеристик та інших значень, над якими необхідно проводити аналіз, здійснювати їх обробку тощо.

Головна мета – розробка програмного забезпечення аналізу криптовалют.

Вхідну інформацію, яка необхідна для розрахунку доходності наведемо в табл. 1.2. Даний калькулятор розроблено для криптовалюти Bitcoin за алгоритмом SHA-256.

Таблиця 1.2

Вхідна інформація для розробки калькулятору окупності криптовалюти BTC

№ п/п	Назва	Характеристика
1	Розрахунковий період	Час, доба, тиждень, місяць, рік
2	Швидкість майнера	kH/s, MH/s, GH/s (кіло-, мега-, гіго- хеши в секунду)
3	Складність майнінгу	Залежить від складності мережі
4	Винагорода за блок	Кількість монет BTC
5	Курс криптовалюти	USD/BTC – кількість доларів за 1 BTC
6	Споживча потужність	Кількість Ватт
7	Тариф	Вартість одного кВт/ч у гривнях
8	Комісія за пул	Відсотки за користування пулом
9	Комісія обміну на USD	Відсотки за обмін криптовалюти BTC на USD.

Саме ця вхідна інформація необхідна для розрахунку. Крім цього, необхідно накопичувати інформацію по курсам валют на певну дату, що можуть коливатися та змінюватися щодня. Вхідна інформація про курси криптовалют наведено в табл. 1.3.

Таблиця 1.3

Перелік і опис вхідних даних таблиці "Курси криптовалют"

№ п/п	Назва вхідного повідомлення	Ідентифікатор	Форма представлення	Термін і частота надходження
1	Код	ID	Таблична	При надходженні нових даних
2	Дата операції	Data_Oper	Таблична	
3	Курс USD/BTC	BTC_USD	Таблична	
4	Курс BTC/LTC	LTC_BTC	Таблична	
	Курс USD/LTC	LTC_USD	Таблична	

Ця вхідна інформація повинна постійно поновлюватися з мережі Інтернет. Повна інформація щодо курсів криптовалют розташована на сайті **https://apirone.com**.

Вихідна інформація є результатом розрахунку доходності криптовалют.

Її представлено у табл. 1.4.

Таблиця 1.4

Результат розрахунку доходності криптовалют

№ п/п	Назва	Характеристика
1	Кількість монет BTC	Кількість монет, які заробить майнер за вказаний період
2	Комісія пулу	Комісія за використання пулу у монетах BTC
3	Кількість монет без BTC без комісії	Кількість монет, які заробить майнер за вказаний період без врахування комісії
4	Комісія за обмін BTC на USD	Комісія за обмін у доларах (USD)
5	Отримані USD	Кількість зароблених доларів
6	Витрати на електроенергію	Витрати на електроенергію у гривнях.

Немає складності перевести криптовалюту у довільну валюту, знаючи відповідні курси.

На рис. 1.2. наведемо сайт **https://bitinfocharts.com/ru/markets**, де знаходиться інформація щодо курсів криптовалют.

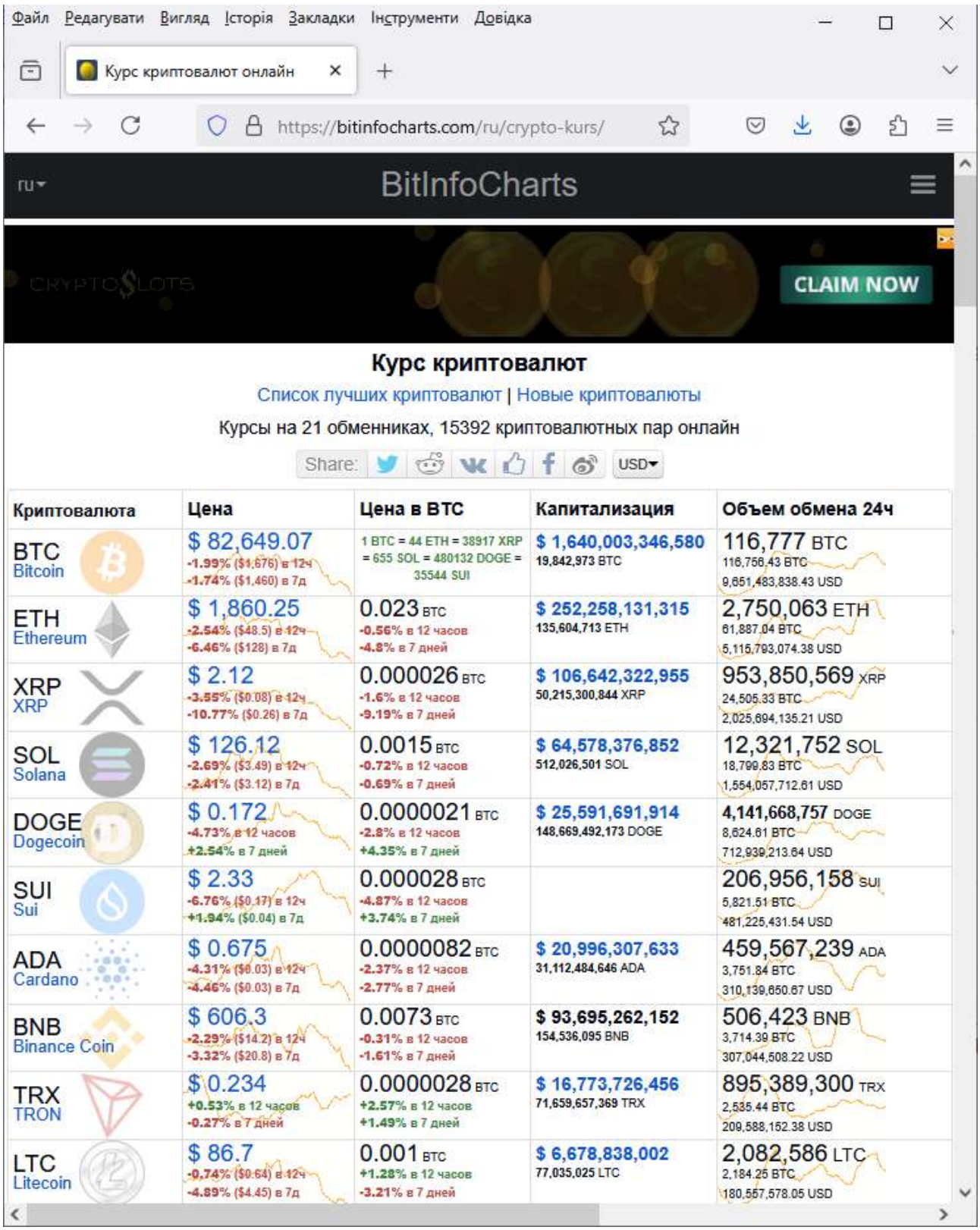


Рис. 1.2. Результат виведення сайту **https://bitinfocharts.com/ru/markets**

Висновки до розділу 1

Отже, у першому розділі детально розглянуті основні види криптовалют: BitCoin, LiteCoin тощо, а також наведено вхідну та вихідну інформацію. Слід сказати, що вхідна інформація повинна постійно поновлюватися з мережі Інтернет. Повна інформація щодо курсів криптовалют розташована на сайті **https://apirone.com**. Вихідна інформація в свою чергу є результатом розрахунку доходності криптовалют.

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

Розробка калькулятору окупності криптовалют є актуальною задачею. Недоліком онлайн-калькуляторів є відсутність наочного представлення даних інформації та одноразове обчислення, тобто відсутнє накопичення інформації.

При розробці калькулятора криптовалют за основу взято валюту bitcoin з алгоритмом шифрування SHA-256.

Розглянемо адресацію та структуру блоків з використанням даного алгоритму.

Передача bitcoin відбувається з виставленням умов для одержувача. Скористатися сумою зможе той, хто зможе виконати всі умови. Стандартним умовою є використання bitcoin-адреси. Але умови можуть бути й іншими, наприклад, послідовне використання декількох bitcoin-адрес або прив'язка до певної IP адреси. Bitcoin-адреса генерується на основі перетворення публічного ключа користувача. Секретні (приватні) ключі авторизують власника. Bitcoin-адреси не містять персональної інформації про власника. Людина може мати багато адрес, створюючи їх за власною ініціативою, для чого навіть не потрібне підключення до інтернету. Створення адреси лише для однієї транзакції або одного кореспондента допомагає зберегти анонімність. Bitcoin-адресу в текстовій формі являє собою рядок Base 58 довжиною до 34 символів, що складається з букв латинського алфавіту і цифр, наприклад **175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W**. Існують варіанти представлення адрес у вигляді QR-кодів та інших двомірних штрих-кодів, придатних для зчитування мобільними пристроями.

На рис 2.1. наведемо ланцюг блоків.

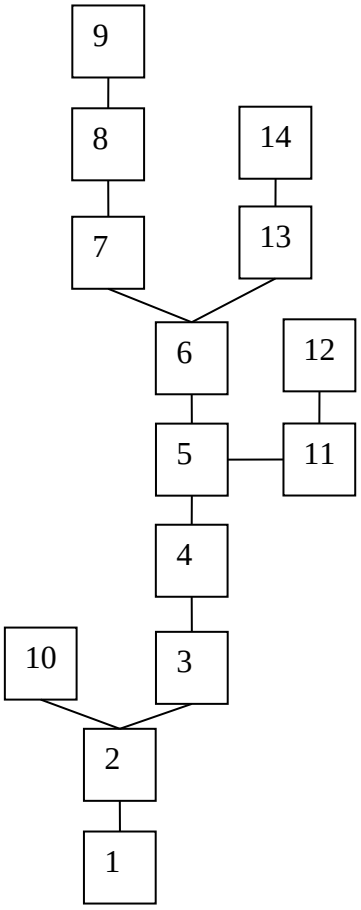


Рис. 2.1. Ланцюг блоків Bitcoin

Основна послідовність блоків 2-9 є найдовшою від початкового 1 до поточного. Побічні гілки (10-14) відсікаються. Будь-які транзакції не вважаються підтвердженими, поки інформація про них не буде згрупована в спеціальні структури – блоки. Структура та інформація в блоках підпорядковується заданими правилами і її можна швидко перевірити. Кожен блок завжди містить інформацію про один попередній блок. Це дозволяє всі існуючі блоки вибудувати в один ланцюжок, який представляє собою розподілену базу даних і містить інформацію про всіх операціях з bitcoin. Блоки одночасно створюють множину майнерів. Регулярно виникають ситуації, коли один і той же блок є попереднім для двох нових блоків. У кожному з нових блоків можуть зустрічатися як однакові транзакції, так і різні, тобто ввійшли тільки в один з них. Через деякий час з'являються чергові блоки, ланцюжок може роздвоїтися. Кожна з гілок рівноправна до тих пір, поки одна з них не отримає більш довге продовження. Зазвичай, при рівності довжини, перевага віддається тому ланцюжку, кінцевий блок якого з'явився раніше. Система

автоматично підтвердженим вважає більш довгий ланцюжок, не звертаючи увагу на час створення останнього блоку. Транзакції, що увійшли виключно в менш довгу гілку (у тому числі з виплати винагороди), втрачають статус підтверджених. Якщо це транзакція з передачі bitcoin, то вона може бути включена в черговий блок. Транзакції отримання винагороди за створення відсічених блоків не дублюються в іншій гілці, то є «зайві» bitcoin за відсічені блоки не отримують подальших підтверджень і «втрачаються». Таким чином, ланцюжок блоків містить історію володіння, з якою можна ознайомитися, наприклад, на спеціалізованих сайтах.

Якщо користувач спробує використовувати раніше витрачені bitcoin-монети знову, мережа не прийме його транзакцію як дійсну. Але в паралельних гілках блоків можуть перебувати транзакції, які по різному витрачають одні й ті ж початкові кошти. Ймовірність існування паралельних ланцюжків блоків вкрай мала і експоненціально зменшується з ростом довжини ланцюжка і кількості незалежних майнерів. Таким чином, чим більше підтверджень має транзакція, тим менш ймовірно скасування транзакції через відмирання її ланцюжка блоків. Однак при наявності у зловмисника контролю над досить великою часткою сумарної потужності майнінгу існує ненульова ймовірність «таємного» вибудовування довгих паралельних ланцюжків блоків. Після їх публікації в мережі головним буде визнаний більш довгий ланцюжок. Скасування ланцюжка блоків може призводити до визнання недійсними транзакцій навіть підтверджених кількома блоками і подальшої повторної витрати коштів. При наявності в одних руках понад 50% сумарної потужності майнінгу така ситуація можлива на будь-якому рівні підтвердження (атака «Double Spending» або «атака 51 %»). Якщо підконтрольна потужність менше 50%, то ймовірність успіху експоненціально знижується з кожним підтвердженням. Проведення успішної атаки не дозволяє:

– змінити розмір винагороди за генерацію блоку;
– отримати необмежену кількість bitcoin;
– знищити мережу;

– витратити bitcoin, які раніше не належали зловмиснику.

На початок 2013 року потужність мережі становила менше 25 THash/s, але за наступні 3 місяці виросла до 55 за рахунок масового розповсюдження спеціалізованих процесорів (ASIC), розроблених спеціально для майнінгу в мережі Bitcoin. У середині липня 2013 потужність мережі перевищила 210,46 THash/s. До вересня потужність перевищила 1000 THash/s , в жовтні потужність подвоїлася, а на 1 грудня 2013 перевищила 6000 THash/s [2]. При цьому користувач з найбільшою продуктивністю має менш 100 THash/s.

Розглянемо структуру блоку. Блок складається з заголовка і списку транзакцій. Тема блоку включає в себе свій хеш, хеш попереднього блоку, хеши транзакцій і додаткову службову інформацію. Першою транзакцією в блоці завжди записується генерація нових bitcoin-монет, які в разі успішної генерації блоку стануть нагородою користувачеві за створений блок. Далі йдуть всі або деякі з останніх транзакцій, які ще не були записані в попередні блоки. Створений блок буде прийнятий іншими користувачами, якщо числове значення хешу заголовка одно або нижче певної мети, величина якої періодично коригується. Якщо блок не задовольняє цілі, то змінюється блок службової інформації в заголовку і хеш перераховується. Зазвичай потрібна велика кількість спроб, так як результат хешування (функції SHA-256) практично непередбачуваний. Коли варіант знайдений, вузол розсилає отриманий блок іншим підключеним вузлам. Інші вузли перевіряють блок. Якщо помилок немає, то блок вважається доданим в ланцюжок і наступний блок повинен включити в себе його хеш. Величина цільового числа, з яким порівнюється хеш, коригується через кожні 2016 блоків. Заплановано, що вся мережа буде витрачати на генерацію одного блоку приблизно 10 хвилин, на 2016 блоків – близько двох тижнів. Якщо 2016 блоків сформовані швидше, то мета трохи зменшується і досягти її стає важче, в іншому випадку мета збільшується. Зміна складності обчислень не впливає на надійність мережі Bitcoin і потрібна лише для того, щоб система генерувала блоки майже з постійною швидкістю, що не залежить від потужності мережі.

Bitcoin передбачає тільки одну можливість для додаткової емісії – нові bitcoin-монети нараховуються в якості винагороди тому, хто згенерував черговий блок, який отримав 120 підтверджень. Спочатку емісія становила 50 bitcoin в кожному блоці . Після 2012-11-28 15:24:38 UTC значення зменшилося до 25 bitcoin, відповідно до алгоритму. На квітень 2013 року в обігу перебувало майже 11 мільйонів bitcoin, що складає більше половини їх максимально можливого граничної кількості в 21 мільйон. Бажають отримати можливо більшу винагороду прагнули задіяти якомога більші обчислювальні потужності. Особливість завдання дозволяла застосувати максимальне розпаралелювання обчислень. У силу специфіки будови, для цього добре підійшли графічні процесори (GPU) з невеликою додатковою програмою (в сотні разів продуктивніше CPU) і плати з FPGA (продуктивність порівнянна з відеокартами, але перевершують їх з енергоефективності). Потім були випущені спеціалізовані процесори (ASIC), що виконують тільки обчислення хешів для Bitcoin, більш продуктивні ніж GPU і FPGA.

За вимогу до хешів блоків відповідає спеціальний параметр, який називається «складність» . Так як обчислювальні потужності мережі непостійні, цей параметр перераховується клієнтами мережі через кожні 2016 блоків таким чином, щоб підтримувати середню швидкість формування розподіленої БД на рівні 2016 блоків на два тижні. Таким чином 1 блок повинен створюватися приблизно раз на десять хвилин. Після створення 210 000 блоків (на що потрібно приблизно 4 роки) удвічі зменшується розмір емісійного винагороди. На практиці, коли обчислювальна потужність мережі зростає – відповідні тимчасові проміжки коротше, а коли знижується – довші. Перерахунок складності з прив'язкою до часу можливий завдяки наявності в заголовках блоків часу їх створення. Вони записані в Unix-форматі і взяті з системних годин автора блоку (якщо блок створений в пулі, то із системних годин сервера цього пулу).

Для зменшення впливу фактора удачі і більш рівномірного і передбачуваного отримання bitcoin майнери використовують пули. Часто

виплати майнера розраховуються виходячи з відправлених ним пулу куль (shares) (блоків з хешем, який підійшов би при складності рівній одиниці). У середньому потрібно 232 операцій хешування для знаходження однієї кулі. Для знаходження блоку в середньому потрібна кількість куль, рівна поточній складності.

Існують 3 основних типи нарахування нагород:

- Proportional. Після знаходження пулом блоку нагорода ділиться пропорційно внеску кожного учасника.
- PPS. Винагороджуються кожна надіслана куля. Оцінюється як поточна винагорода за блок, поділена на поточну складність.
- Score. Оціночна система винагороди кулі, алгоритм визначається організатором пулу.

У цих типів нарахування є наступні популярні варіанти:

- SMPPS. Аналогічно PPS, але пул ніколи не передає користувачам більше, ніж реально отримав сам. Різниця між реальним отриманням нагороди пулом і винагородою кулі в PPS, якщо така є, компенсується поступово.
- ESMPPS. Аналогічно SMPPS, але зрівнює пріоритети винагороди постійним та новим учасникам пулу.
- RSMPPS. Аналогічно SMPPS, але першими в черзі на винагороду ставляться нові користувачі.
- PPLNS. Аналогічно Proportional, але поділ нагороди здійснюється пропорційно внеску в останні складність надісланих на пул куль, помноженому на N, де N зазвичай дорівнює 2.

При отриманні bitcoin-монет новий власник не може відразу ж передати їх. Для зменшення ймовірності подвійного використання, будь-яка транзакція повинна отримати деяку кількість підтверджень. Одним підтвердженням вважається один новий блок, починаючи з того, в якому упакована транзакція. Необхідна кількість підтверджень залежить від програми-клієнта, або від вказівок приймаючої сторони. Отримані за створення блоку bitcoin монети не використовуються, поки кількість підтверджень не досягне 120. Отримані від

інших користувачів bitcoin-монети клієнт «Bitcoin-qt» дозволяє використовувати після шести підтверджень. Різні онлайн-сервіси часто встановлюють свій поріг підтверджень.

В системі Bitcoin не передбачено обов'язкових комісійних зборів. Користувачі можуть добровільно включати в платіж довільну суму комісійного збору, подаючи на вхід транзакції більше коштів, ніж на вихід, що підвищує пріоритет обробки такої транзакції. Різні програми-клієнти мають свої правила щодо розміру та об'єкта комісійних зборів. Комісійний збір дістається вузлу, згенерованого блоку з транзакцією. Новий блок, який генерує користувач, може за своїм розсудом додавати в нього транзакції з черги. Наприклад, він може відібрати лише транзакції з комісійним збором.

У програмах-клієнтах була додана система, що організує записи транзакції таким чином, що користувач може локально видаляти дані зі своєї бази, які йому точно не знадобляться. Після того, як всі транзакції з якимись засобами були упаковані в блоки і підтверджені, попередні транзакції з цими засобами можна відкинути для економії місця на диску. Для того , щоб це можна було здійснити без зміни хешу блоку, транзакції хешуються за допомогою ТТН, і в заголовок блоку поміщається тільки результат даного хешування.

Зараз всі користувачі офіційного клієнтського програмного забезпечення після запуску програми в перший раз отримують повну базу даних (блоки без індексації та оптимізації) . Станом на березень 2013 її розмір становив понад 7 ГБ. Тема блоку має об'єм близько 80 байт. Оскільки блоки генеруються приблизно кожні 10 хвилин, то за рік буде накопичуватися близько 4,2 Мб заголовків блоків.

Програмне забезпечення мережевого вузла існує у двох видах: додаток з графічним інтерфейсом і фоновий додаток . В обох випадках воно може управлятися через програмний інтерфейс по протоколу JSON -RPC (RFC 4627).

Для розрахунку доходності криптовалюти Bitcoin за алгоритмом SHA-256 використовується наступна формула:

$$N = \frac{t \cdot R \cdot H}{D \cdot 2^{32}},$$

де N – дохід в монетах;

t – період майнінгу (добутку монет) в секундах;

R – винагорода за блок у монетах;

H – швидкість майнінгу (H/s – хешрейт в секунду);

D – складність майнінгу.

Комісія за використання пулу у bitcoin-ах N_{kp} розраховується наступним чином

$$N_{kp} = N \cdot \frac{k_p}{100},$$

де N – дохід у монетах;

k_p – комісія за використання пулу у %.

Комісія за обмін BTC на долари (USD) N_{ko} розраховується наступним чином

$$N_{ko} = N \cdot \frac{k_o}{100},$$

де N – дохід у монетах;

k_o – комісія за обмін BTC на USD у %.

Кількість отриманих USD після обміну N_{USD} розраховується так

$$N_{USD} = (1 - \frac{k_o}{100}) \cdot N \cdot k,$$

де

де N – дохід у монетах;

k – курс BTC-USD (кількість доларів за 1BTC);

k_o – комісія за обмін BTC на USD у %.

Крім вище наведених розрахунків з урахуванням комісії пулу та комісії обміну BTC на USD треба врахувати витрати на електроенергію, які витрачаються в процесі майнінгу bitcoin.

Витрати на електроенергію *B* розраховуються наступним чином

$$B = M \cdot k_{ч} \cdot tarif ,$$

де *M* – споживча потужність у кВт;

$k_{ч}$ – кількість годин майнінгу;

tarif – вартість одного кВт/ч у гривнях.

Для реалізації програмного забезпечення калькулятора криптовалют розробимо алгоритм, який представляє майбутню функціональність та послідовність дій програми. Основна частина програмного забезпечення розроблена на мові Visual C# 2019 з використанням технології ADO.Net для обробки баз даних. На мові C++ буде представлено 3D-графіку з використанням бібліотеки OpenGL.

На рис. 2.2. представимо алгоритм програми у вигляді блок-схеми.

На початку програми відбувається ініціалізація поточних налаштувань – шрифт колір вибраної інформації, а також останні дані, які вводив користувач стосовно розрахунку доходності криптовалют. Файл налаштувань має назву "nastr.ini" та знаходиться в поточному каталозі. Якщо такого файлу немає – його буде створено з початковими налаштуваннями.

На наступному етапі створюється головна форма програми, де відбувається підключення бази даних MS ACCESS crypto_valuta.mdb, використовуючи драйвери OleDB та технологію ADO.Net.

Після успішного підключення користувачеві надається можливість обирати опції меню розробленої програми. Це наступні опції:

1. <Статистика по курсам валют> – надається можливість динамічного оновлення інформації з мережі Інтернет та її запис до БД.
2. <Поточний курс> – виведення поточного курсу криптовалют.
3. <Графік> – виведення 2D-графіки по курсам криптовалют: BTC-USD, LTC-BTC, LTC-USD.
4. <Калькулятор> – розрахунок доходності від добутку криптовалюти bitcoin за вказаний період.

5. <Налаштування> – вибір шрифту та кольору для експорту та представлення табличних даних.

6. <Про програму> – виведення інформації про програму (About).

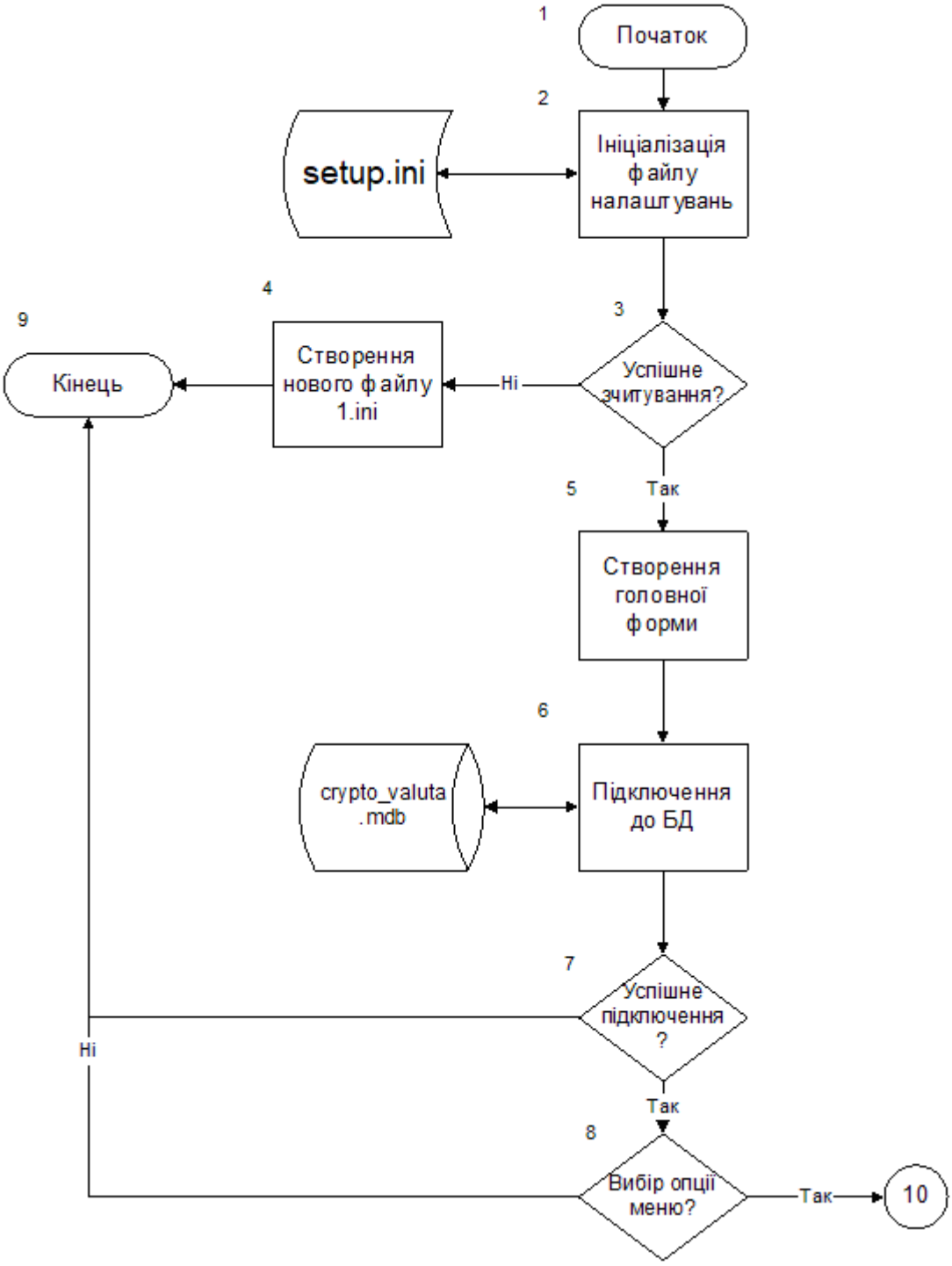


Рис. 2.2. Алгоритм вирішення задачі Calc на мові C#

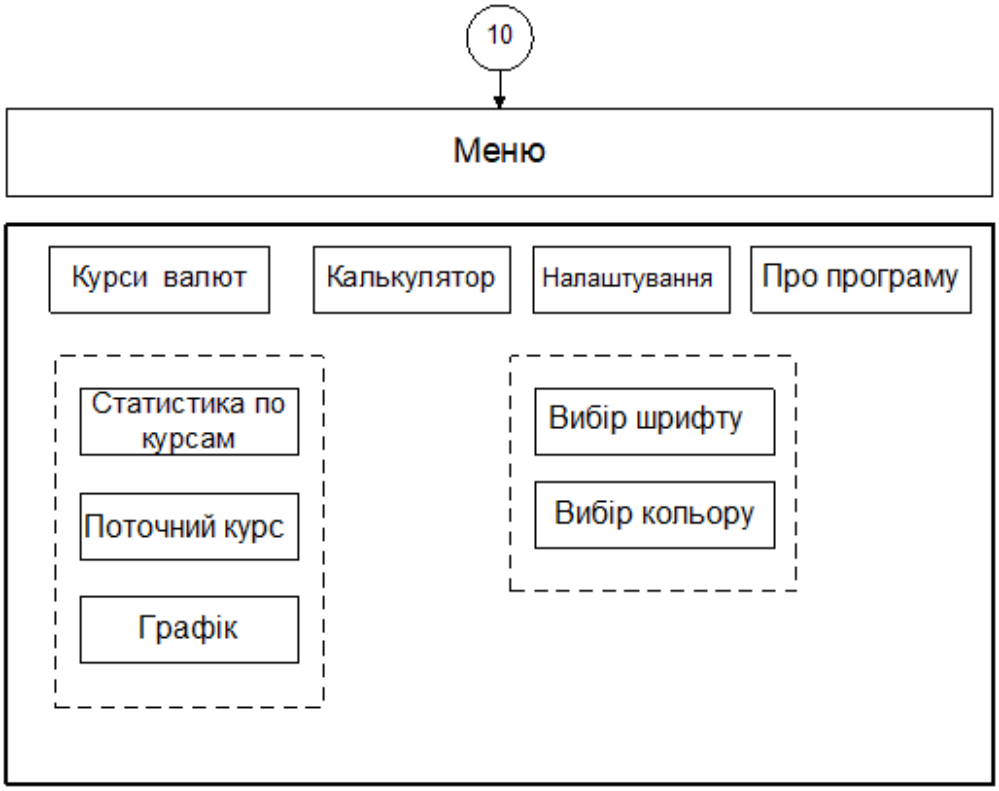


Рис. 2.2. Алгоритм вирішення задачі Calc на мові C# (продовження)

Крім укрупненої блок-схеми задачі доцільно зупинитися на алгоритмі завантаження даних курсів криптовалют з мережі інтернет – ресурс **apirone.com** (рис. 2.3.).

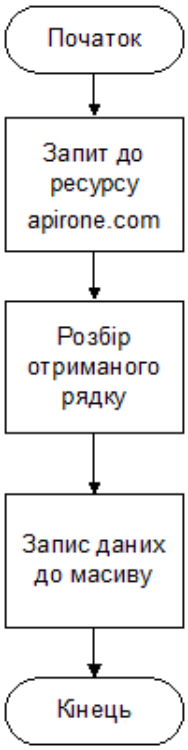


Рис. 2.3. Алгоритм завантаження даних курсів криптовалют з ресурсу apirone.com

На мові C++ розроблено проект Calc_3D, який виводить 3D-графіку по курсам криптовалют з використанням бібліотеки OpenGL.

На рис. 2.4. наведемо укрупнений алгоритм виведення 3D-графіки по курсам криптовалют.

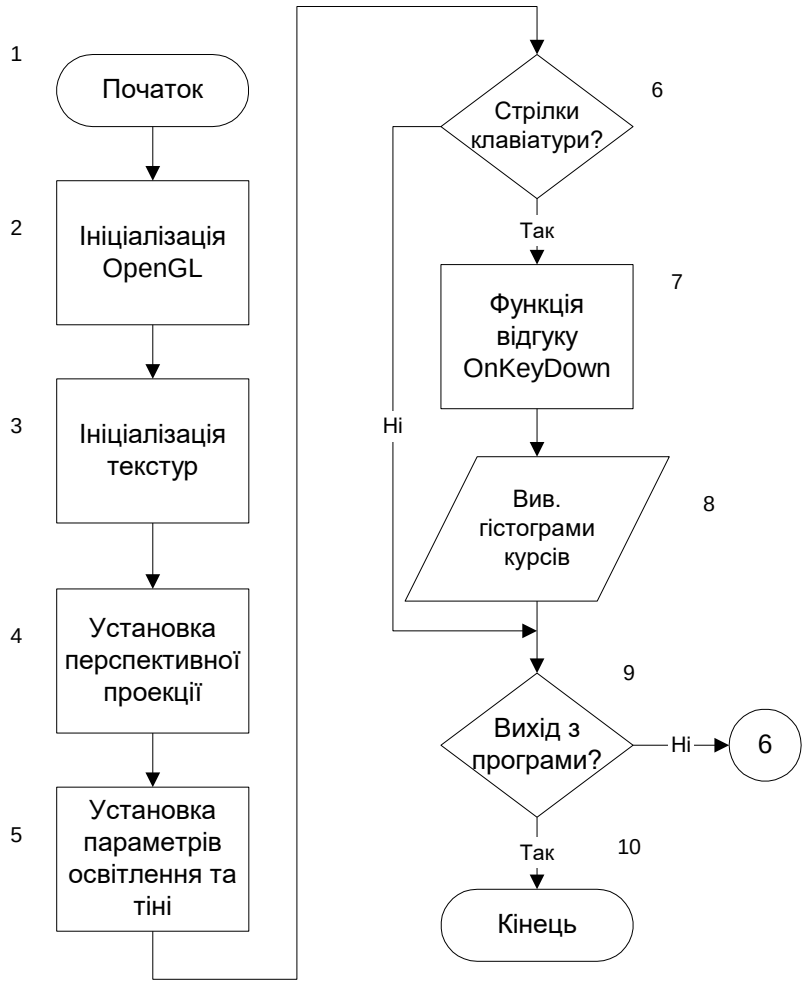


Рис. 2.4. Алгоритм вирішення задачі Calc_3D на мові C++

З рисунку видно, що при роботі з 3D-графікою встановлюються наступні ефекти:

1. Ініціалізація текстур для створення поверхні землі, по якій буде переміщуватись користувач.

2. Установка параметрів освітлення та тіні, які будуть створювати ефекти при виведенні гістограми курсів криптовалют.

3. 3D-графіка побудована за принципом камери та акторів, коли користувач може переміщуватись по сцені та спостерігати різні об’єкти.

Висновки до розділу 2

Отже, у другому розділі розроблено алгоритм розв’язання задачі аналізу криптовалют. Визначено основні показники для розрахунку окупності криптовалют: кількість монет, витрати на електроенергію, комісійні витрати тощо. Крім того визначено алгоритми двох задач: «Calc» та «Calc_3D».

РОЗДІЛ 3
ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Інформаційне забезпечення (ІС) - найважливіший елемент ІС та ІТ - призначено для відображення інформації, що характеризує стан керованого об'єкта і є основою для прийняття управлінських рішень. Інформаційне забезпечення включає сукупність єдиної системи показників:

- потоків інформації - варіантів організації документообігу;
- систем класифікації та кодування економічної інформації;
- уніфіковану систему документації;
- різні інформаційні масиви (файли), що зберігаються в машині і на машинних носіях і мають різну ступінь організації.

Інформаційне забезпечення призначене для відображення інформації, що характеризує стан керованого об'єкта і є основою для прийняття управлінських рішень. У програмах, що розроблюються, усі вхідні дані знаходяться у зовнішніх файлах. Для реалізації задач вся необхідна інформація з вхідними даними міститься у базі даних crypto_valuta.mdb (MS ACCESS), з якою і працює обране програмне забезпечення.

Особливістю інформаційного забезпечення є постійний зв’язок з мережею Інтернет. При виконанні програмного забезпечення з мережі Інтернет (ресурс **https://apirone.com**) постійно завантажується інформація щодо поточного курсу криптовалют bitcoin та litecoin відносно долара (USD). Ця інформація накопичується в БД. Доступ до мережі відбувається за допомогою класу **WebClient** мови C#.

Інформаційною основою задачі є таблиця Kursy бази даних crypto_valuta.mdb. Таблиця динамічно заповнюються даними з ресурсу мережі **https://apirone.com**.

Опишемо структуру таблиці БД Kursy (табл. 3.1).

Таблиця 3.1

Опис властивостей таблиці Kursy

№ п/п	Назва поля таблиці	Тип даних	Характеристика
1	ID	Лічильник (long)	PRIMARY KEY
2	Data_oper	date	Дата поточного курсу криптовалют
3	BTC_USD	double	Курс USD/BTC (скільки доларів коштує bitcoin)
4	LTC_BTC	double	Курс BTC/LTC (скільки bitcoin-ів коштує litecoin)
5	LTC_USD	double	Курс USD/LTC (скільки доларів коштує litecoin)

Крім того, створено клас з вхідними даними для розрахунку калькулятору окупності bitcoin. Наведемо цей клас:

```
public class Raschet
{
    public int kol_sec;
    public double speed;
    public int Hash;
    public double dif;
    public double nagr;
    public double kurs;

    public double Vatt;
    public double tarif;

    public double komis_pul;
    public double komis_obmen;
}
```

Опишемо дані цього класу необхідні для розрахунку (табл. 3.2).

Таблиця 3.2

Опис полів класу Raschet

№ п/п	Назва змінної	Тип даних	Характеристика
1	kol_sec	int	період майнінгу в секундах
2	speed	double	швидкість у хешрейтах в секунду
3	Hash	int	Кількість хешів в секунду: 1 kH/s – 1000 H/s 1 MH/s – 1000000 H/s 1 GH/s – 1000000000 H/s
4	dif	double	складність майнінгу
5	nagr	double	винагорода за блок у монетах
6	kurs	double	Курс USD/BTC (кількість доларів за bitcoin)
7	Vatt	double	Споживча потужність електроенергії (кількість Ватт)
8	tarif	double	Вартість одного кВт/ч у гривнях
9	komis_pul	double	Відсотки за користування пулом
10	komis_obmen	double	Відсотки за обмін криптовалюти BTC на USD

Даної інформації цілком достатньо для створення калькулятору окупності криптовалют та динамічного оновлення курсів з мережі Internet.

Висновки до розділу 3

Отже, інформації, яка описана у третьому розділі, цілком достатньо для створення калькулятору окупності криптовалют та динамічного оновлення курсів з мережі Internet. У даному розділі детально описані властивості таблиці Kursy, а також клас Raschet, призначений для розрахункових операцій.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ

4.1. Опис програмного забезпечення аналізу криптовалют

Програмне забезпечення калькулятора криптовалют розроблено на Visual Studio 2019 на мовах програмування C#, C++.

На мові C# розроблено саме калькулятор окупності криптовалют. Робота з базами даних відбувається на основі технології ADO.NET. Ключовим аспектом програмного забезпечення є взаємодія з мережею Internet для зчитування поточних курсів криптовалют. Це забезпечується за допомогою класу WebClient .

На мові C++ розроблено програму виведення 3D-графіки коливання курсів криптовалют. Робота з базами даних відбувається з використанням технології ADO.

Програмне забезпечення розроблене на мові C# виконано на Windows Forms та складається з таких форм:

1. **FormGl** – головна форма програми.
2. **Form1** – калькулятор окупності криптовалют.
3. **Form2** – виведення результатів розрахунку калькулятору окупності криптовалют.
4. **Form3** – виведення інформації про програму ("About").
5. **Form4** – статистика по курсам криптовалют.
6. **Form5** – поточний курс криптовалют.
7. **Form6** – графік по курсам криптовалют.
8. **Form7** – форма результатів експорту в HTML [додаток Б].

Головна форма програми виглядає наступним чином (рис. 4.1.):

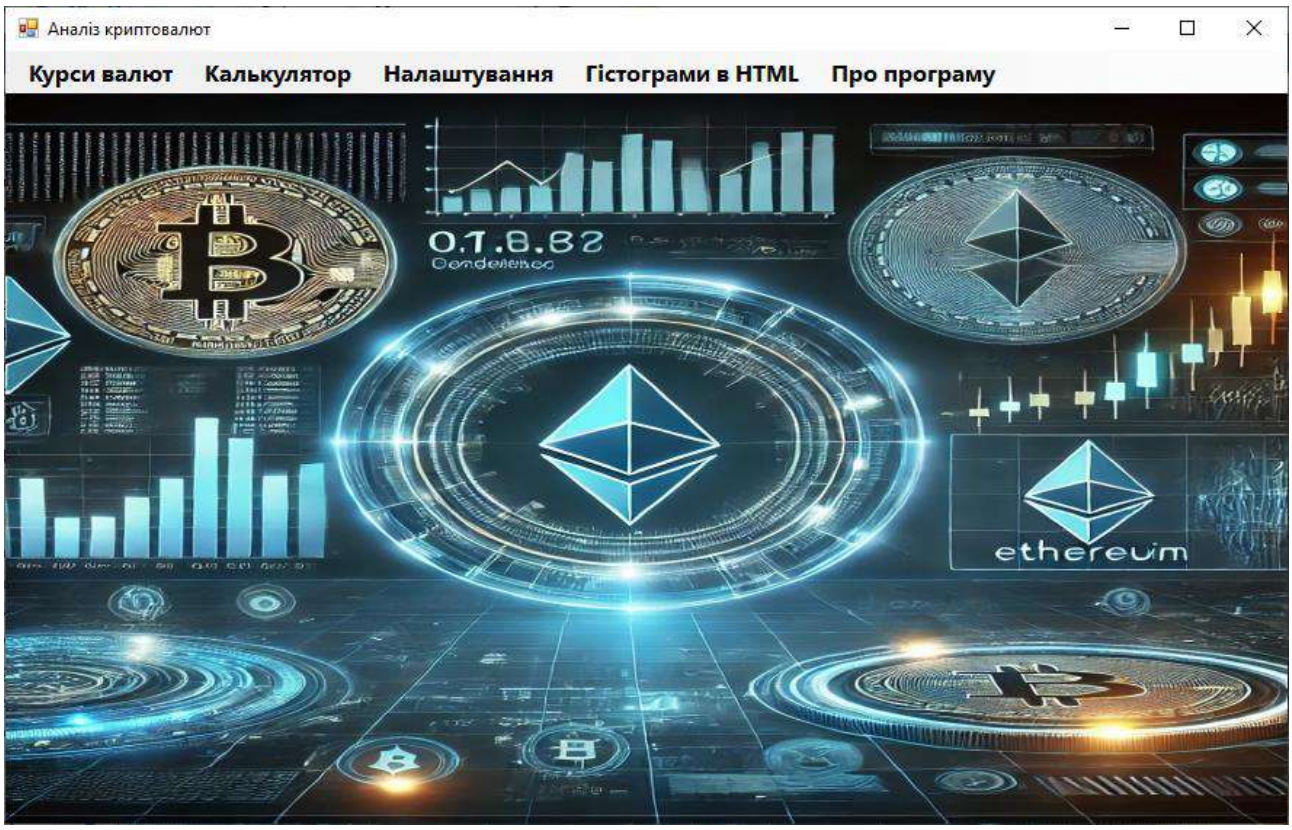


Рис.4.1. Головна форма програми

Головне меню складається з категорій: <Курси валют>, <Калькулятор>, <Налаштування>, <Про програму>.

Категорія <Курси валют> складається з таких опцій:

1. Статистика по курсам.
2. Поточний курс.
3. Графік.

Опція статистика по курсам валют являє собою ведення таблиці **Kursy** БД **crypto_valuta.mdb** (рис. 4.2).

В даній формі розміщено елемент DataGridView для розміщення таблиці Kursy, а також елемент bindingNavigator для зручного переміщення по записам таблиці. Елементи bindingNavigator та DataGridView прив’язані до універсального елементу-зв’язувача bindingSource.

В таблиці присутня інформація щодо поточної дати, а також курсів криптовалют BTC_USD (кількість доларів за bitcoin), LTC_BCD (кількість bitcoin-ів за litecoin), LTC_USD (кількість доларів за litecoin)

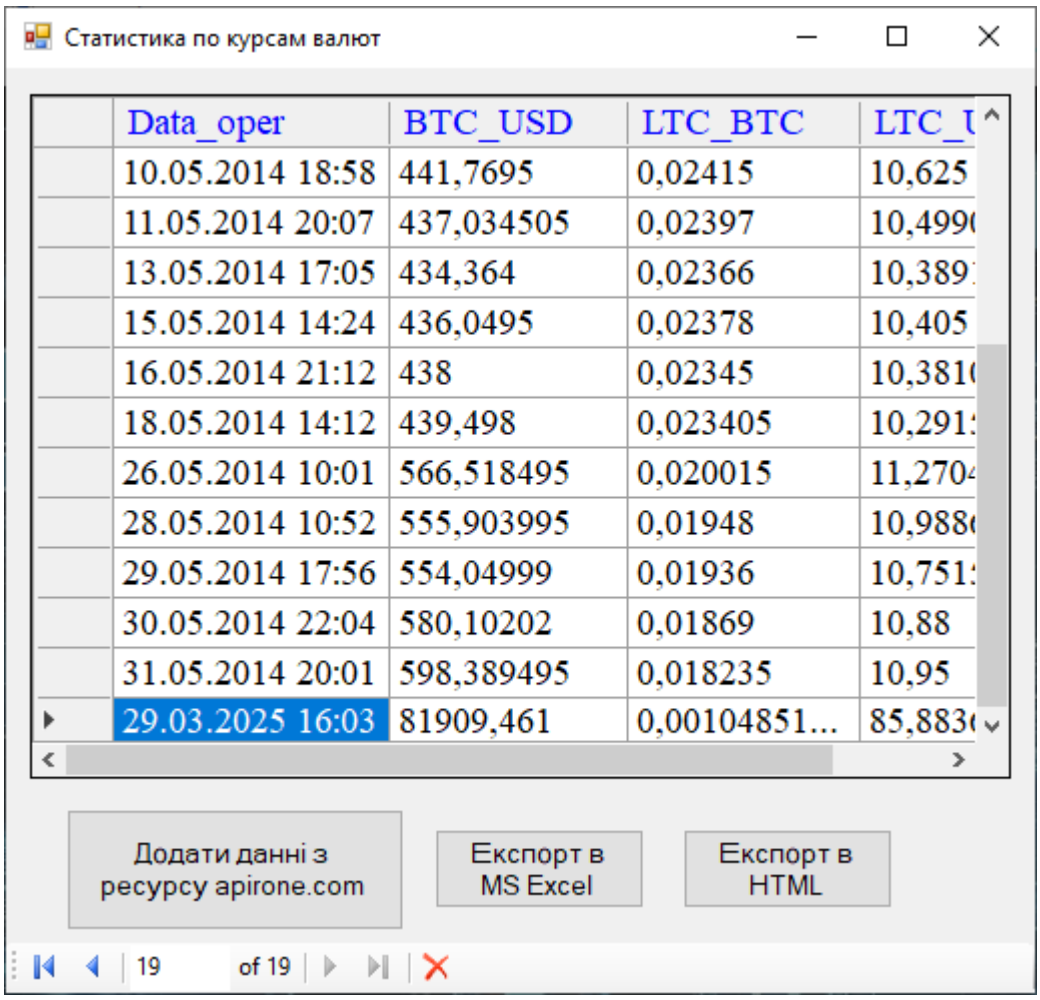


Рис. 4.2. Статистика по курсам криптовалют

Поле ID є лічильником, має місце, але приховане від користувача. Завдяки ньому виконується перше правило нормалізації баз даних – відсутність однакових записів в таблиці. При використанні технології ADO.NET наявність ключового поля є обов’язковим.

На початку спрацьовує конструктор форми.

```
private void Form4_Load(object sender, EventArgs e)
        {
            //Формування запиту
            try
            {
                //Формування запиту і прив'язка до DataGridView через bindingSource
                da = new OleDbDataAdapter("Select * From Kursy", conn);
                bulder = new OleDbCommandBuilder(da);
                ds = new DataSet();
                da.Fill(ds);

                bindingSource1.DataSource = ds.Tables[0];
                dataGridView1.DataSource = bindingSource1;
                bindingNavigator1.BindingSource = bindingSource1;

                dataGridView1.Columns[0].Visible = false;
```

```
            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

            //Налаштування
            dataGridView1.DefaultCellStyle.Font = FormGl.font_text;
            dataGridView1.ColumnHeadersDefaultCellStyle.Font = FormGl.font_text;
            dataGridView1.AutoResizeColumns();
            dataGridView1.AutoResizeRows();
            ////////////////////////////////////////////////////////////////////
            dataGridView1.EnableHeadersVisualStyles = false;
            dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = FormGl.cvet_hapka;
            dataGridView1.DefaultCellStyle.ForeColor = FormGl.cvet_text;

        }
```

Об’єкт da типу OleDbDataAdapter слугує буфером між таблицею бази даних та об’єктом DataTable на стороні клієнта. Об’єкт bulder типу OleDbCommandBuilder формує три запити – на видалення (DELETE), оновлення (UPDATE) додавання даних (INSERT INTO). Всі запити формуються по ключовому полю ID.

Одним з головних моментів в даній формі є поповнення даних з ресурсу Інтернет **apirone.com**. За це відповідає клас WebClient простору імен System.Net.

При натисненні кнопки <Додати данні з ресурсу apirone.com> відбувається додавання курсів криптовалют до таблиці "Kursy".

Доступ до web-ресурсів з використанням класу WebClient буде розглянуто у розділі 4.2. кваліфікаційної роботи.

Наведемо подію додавання нового запису до таблиці "Kursy".

```
//Додавання даних з Інтернету в БД для статистики та розрахунків
        //Ресурс - apirone.com
        private void button1_Click(object sender, EventArgs e)
        {
            double[] tarif = new double[3];
            try
            {
                if (!FormGl.GetValuta(tarif))
                {
                    MessageBox.Show("Відсутній доступ в інтернет", "Увага!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    return;
                }

                //Формування нового рядку
                DataRow ins = ds.Tables[0].NewRow();
```

```
                ins[1] = DateTime.Now;
                ins[2] = tarif[0];
                ins[3] = tarif[1];
                ins[4] = tarif[2];

                ds.Tables[0].Rows.Add(ins);
                //Перехід на потрібну комірку в dataGridView1
                dataGridView1.CurrentCell =
dataGridView1.Rows[ds.Tables[0].Rows.IndexOf(ins)].Cells[1];

                da.Update(ds.Tables[0]);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
```

За допомогою методу Update відбувається оновлення таблиці БД з поточними курсами криптовалют.

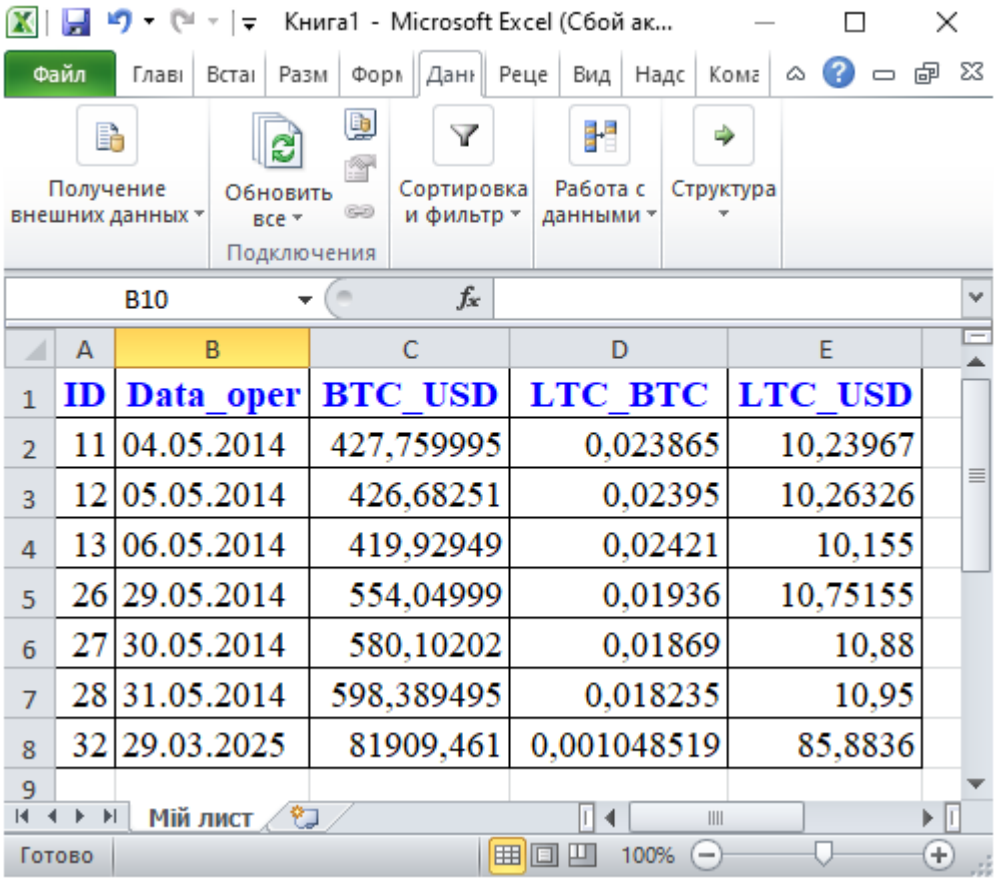
Крім накопичення даних важливим моментом є експорт даних в MS Excel та в HTML. З цією метою розроблено універсальні статичні функції ExportXls, Export_Html, які дозволяють експортувати довільний об’єкт **DataView** – представлення таблиці БД.

Для додавання об’єкту MS Excel до проекту треба скористатися опцією <Add References>.

Функція ExportXls є незалежною (static) і може бути використана користувачами для експорту довільного представлення таблиці БД. Розглянемо головні об’єкти MS Excel:

1. Application – об’єкт додатку MS Excel.
2. WorkBook – об’єкт книги MS Excel.
3. Worksheet – об’єкт листа MS Excel.
4. Range – об’єкт діапазону комірок MS Excel.
5. Border – об’єкт границі комірок MS Excel.

На рис. 4.3. наведено результат експорту в Excel статистики курсів криптовалют.



	A	B	C	D	E
1	ID	Data_oper	BTC_USD	LTC_BTC	LTC_USD
2	11	04.05.2014	427,759995	0,023865	10,23967
3	12	05.05.2014	426,68251	0,02395	10,26326
4	13	06.05.2014	419,92949	0,02421	10,155
5	26	29.05.2014	554,04999	0,01936	10,75155
6	27	30.05.2014	580,10202	0,01869	10,88
7	28	31.05.2014	598,389495	0,018235	10,95
8	32	29.03.2025	81909,461	0,001048519	85,8836
9					

Рис. 4.3. Результат експорту в MS Excel

У будь-який момент часу можна подивитися поточний курс криптовалют (рис. 4.4.)

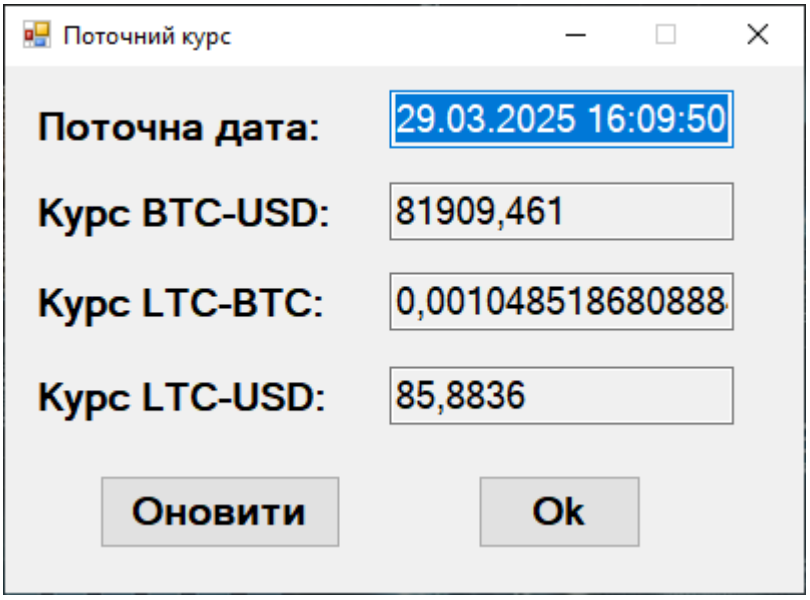
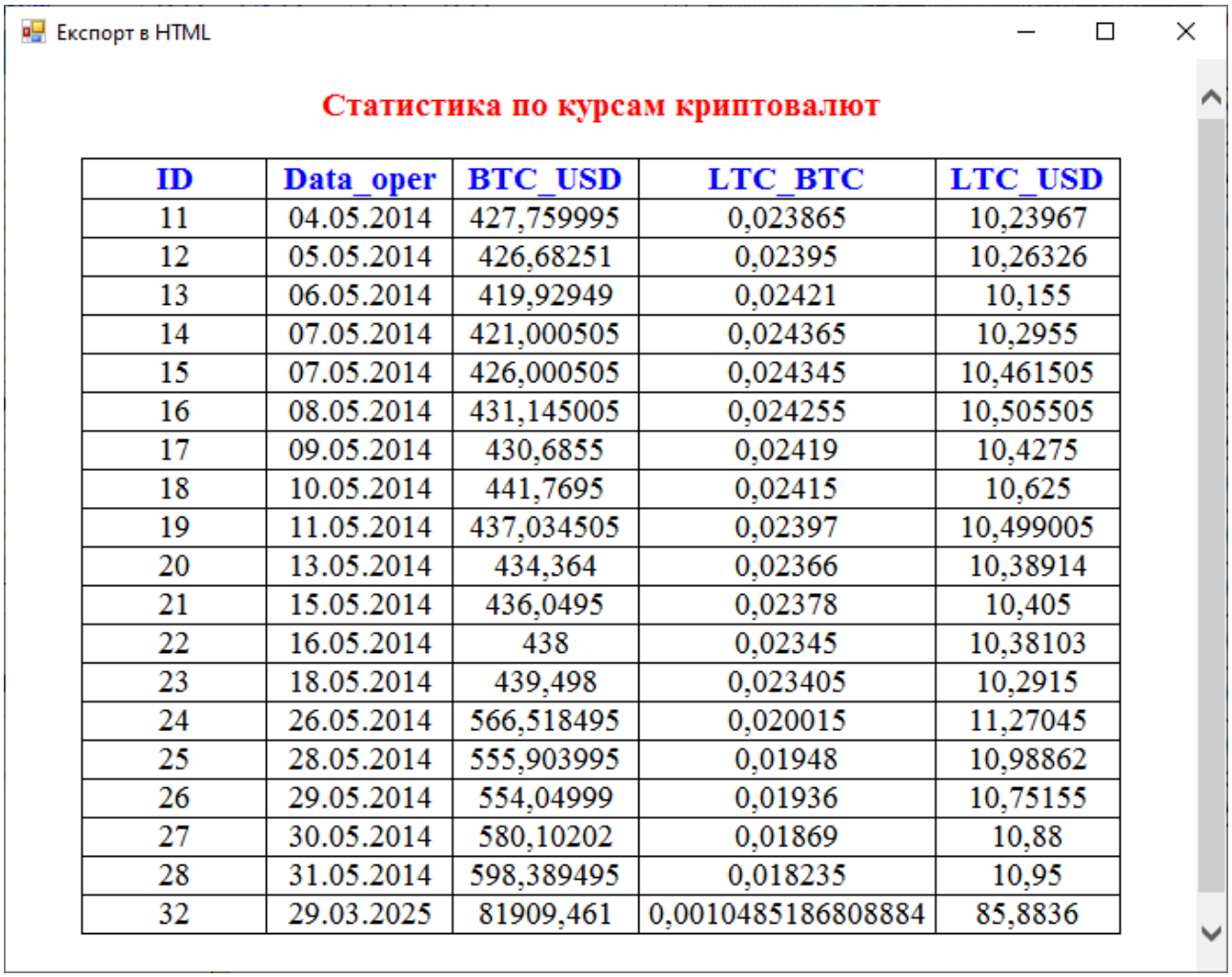


Рис. 4.4. Поточний курс криптовалют

Окрім експорту в Excel, передбачено експорт в HTML (рис. 4.5.). При цьому результат виводиться в незалежній формі **Form7**. В даній формі знаходиться компонент **webBrowser**.



ID	Data_oper	BTC_USD	LTC_BTC	LTC_USD
11	04.05.2014	427,759995	0,023865	10,23967
12	05.05.2014	426,68251	0,02395	10,26326
13	06.05.2014	419,92949	0,02421	10,155
14	07.05.2014	421,000505	0,024365	10,2955
15	07.05.2014	426,000505	0,024345	10,461505
16	08.05.2014	431,145005	0,024255	10,505505
17	09.05.2014	430,6855	0,02419	10,4275
18	10.05.2014	441,7695	0,02415	10,625
19	11.05.2014	437,034505	0,02397	10,499005
20	13.05.2014	434,364	0,02366	10,38914
21	15.05.2014	436,0495	0,02378	10,405
22	16.05.2014	438	0,02345	10,38103
23	18.05.2014	439,498	0,023405	10,2915
24	26.05.2014	566,518495	0,020015	11,27045
25	28.05.2014	555,903995	0,01948	10,98862
26	29.05.2014	554,04999	0,01936	10,75155
27	30.05.2014	580,10202	0,01869	10,88
28	31.05.2014	598,389495	0,018235	10,95
32	29.03.2025	81909,461	0,0010485186808884	85,8836

Рис. 4.5. Результат експорту даних в HTML

Головна опція меню – <Калькулятор криптовалют>.

На рис. 4.6. наведемо форму калькулятора окупності криптовалюти bitcoin.

Особливістю експорту даних є передача вибраних користувачем налаштувань (меню <Налаштування>) кольору заголовка таблиці, кольору шапки таблиці, кольору тексту таблиці, а також вибраний шрифт.

Всі налаштування зберігаються до файлу налаштувань setup.ini.

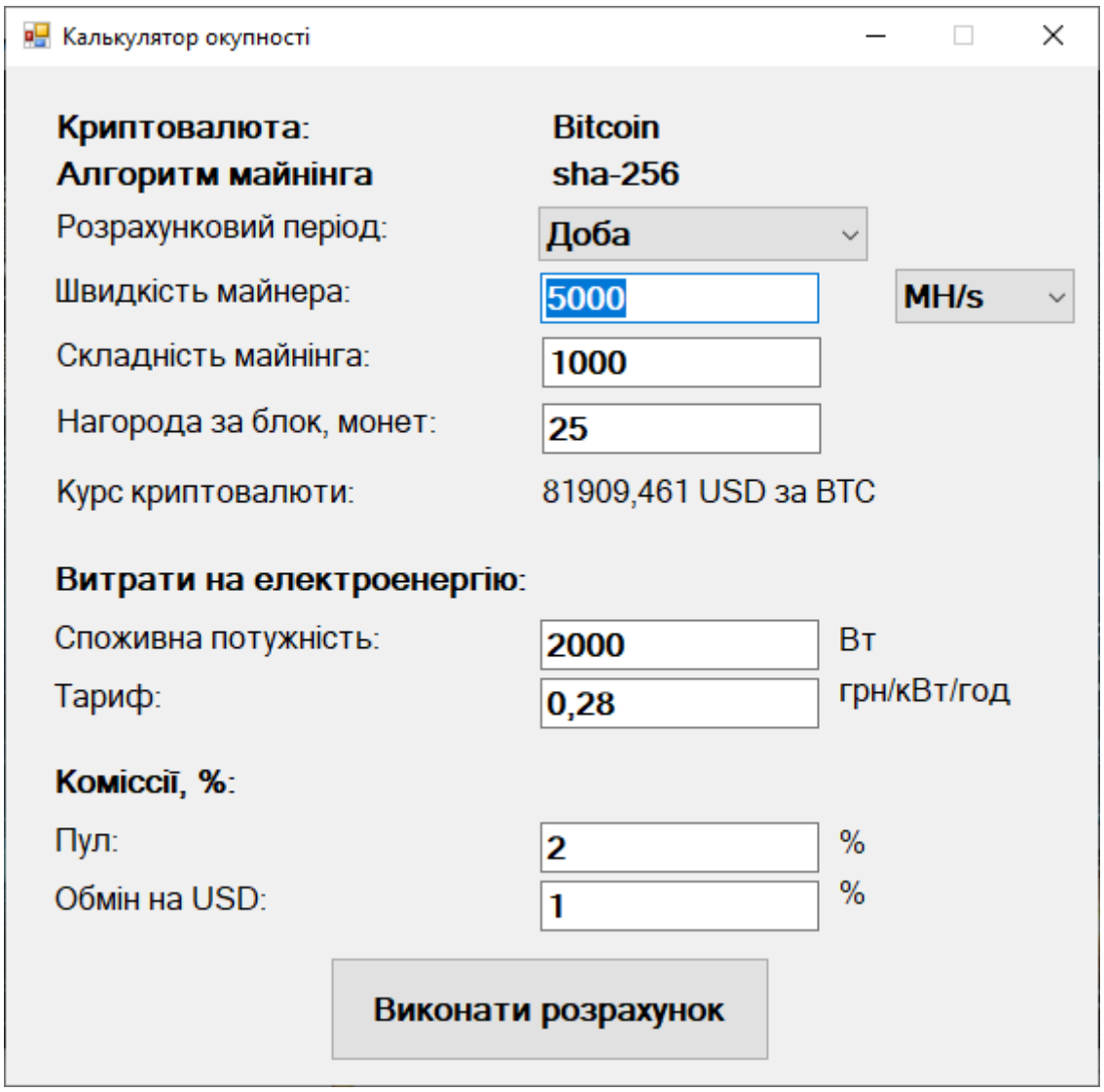


Рис. 4.6. Форма калькулятора окупності криптовалют

Розпишемо вхідні дані для калькулятора криптовалюти:

1. Розрахунковий період для розрахунку доходу. Може приймати наступні значення: година, доба, тиждень, місяць (30 днів), рік (365 днів).

2. Швидкість майнера, яка може бути вибрана: kH/s, MH/s, GH/s (відповідно кіло-, мега-, та гіго-хеши в секунду).

3. Складність майнінгу.

4. Винагорода за блок, кількість монет.

5. Курс криптовалюти BTC_USD, який береться з ресурсу btc-e.com мережі Інтернет.

6. Споживча потужність електроенергії.

7. Тариф – вартість за 1 кВт/ч у гривнях.

8. Комісія за використання пулу у відсотках.
9. Комісія за обмін bitcoin-ів (BTC) на долари (USD) у відсотках.

Так як класи списків в C# ListBox та ComboBox є контейнерами, доцільно створити клас Name_List, який буде основою для елементів керування comboBox1 та comboBox2. Приведемо код класу Name_List.

```
//Клас для заповнення списку
    public class Name_List
    {
        public string str;
        public int ItemData;

        public Name_List(string str, int ItemData)
        {
            this.str = str;
            this.ItemData = ItemData;
        }

        public override string ToString()
        {
            return str;
        }
    }
```

Даний клас має два поля str типу string для відображення інформації в списку та ItemData типу int. В класі обов'язково повинен бути перевантажений метод ToString для відображення вибраної інформації в списку.

Так, наповнення списку comboBox1 відбуватиметься наступним чином:

```
comboBox1.Items.Add(new Name_List("Година",3600));
comboBox1.Items.Add(new Name_List("Доба", 86400));
comboBox1.Items.Add(new Name_List("Тиждень", 7*86400));
comboBox1.Items.Add(new Name_List("Місяць (30 днів)", 30 * 86400));
comboBox1.Items.Add(new Name_List("Рік (365 днів)", 365 * 86400));
```

В поле ItemData буде записано кількість секунд, яка відповідає вибраному часовому проміжку.

Наповнення списку comboBox2 відбуватиметься наступним чином:

```
comboBox2.Items.Add(new Name_List("kH/s", 1000));
comboBox2.Items.Add(new Name_List("MH/s", 1000000));
comboBox2.Items.Add(new Name_List("GH/s", 1000000000));
```

В поле ItemData буде записано кількість хешів в кіло-, мега- та гігохешах.

Це відповідно 1000, 1000000 та 1000000000 хешів.

Вхідні дані, записані в текстових полях (об'єкти TextBox) контролюються у події зміни тексту textBox1_TextChanged. Приведемо код даної функції.

```
//Зміна тексту
        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            string per = ((TextBox)sender).Text;
            double voz;
            bool v = double.TryParse(per, out voz);
            if (per.Length == 0) return;
            if (v == true) { nach_zn = per; return; }
            if (v == false) ((TextBox)sender).Text = nach_zn;
        }
```

При натисканні на кнопку <Виконати розрахунок> відбувається наповнення даних класу **Raschet** для передачі даних до розрахункової форми 2 (Form2) та подальшого розрахунку. Приведемо фрагмент коду наповнення класу **Raschet**.

```
//Наполнение класса Raschet для передачи и дальнейшего расчета
            ras = new Raschet();
            ras.kol_sec = ((Name_List)comboBox1.SelectedItem).ItemData;
            ras.Hash = ((Name_List)comboBox2.SelectedItem).ItemData;
            ras.speed = double.Parse(textBox1.Text);
            ras.dif = double.Parse(textBox2.Text);
            ras.nagr = double.Parse(textBox3.Text);
            ras.kurs = this.kurs_bitcoin;

            ras.Vatt = double.Parse(textBox4.Text);
            ras.tarif = double.Parse(textBox5.Text);

            ras.komis_pul = double.Parse(textBox6.Text);
            ras.komis_obmen = double.Parse(textBox7.Text);
```

Після коректного заповнення даних визивається форма розрахунку показників калькулятора окупності (рис. 4.7.).

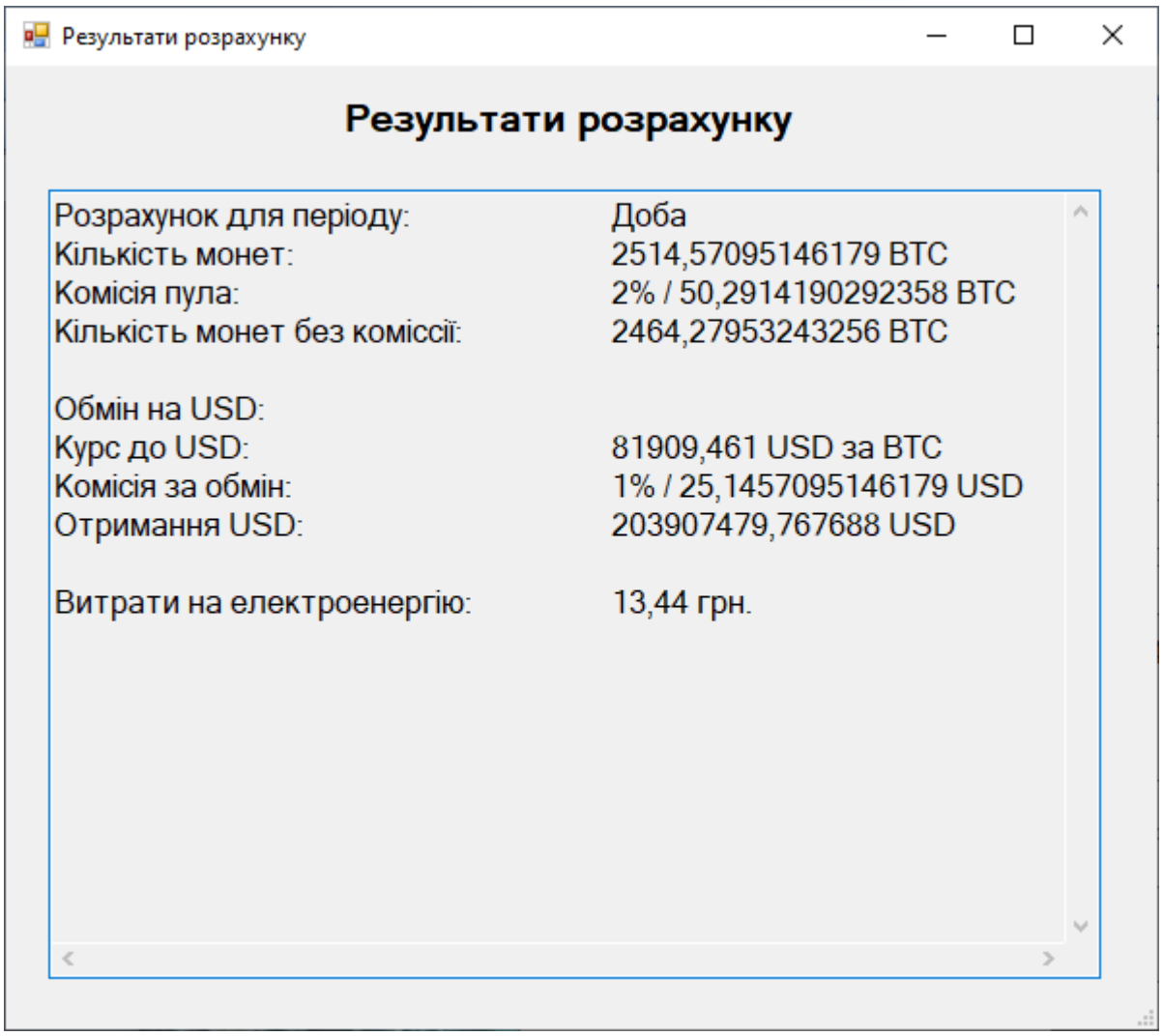


Рис. 4.7. Результати розрахунку

У результаті розрахунків отримуємо наступні показники:

1. Кількість монет bitcoin (BTC), зароблених за вибраний період.
2. Комісія за використання пулу у BTC.
3. Кількість монет без врахування комісії.
4. Комісія за обмін BTC на долари (USD) у валюті USD.
5. Отримані USD без врахування комісії.
6. Витрати на електроенергію у гривнях.

Результат розрахунку виводиться у текстовому полі, яке можна скопіювати до будь-якого редактору.

Приведемо фрагмент коду розрахунку даних доходності криптовалюти, що відбувається у методі завантаження форми Form2_Load.

```
//Метод завантаження форми
        private void Form2_Load(object sender, EventArgs e)
        {
            //Формування розрахунку та строки результату
            double kol_m;
            kol_m = 1.0 * ras.kol_sec * ras.nagr * ras.speed / ras.dif / Math.Pow(2, 32) *
ras.Hash;
            string rez;
            rez = "Розрахунок для періоду:\t\t";
            if (ras.kol_sec == 3600) rez += "Година";
            if (ras.kol_sec == 86400) rez += "Доба";
            if (ras.kol_sec == 7 * 86400) rez += "Тиждень";
            if (ras.kol_sec == 30 * 86400) rez += "Місяць (30 днів)";
            if (ras.kol_sec == 365 * 86400) rez += "Рік (365 днів)";
            rez += "\r\n";
            rez += "Кількість монет:\t\t\t";
            rez += kol_m;
            rez += " BTC\r\n";
            rez += "Комісія пула:\t\t\t";
            rez += ras.komis_pul + "% / " + kol_m * ras.komis_pul / 100;
            rez += " BTC\r\n";
            rez += "Кількість монет без комісії:\t\t";
            rez += (1 - ras.komis_pul / 100) * kol_m + " BTC\r\n\r\n";

            rez += "Обмін на USD:\r\n";
            rez += "Курс до USD:\t\t\t" + ras.kurs + " USD за BTC\r\n";
            rez += "Комісія за обмін:\t\t\t";
            rez += ras.komis_obmen + "% / " + kol_m * ras.komis_obmen / 100;
            rez += " USD\r\n";
            rez += "Отримання USD:\t\t\t";
            rez += (1 - ras.komis_obmen / 100) * kol_m*ras.kurs + " USD\r\n\r\n";

            rez += "Витрати на електроенергію:\t" + (ras.Vatt * ras.tarif / 1000) *
(ras.kol_sec / 3600) + " грн.\r\n";

            textBox1.Text = rez;

        }
```

Формули для розрахунку доходності калькулятору, а також врахування комісій і витрат на електроенергію розписані у розділі 2 кваліфікаційної роботи. Курс криптовалюти bitcoin зчитується з ресурсу **btc-e.com** мережі Інтернет.

4.2. Оновлення курсів криптовалют через мережу Інтернет

Ключовим моментом розробки програмного забезпечення калькулятора окупності криптовалют є оновлення відповідних курсів з мережі Інтернет. Це дає змогу використати поточні курси для розрахунку, а також накопичити статистичну інформацію щодо курсів криптовалют, яка змінюється в часі.

Головний ресурс біржі криптовалют в мережі Інтернет – **https://apirone.com**. Саме цей ресурс вибрано для отримання наступних поточних курсів:

1. BTC_USD – кількість доларів за один bitcoin.
2. LTC_BTC – кількість bitcoin-ів за один litetcoin.
3. LTC_USD – кількість доларів за один litecoin.

Для того щоб в будь-який момент часу мати ці курси розроблено універсальну статичну функцію **GetValuta**, яка зчитує інформацію зазначених курсів. Слід звернути увагу, що ресурс **apirone.com** має власні функції **api** за допомогою, яких можна надіслати GET-запит на зчитування конкретних даних [додаток А]. Так, для зчитування вищезазначених курсів рядок запиту буде виглядати наступним чином:

1. Для зчитування інформації по bitcoin:

"**https://apirone.com/api/v1/ticker?currency=btc**".

2. Для зчитування інформації по litecoin:

"**https://apirone.com/api/v1/ticker?currency=ltc**".

Для зчитування даних з Інтернету використовується клас мови C# **WebClient**, що знаходиться в просторі імен System.Net.

Наведемо код функції GetValuta, яка зчитує дані з ресурсу btc-e.com.

```
//Статична функція, якою будемо користуватися в багатьох формах
        //Функція для зчитування даних з інтернету (ціни на криптовалюти)
        //Зчитування валют відбувається з використанням класу WebClient та api сервису
https://apirone.com/api
        public static bool GetValuta(double[] tarif)
        {

            string stroka1 = "https://apirone.com/api/v1/ticker?currency=btc";
            string stroka2 = "https://apirone.com/api/v1/ticker?currency=ltc";
            string[] mas = { "BSD", "last" };

            try
            {
                ServicePointManager.Expect100Continue = true;
                ServicePointManager.SecurityProtocol = (SecurityProtocolType)3072;

                ServicePointManager.ServerCertificateValidationCallback = delegate { return
true; };

                using (WebClient client = new WebClient())
                {
                    byte[] buffer;
                    string str3;
```

```
            do
            {
                try
                {
                    //MessageBox.Show("1111");
                    buffer = client.DownloadData(stroka1);
                    //MessageBox.Show("2222");
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    buffer = null;
                    return false;
                }
                if (buffer == null)
                {
                    Thread.Sleep(0xea60);
                }
            }
            while (buffer == null);
            str3 = Encoding.UTF8.GetString(buffer);
            //Рзбір рядка з криптовалютою
            int idx1, idx2;
            string ch;
            ////////////////////////////////////////////
            idx1 = str3.IndexOf(mas[0]);
            idx1 += mas[0].Length + 4;
            idx1 = str3.IndexOf(mas[1], idx1) + 7;
            idx2 = str3.IndexOf(" ", idx1);
            ch = str3.Substring(idx1, idx2 - idx1);
            ch = ch.Replace('.', ',');
            tarif[0] = double.Parse(ch);
            ////////////////////////////////////////////
        }
    }

    catch
    {
        return false;
    }

    try
    {
        using (WebClient client = new WebClient())
        {
            byte[] buffer;
            string str3;

            //int num;
            do
            {
                try
                {
                    buffer = client.DownloadData(stroka2);
                }
                catch
                {
                    buffer = null;
                    return false;
                }
                if (buffer == null)
                {
                    Thread.Sleep(0xea60);
                }
            }
```

```
                while (buffer == null);
                str3 = Encoding.UTF8.GetString(buffer);

                //Розбір рядка з криптовалютою
                int idx1, idx2;
                string ch;
                ////////////////////////////////////////
                idx1 = str3.IndexOf(mas[0]);
                idx1 += mas[0].Length + 4;
                idx1 = str3.IndexOf(mas[1], idx1) + 7;
                idx2 = str3.IndexOf(" ", idx1);
                ch = str3.Substring(idx1, idx2 - idx1);
                ch = ch.Replace('.', ',');
                tarif[2] = double.Parse(ch);
                tarif[1] = tarif[2] / tarif[0];
                ////////////////////////////////////////
            }
        }
        catch
        {
            return false;
        }
        return true;
    }
```

Функція контролюється за допомогою обробки виключень try … catch та повертає значення типу bool. Зчитування трьох курсів криптовалют відбувається в масив tarif типу double.

Вся інформація зчитується в масив байтів buffer за допомогою методу DownloadData класу WebClient.

На рис. 4.8. приведемо результат запиту щодо курсів BTC_USD, LTC_BTC, LTC_USD.

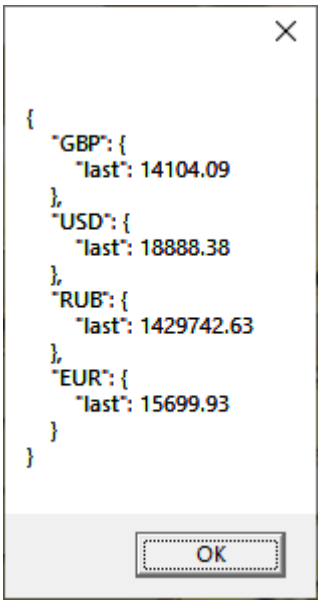


Рис. 4.8. Результат запиту щодо поточних курсів криптовалют apirone.com

З цього результату ми вибираємо значення курсів криптовалюти bitcoin в доларах (USD) – **USD**. Розбір рядку зробимо використовуючи методи класу string, зокрема IndexOf – пошук підрядка в рядку з заданої позиції, Substring – виділення підрядка з рядку, Replace – заміна символів в рядку.

Функція **GetValuta** використовується при розрахунку доходності криптовалют (опція <Калькулятор>), статистики по курсам криптовалют (опція <Статистика по курсам> підменю <Курси валют>), виведення поточного курсу криптовалют (<Поточний курс> підменю <Курси валют>).

4.3. Виведення статистичної інформації по криптовалютам з використанням 2D і 3D-графіки

При розробці програмного забезпечення великого значення набуває використання 2D і 3D-графіки.

Робота з 2D-графікою передбачена у додатку Calc, розробленого на мові C# – опція меню <Графіка> підменю <Курси валют>.

Для виведення статистичної інформації по курсам криптовалют на мові C# використано елемент керування **Chart**, який дозволяє графічно вивести довільну інформацію.

Графічне відображення статистики передбачено по наступним курсам валют: BTC-USD (кількість доларів за 1 BTC), LTC-BTC (кількість BTC за 1 LTC), LTC-USD (кількість доларів за 1 LTC).

На рис. 4.9. приведемо результат виведення 2D-графіки по курсам BTC-USD. Дослідження по курсах поповнювались з ресурсу btc-e.com щодня.

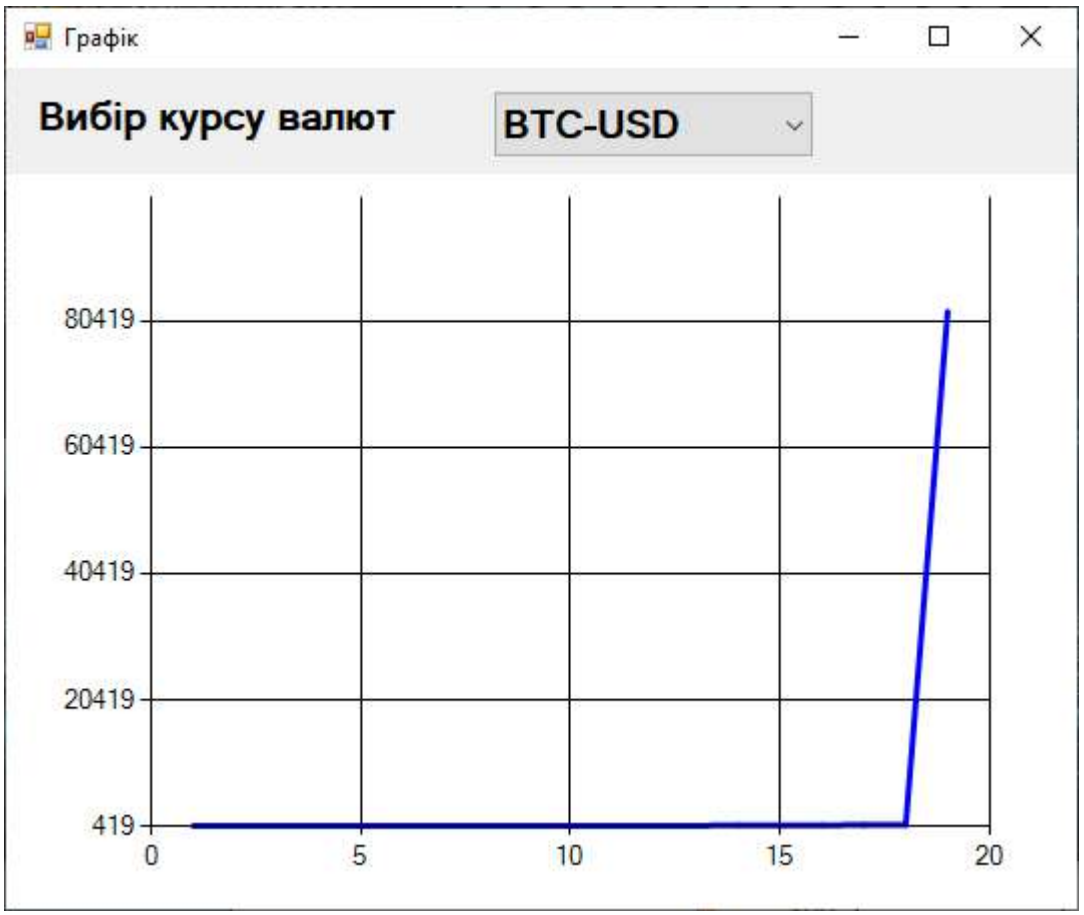


Рис. 4.9. Результат виведення 2D-графіки

Дані для виведення графіків беруться з таблиці Kursy БД crypto_valuta.mdb.

Приведемо метод Paint_Graf заповнення даних компоненту Chart для роботи з 2D-графікою.

```
//Функція малювання графіку, використовуючи компонент Chart
        void Paint_Graf(DataTable table, int index)
        {
            //Малювання графіку функції
            int i;
            double zn, min = 1000000000;
            chart1.Series[0].Points.Clear();
            for (i = 0; i < table.Rows.Count; i++)
            {
                zn = Convert.ToDouble(table.Rows[i][index]);
                if (zn < min) min = zn;
                chart1.Series[0].Points.AddY(zn);
            }

            chart1.ChartAreas[0].AxisY.Minimum = (int)min;
        }
```

В даній функції до 1-го елементу сімейства Series додаються точки з координатами *X* та *Y* по яким буде будуватися графік. Крім того, розраховується мінімальне значення координати *Y* для коректного

відображення графіку. За умовчанням мінімальне значення координати Y дорівнює нулю, максимальне значення розраховується автоматично.

Робота з 3D-графікою здійснюється у додатку Calc_3D, розробленому на мові C++ Visual Studio 2019.

Проект Calc_3D реалізований з використанням бібліотеки MFC (Microsoft Foundation Classes) на основі архітектури Document/View.

Базовий проект складається з наступних класів:

1. CCalc_3DApp – клас запуску вікна, похідний від CWinAppEx. Саме в цьому класі, у функції формується шаблон документу SDI CSingleDocTemplate(), в якому об'єднуються три класи – CMainFrame, CCalc_3DDoc, CCalc_3DView.

2. CMainFrame – клас головного вікна програми, є свого роду контейнером, в який вставляються дочірні вікна. Сам клас CMainFrame призначений для роботи з меню, панеллю інструментів, рядком стану, а також для підключення інших дочірніх вікон або інших документів.

3. CCalc_3DDoc – клас документу для зберігання, зчитування та запису даних на диск. Може мати декілька видів (класів похідних від CView).

4. CCalc_3DView – клас виду, де користувач, використовуючи функцію OnDraw() може виводити довільну інформацію – текстову або графічну.

Крім основних класів додамо створимо новий клас GR_3D для роботи з 3D-графікою. Даний клас є похідним від класу CFrameWnd.

При запуску додатку дані щодо курсів валют зчитуються з бази даних, використовуючи технологію ADO, та записуються до масивів. Це відбувається у класі CCalc_3DDoc.

Графіка у класі GR_3D побудована по принципу камери і акторів, які мають власну систему координат. Так, кожен об'єкт має структуру з трьох векторів vUp – вектор, спрямований вверх, vForward – вектор, спрямований вперед та vLocation – розташування об'єкту у 3D-просторі.

На рис. 4.10. приведемо результат виведення 3D-графіки по курсам криптовалют.

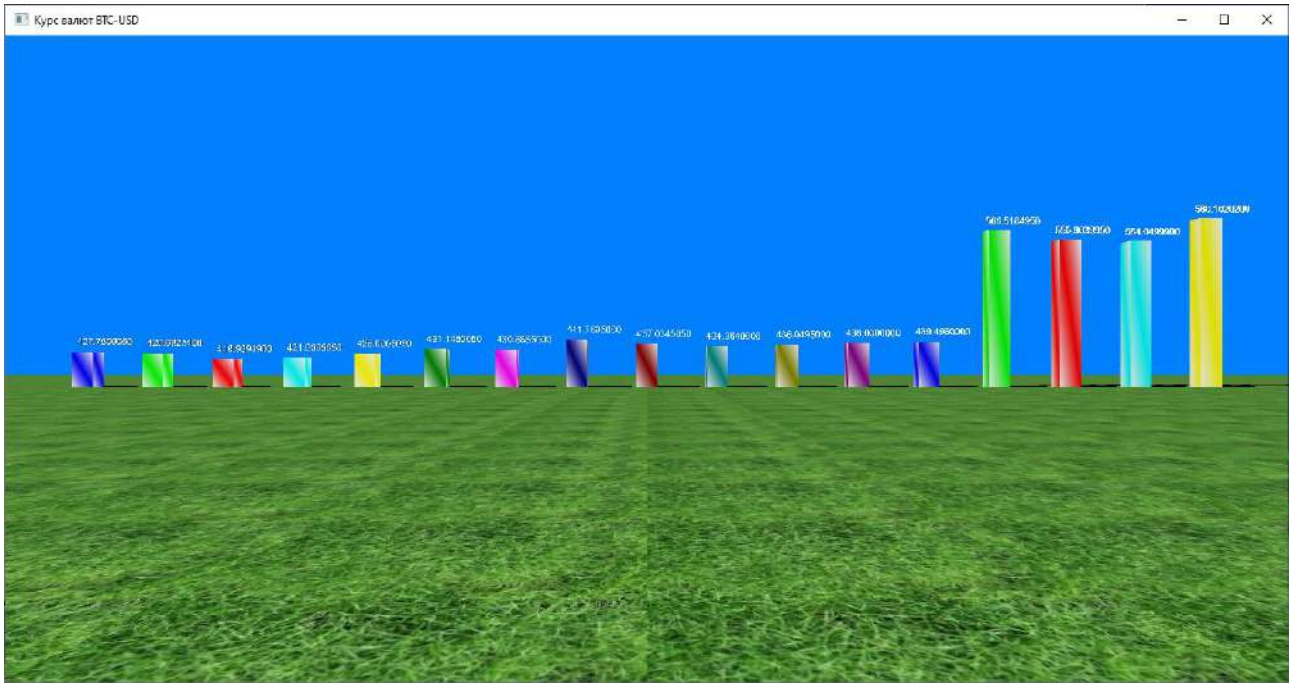


Рис. 4.10. Результат виведення 3D-графіки по курсу BTC-USD

При виведенні 3D-графіки передбачені наступні властивості:

1. Накладення текстури на землю.
2. Використання джерел світла.
3. Побудова тіні для кожного стовпчику.
4. Градієнтна заливка стовпчику.
5. Виведення надписів курсів валют.

Так, користувач може рухатися по сцені і спостерігати необхідно йому інформацію. Так можна роздивитись з будь-якого ракурсу динаміку зміни криптовалют.

При побудові тіні об’єкту виводиться той самий об’єкт тільки чорним кольором. При цьому матриця моделі перераховується таким чином, щоб була побудована проекція на площину *XOZ*.

Нижче наведемо функцію виведення діаграми та відповідної тіні до неї.

```
//Функция вывода диаграммы и тени к ней
void GR_3D::DrawInhabitants(GLint nShadow)
    {
 // включение режима работы с текстурами
```

```
      if(text_nal&&nShadow==0&&kod_text!=3)
   {
      glEnable(GL_TEXTURE_2D);
   // Вывод текстуры с учетом цвета объекта
   if(kod_text==1)glTexEnvf(GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE,GL_MODULATE);
   // Вывод текстуры без учета цвета объекта
   if(kod_text==2)glTexEnvf(GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE,GL_DECAL);
   glBindTexture(GL_TEXTURE_2D, text1);
   }

      //Указатель на документ
      CCalc_3DDoc* pDoc = (CCalc_3DDoc*)((CFrameWnd*)
      AfxGetMainWnd())->GetActiveDocument();

      long c1,c2;
      double y,max,min;
      int i,n = pDoc->n;
      double* mas = pDoc->mas[kod_kurs-1];
      char str[20];

      //Находи максимум массива
      min = mas[0];
      max = mas[0];
      for(i=1;i<n;i++)
      {
            if (mas[i]<min)min = mas[i];
            if (mas[i]>max)max = mas[i];
      }

    for(i = 0; i < n; i++)
        {
                glPushMatrix();
                gltApplyActorTransform(&prizma[i]);

                if(nShadow == 0)
                {
                     c1 = colorArray[i+2];
                     c2 = colorArray[1];
                }
                else c1 = c2 = 0;

                glBegin(GL_QUADS);
                     y = (mas[i]-min)/(max-min)*2;
                     truba_y(0, 0, -0.4, y, 0.15,c1,c2);
                glEnd();

                //Вывод текста в 3D

                if(nShadow == 0)
```

```
                {
                        glPushAttrib(GL_ALL_ATTRIB_BITS);
                        glTranslatef(0.15,y+0.1,0);
                        glRotatef(180,0,1,0);
                        glScalef(0.15f,0.15f,0.15f);
                        sprintf_s(str,"%0.7f",mas[i]);
                        glColor3f(1,1,1);
                        glDisable(GL_TEXTURE_2D);
                        OutText(str);
                        glEnable(GL_TEXTURE_2D);
                        glPopAttrib();
                }


                glPopMatrix();
        }

   glDisable(GL_TEXTURE_2D);

}
```

У даній функції змінна nShadow характеризує виводити тінь, або ні. Якщо nShadow дорівнює нулю – виводиться безпосередньо об’єкт, в іншому випадку – виводиться тінь об’єкта.

У якості web-частини даного програмного забезпечення вибрано автоматичне формування скриптів на основі елементу керування chart.js. З цією метою розроблено універсальну функцію, до якою треба передати два масиви вхідних параметрів – числові параметри по яким будується діаграма, а також рядкові параметри для виведення підказок та легенди стовпчиків.

Наведемо код даної функції:

```
        //Універсальна функція скрипта для побудови діаграм в Web
        public static void SaveHtmlFile(string fileName, string title, string chartLabel,
string[] countries, double[] population)
        {
            // Формуємо HTML-контент
            string htmlContent = $@"
<!DOCTYPE html>
<html lang='uk'>
<head>
    <meta charset='UTF-8'>
    <meta name='viewport' content='width=device-width, initial-scale=1.0'>
    <title>{title}</title>
    <script src='https://cdn.jsdelivr.net/npm/chart.js'></script>
    <style>
        body {{
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 0;
            display: flex;
            justify-content: center;
```

```
            align-items: center;
            height: 100vh;
            background-color: #f4f4f4;
        }}
        canvas {{
            width: 100vw !important;
            height: 100vh !important;
        }}
    </style>
</head>
<body>
    <canvas id='populationChart'></canvas>
    <script>
        const ctx = document.getElementById('populationChart').getContext('2d');
        new Chart(ctx, {{
            type: 'bar',
            data: {{
                labels: {GenerateArrayForJS(countries)},
                datasets: [{{
                    label: '{chartLabel}',
                    data: {GenerateArrayForJS(population)},
                    backgroundColor: 'rgba(255, 0, 0, 0.6)',
                    borderColor: 'rgba(75, 192, 192, 1)',
                    borderWidth: 1
                }}]
            }},
            options: {{
                responsive: true,
                maintainAspectRatio: false,
                scales: {{
                    y: {{
                        beginAtZero: true
                    }}
                }}
            }}
        }});
    </script>
</body>
</html>";

            // Записуємо HTML в файл
            File.WriteAllText(fileName, htmlContent);
        }

        private static string GenerateArrayForJS(string[] array)
        {
            return $"['{string.Join("', '", array)}']";
        }

        private static string GenerateArrayForJS(double[] array)
        {
            return $"[{string.Join(", ", array)}]";
        }
```

Дана функція є універсальною і дозволяє автоматично формувати Java-скрипти по даним які беруться з бази даних та відповідна інформація відразу відкривається з браузеру. Це дозволяє користуватися статистикою аналізу криптовалют не тільки з дек стоп-додатку, а і з мобільних пристроїв.

Гістограма в HTML статистики ціни криптовалют курсу BTC-USD

продемонстрована на рис. 4.11.

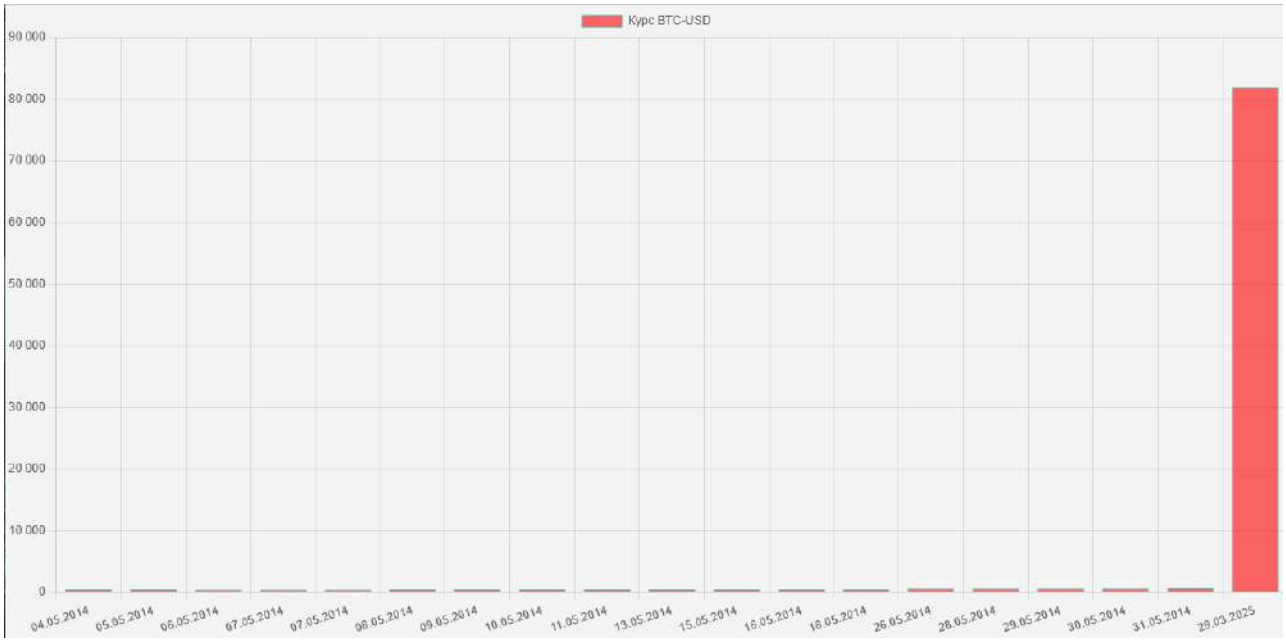


Рис. 4.11. Гістограма в HTML курсу криптовалют BTC-USD

Гістограма в HTML статистики ціни криптовалют курсу LTC-BTC продемонстрована на рис. 4.12.

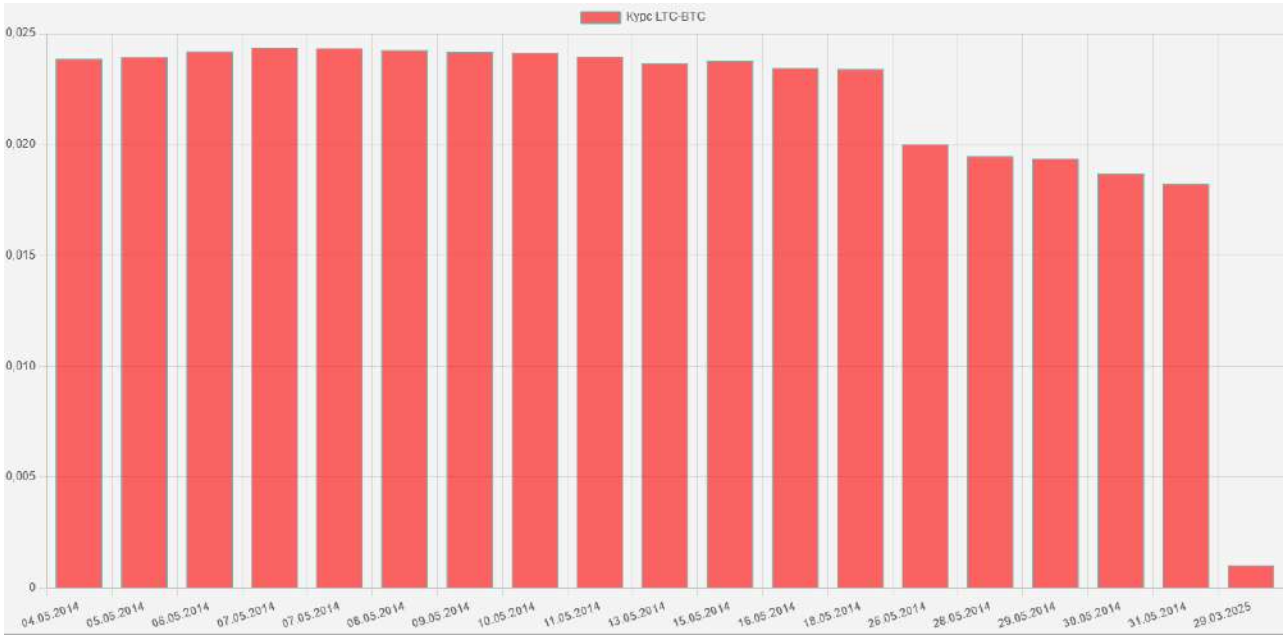


Рис. 4.12. Гістограма в HTML курсу криптовалют LTC-BTC

Гістограма в HTML статистики ціни криптовалют курсу LTC-USD продемонстрована на рис. 4.13.

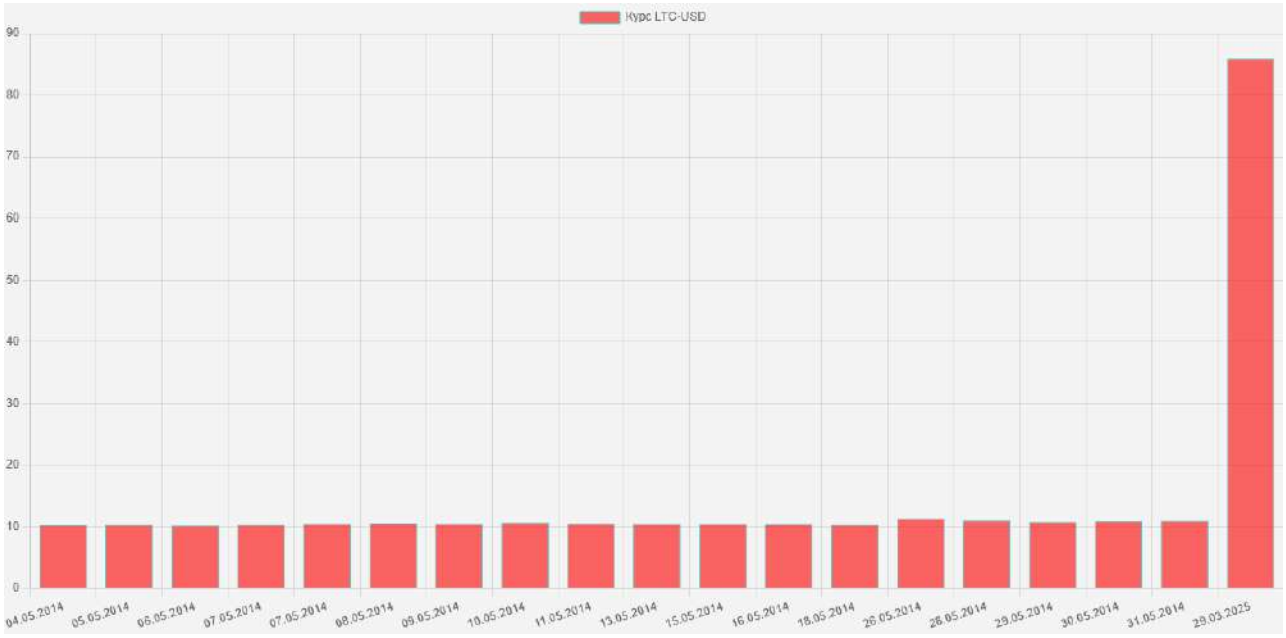


Рис. 4.13. Гістограма в HTML курсу криптовалют LTC-USD

Висновки до розділу 4

Отже, у четвертому розділі розроблено калькулятор криптовалют на мові програмування C# Visual Studio 2019. В даному проекті передбачено динамічне оновлення курсів з ресурсу **apirone.com** за допомогою класу **WebClient**. Для зчитування поточного курсу криптовалюти bitcoin розроблено універсальну функцію GetValuta. За допомогою елементу керування Chart реалізовано виведення статистичної інформації по курсам криптовалют у 2D-графіці. Інформація вивдиться по курсам BTC-USD (кількість доларів за 1BTC), LTC-BTC (кількість BTC за 1LTC), LTC-USD (кількість доларів за 1LTC). Розроблене програмне забезпечення враховує витрати на електроенергію, а також всі види комісій при розрахунку доходності криптовалют.

ВИСНОВКИ

У ході вирішення поставлених на початку нашої роботи задач отримані такі основні результати:

1. Розроблено калькулятор криптовалют на мові програмування C# Visual Studio 2019. В даному проекті передбачено динамічне оновлення курсів з ресурсу **apirone.com** за допомогою класу **WebClient**. Для зчитування поточного курсу криптовалюти bitcoin розроблено універсальну функцію GetValuta.

Для роботи з базою даних використано технологію ADO.Net. Для експорту даних розроблено універсальну статичну функцію ExportXls, яка виводить довільне представлення таблиці DataView.

2. За допомогою елементу керування Chart реалізовано виведення статистичної інформації по курсам криптовалют у 2D-графіці. Інформація вивдиться по курсам BTC-USD (кількість доларів за 1BTC), LTC-BTC (кількість BTC за 1LTC), LTC-USD (кількість доларів за 1LTC).

3. На мові C++ з використанням бібліотеки OpenGL реалізовано виведення 3D-графіки по курсам криптовалют. При цьому передбачено переміщення камери відносно сцени, накладення текстур, роботу зі світлом, роботу з тінями, виведення тексту тощо.

Розроблене програмне забезпечення враховує витрати на електроенергію, а також всі види комісій при розрахунку доходності криптовалют.

Застосування універсальних Java-скриптів на основі елементу керування Chart.js дають можливість гнучко передавати дані з бази до створюваного html-файлу та користуватися ним в будь-яких браузерах. Це дозволило зробити гістограми в html для аналізу різних варіантів криптовалюти, зокрема BTC та LTC.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Криптовалюта [Електронний ресурс] – Режим доступу до матеріалу: http://ru.wikipedia.org/wiki/Криптовалюта.

2. Bitcoin [Електронний ресурс] – Режим доступу до матеріалу: http://ru.wikipedia.org/wiki/Bitcoin.

3. Litecoin [Електронний ресурс] – Режим доступу до матеріалу: http://ru.wikipedia.org/wiki/Litecoin.

4. Дэвис Т. OpenGL. Руководство по программированию. Библиотека программиста. / Дэвис Т., Нейдер Дж., Шрайндер Д. – Санкт-Петербург: «Питер», 2006. – 863 с.

5. Зеленский А.С. Методические указания для самостоятельного изучения программирования баз данных на Visual Basic с использованием объекта DAO. / А. Зеленский, С. Баран. – Кривой Рог: КЭИ, 2004. – 46 с.

6. Зеленский А.С. Методические указания для самостоятельного изучения программирования баз данных на Visual Basic с использованием элемента управления ADO / А.С. Зеленский. – Кривой Рог: КЭИ, 2005. – 59 с.

7. Зеленский А.С. Методические указания для самостоятельного изучения работы с базами данных на VISUAL C++ с использованием объектов ActiveX Data Object (ADO). / Зеленский А.С., Лысенко В.С., Баран С.В. – Кривой Рог: КЭИ, 2008. 54 с.

8. Зеленский А.С. Методические указания для самостоятельного изучения работы с базами данных на Visual C++ с использованием объектов ActiveX Data Object (ADO). / Зеленский А.С., Лысенко В.С., Баран. С.В. – Кривой Рог: КЭИ, 2008. – 65 с.

9. Зеленский А.С. Методические указания использования объектов ADO при работе с базами данных на Visual C++6.0. / Зеленский А.С., Лысенко В.С., Баран. С.В. – Кривой Рог: КЭИ, 2008. – 65 с.

10. Зеленский А.С. Методические указания к выполнению лабораторных и индивидуальных работ на основе примеров разработки программного

обеспечения в VISUAL C++ 6.0. / Зеленский А.С., Лысенко В.С., Баран С.В. – Кривой Рог: КЭИ, 2007. – 63 с.

11.Зеленский А.С. Методические указания к выполнению лабораторных и индивидуальных работ на основе примеров разработки программного обеспечения в VISUAL C++ 6.0. / Зеленский А.С., Лысенко В.С., Баран. С.В. – Кривой Рог: КЭИ, 2008. – 46 с.

12. Зеленський О.С. Методичні вказівки для самостійного вивчення основ класичного програмування на мові С++, з дисциплін "Інструментальні засоби прикладного програмування", "Об’єктно-орієнтоване програмування", "Прикладне програмне забезпечення". / А. Зеленський, В. Лисенко. – Кривий Ріг: КЕІ КНЕУ, 2008. – 94 с.

13. Малик. С. Microsoft ADO.NET 2.0 для профессионалов. Учебное руководство. / С. Малик. – М.: Издательский дом "Вильямс", 2006. – 560 с.

14. Нейгел К. Visual C#. Учебное пособие. / К. Нейгел., К. Уотсон. – М.: Диалектика, 2010. – 710 с.

15. Нэш Т. C# 2010: ускоренный курс для профессионалов. / Т. Нэш. – М.: Издательский дом "Вильямс", 2010. – 592 с.

16. Олафсен Ю. MFC и Visual C++ 6. Энциклопедия программиста. / Олафсен Ю., Скрайбнер К., Уайт К. – Санкт-Петербург: ООО "ДиаСофтЮП", 2003. – 992 с.

17. Павловская Т.А. C # и C++. Программирование на языке высокого уровня. Учебное пособие. / Т.А. Павловская. – Санкт-Петербург: «Питер», 2003. – 463 с.

18. Постолит А.В. Visual Studio.NET. / А.В. Постолит – СПб.: БХВ-Петербург, 2003. –544 с.

19. Ричард С. OpenGL. Суперкнига. Учебное пособие. / С. Ричард. – [3-тє вид.]. – М.: Издательский дом "Вильямс", 2006. – 1040 с.

20. Рост Р. OpenGL. Трёхмерная графика и язык программирования шейдеров. Учебное пособие. / Р. Рост. – Санкт-Петербург: «Питер», 2005. – 863 с.

21. Секунов Н.Ю. Самоучитель Visual C++ 6. Учебное пособие. / Н.Ю. Секунов. – Санкт-Петербург: БХВ, 1999. – 960 с.

22. Скит Дж. C#: программирование для профессионалов. / Дж. Скит. – [2-ге вид.]. – М.: Издательский дом "Вильямс", 2011. – 544 с.

23. Тарасов И.А. Основы программирования OpenGL. Учебное пособие. / И.А. Тарасов. – М.: Издательский дом "Вильямс", 2001. – 778 с.

24. Тихомиров Ю.В. OpenGL. Программирование трёхмерной графики. Учебное пособие. / Ю.В. Тихомиров. – [2-ге вид.]. – СПб.: БВЧ-Петербург, 2002. – 304 с.

25. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. / Э. Троелсен. – [3-тє вид.]. – М.: Издательский дом "Вильямс", 2010. – 1392 с.

26. Херн Д. Компьютерная графика и стандарт OpenGL. Учебное пособие. / Д. Херн, М. Паулин Бейкер. – [3-тє вид.]. – М.: Издательский дом "Вильямс", 2005. – 1168 с.

27. Холзнер С. Visual C++. Учебное пособие. / С. Холзнер. – Санкт-Петербург: ЗАО "Издательство «Питер»", 1999. – 576 с.

ДОДАТКИ

ДОДАТОК А

Лістинг універсальної функції оновлення курсів валют з мережі Інтернет GetValuta

```
//Статична функція, якою будемо користуватися в багатьох формах
        //Функція для зчитування даних з інтернету (ціни на криптовалюти)
        //Зчитування валют відбувається з використанням класу WebClient та api сервису
https://apirone.com/api
        public static bool GetValuta(double[] tarif)
        {

            string stroka1 = "https://apirone.com/api/v1/ticker?currency=btc";
            string stroka2 = "https://apirone.com/api/v1/ticker?currency=ltc";
            string[] mas = { "BSD", "last" };

            try
            {
                ServicePointManager.Expect100Continue = true;
                ServicePointManager.SecurityProtocol = (SecurityProtocolType)3072;

                ServicePointManager.ServerCertificateValidationCallback = delegate { return
true; };

                using (WebClient client = new WebClient())
                {
                    byte[] buffer;
                    string str3;

                    do
                    {
                        try
                        {
                            //MessageBox.Show("1111");
                            buffer = client.DownloadData(stroka1);
                            //MessageBox.Show("2222");
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.Message);
                            buffer = null;
                            return false;
                        }
                        if (buffer == null)
                        {
                            Thread.Sleep(0xea60);
                        }
                    }
                    while (buffer == null);

                    str3 = Encoding.UTF8.GetString(buffer);

                    // MessageBox.Show(str3);

                    //Рзбір рядка з криптовалютою
                    int idx1, idx2;
                    string ch;
                    ////////////////////////////////////////
                    idx1 = str3.IndexOf(mas[0]);
                    idx1 += mas[0].Length + 4;
                    idx1 = str3.IndexOf(mas[1], idx1) + 7;
                    idx2 = str3.IndexOf(" ", idx1);
                    ch = str3.Substring(idx1, idx2 - idx1);
```

```
                ch = ch.Replace('.', ',');
                tarif[0] = double.Parse(ch);
                ////////////////////////////////////////
            }
        }

        catch
        {
            return false;
        }

        try
        {
            using (WebClient client = new WebClient())
            {
                byte[] buffer;
                string str3;

                //int num;
                do
                {
                    try
                    {
                        buffer = client.DownloadData(stroka2);
                    }
                    catch
                    {
                        buffer = null;
                        return false;
                    }
                    if (buffer == null)
                    {
                        Thread.Sleep(0xea60);
                    }
                }
                while (buffer == null);

                str3 = Encoding.UTF8.GetString(buffer);

                //Розбір рядка з криптовалютою
                int idx1, idx2;
                string ch;
                ////////////////////////////////////////
                idx1 = str3.IndexOf(mas[0]);
                idx1 += mas[0].Length + 4;
                idx1 = str3.IndexOf(mas[1], idx1) + 7;
                idx2 = str3.IndexOf(" ", idx1);
                ch = str3.Substring(idx1, idx2 - idx1);
                ch = ch.Replace('.', ',');
                tarif[2] = double.Parse(ch);
                tarif[1] = tarif[2] / tarif[0];
                ////////////////////////////////////////
            }
        }

        catch
        {
            return false;
        }
        return true;
    }
```

ДОДАТОК Б

Лістинг універсальних функцій експорту даних до MS Excel, а також до HTML

```
//Универсальная функция экспорта в Excel произвольных данных
        public static bool ExportXLS(DataView dt)
        {
            try
            {
                Excel.Application ExcelApp = new Excel.Application();

                Excel.Workbook book = ExcelApp.Workbooks.Add(Type.Missing);
                ExcelApp.SheetsInNewWorkbook = 1;
                Excel.Worksheet sheet = (Excel.Worksheet)ExcelApp.Sheets.get_Item(1);
                sheet.Name = "Мой лист";

                Excel.Range range, range1;

                range = sheet.Cells;

                string str = (char)('A' + dt.Table.Columns.Count - 1) + "";

                //A1 - D1
                range1 = range.get_Range("A1", ((char)('A' + dt.Table.Columns.Count - 1)) +
"1");

                range1.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
                range1.Font.Bold = true;

                int row = 1, col;
                string v;

                Excel.Border border;
                border = range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeTop);

                //Рисование шапки
                for (col = 0; col < dt.Table.Columns.Count; col++)
                {
                    v = dt.Table.Columns[col].ColumnName;
                    if (v == "") continue;
                    range.set_Item(row, col + 1, v);
                }


                row++;

                foreach (DataRowView rr in dt)
                {
                    for (col = 0; col < dt.Table.Columns.Count; col++)
                    {
                        v = rr[col].ToString();
                        if (v == "") continue;
                        double dd;

                        if (dt.Table.Columns[col].DataType.ToString() == "System.DateTime")
                            v = Convert.ToDateTime(rr[col]).ToShortDateString();

                        if (double.TryParse(v, out dd) == true)
                            range.set_Item(row, col + 1, dd);
                        else
                            range.set_Item(row, col + 1, v);
                    }
```

```
                    row++;
                }

                range1 = range.get_Range("A1", ((char)('A' + dt.Table.Columns.Count - 1)) +
(dt.Count + 1).ToString());
                range1.VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeBottom).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeLeft).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeRight).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeTop).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlInsideHorizontal).LineStyle =
1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlInsideVertical).LineStyle =
1;

                range1.Font.Name = FormGl.font_text.Name;
                range1.Font.Size = FormGl.font_text.SizeInPoints;
                range1.Font.Color = FormGl.cvet_text;

                range1 = range.get_Range("A1", ((char)('A' + dt.Table.Columns.Count - 1)) +
"1");
                range1.Font.Color = FormGl.cvet_hapka;


                sheet.Columns.AutoFit();
                ExcelApp.Visible = true;

            }

            catch (Exception)
            {
                return false;
            }

            return true;
        }

        //Универсальная функция экспорта в HTML произвольных данных
        // ф-я экспорта любой таблицы в файл html.
        // Вход:
        // filename - имя html-файла в который производится экспорт
        // dt - представление записей DataView
        // zaglav - заголовок таблицы
        // ff - задаваемый шрифт
        // cvetz - цвет заголовка таблицы
        // cveth - цвет шапки таблицы
        // cvett - цвет текста ячеек таблицы


    public static bool Export_Html(string filename,DataView dt,string zaglav,Font ff, Color
cvetz, Color cveth, Color cvett)
    {

      int i,kol_p;
      string vt;

      try
      {

          StreamWriter writer = new StreamWriter(filename, false, Encoding.Default);
          writer.Write("<html>\n");
          writer.Write("<head>\n");
```

Продовження додатку Б

```
            writer.Write("<meta http-equiv=Content-Type \"content=text/html; charset=windows-
1251\">\n");

            writer.Write("<title>");
            writer.Write("Экспорт таблицы");
            writer.Write("</title>\n");
            writer.Write("<style>\n");

            writer.Write("table{border-style:solid; ");
            writer.Write("border-top-style:none;font-family:{0};font-size:{1}pt;", ff.Name,
(int)ff.SizeInPoints);


            //"{0:x2}{1:x2}{2:x2}"
            writer.Write("background-color:white;}\np.zag{color:");
            writer.Write("#{0:x2}{1:x2}{2:x2}; text-align:center; font-weight:bold; font-
family:{3}; font-size:{4}pt; color:#{0:x2}{1:x2}{2:x2};",
                cvetz.R, cvetz.G, cvetz.B, ff.Name, (int)ff.SizeInPoints);
            writer.Write("}\n");

            writer.Write("td{border-style:none;border-right-style:solid;border-top-
style:solid;border: solid windowtext 1px;}\n");
            writer.Write("td.text{color:");
            writer.Write("#{0:x2}{1:x2}{2:x2};", cvett.R, cvett.G, cvett.B);
            writer.Write("}\n");

            writer.Write("td.shapka{color:");
            writer.Write("#{0:x2}{1:x2}{2:x2}; font-weight:bold;", cveth.R, cveth.G, cveth.B);
            writer.Write("}\n");

            ////////////////////////

            writer.Write("</style>\n");
            writer.Write("</head>\n");
            writer.Write("<body>\n");

            writer.Write("<center>\n");
            writer.Write("<p class=zag align=center>\n{0}\n</p>\n", zaglav);

            writer.Write("<table width=90% border=1 bordercolor=#000000 cellspacing=0
cellpadding=0 valign=center style='width:90.0%;border-collapse:collapse;border:none;'>\n");

            kol_p = dt.Table.Columns.Count;

            writer.Write("<tr>\n");

            //Экспорт шапки
            for (i = 0; i < kol_p; i++)
            {
                vt = string.Format("    <td class=shapka width={0:F2}% align=center>", 100.0 /
kol_p);
                vt = vt.Replace(',', '.');
                writer.Write(vt);
                writer.Write(dt.Table.Columns[i].ColumnName);
                writer.Write("</td>\n");
            }

            writer.Write("</tr>\n");

            //Экспорт данных
            foreach (DataRowView row in dt)
            {
                writer.Write("<tr>\n");
```

Продовження додатку Б

```
            for (i = 0; i < kol_p; i++)
            {
                writer.Write("    <td class=text align=center>");
                writer.Write(" ");

                vt = row[i].ToString();

                if (dt.Table.Columns[i].DataType.ToString() == "System.DateTime")
                    vt = Convert.ToDateTime(row[i]).ToShortDateString();

                if (vt != "")
                {
                    writer.Write(vt);
                    writer.Write(" ");
                }
                writer.Write("</td>\n");
            }

            writer.Write("</tr>\n");
        }


        writer.Write("</table>\n<br>\n");

        writer.Write("\n");
        writer.Write("</body>\n");
        writer.Write("</html>\n");

        writer.Close();
        return true;
    }
    catch (Exception)
    {
        return false;
    }
}
```