МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Предметно-циклова комісія інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Голова ПЦК _____
спеціаліст в/к Сабанов С.О.

ВИПУСКНА РОБОТА МОЛОДШОГО СПЕЦІАЛІСТА

РОЗРОБКА ПРОГРАМИ СИСТЕМИ ОБЛІКУ НЕСПРАВНОСТЕЙ КОМП'ЮТЕРНОГО ОБЛАДНАННЯ ПІДПРИЄМСТВА

Виконав
ст. гр. ІПЗ-119К9 _____ Зарудинський О.Ю.

Керівник
викладач _____ Костерной Д.О.

Запоріжжя
2023

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Предметно-циклова комісія інформаційних технологій

ЗАТВЕРДЖУЮ

Голова ПЦК спеціаліст в/к

Сабанов С.О._____

«21» лютого 2023 року

ЗАВДАННЯ

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

студенту гр. *ІПЗ-119К9*

спеціальності: *121 – Інженерія програмного забезпечення*

Зарудинському Олександру Юрійовичу

(прізвище, ім'я, по батькові)

1. Тема: *Розробка програми системи обліку несправностей комп'ютерного обладнання підприємства*

затверджена наказом по інституту: № 09.2-14 від 20 лютого 2023 року

2. Термін здачі студентом закінченої роботи: 20 червня 2023 року

3. Перелік питань, що підлягають розробці:

1. *Провести огляд літератури та інтернет-джерел, присвячених тематиці випускної роботи.*
2. *Виконати огляд найбільш популярних програм діагностування комп'ютерного обладнання.*
3. *Розглянути вимоги до облікових карток обчислювальної техніки.*
4. *Розробити програму «Система обліку комп'ютерного обладнання».*
5. *Реалізувати алгоритм роботи програми.*
6. *Протестувати розробку, надати керівництво користувача.*
7. *Оформити результати роботи у вигляді пояснювальної записки, що відповідає стандартам підприємства щодо оформлення випускних робіт.*

4. Календарний графік

№ етапу	Зміст	Термін виконання	Готовність (%), підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Збір практичного матеріалу за темою	16.01.23-17.02.23		
2	І атестація. І розділ випускної роботи	27.03.23-01.04.23		
3	ІІ атестація. ІІ розділ випускної роботи	01.05.23-06.05.23		
4	ІІІ атестація. ІІІ розділ випускної роботи, висновки та рекомендації, додатки, реферат.	29.05.23-03.06.23		
5	Перевірка випускної роботи на оригінальність	15.05.23-12.06.23		
6	Доопрацювання випускної роботи, підготовка презентації, отримання відгуку керівника та рецензії	05.06.23-10.06.23		
7	Попередній захист випускної роботи	12.06.23-18.06.23		
8	Подача випускної роботи на кафедру	за 3 дні до захисту		
9	Захист випускної роботи	19.06.23-24.06.23		

Дата видачі завдання: 22 лютого 2023 р.

Керівник випускної роботи_____ Костерной Д.О.
(підпис)

Завдання прийняв до виконання_____ Зарудинський О.Ю.
(підпис студента)

РЕФЕРАТ

Випускна робота молодшого спеціаліста містить 75 сторінок, 2 таблиці, 35 рисунків, 21 бібліографічне посилання.

Об’єкт розробки – програмний облік несправностей комп’ютерного обладнання.

Метою роботи є розробка програми, яка дозволяє діагностувати несправності комп’ютерного обладнання підприємства.

У випускній роботі розглядаються класифікація і задачі діагностування комп’ютерної техніки, здійснюється їх аналітичний огляд. Детально описано процес розробки програми «Система обліку комп’ютерного обладнання».

У даній дипломній роботі мовою програмування обраний C++. В якості середовища розробки найкраще підходить пакет Microsoft Visual Studio 2008, який допомагає оптимізувати та спрощує процес розробки високоефективних програм. Для отримання достовірної інформації про використання застосовуваних технологій і підходів, тестування проводилося на понад 50 комп’ютерах різної конфігурації.

C++, MICROSOFT VISUAL STUDIO, ДІАГНОСТИКА, НЕСПРАВНОСТІ, КОМП’ЮТЕРНЕ ОБЛАДНАННЯ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Слово / словосполучення	Скорочення
C	
Component Object Model	COM
S	
Standard Template Library	STL
W	
Windows Management Instrumentation	WMI

ВСТУП

Розробка програми для системи обліку несправностей комп’ютерного обладнання підприємства є важливим завданням, яке допомагає забезпечити ефективне управління технічними проблемами і зберегти продуктивність комп’ютерних ресурсів. Така програма дозволяє систематизувати та відстежувати інформацію про виявлені несправності, а також вести контроль над процесом їх вирішення.

Розробка програми системи обліку несправностей комп’ютерного обладнання підприємства по суті виявляється діагностикою системи, тобто це процес ретельного тестування всіх компонентів комп’ютера з метою визначення відповідності їх характеристик заявленим виробником. Цей процес також включає вимірювання реальної продуктивності, такої як швидкість роботи, і порівняння цих показників з еталонними значеннями.

У випускній роботі розглядаються класифікація і задачі діагностування комп’ютерної техніки, здійснюється їх аналітичний огляд. Детально описано процес розробки програми «Система обліку комп’ютерного обладнання».

У даній дипломній роботі мовою програмування обраний C++. В якості середовища розробки найкраще підходить пакет Microsoft Visual Studio 2008, який допомагає оптимізувати та спрощує процес розробки високоефективних програм. Для отримання достовірної інформації про використання застосовуваних технологій і підходів, тестування проводилося на понад 50 комп’ютерах різної конфігурації.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Класифікація та задачі програм діагностування обчислювальної техніки

Розробка програми для системи обліку несправностей комп’ютерного обладнання підприємства є важливим завданням, яке допомагає забезпечити ефективне управління технічними проблемами і зберегти продуктивність комп’ютерних ресурсів. Така програма дозволяє систематизувати та відстежувати інформацію про виявлені несправності, а також вести контроль над процесом їх вирішення.

Основні функціональні можливості програми системи обліку несправностей комп’ютерного обладнання можуть включати:

Реєстрацію несправностей: користувачі можуть додавати записи про виявлені проблеми, вказуючи деталі, такі як тип несправності, дата, час, місце виникнення і опис проблеми.

Класифікацію несправностей: можливість встановлення категорій або тегів для розподілу несправностей за типами або відділами.

Стеження за статусом: система дозволяє відстежувати поточний статус кожної несправності, наприклад, «в очікуванні вирішення», «у процесі виправлення» або «вирішено».

Призначення відповідальних осіб: можливість призначати відповідальних співробітників для вирішення кожної несправності і встановлення термінів виконання.

Звіти і аналітику: генерація звітів про кількість несправностей, тривалість їх вирішення, вартість ремонтних робіт та інші аналітичні дані, що допомагають управлінню приймати обґрунтовані рішення.

Сповіщення та нагадування: можливість надсилання сповіщень та нагадувань про важливі терміни вирішення несправностей, що допомагає забезпечити своєчасне реагування.

Розробка такої програми допомагає підприємствам зберігати належну документацію про несправності, ефективно планувати та координувати ремонтні роботи, а також забезпечує можливість аналізу та вдосконалення процесу управління несправностями.

Розробка програми системи обліку несправностей комп'ютерного обладнання підприємства по суті виявляється діагностикою системи, тобто це процес ретельного тестування всіх компонентів комп'ютера з метою визначення відповідності їх характеристик заявленим виробником. Цей процес також включає вимірювання реальної продуктивності, такої як швидкість роботи, і порівняння цих показників з еталонними значеннями.

В залежності від різних потреб тестування обчислювальної техніки, можна виділити кілька основних типів тестування:

Навантажувальне тестування, яке використовується для оцінки роботи комп'ютера при підвищеному навантаженні. Цей тип тестування дозволяє виявити, як система поводиться під час інтенсивного використання та перевірити її стабільність.

Стрес-тестування, яке спрямоване на перевірку межових можливостей системи шляхом створення екстремальних умов навантаження. Воно дозволяє виявити проблеми, пов'язані з перегрівом, падінням продуктивності, нестабільною роботою апаратних компонентів тощо.

Тестування продуктивності, яке має на меті оцінити швидкість роботи системи і здатність виконувати завдання з необхідною ефективністю. Цей тип тестування дозволяє виявити можливі проблеми з продуктивністю, такі як повільна відповідь, довгий час завантаження тощо.

При веденні господарської діяльності ведеться облік матеріально–технічних ресурсів. Одним із ресурсів в даний час є обчислювальна техніка, яка має певну вартість і знаходиться на балансі підприємств. Обчислювальна техніка має характеристики, які необхідно враховувати при адмініструванні парку обчислювальної техніки. Особлива увага до обліку обчислювальної техніки приділяється в бюджетних організаціях. Залежно від структури

організації ведеться різна форма карток обліку обчислювальної техніки. У більшості бюджетних організацій картки обліку обчислювальної техніки представляють собою таблицю або список з інвентарними номерами та опис характеристик обчислювальних пристроїв. Крім ведення карток обліку обчислювальної техніки, необхідно стежити за їх актуальністю. Для таких цілей існує інвентаризація, в процесі якої звіряється обчислювальна техніка за фактом з поточними картками обліку обчислювальної техніки.

Для діагностики та тестування комплектуючих і, взагалі всього комп'ютера, є спеціальні програми. Одна з основних задач зазначених програм: відображати основні властивості комплектуючих, а саме: модель, ім'я виробника, об'єм накопичувальних пристроїв, поточну версію і т.д. Також можна протестувати повний набір комплектуючих, визначити їх мінімальні та максимальні можливості.

Найбільш популярними серед програм для тестування комп'ютера є: Everest, WinAudit, CPU-Z, GPU-Z, PC Wizard, HDDScan. Усі вони здатні визначити та відобразити характеристики програмних та апаратних модулів, а саме: інформацію про встановлене програмне забезпечення, версію драйверів, налаштування TCP/IP, параметри монітору, інформацію про відео підсистеми, встановлені принтери, версію BIOS, системні пристрої, процесори, інформацію про бази даних та багато іншого.

1.2. Аналітичний огляд програм діагностування комп'ютерного обладнання

1.2.1. Опис та можливості програми Everest

Everest – програма призначена для перегляду інформації про апаратну та програмну конфігурацію комп'ютера (рис. 1.1). Програма Everest, являє собою потужній інструмент для моніторингу багатьох вузлів апаратної частини та програмних модулів комп'ютера.

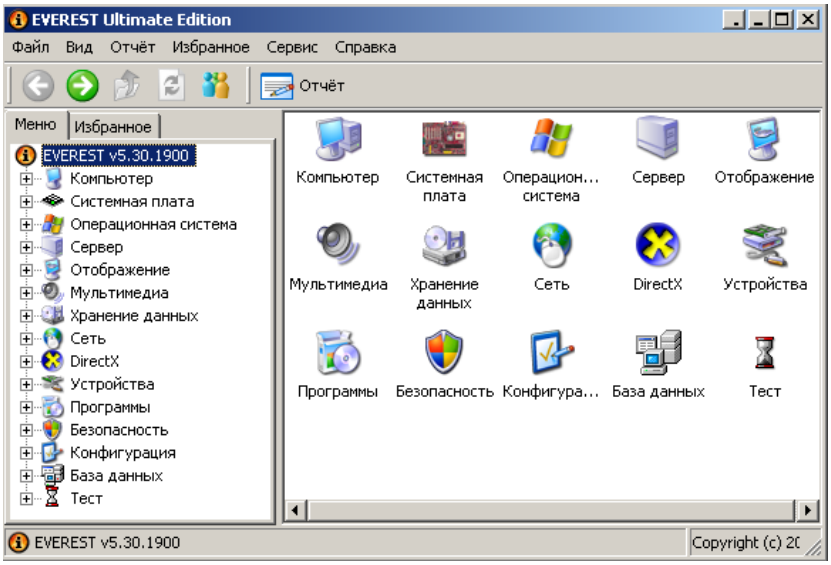


Рисунок 1.1. – Головне вікно програми Everest

Everest має можливість аналізувати конфігурацію комп’ютера та надає докладну інформацію:

– про встановлені в системі пристрої – процесори, системні платі, відеокарти, аудіокарти, модулі пам’яті;

– про набір підтримуваних команд і режимів роботи;

– про виробника компонентів;

– про конфігурацію операційної системи;

– про встановлені драйвери;

– про програми, що завантажуються автоматично;

– про процеси, шо виконуються;

– про ліцензії, які встановлені в системі.

В даній програмі є широкий набір тестів, які порівнюють технічні характеристики з еталонними показниками:

Тестування швидкості зчитування з пам’яті оцінює швидкість передачі даних з оперативної пам’яті до процесора.

– тестування швидкості запису в пам’ять вимірює швидкість передачі даних з процесора до оперативної пам’яті.

– тестування швидкості копіювання в пам’ять оцінює швидкість передачі даних з одного сегмента пам’яті в інший через кеш процесора.

– тестування затримки пам’яті вимірює середній час, який процесор витрачає на зчитування даних з оперативної пам’яті.

– тестування продуктивності процесора CPU Queen визначає його продуктивність у цілочислових операціях шляхом розв’язування класичного «Завдання з ферзями».

– тестування продуктивності процесора CPU PhotoWorxx оцінює продуктивність блоків цілочислових арифметичних операцій, множення та підсистеми пам’яті при виконанні стандартних операцій з RGB-зображеннями.

– тестування продуктивності процесора CPU ZLib вимірює його продуктивність та продуктивність підсистеми пам’яті при створенні ZIP-архівів з використанням популярної відкритої бібліотеки ZLib, використовуючи цілочислові операції.

– тестування продуктивності процесора CPU AES оцінює швидкість процесора при виконанні шифрування за криптоалгоритмом AES. Використовує низькорівневі команди шифрування процесорів VIA C3 і C7, що дає йому перевагу у продуктивності над багатоядерними процесорами Intel та AMD.

– тестування продуктивності блоків процесора FPU Julia вимірює продуктивність блоків процесора, які виконують операції з плаваючою комою з 32-розрядною точністю. Моделює кілька фрагментів фрактала Жюліана і може використовувати інструкції MMX, SSE і 3DNow!.

– тестування продуктивності блоків процесора FPU Mandel вимірює продуктивність блоків процесора, які виконують операції з плаваючою комою з 64-розрядною точністю шляхом моделювання декількох фрагментів фрактала Мандельброта. Може використовувати інструкції SSE2.

Загальну інформацію про комп’ютер можна подивитись у вкладці «Сумарна інформація» в меню «Комп’ютер» (рис. 1.2).

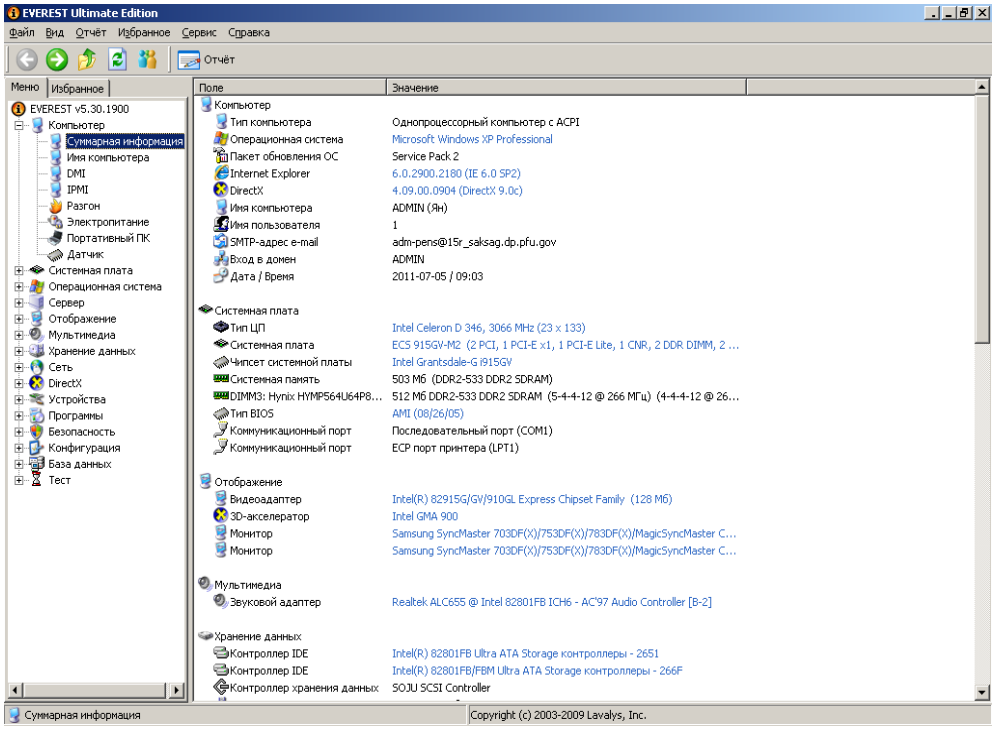


Рисунок 1.2. – Вкладка «Сумарна інформація» в меню «Комп'ютер»

Після закінчення тестування на екран монітора виводяться відповідні результати та конфігурація комп'ютера у порівнянні з еталонними конфігураціями (рис. 1.3).

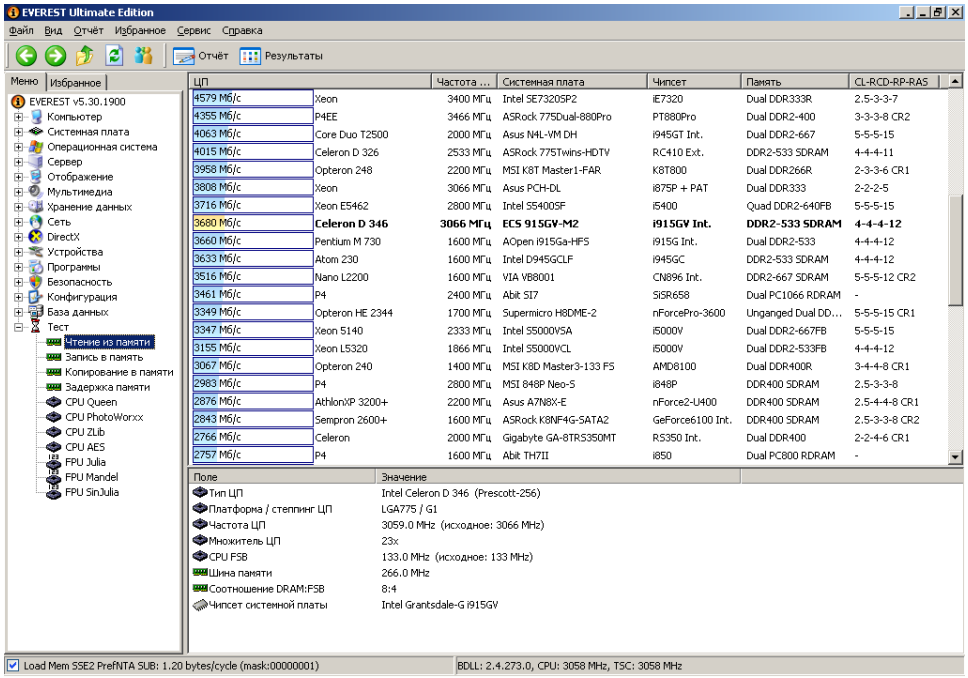


Рисунок 1.3. – Вкладка «Зчитування з пам'яті» в меню «Тест»

При тестуванні слід враховувати версію програми Everest, і тому отримані результати можуть відрізнятися.

Визначення апаратних модулів також залежить від версії програми.

1.2.2. Технічні характеристики програми WinAudit

Програма WinAudit призначена для інвентаризації персонального комп'ютера (рис. 1.4). Також WinAudit надає можливість переглянути інформацію і характеристики програмних та апаратних модулів комп'ютера, а саме: дані про встановлене програмне забезпечення, параметри монітору, технічні характеристики відеоадаптера, дані про встановлені принтери, версію BIOS, системні пристрої, налаштування TCP/IP, назву та кількість процесорів, об'єм пам'яті жорстких та логічних дисків і т.д.

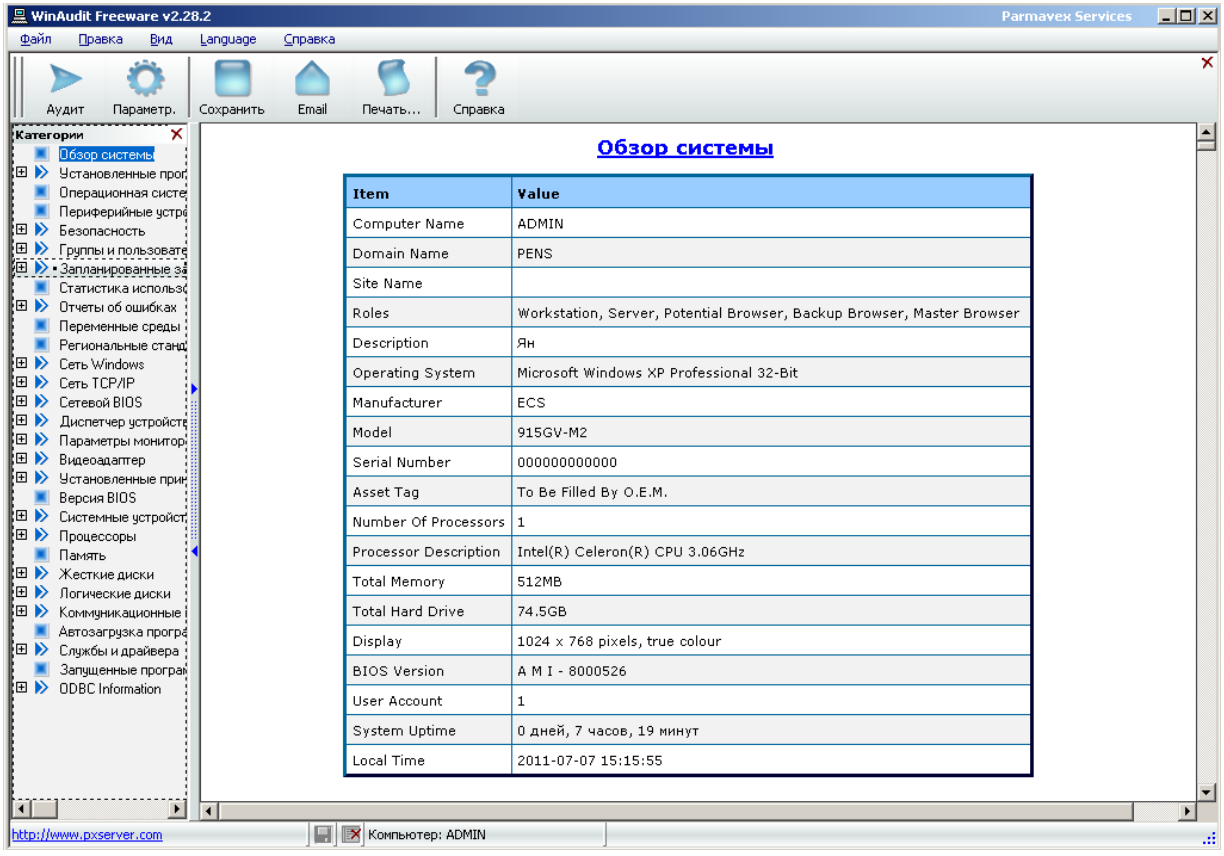


Рисунок 1.4. – Програма WinAudit, та основна інформація в категорії «Огляд системи»

Програма WinAudit проста у використанні, не потребує установки на комп'ютер, що є одною з переваг. Має зрозумілий російський інтерфейс і безліч варіантів для створення файлу звіту в різних форматах.

1.2.3. Загальні відомості про програму GPU–Z

Програма GPU–Z (рис. 1.5) надає детальну інформації про графічний процесор, технологічні процеси, частоти графічного процесора та відеопам'яті, а також відображає версії драйверів, тип і характеристики графічної пам'яті, інформацію з термодатчика на графічному процесорі.

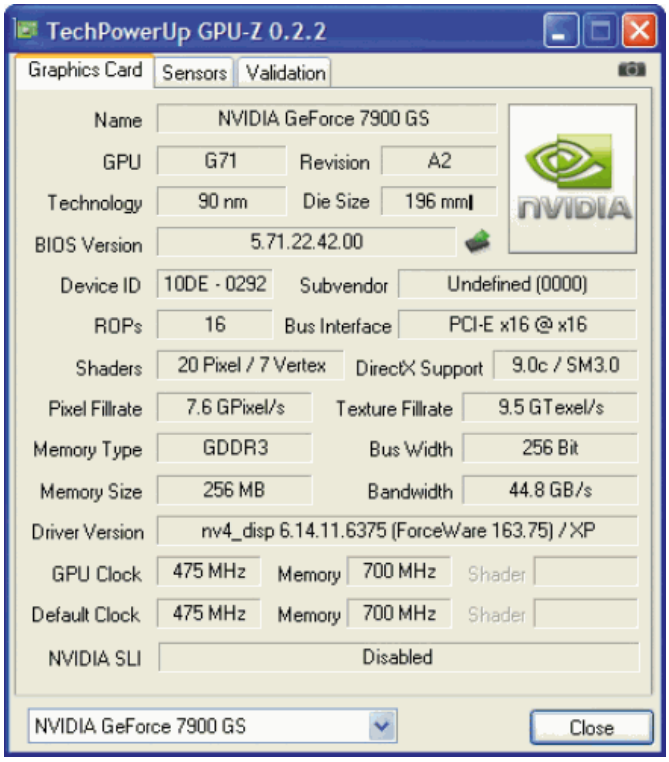


Рисунок 1.5. – Головне вікно програми GPU–Z

Незначним мінусом програми є не завжди коректне визначення деяких моделей відеокарт, але, з огляду на жвавий старт розроблювачів утиліти, можна не сумніватися – всі виявлені помилки будуть виправлені в найкоротший термін.

1.2.4. Опис та можливості програми HDDScan

HDDScan – це програма для низькорівневої діагностики накопичувачів HDD в операційній системі Windows. Програма підтримує диски IDE, SATA, SCSI, RAID масиви, зовнішні накопичувачі USB, Firewire, флеш–карти. Основні можливості програми:

– перегляд інформації S.M.A.R.T.;
– перевірка поверхні диска в трьох режимах: Verify, Read, Erase;
– управління шумовими характеристиками жосткого диску;
– запуск і зупинка шпиндельного двигуна.

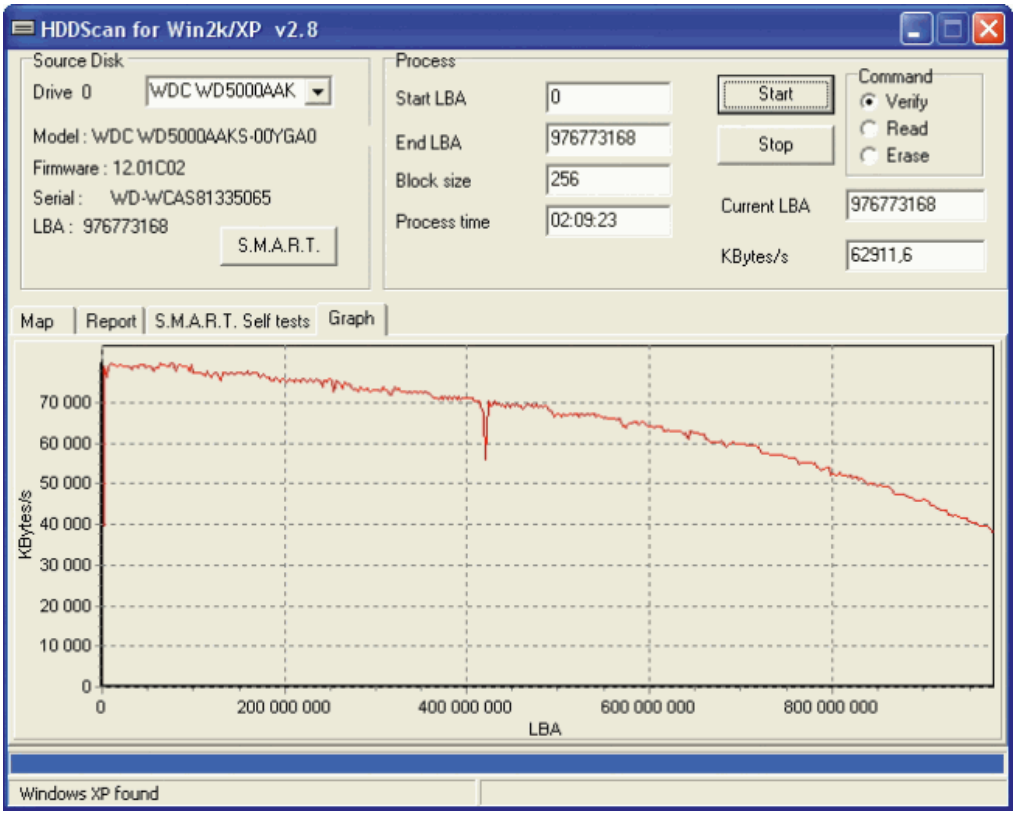


Рисунок 1.6. – Зовнішній вигляд програми HDDScan

Програма HDD Scan не вимагає інсталяції. Інтерфейс програми виконаний повністю англійською мовою. Тим не менше, програма дуже проста у використанні.

1.2.5. Програма CPU-Z та її технічні можливості

CPU-Z – це програмне забезпечення, яке надає інформацію про технічні характеристики різних пристроїв (таких як центральний процесор, відеокарта, материнська плата, оперативна пам’ять) у персональному комп’ютері, що працює під управлінням операційної системи Windows (рис. 1.7).

Основні можливості програми CPU-Z включають:

Визначення параметрів процесора, такі як назва, архітектура, сокет, технологічний процес, напруга живлення ядра, сімейство, підтримувані набори інструкцій, тактова частота, обсяг кешу на різних рівнях, фізична організація кешу, кількість процесорів та ядер.

Визначення параметрів материнської плати, включаючи виробника, модель, чіпсет, південний міст, версію BIOS, графічний інтерфейс та кількість ліній для PCI-Express.

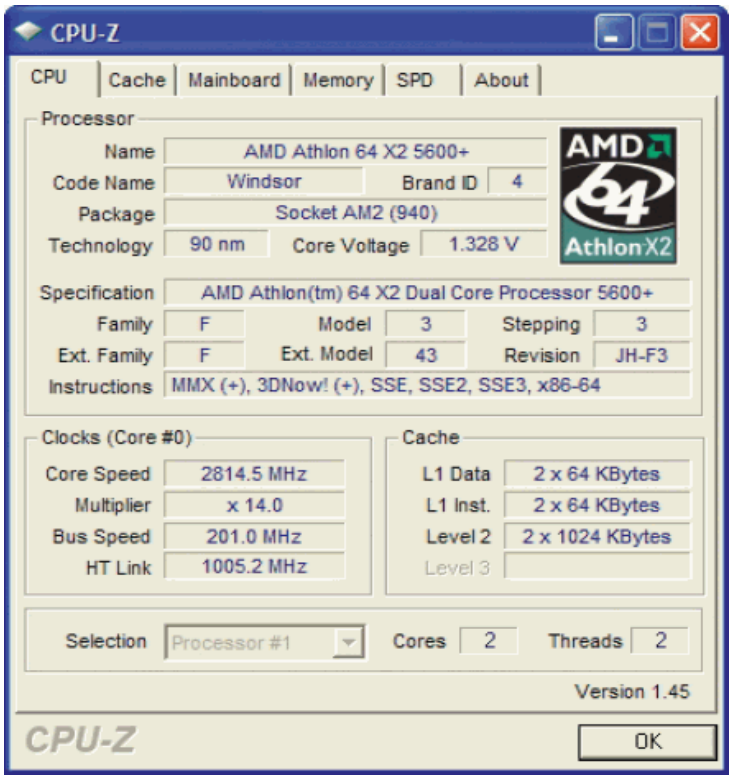


Рисунок 1.7. – Головне вікно програми CPU–Z

Програма CPU-Z має можливість визначати параметри оперативної пам’яті: тип, обсяг, тактова частота, кількість каналів пам’яті.

1.2.6. Функціональні можливості програми PC Wizard 2008

Функціонально програма PC Wizard 2008 (рис. 1.8) складається з п’яти компонентів (Hardware, Configuration, System Files, Resources, Benchmark), разом вони надають максимально точну і детальну характеристику комп’ютера. Розділ Hardware повністю присвячений загальній діагностиці системи та окремих її комплектуючих, напругу живлення, швидкості обертання вентиляторів і показників термодатчиків CPU, GPU та HDD. Надзвичайно корисним є модуль Configuration, де зібрана вся інформація про програмні компоненти комп’ютера. В System Files доступний перегляд завантажувального файлу boot.ini, конфігураційних файлів системи win.ini, system.ini і навіть значень CMOS.

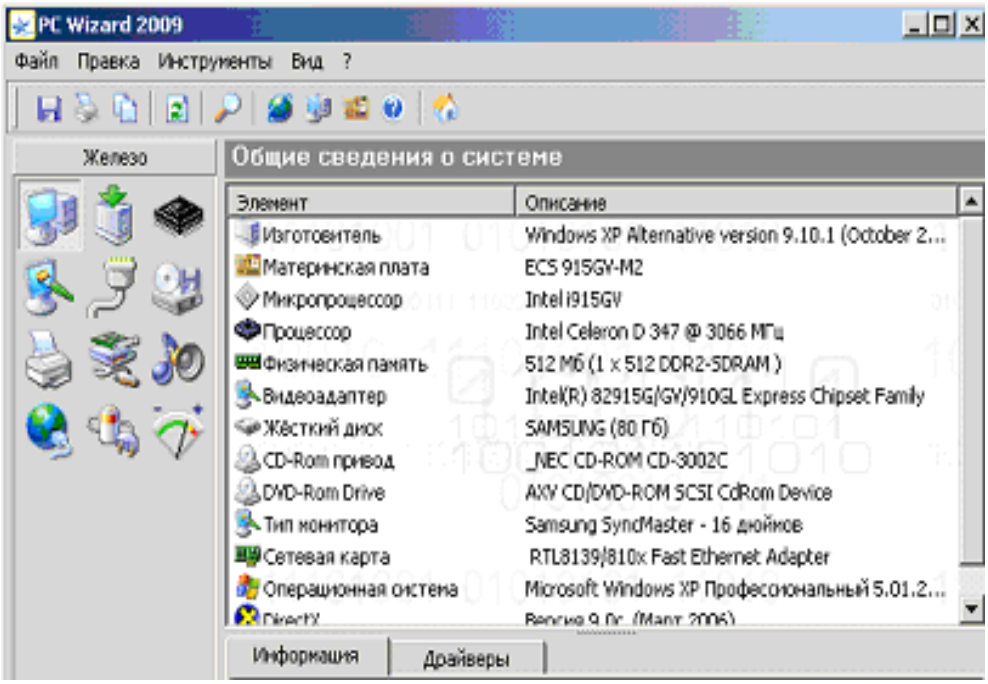


Рисунок 1.8. – Головне вікно програми PC Wizard 2008

При виникненні апаратних конфліктів після встановлення нових комплектуючих є можливість детально вивчити список запитів переривання (IRQ) у закладці Resources.

1.3. Вимоги до облікових карток обчислювальної техніки.

Програма Everest надає можливість зберігати файл звіту у форматі веб–сторінки та текстового файлу. На рис. 1.9 зображена частина файлу звіту у форматі html, повний звіт за вибором профілю «Сумарна інформація» формується обсягом 9 сторінок, але дані, що містяться у ньому можуть виявитися неважливими для обліку обчислювальної техніки.

EVEREST Ultimate Edition

Версия	EVEREST v5.30.1900/ru
Тестовый модуль	2.4.273.0
Домашняя страница	http://www.lavalys.com/
Тип отчёта	Мастер отчётов
Компьютер	ADMIN (Ян)
Генератор	1
Операционная система	Microsoft Windows XP Professional 5.1.2600 (WinXP RTM)
Дата	2011-07-05
Время	09:46

Суммарная информация

Компьютер:

Тип компьютера	Однопроцессорный компьютер с ACPI
Операционная система	Microsoft Windows XP Professional
Пакет обновления ОС	Service Pack 2
Internet Explorer	6.0.2900.2180 (IE 6.0 SP2)
DirectX	4.09.00.0904 (DirectX 9.0c)
Имя компьютера	ADMIN (Ян)
Имя пользователя	1
SMTP-адрес e-mail	adm-pens@15r_saksag.dp.pfu.gov
Вход в домен	ADMIN
Дата / Время	2011-07-05 / 09:46

Системная плата:

Тип ЦП	Intel Celeron D 346, 3066 MHz (23 x 133)
Системная плата	ECS 915GV-M2 (2 PCI, 1 PCI-E x1, 1 PCI-E Lite, 1 CNR, 2 DDR DIMM, 2 DDR2 DIMM, Audio, Video, LAN)
Чипсет системной платы	Intel Grantsdale-G i915GV
Системная память	503 Мб (DDR2-533 DDR2 SDRAM)
DIMM3: Hynix HYMP564U64P8-C4	512 Мб DDR2-533 DDR2 SDRAM (5-4-4-12 @ 266 МГц) (4-4-4-12 @ 266 МГц) (3-3-3-9 @ 200 МГц)
Тип BIOS	AMI (08/26/05)

Рисунок 1.9. – Частина звіту за вибором профілю «Сумарна інформація»

Програма WinAudit зберігає файл звіту у багатьох форматах (pdf, chm, csv, txt, html, xml) та експортує в реальну базу даних, а також є можливість відправляти результати по електронній пошті.

На рисунку 1.10 зображена невелика частина файлу звіту у форматі pdf, в якому чітко видно апаратні і програмні компоненти комп'ютера.

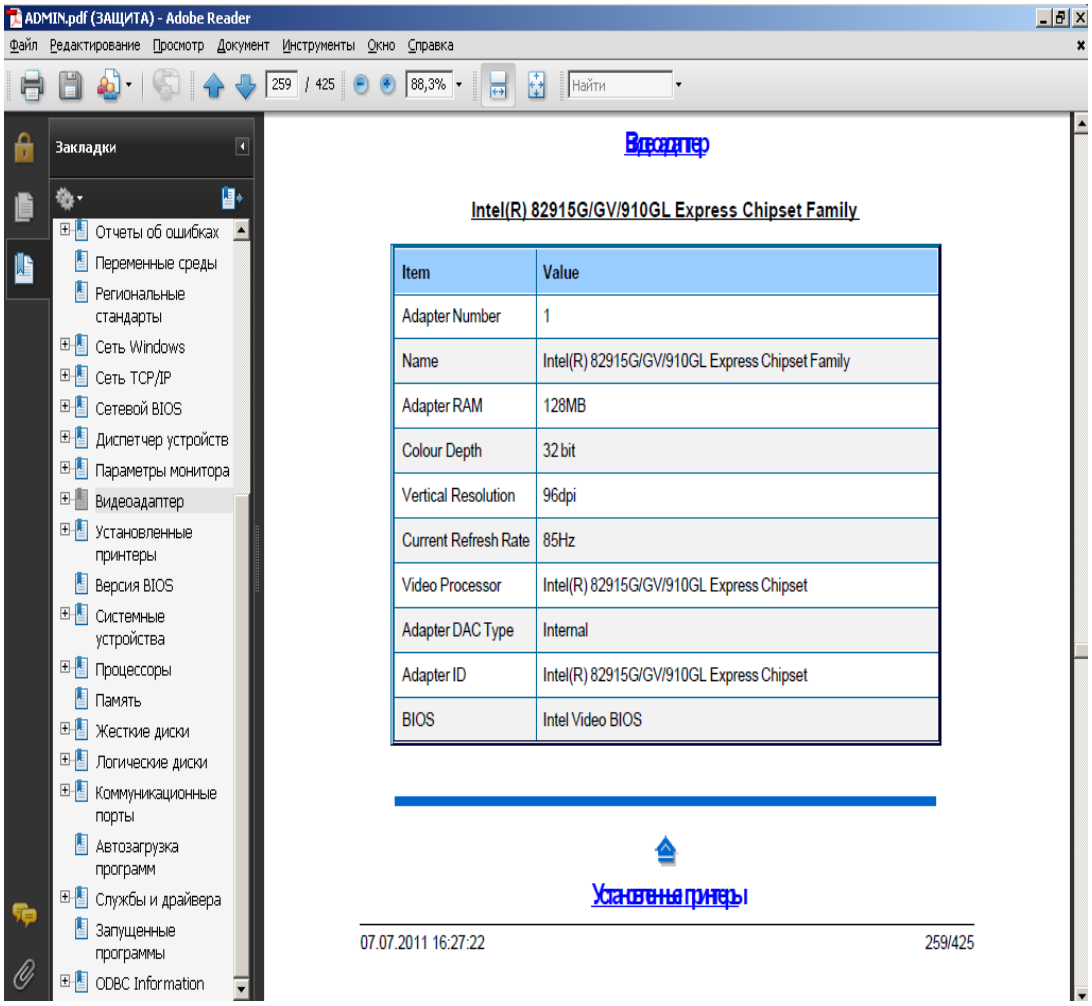
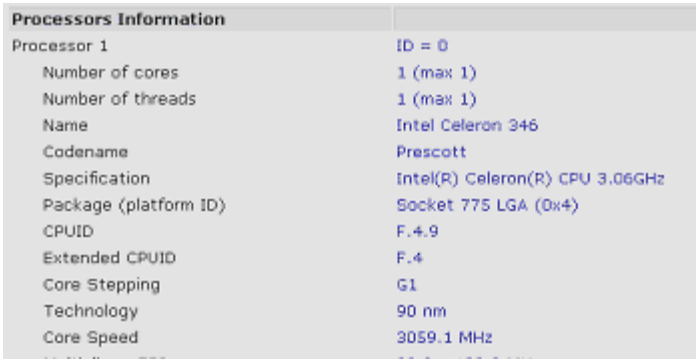


Рисунок 1.10. – Частина звіту у форматі pdf

Програма HDDScan не надає можливості зберігати файл звіту. Подібні утиліти застосовуються лише для конкретного вузлу комп'ютера, відображають детальну інформацію та роблять різноманітні тести для цього вузла. Програма CPU–Z зберігає файл звіту у форматі веб–сторінки та текстового файлу, як показано на рисунку 1.11.



Processors Information	
Processor 1	ID = 0
Number of cores	1 (max 1)
Number of threads	1 (max 1)
Name	Intel Celeron 346
Codename	Prescott
Specification	Intel(R) Celeron(R) CPU 3.06GHz
Package (platform ID)	Socket 775 LGA (0x4)
CPUID	F.4.9
Extended CPUID	F.4
Core Stepping	G1
Technology	90 nm
Core Speed	3059.1 MHz

Рисунок 1.11. – Частина звіту у html форматі, створений програмою CPU–Z

Програма GPU–Z не формує файл звіту, замість цього вона робить знімок з екрану та зберігає у форматі зображення .

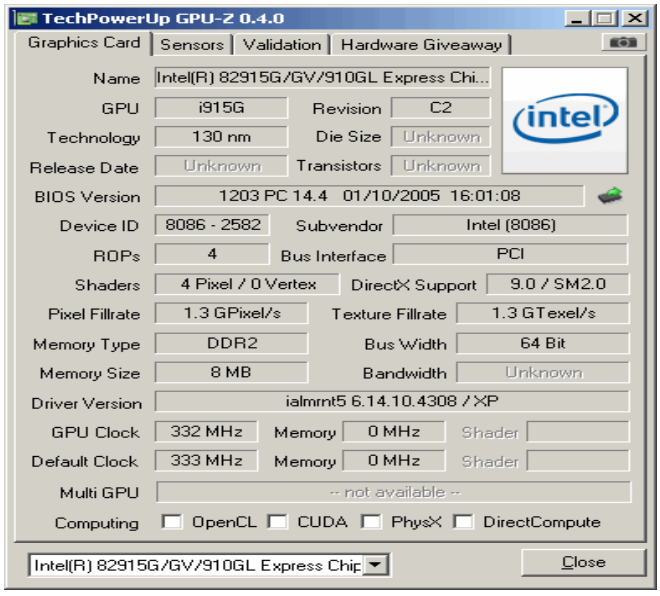


Рисунок 1.12. – Знімок з екрану зроблений програмою GPU–Z

Програма PC Wizard 2008 надає можливість також зберегти звіт у файл за вибором категорії, у багатьох форматах (рис 1.13). На рисунку 1.14 зображений звіт у форматі csv. Програма PC Wizard 2008 формує дуже короткий і інформативний звіт, що в свою чергу є дуже важливою властивістю.

Усі вищеперераховані та наведені програми формують звіт лише за власним шаблоном. Це призводить до того, що отримані звіти необхідно коригувати у відповідності до вимог.

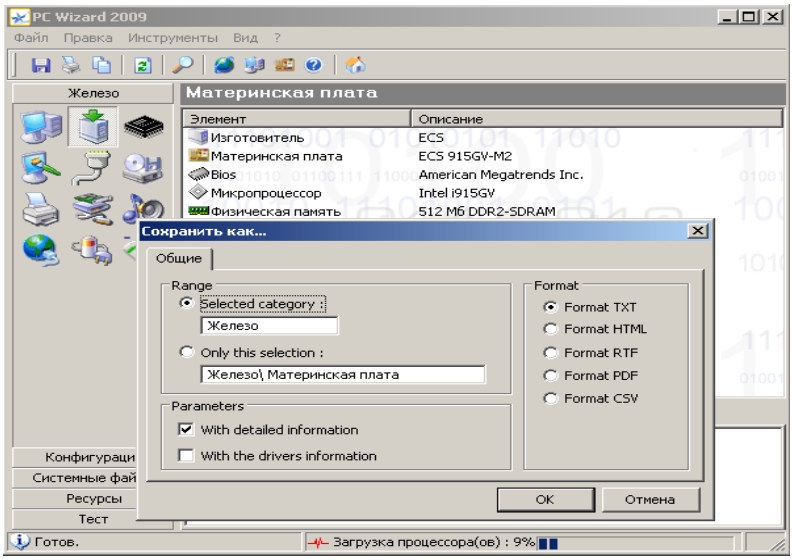


Рисунок 1.13. – Збереження файла звіту у програмі PC Wizard 2008.

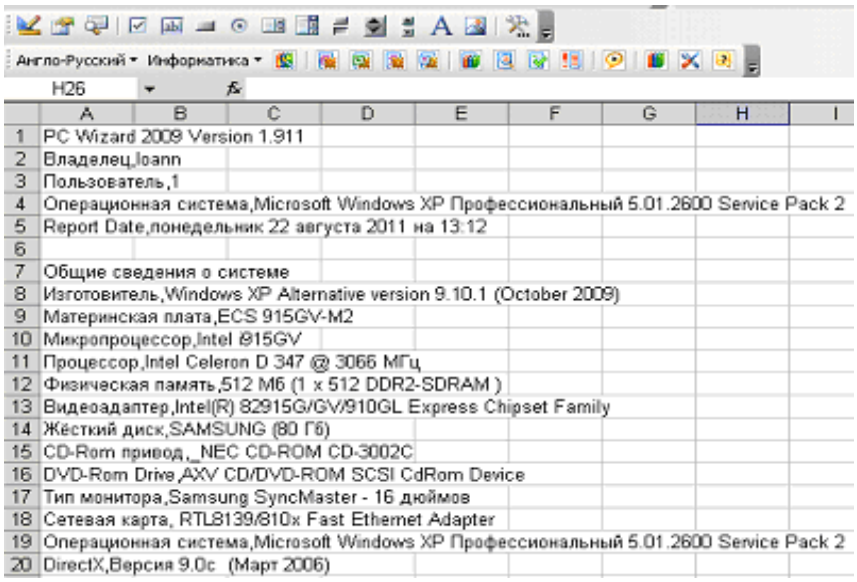


Рисунок 1.14. – Звіт програми PC Wizard 2008 у форматі csv

При цьому, слід зазначити, що для заповнення розділів картки спеціаліст повинен мати базові знання та навички роботи з апаратними і програмними компонентами комп’ютера.

Для більш повної та точної інформації про наявну техніку в форму облікової картки треба додати деякі характеристики комп’ютера, а саме: дату випуску, дату гарантійного обслуговування, дату закінчення гарантійного обслуговування по ремонту монітору та системного блоку.

Приведені зразки звітів у різноманітних програмах не забезпечують збереження звіту у відповідній формі, тому необхідна розробка спеціального програмного забезпечення.

Програма повинна визначати та оброблювати, відповідно до заданих параметрів пошуку, наступні дані:

– виробника та модель материнської плати;
– назву та частоту процесора;
– назву жорстких дисків та їх об’єм;
– об’єм оперативної пам’яті;
– назву відеокарти;
– назву монітору;
– назву принтерів;
– назву сканера;
– версію та модифікацію операційної системи;
– ім’я у мережі;
– IP адреса у мережі.

Після автоматизованого визначення конфігурації комп’ютера користувач або адміністратор повинен заповнити в ручному режимі наступні дані: серійні номери, дати гарантійного обслуговування, П.І.Б. користувача та адміністратора, наявність блоку безперебійного живлення.

Наступним кроком стане обробка інформації програмою та формування звіту в документ обліку Пенсійного фонду України.

Вимоги до операційної системи – Microsoft Windows 2000/XP/Vista/Seven.

Для зручності у використані програма повинна складатися з одного виконавчого файлу, без інсталяційного пакету та допоміжних бібліотек. Перевагою цього є запуск програми без установки та перенос її на знімних носіях або запуск по мережі. Файл звіту повинен бути зручним у коригуванні та друку, найоптимальніший формат для визначених завдань – doc або xls.

РОЗДІЛ 2 РОЗРОБКА ПРОГРАМИ «СИСТЕМА ОБЛІКУ КОМП'ЮТЕРНОГО ОБЛАДНАННЯ»

2.1. Загальна структура інтерфейсу програми «Система обліку комп'ютерного обладнання»

Розглянуте в першому розділі програмне забезпечення для діагностування та тестування комп'ютерного обладнання дозволило виявити ряд моментів, що потребують удосконалень.

Доцільно, щоб головне вікно програми «Комп'ютерна система обліку обчислювальної техніки» складалось з модального діалогового вікна (рис. 2.1), яке, в свою чергу, повинно містити древоподібний елемент типу `CtreeCtrl` та чотири кнопки для навігації: «Введення даних», «Створити та зберегти звіт», «Відкрити» та «Вихід».

Після ініціалізації всіх необхідних змінних у програмі в об'єкт типу `CtreeCtrl` додаються всі необхідні вузли, картинки для візуалізації та відповідні іконки вузлів, які відображають апаратну та програмну інформацію про комп'ютер:

- назву виробника та модель материнської плати;
- назву та частоту процесора;
- назву жорстких дисків та їх об'єм;
- об'єм оперативної пам'яті;
- назву відеокарти;
- назву монітору;
- назву принтерів;
- назву сканера;
- версію та модифікацію операційної системи;
- ім'я у мережі;
- IP адреса у мережі.

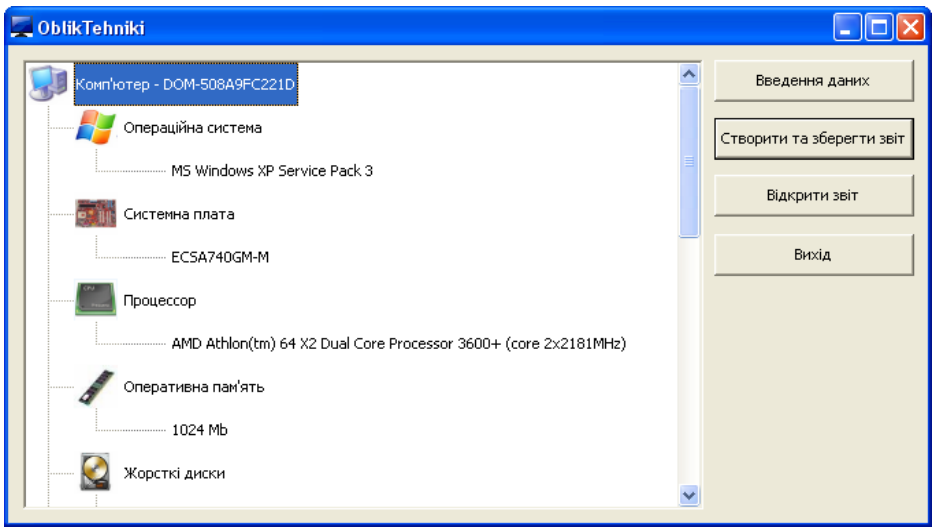


Рисунок 2.1. – Інтерфейс програми «Облік обчислювальної техніки»

Введення додаткової інформації виконується при натисканні кнопки «Введення даних» (рис. 2.2).

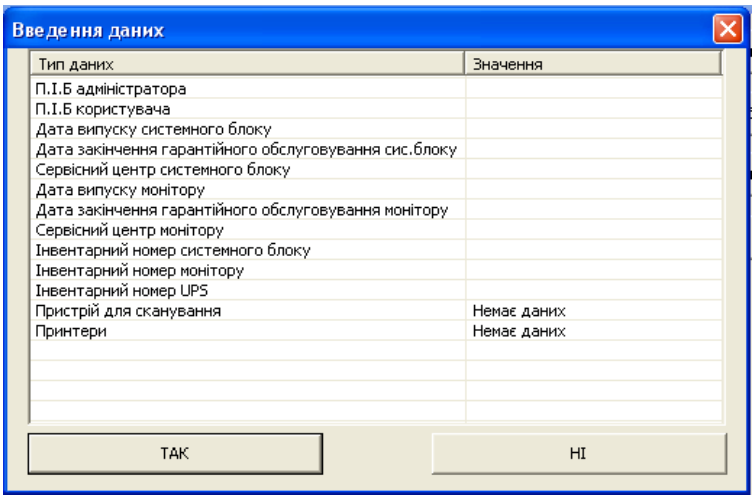


Рисунок 2.2. – Діалогове вікно «Введення даних»

Для подальшої роботи та вивантаження звіту у потрібний формат необхідно ввести додаткову інформацію.

Для більшої зручності перегляду звіту використовується кнопка «Відкрити звіт», яка відкриває щойно збережений файл звіту. Для виходу з прогами «Облік обчислювальної техніки » передбачена кнопка «Вихід».

2.2. Опис класів програми та вибір інструментальних засобів

Для реалізації графічного інтерфейсу дуже добре підходить бібліотека мови C++ MFC. Microsoft Foundation Classes (MFC) – бібліотека на мові програмування C ++, розроблена компанією Microsoft, яка спрощує розробку GUI-додатків для Microsoft Windows шляхом використання різноматніго набору бібліотечних класів.

У даній дипломній роботі мовою програмування обраний C++.

В якості середовища розробки найкраще підходить пакет Microsoft Visual Studio 2008, який допомагає оптимізувати та спрощує процес розробки високоефективних програм.

Для отримання достовірної інформації про використання застосовуваних технологій і підходів, тестування проводилося на понад 50 комп'ютерах різної конфігурації.

2.2.1. Опис класу визначення параметрів комп'ютера

В якості властивостей деяких класів для реалізації динамічних масивів використовується стандартна бібліотека шаблонів (Standard Template Library) та його контейнерний клас Vector.

Standard Template Library (STL) – набір контейнерів, засобів доступу до їхнього вмісту, погоджених узагальнених алгоритмів і різних допоміжних функцій в C ++. У контейнерах для зберігання елементів використовується семантика передачі об'єктів за значенням.

Vector – контейнерний клас, в якому доступ до його елементів здійснюється за індексом. У силу цього вектори багато в чому нагадують одномірні масиви. Клас vector поводиться подібно масиву, проте, надає більше можливостей керування ним. Зокрема, вектор можна нарощувати в міру необхідності. Крім того, він забезпечує контроль значень індексів. Доступ до членів вектора здійснюється за константний час, додавання елементів до

вектора відбувається за амортизований константний час, оскільки визначення місцезнаходження або вставка елементів в векторі займає лінійний час.

Для зберігання рядків у програмі використовується клас `CString` бібліотеки MFC. Використання операцій конкатенації і порівняння рядків, а також спрощення операцій по роботі з пам'яттю, полегшують роботу з об'єктом класу `CString` порівняно зі звичайним масивом символів. Для символів в об'єкті класу `CString` використовується тип TCHAR.

Якщо для програми визначено символ `_UNICODE`, то типу `TCHAR` відповідає тип `wchar_t` (16-розрядний символ), в іншому випадку цього типу відповідає тип char (8-розрядний символ). При використанні кодування Unicode об'єкти класу `CString` складаються з 16-розрядних символів.

`Chwdata` – основний клас, який містить у собі всі необхідні функції (методи) та властивості для відображення інформації про програмні та апаратні модулі комп'ютера. Властивості класу `Chwdata` та їх опис вказані (табл. 2.1.)

Таблиця 2.1. – Опис властивостей класу `Chwdata`

Назва	Тип	Опис
mboard	CString	інформація про материнську плату
pcname	CString	ім'я комп'ютера в мережі
video	CString	назва відеоадаптеру
monik	CString	назва монітору
processor	CString	данні про центральний процесор комп'ютера
memory	CString	об'єм оперативної пам'яті
os_str	CString	данні про операційну систему
scanner	CString	назва скануючого пристрою
netif	vector <CString>	масив для збереження назви мережевих інтерфейсів
netip	vector <CString>	масив ip-адресів у вигляді
hdd	vector <CString>	масив для збереження інформації про жорсткі диски
rom	vector <CString>	назва оптичних пристроїв
printers	vector <CString>	Масив для збереження назви принтерів

Даний клас інкапсулює кілька варіантів отримання даних про комп’ютер: зчитування з реєстру, використання стандартних Windows API структур та за допомогою технології WMI.

Працює клас настуним чином: у конструкторі, вихідний код якого наведено нижче, при будуванні об’єкту активізуються функції (методи), які заповнюють даними всі властивості класу, що оголошені з модифікатором доступу public.

```
Chwdata::Chwdata(void){
     get_os(os_str);
     get_MBoard(mboard);
     get_hostname(pcname);
     get_mon_video(monik,video);
     get_processor(processor);
     get_mem_hdd_rom(memory,hdd,rom);
     get_net_param(netif,netip);
     scan_find = get_scaner(scanner);
     prn_find = get_printers(printers); }
```

Функція `get_os` дозволяє одержати версію та модифікацію операційної системи, яка встановлена на комп’ютері; приймає строковий об’єкт типу `CString` для заповнення, нічого не повертає. Код функції наведен нижче.

```
void Chwdata::get_os(CString &OperSystem){
CString TempOS;
OperSystem="MS Windows ";
OSVERSIONINFO os;
memset (&os,0x00, sizeof(OSVERSIONINFO));
os.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
GetVersionEx (&os);
...
}
```

Визначення версії Windows відбувається порівнянням значень властивостей `dwMajorVersion` та `dwMinorVersion`. Наприклад: якщо `dwMajorVersion` = 5 та `dwMinorVersion` = 1, це Windows XP. Модифікацію (сервісний пакет оновлень) операційної системи можно визначити за

допомогою значення властивості szCSDVersion, яка являє собою масив символів типу char.

Для зручності необхідно перетворити цей масив до типу CString, за допомогою методу Format.

```
     {
     ...
     if(os.dwMajorVersion  ==  5  &&  os.dwMinorVersion  ==  0)
{OperSystem+="2000 ";}
     if(os.dwMajorVersion  ==  5  &&  os.dwMinorVersion  ==  1)
{OperSystem+="XP ";}
     if(os.dwMajorVersion  ==  5  &&  os.dwMinorVersion  ==  2)
{OperSystem+="2003 ";}
     if(os.dwMajorVersion  ==  6  &&  os.dwMinorVersion  ==  0)
{OperSystem+="VISTA ";}
     if(os.dwMajorVersion  ==  6  &&  os.dwMinorVersion  ==  1)
{OperSystem+="SEVEN ";}
     TempOS.Format ("%s",os.szCSDVersion);
     OperSystem+=TempOS;
     }
```

Функція get_os оголошена з модифікатором private, для того щоб уникнути доступу із зовні. Реалізована за допомогою стандартної Windows API структури OSVERSIONINFO та функції GetVersionEx. Спочатку створюємо об’єкт os типу OSVERSIONINFO, заповнюємо його нулями функцією memset, з’ясовуємо необхідний розмір структури і присвоюємо це значення властивості dwOSVersionInfoSize, потім передаємо адресу об’єкту os у функцію GetVersionEx для заповнення інформацією. Після заповнення з’являється можливість зчитування властивостей об’єкту os, в яких є вся необхідна інформація.

Функція get_MBoard дає можливість одержати модель та назву виробника материнської плати, оголошена як private, для того, щоб уникнути доступу з зовні. Приймає строковий об’єкт типу CString для заповнення, нічого не повертає.

У даній функції використовується метод читання даних з реєстру та технологія WMI.

Технологія WMI є розширеною та адаптованою реалізацією стандарту WBEM, який був прийнятий різними компаніями. Ця технологія заснована на ідеї створення універсального інтерфейсу для моніторингу та управління різними системами і компонентами розподіленого інформаційного середовища підприємства. Вона використовує об'єктно-орієнтовані принципи і протоколи HTML і XML. Зокрема, технологія WMI спеціально розроблена для операційної системи Windows.

Для звернення до об'єктів WMI використовується специфічна мова запитів WMI Query Language (WQL), яка є одним з різновидів SQL. Основна її відмінність від ANSI SQL - це неможливість зміни даних, тобто за допомогою WQL можлива лише вибірка даних за допомогою команди `SELECT`. Для визначення моделі та виробника материнської плати в ноутбуках (нетбуках та інших переносних комп'ютерах) використовується зчитування з реєстру SMBiosData даних.

System Management BIOS (SMBIOS) (Системне управління BIOS) визначає структуру даних (метод доступу) в BIOS, що дозволяє користувачеві програми зберігати і отримувати специфічну інформацію для даного комп'ютера.

Алгоритм роботи фунції по визначенню назви виробника та моделі материнсткої плати представлений на рис. 2.3.

Функція працює наступним чином, спочатку зчитуємо `SMBiosData` дані, які допожуть встановити форм-фактор комп'ютера, так як на стаціонарних комп'ютерах та ноутбуках форм фактор визначається різними шляхами.

Після цього встановлюємо змінну бульового типу `IsNoteBook` класу `Chwdata` у відповідний стан. На звичайних комп'ютерах при встановленій операційній системі Windows Vista/Seven модель та виробника материнської плати можно зчитати з реєстру, на операційних системах Windows 2000/2003/XP такої можливості немає.

Рисунок 2.3. – Блок схема функції `get_Mboard`.

Вихідний код функції `get_Mboard` наведено нижче. З початку ініціалізуємо необхідні змінні.

```
void Chwdata::get_MBoard(CString &MBoard){
IsNoteBook = false;
bool ErroReadSMBIOS=true;
HKEY hkData; LPSTR lpString = 0; BYTE *lpData = 0;
DWORD dwType = 0, dwSize = 0;
UINT uIndex, uStart, uEnd, uString, uState = 0;
```

Зчитуємо дані SMBIOS з реєстру, якщо функція повернула `SUCCESS` читаємо ці дані.

```
     if (ERROR_SUCCESS == RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\mssmbios\\Data",
0, KEY_QUERY_VALUE, &hkData))
     {
     RegQueryValueEx(hkData,
```

```
     "SMBiosData", 0, &dwType, 0, &dwSize);
     lpData = new BYTE [dwSize];
     if(ERROR_SUCCESS == RegQueryValueEx(hkData,
     "SMBiosData", 0,    0, lpData, &dwSize))
     { ErroReadSMBIOS=false; }
     RegCloseKey(hkData);
     }
```

Зчитування даних SMBIOS відбувається побитово.

```
if(!ErroReadSMBIOS){
     uIndex = 8 + *(WORD *)(lpData + 6);
     uEnd = 8 + *(WORD *)(lpData + 4);
     while(lpData[uIndex] != 0x7F && uIndex < uEnd){
     uIndex += lpData[(uStart = uIndex) + 1];
     uString = 1;
...
}
```

Тому для універсальності зчитування інформації на стаціонарних комп’ютерах у даному диплоному проекті використовується технологія WMI і зчитування даних з `SMBiosData` для встановлення форм фактору комп’ютера.

Якщо це ноутбук (нетбук та інші переносні комп’ютери) зчитуємо інформацію з першої таблиці специфікації `SMBIOS` 4 та 5 біт, в яких знаходиться модель та виробник материнської плати.

```
{
...
do{
if(lpData[uStart] == 0x03 && uString == 1)
     {
     switch(lpData[uStart + 5])
          {
```

Визначаемо форм фактор комп’ютера, шляхом порівняння значеннь.

```
          case 0x08: IsNoteBook=true; break;
          case 0x09: IsNoteBook=true; break;
          case 0x0A: IsNoteBook=true; break;
          case 0x0B: IsNoteBook=true; break;
          case 0x0C: IsNoteBook=true; break;
          case 0x0D: IsNoteBook=true; break;
```

```
            case 0x0E: IsNoteBook=true; break;
            default: IsNoteBook=false; break;
            }
}
while(lpData[uIndex++]);
     }
while(lpData[uIndex] && uIndex < uEnd);
     uIndex++;
          }
}
```

Якщо змінна в стані `false`, форм фактор – переносний комп'ютер. У такому випадку використовуємо технологію WMI.

```
if(!IsNoteBook){
HRESULT hr;
...}
```

Оголошуємо всі необхідні змінні для роботи з сервісами WBEM та простіром імен cimv2.

Після визначеннях всіх необхідних змінних треба встановити рівень безпеки за допомогою функції `CoCreateInstance`.

```
IWbemClassObject * pObject = 0;
IWbemServices * pService = 0;
IWbemLocator * pLocator = 0;
IEnumWbemClassObject * pEnum = 0;
CoInitializeEx(0, COINIT_APARTMENTTHREADED);
CoInitializeSecurity(0, -1, 0, 0,2,3,0, EOAC_NONE, 0);
```

Наступним кроком буде підключення до простору імен `cimv2` за допомогою функції `ConnectServer`.

```
hr = CoCreateInstance( CLSID_WbemLocator, 0, CLSCTX_ALL,
IID_IWbemLocator, (void**) &pLocator);
if (FAILED(hr))
{ ::CoUninitialize();}
hr = pLocator->ConnectServer
(_bstr_t(L"root\\cimv2"),0,0,0,0,0,0,&pService);
if (FAILED(hr))
{
pLocator->Release();
```

```
::CoUninitialize();
}
```

Виконуємо запит WQL до класу Win32_baseboard операційної системи, заповнюєм необхідні поля, а саме Manufacturer,Product.

```
hr = pService->ExecQuery( _bstr_t(L"WQL"),_bstr_t
(L"Select Manufacturer,Product from Win32_baseboard"),
WBEM_FLAG_RETURN_IMMEDIATELY | WBEM_FLAG_FORWARD_ONLY, 0,
 &pEnum );
if (FAILED(hr)) {
pService->Release();
pLocator->Release();
::CoUninitialize();}
ULONG uReturn2 = 0;
```

У циклі while поки покажчик pEnum буде TRUE зчитуємо модель і виробника материнської плати.

За допомогою функції Get виконуємо запит про виробника материнської плати.

```
while (pEnum) {
hr = pEnum->Next(WBEM_INFINITE, 1, &pObject, &uReturn2);
if(0 == uReturn2)
{ break;
}
VARIANT vtProp_mb;
VariantInit(&vtProp_mb);
hr = pObject->Get(_bstr_t(L"Manufacturer"), 0, &vtProp_mb,
 0, 0);
```

Потім зчитуємо значення із змінної vtProp і присвоюємо це значення змінній Mboard.

```
MBoard = vtProp_mb.bstrVal;
hr = pObject->Get(_bstr_t(L"Product"),
 0, &vtProp_mb, 0, 0);
MBoard += vtProp_mb.bstrVal;
VariantClear(&vtProp_mb);
pObject->Release();
pService->Release();
pLocator->Release();
```

```
pEnum->Release();
::CoUninitialize();
}else {
```

Якщо форм фактор - переносний комп’ютер отримуємо дані з SMBIOS

```
uIndex = 8 + *(WORD *)(lpData + 6);
uEnd = 8 + *(WORD *)(lpData + 4);
while(lpData[uIndex] != 0x7F && uIndex < uEnd){
     uIndex += lpData[(uStart = uIndex) + 1];
     uString = 1;
```

Згідно специфікації SMBIOS інформація про материнську плату знаходиться в першій таблиці, а в її четвертому і п’ятому біті знаходяться дані про модель і виробника.

Після виконання всіх необхідних операцій очищаємо пам’ять оператором delete.

```
do{ if(lpData[uStart] == 0x1 && uState == 0){
                if(lpData[uStart   +   4]   ==   uString   ||
lpData[uStart + 5] == uString ){
                          lpString = (LPSTR)(lpData + uIndex);
                               MBoard+=lpString; }}
                                    uString++;
                               while(lpData[uIndex++]); }
                     while(lpData[uIndex] && uIndex < uEnd);
          uIndex++;
          }
     }
delete lpData; }
```

В багатьох функціях класу Chwdata для роботи зі змінними тип яких в процесі виконування може змінюватися використовується класс VARIANT.

VARIANT – тип, що дозволяє створити змінну, тип якої визначається під час виконання програми і може змінюватися з часом. По суті, VARIANT – це структура, яка містить поле vt, що визначає тип, і поле, створене на базі об’єднання, та містить доступні типи даних. VARIANT дозволяє зберігати в собі навіть структуру (VT_RECORD), масив (VT_ARRAY) або покажчик на інтерфейс. Використання масивів VARIANT’ов і масивів структур дозволяє створювати

моделі даних будь-якої складності. Єдина проблема типу VARIANT - розмір. VARIANT займає 16 байт.

Функція get_mem_hdd_rom надає можливість одержати інформацію про об'єм та кількість модулів оперативної пам'яті, назву та об'єм жорстких дисків та назву оптичних пристроїв, які встановлені на комп'ютері. Оголошена з модифікатором доступу private з аргументів функція, приймає строковий об'єкт типу CString та два об'єкти типу vector з шаблоном типу CString, нічого не повертає.

Спочатку для роботи з WBEM сервісами необхідно ініціалізувати всі необхідні змінні, а саме: покажчики pObject типу IwbemClassObject, pService типу IwbemServices, pLocator типу IwbemLocator та pEnum типу IenumWbemClassObject. Фрагмент коду наведено нижче.

```
HRESULT hr; // инициализация необходимых переменных
bool FindROM=false,FindHDD=false;
vector<CString> MemoryArr;
IWbemClassObject * pObject = 0;
IWbemServices * pService = 0;
IWbemLocator * pLocator = 0;
IEnumWbemClassObject * pEnum = 0;
```

Оскільки WMI заснована на COM технології необхідно виконати виклики CoInitializeEx і CoInitializeSecurity функцій для доступу до WMI.

Функція CoInitializeEx повинна бути викликана хоча б один раз для кожного потоку, який використовує COM бібліотеки.

```
CoInitializeEx(0, COINIT_APARTMENTTHREADED);
CoInitializeSecurity(0, -1, 0, 0,RPC_C_AUTHN_LEVEL_CONNECT,
RPC_C_IMP_LEVEL_IMPERSONATE, 0, EOAC_NONE, 0);
hr = CoCreateInstance( CLSID_WbemLocator, 0,
CLSCTX_ALL,IID_IWbemLocator, (void**) &pLocator);
if (FAILED(hr))
{
     ::CoUninitialize();
}//подключаемся к пространству имен wbem
```

Функція `CoInitializeSecurity` ініціалізує рівень безпеки. Якщо процес не викликає `CoInitializeSecurity`, COM викликає його автоматично.

```
hr = pLocator->ConnectServer
( _bstr_t(L"root\\cimv2"),0,0,0,0,0,0,&pService);
     if (FAILED(hr))
{
```

Потім необхідно підключитися до простору імен WMI. Для цього викликаємо функцію CoCreateInstance.

Резальтатом виклику є значення змінної `hr`, перевіряємо її на коректність.

```
     pLocator->Release();
     ::CoUninitialize();
}
hr = pService->ExecQuery( _bstr_t(L"WQL"),
     _bstr_t(L"Select capacity from Win32_PhysicalMemory"),
     WBEM_FLAG_RETURN_IMMEDIATELY | WBEM_FLAG_FORWARD_ONLY,
     0, &pEnum );
     if (FAILED(hr))
{
     pService->Release();
     pLocator->Release();
     ::CoUninitialize();
}
```

Якщо результат FALSE необхідно викликати `CoUninitialize`, при результаті TRUE – створюємо з’єднання через DCOM до простору імен WMI на комп’ютері за допомогою методу `ConnectServer`.

```
{ hr = pEnum->Next(WBEM_INFINITE, 1, &pObject, &uReturn);
if(0 == uReturn)
{
     break;
}
VARIANT vtProp;
VariantInit(&vtProp);
     hr = pObject->Get(_bstr_t(L"capacity"),0,&vtProp,0, 0);
     TempMem = vtProp.bstrVal; //считываем объем озу
     MemoryArr.push_back(TempMem);
```

```
VariantClear(&vtProp);
pObject->Release(); }
hr = pService->ExecQuery( _bstr_t(L"WQL"),
_bstr_t(L"Select model,
```

Далі присвоюємо змінній hr результат виклику, при негативному результаті – викликаємо `CoUninitialize` та `Release` методи.

Наступним кроком буде запит на мові WQL, який передається в якості параметру у функцію `ExecQuery`. Частина коду функції наведено нижче.

```
size from win32_diskdrive where InterfaceType !=\"USB\""),
WBEM_FLAG_RETURN_IMMEDIATELY | WBEM_FLAG_FORWARD_ONLY,
0, &pEnum );
if (FAILED(hr))
{
pService->Release();
pLocator->Release();
::CoUninitialize();
}
```

Після всіх виконаних маніпуляцій є можливість зчитати значення з властивості об’єкт типу `VARIANT`, а саме basic string рядок `bstrVal`. Присвоюємо `bstrVal` змінній типу `CString`. Один з великих плюсів використання для рядків саме цього типу: в ньому багато перевантажень для різних типів рядків. У такий же спосіб і проводиться зчитування інформації про жорсткі диски та оптичні носії, з однією відмінністю: для збереження отриманих рядків використовується масив vector з шаблоном `CString [9]`. Нижче наведено фрагмент коду, де додаються рядки з назвами оптичних пристроїв до масиву.

```
hr = pEnum->Next(WBEM_INFINITE, 1, &pObject, &uReturnHDD);
if(0 == uReturnHDD)
{ break;}
VARIANT vtProp;
VariantInit(&vtProp); //выполняем запрос
hr = pObject->Get(_bstr_t(L"model"), 0, &vtProp, 0, 0);
TempHdd = vtProp.bstrVal;
hr = pObject->Get(_bstr_t(L"size"), 0, &vtProp, 0, 0);
SizeHdd = vtProp.bstrVal;
```

Для подальшого використання рядки з ім’ям жорсткого диска і його обсягу, додаємо їй одиницю виміру інформації Gb.

Після формування кінцевого виду рядки інформації по жорсткому диску додаємо в масив для збереження.

```
        SizeHdd = SizeHdd.Mid(0,SizeHdd.GetLength()-9);
        TempHdd +=" "+SizeHdd+"Gb";
        HDDarray.push_back(TempHdd); //добавляем в массив
        FindHDD=true;
        VariantClear(&vtProp);
        pObject->Release();
}
hr = pService->ExecQuery( _bstr_t(L"WQL"),
_bstr_t(L"Select name from Win32_CDROMDrive"),
WBEM_FLAG_RETURN_IMMEDIATELY | WBEM_FLAG_FORWARD_ONLY,
0, &pEnum );
if (FAILED(hr)){
          pService->Release();
          pLocator->Release();
          ::CoUninitialize();}
```

При успішному результаті за допомогою функції `Next` зчитуємо всі значення з об’єкту у циклі `while`, поки змінна `pEnum` типу `IenumWbemClassObject` не буде `FALSE`. Для того, щоб отримати необхідне значення у вигляді строкової змінної, треба створити об’єкт типу `VARIANT` та ініціалізувати його функцією `VariantInit`, передати його адресу в якості параметру до функції `Get`.

Після закінчення роботи з об’єктами WBEM, треба їх вивантажити з пам’яті методами Release. У кінці функції робимо деякі маніпуляції з рядками для подальшого використання їх у звіті. Форматування рядків у зручний формат для подальшого відображення в програмі реалізовано в коді, який показаний нижче.

```
MemoryInt +=((atoi(MemoryArr[i]))/1048576);
Memory.Format("%d",MemoryInt);
Memory+=" Mb"; //переводим в мегабайты
```

Перевіряємо, якщо кількість модулів більше 1 - формуємо рядок певного типу.

Потім доблавяем необхідні знаки для відділення кількості модулів.

```
if(1 < SizeMemoryArr){
     CString TempRow;
     Memory+=" (";
     for(int i=0; i < SizeMemoryArr; i++)
          {
          TempRow.Format("%d",(atoi(MemoryArr[i]))/1048576);
          if(i==0)
          {
          Memory+=TempRow;
          } else {
     Memory+="+"+TempRow;
               }
          }
     Memory+=")";
}
if (!FindROM){ //проверка
ROMarray.push_back("Немає даних"); }
if (!FindHDD){
HDDarray.push_back("Немає даних"); }
}
```

За допомогою функції `get_hostname` можливо одержати ім’я комп’ютера у локальній мережі. Код функції наведен нижче. Це ім’я необхідне для ідентифікації його у локальній мережі, та для обміну інформацією між клієнтами мережі.

Оголошена з модифікатором доступу private, з аргуметів функція приймає строковий об’єкт типу `CString` і заповнює його ім’ям комп’ютера, нічого не повертає.

Реалізована зазначена функція за допомогою стандартної Windows API функції `GetComputerName`, яка приймає в свою чергу покажчик на масив типу char та покажчик на розмір цього масиву.

Після одержання масиву з ім’ям комп’ютера у локальній мережі, для зручності необхідно перетворити його у рядок типу `CString`, за допомогою методу `Format`.

```
void Chwdata::get_hostname (CString &host) {
     char HostNameBuf[MAX_COMPUTERNAME_LENGTH + 1]={0};
     DWORD size = sizeof(HostNameBuf);
     GetComputerName(HostNameBuf, &size);
     host.Format("%s",HostNameBuf);
 }
```

Функція `get_processor` дає інформацію про центральний процесор комп'ютера, а саме його назву, кількість ядер та поточну тактову частоту. Оголошена з модифікатором доступу private, з аргументів функція приймає строковий об'єкт типу `CString`, нічого не повертає. Основна частина коду функції `get_processor` наведена нижче.

```
SYSTEM_INFO SysINFO;
memset(&SysINFO, 0x00, sizeof(SYSTEM_INFO));
GetSystemInfo(&SysINFO);
if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0",
0,KEY_READ,&hKeyCPU)){RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0",
0,KEY_READ,&hKeyCPU);
RegQueryValueEx(hKeyCPU,
"ProcessorNameString", 0, &dwTypeCPU,
(LPBYTE)&valueCPU, &dwCountCPU);
RegCloseKey(hKeyCPU);
CPU=valueCPU;
}else { CPU="Немає даних";
}
```

За допомогою стандартної Windows API структури `SYSTEM_INFO` та функції `GetSystemInfo` отримуємо кількість ядер центрального процесора. Для цього треба створити об'єкт типу `SYSTEM_INFO`, заповнити його нулями за допомогою функції `memset` та визначити розмір самої струтури `SYSTEM_INFO`, а потім передати адресу об'єкту до списка аргументів функції `GetSystemInfo`. Після чого вона заповнюється інформацією про центральний процесор. Зчитуємо кількість ядер із значення властивості `dwNumberOfProcessors`.

```
DWORD dwTimerHi, dwTimerLo;
_asm
 {
```

```
_emit 0x0F
_emit 0x31
mov dwTimerLo, EAX
mov dwTimerHi, EDX
}
Sleep (500);
 _asm
 {
_emit 0x0F
_emit 0x31
sub EAX, dwTimerLo
sub EDX, dwTimerHi
mov dwTimerLo, EAX
mov dwTimerHi, EDX
 }
 strCORE.Format("%d",SysINFO.dwNumberOfProcessors);
 tempCPU.Format("%d",dwTimerLo/(1000*500));
 CPU+=" (core "+ strCORE+"x"+tempCPU+"MHz)";
}
```

Функція зчитує інформацію з гілки реєстру, в якій є не тільки назва, але й частота, но як показали тести, частота незавжди правдива, тому в зазначеній функції є асемблерні вставки коду для визначення поточної частоти центрального процесору.

Під час розробки дипломного проекту та написання програми було декілька варіантів отримання інформації про центральний процесор комп’ютера, використання технології WMI або зчитування унікального ID пристрої - vendor id. У підсумку проведення довгих тестів та експериментів вибір зупинився саме на реєстрі, в якому в більшості випадків зберігається правдива інформація. Обчислення частоти процесору проводиться за допомогою асемблерних команд.

Для отримання інформації про мережні інтерфейси та їх IP-адреси використовується функція `get_net_param`, яка приймає в якості аргументів два масива vector з шаблоном `CString`. Оголошена з модифікатором доступу private, нічого не повертає. У зазначеній функції реалізовано два варіанти отримання інформації про мережні інтерфейси та їх IP-адреси.

```
Void Chwdata::get_net_param
(vector<CString> &NetwrkAdaptrArr,
```

```
vector<CString> &IPAdrArr){
bool NetwrkData = false;ULONG adapter_info_size = 0;
PIP_ADAPTER_INFO
ptr_adapter_info = 0,ptr_adapter_info_first = 0;
GetAdaptersInfo(ptr_adapter_info, &adapter_info_size);
ptr_adapter_info_first = ptr_adapter_info =
(PIP_ADAPTER_INFO) new(char[adapter_info_size]);
if (GetAdaptersInfo
(ptr_adapter_info, &adapter_info_size) != ERROR_SUCCESS){
delete(ptr_adapter_info); }else{
while(ptr_adapter_info){
CString strServiceName,strNetwAdptr;
HKEY hKeyNetwrk;
DWORD dwType_reg_sz = REG_SZ;
strServiceName=ptr_adapter_info->AdapterName;
```

З початку створюємо два об’єкт типу `PIP_ADAPTER_INFO` та визначаємо розмір структури за домогою функції `GetAdaptersInfo`. Після визначення розміру присвоюємо `ptr_adapter_info_first` та `ptr_adapter_info` відповідний розмір масиву типу char. Далі викликаємо функцію `GetAdaptersInfo` ще раз, але вже для заповнення об’єктів структури `PIP_ADAPTER_INFO`.

У разі наявності помилок видаляємо об’єкт ptr_adapter_info. Якщо помилки відсутні, керування передається у блок циклу while, в якому при кожній ітерації береться клас мережного інтерфейсу і передається в цикл for. У цьому циклі здійснюється пошук збігів інтерфейсів по класу, так як в реєстрі зберігається інформація про всі інтерфейси, коли-небудь встановлені.

```
for(int i=1; i <=100; i++)
{
     strNetwAdptr.Format("SOFTWARE\\Microsoft\\
     Windows NT\\CurrentVersion\\NetworkCards\\%d",i);
     if( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
     strNetwAdptr,0,KEY_READ,&hKeyNetwrk) == ERROR_SUCCESS)
     {
          char strNtwrkApdtr[256]={0},ServiceName[256]={0};
          DWORD dwCount_service = sizeof(ServiceName),
          dwCount_adap_name = sizeof(strNtwrkApdtr);
RegQueryValueEx(hKeyNetwrk, "ServiceName", 0,
&dwType_reg_sz, (LPBYTE)&ServiceName, &dwCount_service);
if( strServiceName== ServiceName)
{
```

```
          RegQueryValueEx(hKeyNetwrk,"Description",0,
&dwType_reg_sz,
     (LPBYTE)&strNtwrkApdtr, &dwCount_adap_name);
     CString TempNtwrkAdptr=strNtwrkApdtr;
```

Такий спосіб найбільш оптимальний для отримання даних про фізично встановлені мережні інтерфейси в комп'ютері. У вкладеному блоці оператора `if` фільтруються віртуальні пристрої.

IP-адресу отримуємо з властивостей об'єкта `ptr_adapter_info` [16], а саме `String`, який є масивом типу `char`. Усі значення додаємо в масив `vector` за допомогою функції push_back. Після закінчення роботи видаляємо об'єкт `ptr_adapter_info_first`.

Частина вихідного коду додавання назви мережевих інтерфейсів та їх IP-адресів до відповідних масивів , наведено нижче.

Перевіряємо кожен інтерфейс на фізичну доступність, деякі програми при установці додають віртуальні пристрої, які нам не потрібні.

```
     TempNtwrkAdptr.MakeLower();
     if (TempNtwrkAdptr.Find("virtual",0) <= 0 ){
               NetwrkAdaptrArr.push_back(strNtwrkApdtr);
               IPAdrArr.push_back
     (ptr_adapter_info->IpAddressList.IpAddress.String);
     NetwrkData=true;
          }
     }
     RegCloseKey(hKeyNetwrk);
     }
     ptr_adapter_info = ptr_adapter_info->Next;}
     delete(ptr_adapter_info_first);
     }
     if (!NetwrkData){
          NetwrkAdaptrArr.push_back("Немає даних");
          IPAdrArr.push_back("Немає даних");
          }
     }
```

Функція get_mon_video надає інформацію про монітор та відеоадаптер, оголошена з модифікатором доступу private, нічого не повертає. Вихідний код функції наведено нижче. Реалізація цієї функції трохи відрізняється від інших.

У випадку не визначення монітору через стандартну Windows API структуру DISPLAY_DEVICE, використовується пошук у базі ID моніторів за допомогою функції SearchByID, яка знаходиться у класі MonitorsID. Детальніше це можно побачити на блок-схемі, яка зображена на рис. 2.4. Клас `MonitorsID` являє собою базу `ID` значень та назву моніторів.

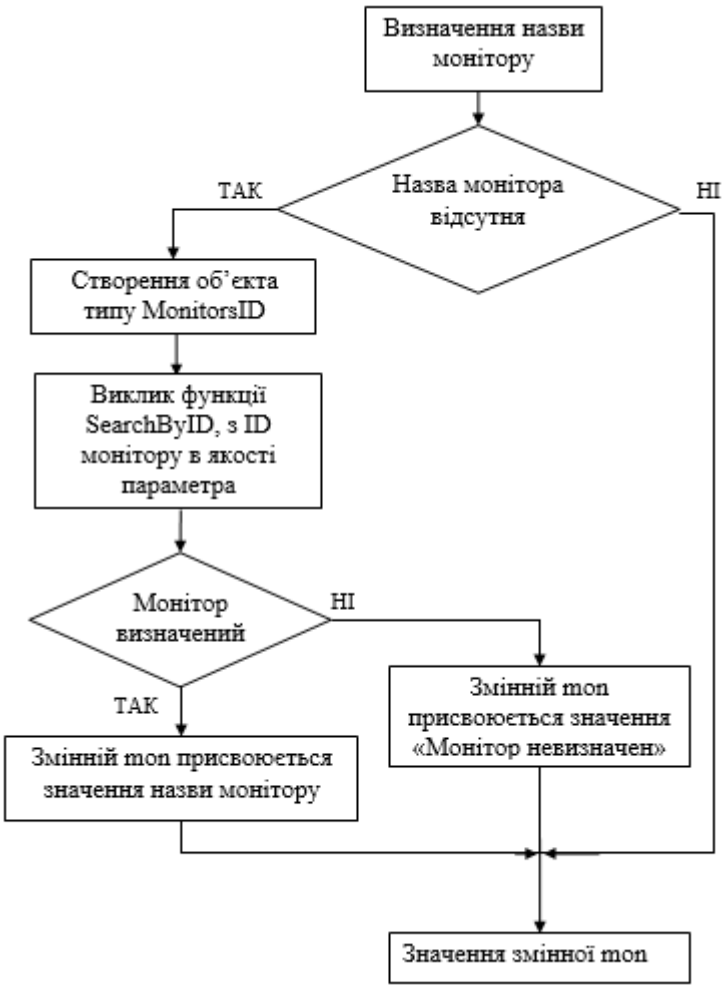


Рисунок 2.4. – Блок-схема визначення назви монітору

Пошук у базі `ID` моніторів виконується за допомогою метода SearchByID. Спочатку ініціалізується двомірний масив, в якому перше поле являє собою ID монітору, а друге його назву.

В кінце функції реалізован спеціальний конвеєр для пошуку. Код реалізації конвеєра наведено нижче.

При значенні змінної count, яка дорівнює розміру масиву, змінній mon_out присвоюється значення «Монітор невизначений».

В обліку техніки монітор враховується за одну одиницю техніки в порівнянні з системним блоком, який складається з безлічі комплектуючих. Такий підхід дає більший відсоток ймовірності визначення монітору.

```
int sz_mon_array
= (((sizeof(mon_array)/sizeof(CString))/2)-1);
 int count = 0;
 for(; count < sz_mon_array; count++) {
     if (monID == mon_array[count].ID) {
          mon_out = mon_array[count].Name;
               break;
           }
     }
if (count == sz_mon_array)
     {
     mon_out = "Монітор невизначений";
}
```

Зазначена функція працює наступним чином: спочатку створюємо два об’єкти `DDvideo` та `DDmonitor` типу `DISPLAY_DEVICE`, потім об’єкт відеоадаптера `DDvideo` заповнюємо нулями за допомогою функції `memset`. Наступним кроком буде встановлення властивості `cb` об’єкту `DDvideo` значенням розміру самого об’єкту, потім через енумератор `EnumDisplayDevices` заповняємо `Ddvideo`. Після цього виконуємо аналогічну операцію за винятком того, що в якості першого аргументу в енумератор `EnumDisplayDevices` передаємо рядок імені відеоадаптера для визначення заповнення об’єкта `DDmonitor` даними. Доступ до тих значень, які нас цікавлять, а саме ім’я відеоадаптера і монітора можна отримати через властивість `DeviceString` об’єктів структури `DISPLAY_DEVICE`. Отримані дані присвоюємо відповідним рядковим змінним `mon` і `vid` типу `CString`. Не можна залишити поза увагою визначення дисплея в ноутбуках та інших портативних комп’ютерах. Якщо у функції `get_MBoard` бульова змінна `IsNoteBook` класу `Chwdata` дорівнює TRUE, рядковій змінній mon присвоюємо значення

«Дісплей ноутбука», а якщо FALSE – шукаємо в рядковій змінній `mon` монітору деякі стандартні фрази, що означає відсутність коректної інформації про засобі відображення інформації. Для цього треба створити об'єкт `m_mID` класу `MonitorsID`, який в якості аргументів приймає ID монітора і рядок імені монітора.

```
void Chwdata::get_mon_video(CString &mon, CString &vid)
{
     DISPLAY_DEVICE DDvideo, DDmonitor;
     memset(&DDvideo, 0x00, sizeof(DISPLAY_DEVICE));
     DDvideo.cb = sizeof(DISPLAY_DEVICE);
     EnumDisplayDevices(0, 0, &DDvideo, 0);
     memset(&DDmonitor, 0x00, sizeof(DISPLAY_DEVICE));
     DDmonitor.cb = sizeof(DISPLAY_DEVICE);
     EnumDisplayDevices(DDvideo.DeviceName,0,&DDmonitor, 0);
     mon.Format("%s",DDmonitor.DeviceString);
     vid.Format("%s",DDvideo.DeviceString);
if(IsNoteBook)
{
     mon="Дісплей ноутбука";
}else{
     if (mon.Find("Plug") >= 0 || mon.Find("он") > 0 ||
mon.Find("efault") > 0 || mon.GetLength() == 0){
     CString MonikID;
     MonikID.Format ("%s", DDmonitor.DeviceID);
     MonikID = MonikID.Mid(8,7);
     MonitorsID m_mID;
     m_mID.SearchByID(MonikID, mon);
          }
     }
}
```

Функція визначення пристроїв сканування `get_scaner`. Вихідний код наведено нижче. Оголошена з модифікатором доступу private, в якості аргументів приїмає рядкову змінну типу `CString`, повертає TRUE або FALSE в залежності від наявності пристрою.

Реалізована за допомогою стандартних функцій зчитування інформації з реєстру операційної системи. Вихідний код наведени нижче.

```
bool Chwdata::get_scaner(CString &scaner){
     bool find_scan=false;
     HKEY hkey_scan_enum, hkey_scan;
```

```
     char value_scan_enum[256]={0}, value_scan[256]={0};
     DWORD dwType_scan_enum = REG_SZ;
     DWORD dwType_scan = REG_SZ;
     DWORD dwCount_scan_enum = sizeof(value_scan_enum);
     DWORD dwCount_scan = sizeof(value_scan);
     LONG result_enum, result_scan,result_empty;
     CString str_enum,cout_enum,str_path_scan;
     result_enum = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
     "SYSTEM\\CurrentControlSet\\Services\\usbscan\\Enum",
     0,KEY_READ,&hkey_scan_enum);
     if(result_enum == ERROR_SUCCESS){
          for (int i=0; i < 50; i++){
               cout_enum.Format("%d",i);
               RegQueryValueEx(hkey_scan_enum,cout_enum,0,
                    &dwType_scan_enum,
          (LPBYTE)&value_scan_enum,&dwCount_scan_enum);
}
     RegCloseKey(hkey_scan_enum);
}
     str_enum = value_scan_enum;
     str_enum = str_enum.Mid(4);
     str_path_scan=
     "SYSTEM\\CurrentControlSet\\Enum\\USB\\"+str_enum;
result_scan = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
str_path_scan,0,KEY_READ,&hkey_scan);
```

Спочатку намагаємося зчитати дані з гілки реєстру `SYSTEM\CurrentControlSet\Services\usbscan\Enum` функцією `RegOpenKeyEx`. У випадку успішного результату функції, запускаємо цикл, шукаємо номер пристрою, зазвичай це від 0 і не більше 50.

Потім до строкової змінної `str_path_scan` додаємо цей номер та із значення `DeviceDesc` зчитуємо назву пристрою для сканування.

```
if(result_scan == ERROR_SUCCESS){
     result_empty = RegQueryValueEx(hkey_scan,"DeviceDesc",
0,&dwType_scan,
     (LPBYTE)&value_scan,&dwCount_scan);
     if(result_empty != ERROR_SUCCESS)
          {
               scaner = "Немає даних";
     }else{
          scaner = value_scan;
          find_scan=true;
     }
     RegCloseKey(hkey_scan);
          }
return find_scan;
```

```
    }
}
```

Функція класу `Chwdata get_printers`, дозволяє отримати інформацію про встановлені на визначеному комп'ютері принтери. У функції `get_printers` є допоміжна функція `PrnCompare` та масив `IzgotovPrnts`.

Функція `PrnCompare` призначена для того, щоб не записувати до масиву принтерів віртуальні пристрої, які встановлюються з деякими програмними засобами. Вихідний код функції `PrnCompare` наведено нижче. Працює функція `PrnCompare` наступним чином: приймає масив з назвою принтерів і в залежності від результата пошуку у масиві IzgotovPrnts повертає true або false. У масиві `IzgotovPrnts` знаходяться назви виробників усіх популярних принтерів.

При виконанні всіх цих умов додається значення `pDriverName`. Важливим є визначення принтерів стосовно імені драйвера. Частіш за все ім'я принтера задається автоматично або в ручному режимі і, як правило, це лише марка без зазначення моделі.

```
bool Chwdata::PrnCompare (vector<CString> &arr_prn,
CString &str_prn){
bool cmpr=false;
CString temp = str_prn;
temp.MakeLower();
for(int i=0; i <sizeof(IzgotovPrnts)/sizeof(CString); i++)
{
if (temp.Find(IzgotovPrnts[i],0) >= 0) {cmpr=true;}
} return cmpr;
}
```

Оголошена функція `get_printers` з модифікатором доступу private, в якості аргументів приїмає масив типу vector з шаблоном `CString`, повертає TRUE або FALSE в залежності від наявності принтерів. Вихідний код наведено нижче.

```
bool Chwdata::get_printers(vector<CString> &prn_array)
{
```

```
          bool find_prn=false;
          CString prn_port,prn_drv_name;
          DWORD needed, returned;
          EnumPrinters(PRINTER_ENUM_LOCAL,0,2,0,0,&needed,&return
ed);
          PRINTER_INFO_2 *prn_info = new PRINTER_INFO_2[needed];
     if (EnumPrinters(PRINTER_ENUM_LOCAL, 0,
          2, (LPBYTE) prn_info, needed, &needed, &returned)){
           for (int i=0;i < (int)returned; i++){
               prn_port = prn_info[i].pPortName;
          prn_drv_name=prn_info[i].pDriverName;
```

Реалізована функція за допомогою стандартної Windows API структури `PRINTER_INFO_2` та функції `EnumPrinters`. Ця функція працює таким чином: спочатку з'ясовуємо необхідний розмір для буфера за допомогою функції EnumPrinters з флагом `PRINTER_ENUM_LOCAL` в якості першого аргументу і встановлюється значення змінної returned типу `DWORD`, яка дорівнює кількості принтерів.

```
     if((prn_port.Find("USB") >= 0 ||
          prn_port.Find("LPT") >= 0 ||
          prn_port.Find("COM") >= 0 )
          && PrnCompare(prn_array,prn_drv_name)) {
          prn_array.push_back(prn_drv_name);
          find_prn=true;
               }
          }
     }
     if (!find_prn){
          prn_array.push_back("Немає даних");
     }
          delete prn_info;
          return find_prn;
     }
```

Флаг PRINTER_ENUM_LOCAL означає, що нам потрібні тільки локальні принтери. Потім створюємо об'єкт prn_info типу `PRINTER_INFO_2` з відповідним розміром і викликаємо знову функцію EnumPrinters, якщо результат функції TRUE, управління передаємо в блок циклу. У циклі залишаємо принтери, у яких інтерфейс підключення до комп'ютера usb, com

або lpt. Паралельно передаючи ім’я драйвера принтера в допоміжну функцію PrnCompare класу `Chwdata`.

2.2.2. Опис класів інтерфейсу користувача

Класс Cuserdata призначений для зберігання даних, які вводяться в ручному режимі користувачем програми. Тобто інформація, яку неможливо одержати програмними шляхами, наприклад інвентарні номери монітору та системного блоку. Дані, які вводяться та властивості класу вказано в табл. 2.2.

Таблиця 2.2. – Властивості класу Cuserdata

Назва	Тип	Опис
adm_name	CString	прізвище адміністратора
usr_name	CString	прізвище користувача
date_sblock	CString	дата системного блоку
garant_sblock	CString	дата закінчення гарантійного обслуговування системного блоку
srvs_sblock	CString	назва сервісного центру системного блоку
date_mon	CString	дата випуску монітору
garant_mon	CString	дата закінчення гарантійного обслуговування монітору
srvs_mon	CString	назва сервісного центру монітору
inv_sblock	CString	інвентарний номер системного блоку
inv_mon	CString	інвентарний номер монітору
inv_ups	CString	інвентарний номер UPS
inv_scan	CString	інвентарний номер пристрою для сканування
printers	vector <CString>	інвентарні номери принтерів

При відсутності пристрою для сканування або принтера, виставляється відповідна булева змінна класу `Chwdata scan_find` і `prn_find`. Наявність такої перевірки дозволяє автоматично встановити значення відсутності пристрою. Всі перераховані вище дані вводяться за допомогою модального діалогового вікна, яке викликається натисканням на кнопку «Введення даних» (рис. 2.5).

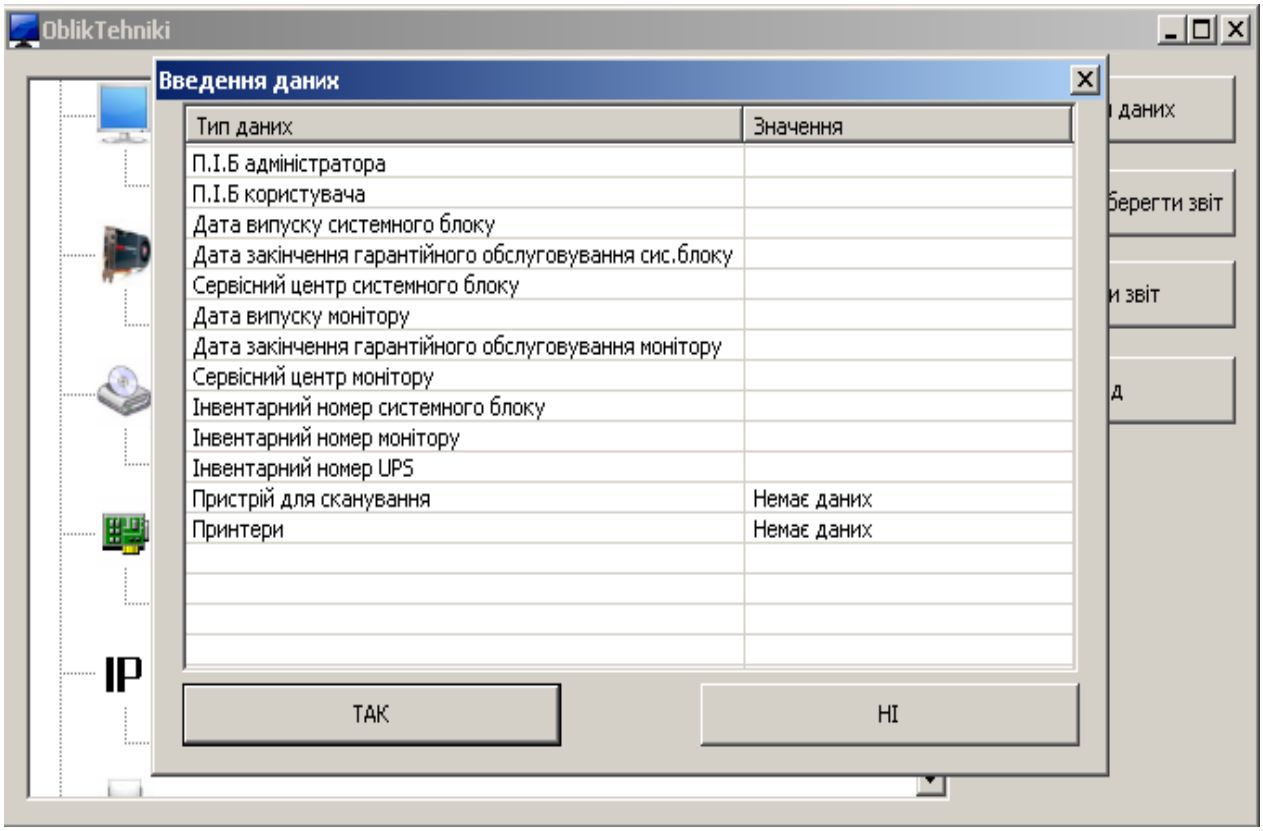


Рисунок 2.5. – Діалогове вікно «Введення даних»

Клас `InUsrDatadlg` є успадкованим від класу діалогових вікон CDialog. З його допомогою користувач може вводити дані, які в свою чергу зберігаються в класі Cuserdata для подальших маніпуляцій у вихідному документі.

Введення даних здійснюється за допомогою об’єкта `Lst` класу `CEditList`. Цей тип є успадкованим від класу `CListCtrl`. Редагування рядків та внесення даних в них в базовому класі `CListCtrl` неможливо зробити. Реалізація метода редагування складається з двох сторонніх класів `EditList` та `EditItem`..

В таблиці два стовпчика: «Тип даних» та «Значення». Перший стовпчик неможливо редагувати, програмно це виглядає так.

```
void CEditList::OnClick(NMHDR* pNMHDR, LRESULT* pResult){
     if( GetFocus() != this)
          SetFocus();
```

```
                NM_LISTVIEW* pNMListView = (NM_LISTVIEW*)pNMHDR;
                if (pNMListView->iSubItem == 1)
                     EditItem   (pNMListView->iItem,   pNMListView-
>iSubItem);
          *pResult = 0;
     }
```

При співпаданні фокуса миші з номером стовпчика «Значення», активується функція відтворення прямокутника і його контурів. Прямокутник являє собою поле для введення тексту. Після переміщення курсора в будь-яке місце, записуються дані в елемент списку `Lst`.

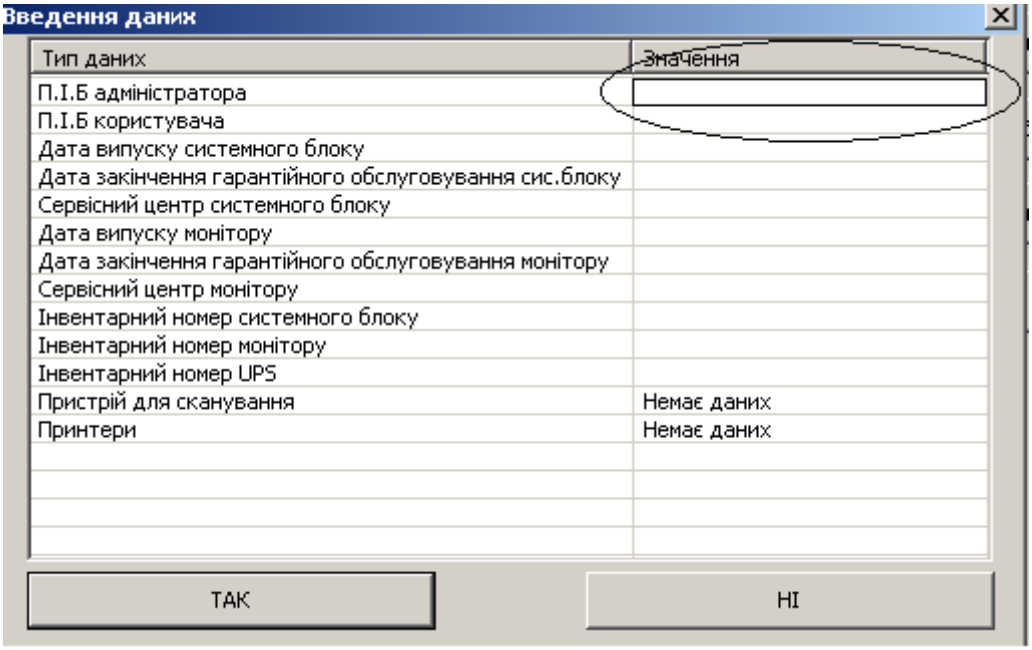


Рисунок 2.6. – Процес введення даних

Для того, що б ввести дані необхідно: навести курсор на потрібну комірку і виконати одинарне натискання лівої кнопки миші, після цього активується функція редагування (рис. 2.6.), вводимо дані і переводимо фокус миші на іншу клітинку або ж на інше місце.

Після заповнення полів, при натисканні кнопки «ТАК» відбувається перевірка на порожні поля. На рисунку 2.7 зображена блок-схема алгоритму заповнення рядків користувачем програми.

Рисунок 2.7. – Блок-схема алгоритму заповнення рядків користувачем програми

Бульова зміна `EmptyFields` класу `InUsrDatadlg` встановлюється в значення TRUE та генерується вікно попереджень з проханням заповнити всі поля. За допомогою функції `IsEmpty`, яка повертає `true` при порожній клітинці. Після цього булева змінна `EmptyFields` встановлюється в `true` і відбувається вихід з циклу за допомогою оператора `break`. У випадку заповнених всіх комірок, булева змінна `EmptyFields` встановлюється в `false` і можна зберегти звіт. Помилка про пусті поля буде з'являтись до тих пір поки не будуть заповнені всі поля. Код для перевірки порожніх комірок наведено нижче.

```
for(int i=0; i< Lst.GetItemCount(); i++)
{
     if(Lst.GetItemText(i,1).IsEmpty()) {
          EmptyFields=true;
          break;
     } else {
     EmptyFields=false;
     }
}
```

При успішному результаті відбувається заповнення властивостей класу `Cuserdata`.

2.2.3. Опис класу створення Excel листа

Клас `Cxls` призначений для формування вихідного файла звіту у форматі електроних таблиць Excel. Цей формат був обран, як стандартний формат з формування інформації будь-якого типу.

Даний клас являє собою оболонку стороньої бібліотеки Libexcel. Бібліотека Libexcel інкапсулює в собі багато методів та властивостей для створення вихідного файлу у форматі електронних таблиць Excel. Дуже зручна та проста у використані, має в своему складі алгоритми виявлення помилок та обробку винятків. Для подібного завдання є можливість працювати напряму з COM технологією для створення електроних таблиц Excel, але вона не дає такої простити у використанні та не надає можливості детально корегувати обрані комірки в таблици, що дуже ускладнює процес створення вихідного файлу звіту. Однією з переваг у використані бібліотеки Libexcel є те що, створення вихідного файла звіту відбувається навіть при відсутності встановленого потрібного програмного забезпечення, а саме Microsoft Office. COM технологія не надає такої можливості. Створення звіту без потрібного встановленого на комп’ютері програмного забезпечення важливо для адміністратора, так як для встановлення Microsoft Office потрібний час. Даний дипломний проект та сама програма OblikTehniki орієнтовані на простоту у використані при мінімальних потребах. Виходячи з цього оптімальним рішенням для створення вихідного файлу звіту у дипломному проекті та програмі використовується бібліотека Libexcel.

Бібліотека Libexcel статична і тому складається з декількох файлів типу lib для різних бібліотек часу виконування (libexcel-MDd.lib, libexcel-MD.lib, libexcel-MTd.lib, libexcel-MT.lib) та заголовних файлів: Libexcel.h, Error.h, Format.h, Workbook.h, Worksheet.h. У файлі Libexcel.h визначається, яку треба підключити статичну бібліотеку для подальшої работи. Для того щоб

сформувати кінцевий файл звіту, потрібно задати всі необхідні значення для кожної клітинки звіту.

Бібліотека Libexcel використовує в якості строкових змінних тип wstring, який не дуже зручний. Перетворення `CString` в `wstring` реалізовано в наступному фрагменті коду.

```
wstring
MBoardW = CStringW(hwdataIN.mboard),
procW = CStringW(hwdataIN.processor),
memoryW = CStringW(hwdataIN.memory),
videoW = CStringW(hwdataIN.video),
monitorW = CStringW(hwdataIN.monik),
scanW = CStringW(hwdataIN.scanner),
inv_upsW = CStringW(userdataIN.inv_ups),
osW = CStringW(hwdataIN.os_str),
hostnameW = CStringW(hwdataIN.pcname),
inv_monitorW = CStringW(userdataIN.inv_mon),
```

У всій програмі використовуються рядкові змінні типу `CString`, тому для коретної роботи функції бібліотеки Libexcel необхідно перетворити `CString` в `wstring`. Кінець фрагмент коду перетворення `CString` в wstring наведено нижче.

Для перетворення массивів рядкових масивів використовується функція `CstrArrToWstrArr`, код якої наведено нижче.

```
void Cxls::CstrArrToWstrArr(vector<CString> &inC,
vector<wstring> &outW){
     for(int i = 0; i < (int)inC.size(); i++ ){
          wstring TempW = CStringW(inC[i]);
          outW.push_back(TempW);
     }
}
```

Функція `CreateZvit` класу Cxls призначена для створення файл звіту у форматі електроних таблиц Excel, нічого не повертає, приймає об’єкт типу Chwdata та об’єкт типу Cuserdata. В цих об’єктах міститься вся необхіодна інформація: дані про апаратну та програмну частину комп’ютера і дані, які ввів користувач.

```
Worksheet* sheet = xls->addWorksheet(L"zvit");
Format* format = xls->addFormat();
sheet->setColumn(0,0,14); //A
sheet->setColumn(0,1,13); //B
sheet->setColumn(0,2,9); //C
sheet->setColumn(0,3,18); //D
sheet->setColumn(0,4,16); //E
sheet->setColumn(0,5,28); //F
sheet->setColumn(0,6,14); //G
sheet->setPrintScale(75);
sheet->showGridlines(false);
format->setSize(18);
format->setBold();
sheet->write(L"D1",L"Облікова картка комп’ютера", format);
format = xls->addFormat();
format->setSize(12);
```

Спочатку треба створити робочий лист за допомогою функції addWorksheet, яка в якості параметрів приймає ім’я робочого листа. Наступним кроком буде встановлення необхідних розмірів для рядків і стовпців, для цього треба створити об’єкт format типу Format та присвоєти йому результати виконання функції addFormat покажчика xls.

```
format->setLeft(Format::BORDER_THIN);
format->setTop(Format::BORDER_THIN);
format->setBottom(Format::BORDER_THIN);
sheet->write(L"A3",L"П.І.Б користувача", format);
format = xls->addFormat();
format->setTop(Format::BORDER_THIN);
format->setBottom(Format::BORDER_THIN);
sheet->write(L"B3",L"", format);
format = xls->addFormat();
format->setTop(Format::BORDER_THIN);
format->setBottom(Format::BORDER_THIN);
sheet->write(L"C3",L"", format);
format = xls->addFormat();
```

Потім встановлюємо розмір шрифту, стиль і позиціонування тексту всередині самої комірки. Запис у комірку відбувається за допомогою функції write, яка в якості параметрів приймає ім’я комірки та текст.

2.2.4. Опис класу головного вікна програми

Клас `OblikTehnikiDlg` являє собою клас головного діалогового вікна. За допомогою класу `OblikTehnikiDlg` відбуваються всі операції для введення даних та створення звіту. Додавання іконок в дерево з апаратними та програмними модулями, здійснюється за допомогою функції TreeInit. Фрагмент коду процесу додавання іконок до вузлів дерева наведено нижче.

```
m_img.Create( 32, 32, ILC_COLOR24,0,0);
m_img.Add(AfxGetApp()->LoadIcon(IDI_ICON1));
m_img.Add(AfxGetApp()->LoadIcon(IDI_ICON2));
m_img.Add(AfxGetApp()->LoadIcon(IDI_ICON3));
m_img.Add(AfxGetApp()->LoadIcon(IDI_ICON5));
```

Спочатку треба завантажити колекцію іконок, вказавши при цьому вказавши розміри. Потім встановити іконки в необхідні вузли дерева. Встановлення іконок на свої місця показано в фрагменті коду, який зображений нижче.

```
m_tree.SetItemImage(tree_pc,0,0);
m_tree.SetItemImage(tree_os,1,1);
m_tree.SetItemImage(tree_os1,13,13);
m_tree.SetItemImage(tree_mb,2,2);
m_tree.SetItemImage(tree_mb1,13,13);
m_tree.SetItemImage(tree_proc,3,3);
m_tree.SetItemImage(tree_proc1,13,13);
m_tree.SetItemImage(tree_mem,4,4);
```

Процес додавання статичних значень та інформацію про апаратну і програмну частину комп’ютера, у фрагменті коду, який наведено нижче. Спочатку створюється змінна `tree_pc` типу `HTREEITEM`, для додавання нового вузла в дерево відображуваних апаратних та програмних модулів.

```
HTREEITEM tree_pc = m_tree.InsertItem
("Комп’ютер - "+hw_DATA.pcname);
HTREEITEM tree_os = m_tree.InsertItem
("Операційна система",tree_pc);
```

```
HTREEITEM tree_os1 = m_tree.InsertItem
(hw_DATA.os_str,tree_os);
HTREEITEM tree_mb = m_tree.InsertItem
("Системна плата",tree_pc);
HTREEITEM tree_mb1 = m_tree.InsertItem
(hw_DATA.mboard,tree_mb);
HTREEITEM tree_proc = m_tree.InsertItem
("Процессор",tree_pc);
HTREEITEM tree_proc1 = m_tree.InsertItem
(hw_DATA.processor,tree_proc);
HTREEITEM tree_mem = m_tree.InsertItem
("Оперативна пам’ять",tree_pc);
```

Наступним кроком буде створення безпосередньо вузлів для виведення значень, для цього змінної `tree_pc` присвоюємо результат виконання функції `InsertItem` об’єкту `m_tree`.

Програмна реалізація кнопки «Створити та зберегти» складається з наступних кроків:

– визначення поточної дати;
– створення діалогового вікна збереження;
– отримання шляху для збереження звіту;
– створення змінної класу `Cxls`;
– перевірка при створенні звіту.

Визначення поточної дати відбувається за допомоги стандартної WINDOWS API структури `SYSTEMTIME` та функції `GetLocalTime`.Після отримання поточної дати, треба перетворити в рядок, який складається з імені комп’ютера, поточної дати та розширення xls. Фрагмент коду наведено нижче.

```
if(usr_DATA.ok_usrdata)
{
     SYSTEMTIME pTime;
     GetLocalTime(&pTime);
     CString month,day,year;
     if (pTime.wDay< 10)
     {
          CString day_temp;
          day_temp.Format("%d",pTime.wDay);
          day="0"+day_temp;}
          else {day.Format("%d",pTime.wDay);}
          if (pTime.wMonth < 10)
           {
```

```
                CString month_temp;
                month_temp.Format("%d",pTime.wMonth);
                month="0"+month_temp;}
                else {month.Format("%d",pTime.wMonth);}
                year.Format("%d",pTime.wYear);
                CString fname
      = hw_DATA.pcname+"_"+day+"_"+month+"_"+year+".xls";
```

Наступним кроком буде створення діалогового вікна вибору шляху для збереження файлу звіту.

Програмна реалізація наведена нижче.

```
CFileDialog fsave(0,0,fname,0,"
Microsoft Office Excel (*.xls) |*.xls|",0,0,0);
fsave.DoModal();
fpath=fsave.GetPathName();
```

Потім, якщо шлях збереження обраний вірно або користувач програми «Облік комп'ютерної техніки» не натиснув «Скасувати» відбувається створення звіту.

При створенні нової книги функцією Workbook їй передається параметр ім'я звіту, який було сформовано раніше. Далі викликається функція `CreateZvit`, яка приймає об'єкт типу Chwdata та об'єкт типу `Cuserdata`. Це два головних об'єкта, тому як містять в собі всю необхідну інформацію для створення звіту. Вихідний реалізації створення звіту наведено нижче.

```
if(fname !=fpath)
{
     Cxls m_xls;
     try {
     m_xls.xls = new Workbook(string(fname));
     m_xls.CreateZvit(hw_DATA,usr_DATA);
     m_xls.xls->close();
     delete m_xls.xls;
     }
catch(const Error& e)
     {
          wstring errW = e.message();
          CString Cerr(errW.c_str());
          AfxMessageBox(Cerr);
          ErrorSave = true;
```

```
     }
}
```

Перевіряємо успішність збереження звіту, за допомогою класу `CFileFind` і методом його об’єкту `FindFile`. Метод `FindFile` в якості параметів приймає шлях до файлу, який треба перевірити на наявність.

Програмна реалізація у вигляді коду наведена нижче.

```
if (!ErrorSave){
     CFileFind ff;
     if(ff.FindFile(fpath) && fname !=fpath) {
          MessageBox("Файл звіту збережен!","Увага!",0);
          ErrorOpenZvit = false;
} else {
     MessageBox("Файл звіту НЕ збережен!","Увага!",0);
     }
}
```

Результат збереження генерується у вигляді діалогового вікна, яке повідомляє про успішність або невдачу операції.

РОЗДІЛ 3 КЕРІВНИЦТВО КОРИСТУВАЧА

3.1. Запуск програми «Облік комп’ютерної техніки»

Програма «Облік комп’ютерної техніки» не потребує встановлення та налаштування, що робить її мобільною та простою в експлуатації. Для запуску програми необхідно подвійним натисканням лівої кнопки миші на виконуючий файл (рис. 3.1).

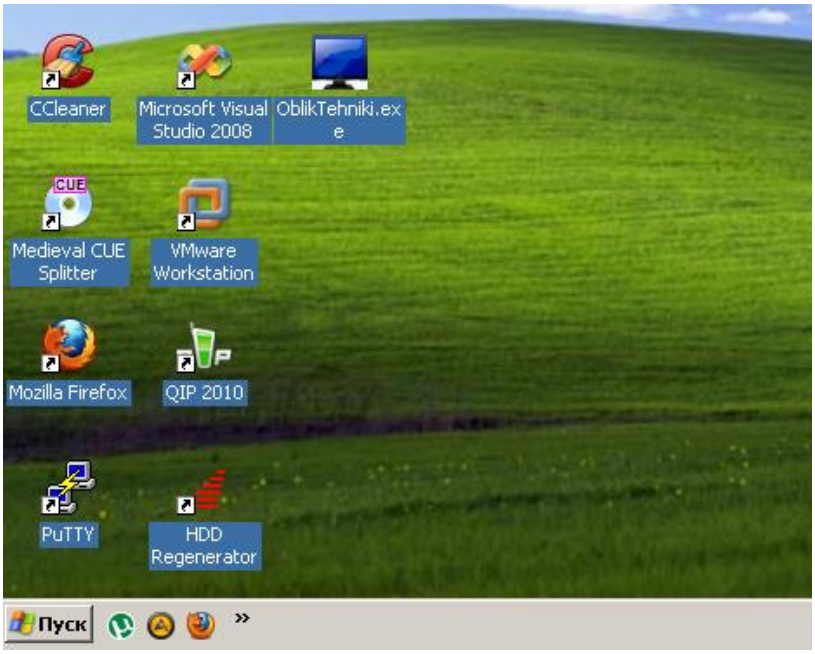


Рисунок 3.1. – Робочій стіл з програмою «Облік комп’ютерної техніки»

Також є можливість запускати програму по мережі (рис. 3.2), працювати в ній і зберігати звіти як на локальному комп’ютері так і на віддаленому. Для раціонального використання ресурсів жорстких дисків комп’ютерів, виконавчий файл програми потрібно розмістити на файловому сервері, або ж на загальному ресурсі одного з комп’ютерів.

Після запуску програми, відкривається головне діалогове вікно (рис. 3.3).

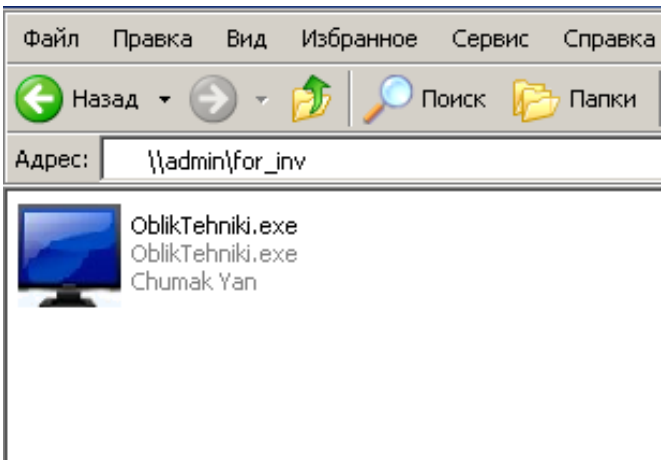


Рисунок 3.2. – Запуск програми «Облік комп’ютерної техніки»в мережевому режимі

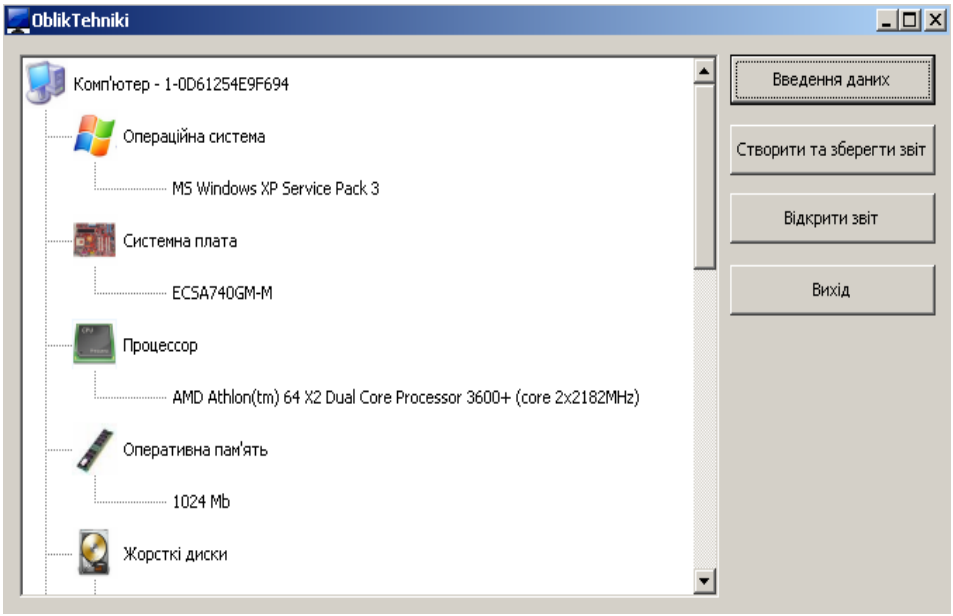


Рисунок 3.3. – Головне діалогове вікно програми «Облік обчислювальної техніки»

3.2. Робота в програмі

Для візуалізації даних в вікні програми відображені основні компоненти для інвентаризації техніки, які потрапляють у файл звіту. Для керування програмою , на головному діалоговому вікні є кнопки:

– «Введення даних»;

– «Створити та зберегти звіт»;

– «Відкрити звіт»;

– «Вихід».

Програма «Облік обчислювальної техніки» виконує перевірку на заповнення всіх даних вхідної форми, якщо дані не були занесені, сформувати звіт, зберегти і відкрити його буде неможливо (рис. 3.4).

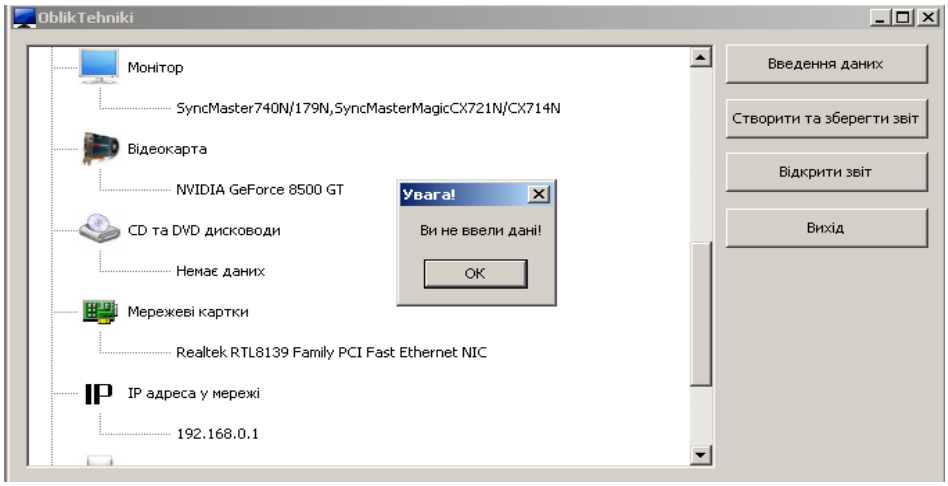


Рисунок 3.4. – Помилка при створенні звіту

Після заповнення даних, потрібно створити та зберегти звіт, якщо цього не зробити і спробувати відкрити звіт, автоматично згенерується повідомлення про помилку (рис. 3.5).

В програмі «Облік обчислювальної техніки» реалізована перевірка на порожні значення.

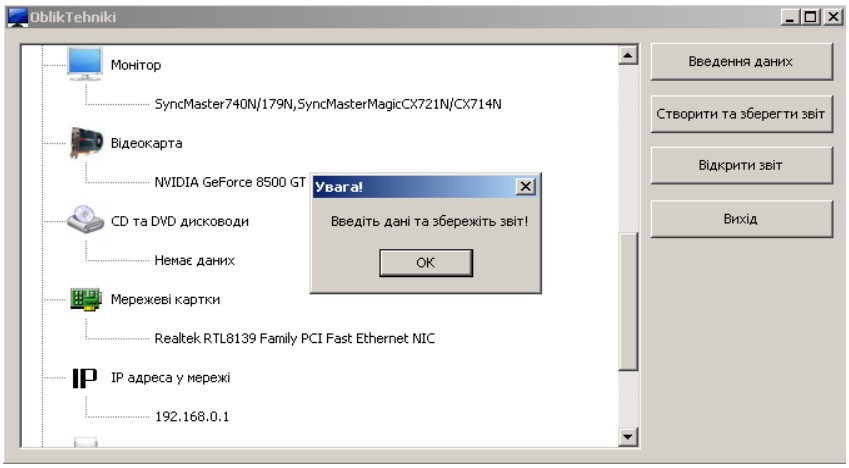


Рисунок 3.5. – Помилка при відкритті звіту

У разі натискання кнопки «ТАК» при порожніх полях з’являється повідомлення у вигляді діалогового вікна з текстом повідомлення «Ви не ввели дані».

3.2.1. Введення даних

На початку роботи з програмою «Комп’ютерний облік обчислювальної техніки» потрібно ввести дані користувача комп’ютера (рис.3.6). Це дані, які користувач вводить до програми власноруч:

– інвентарні номери пристроїв;
– ім’я користувача та адміністратора;
– назву сервісних центрів;
– терміни дії гарантійного обслуговування.

За правильність даних відповідальний користувач програми «Комп’ютерний облік обчислювальної техніки».

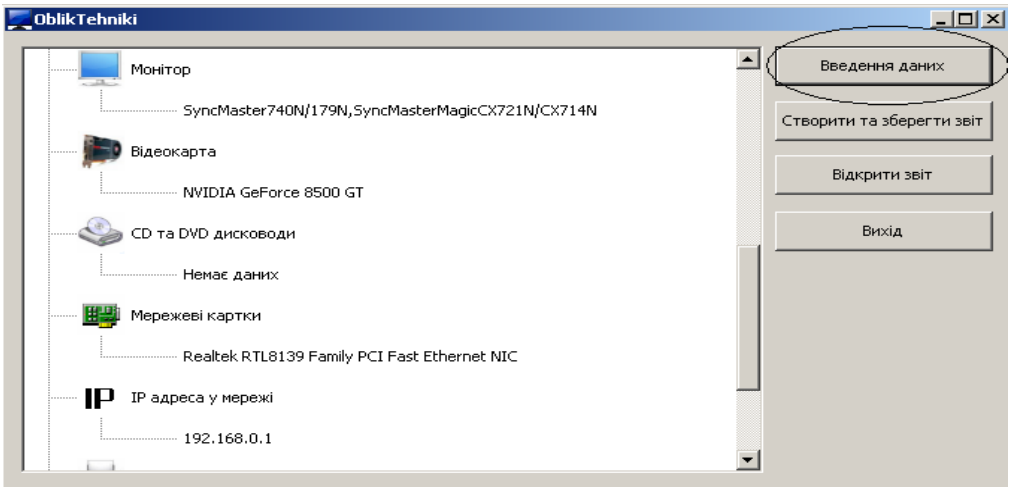


Рисунок 3.6. – Введення даних

Для внесення даних необхідно активувати функцію редагування рядків, для цього потрібно навести курсор на необхідне поле та один раз натиснути ліву кнопки миші.

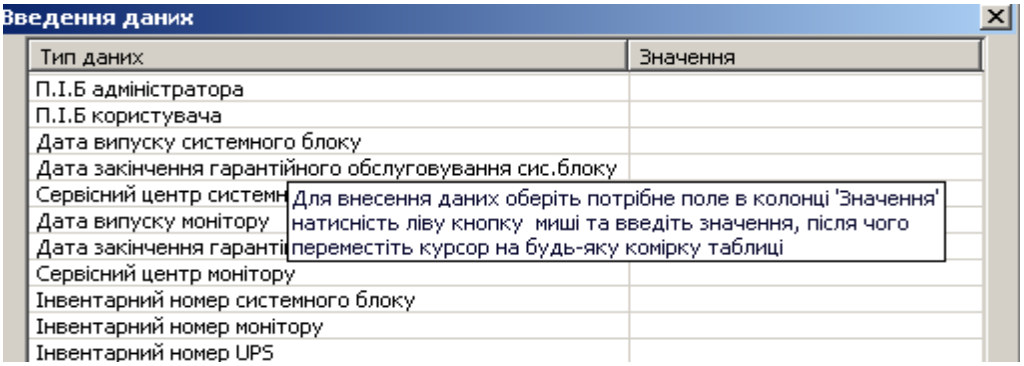


Рисунок 3.7. – Діалогове вікно для вводу даних

В якості допомоги користувачу, у програмі «Облік обчислювальної техніки» є спливаючі підказки. Вони з’являються при наведенні курсору миші на діалогове вікно «Введення даних» (рис. 3.7).

Після заповнення всіх рядків, потрібно натиснути «ТАК». Для перевірки рядків на порожні значення, в програмі є алгоритм перевірки рядків (рис. 3.8).

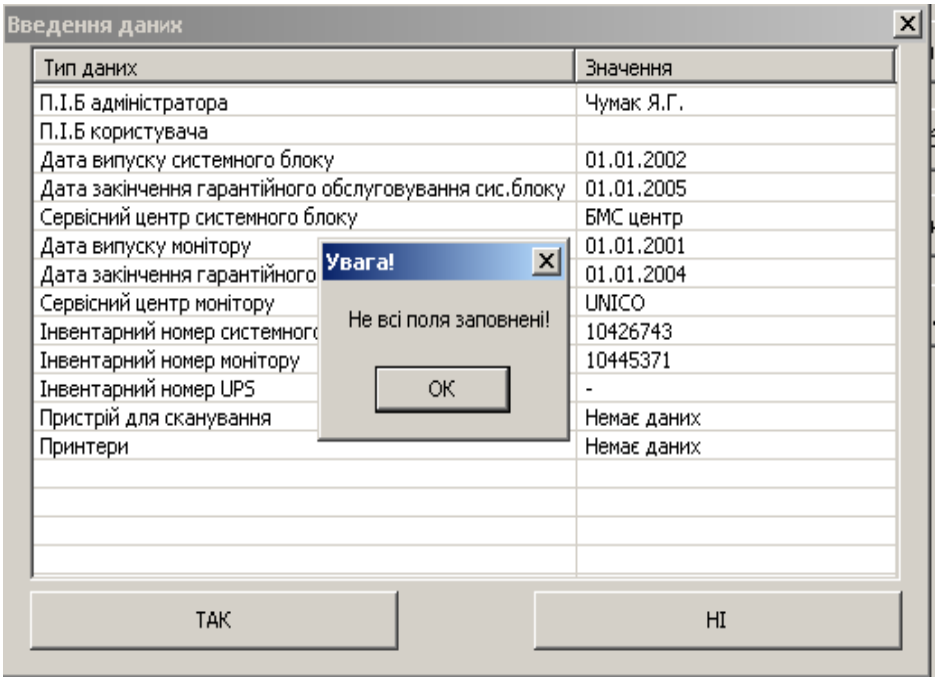


Рисунок 3.8. – Перевірка рядків на порожні значення

У випадку заповнення невірними даними є можливість коригування без введення всіх даних спочатку. Для цього в програмі організовано тимчасове збереження даних.

3.2.2. Створення та збереження файлу звіту

Файл звіту програми облік обчислювальної техніки, формуються в форматі електронних книг Microsoft Excel. Шаблон за яким генеруються звіт, повністю відповідає всім вимогам для використання його в Пенсійних фондах Україні. Шаблон у програмі статичний і не може бути змінений.

Після коректного введення всіх даних користувача, потрібно натиснути на кнопку «Створення та збереження звіту» (рис. 3.9).

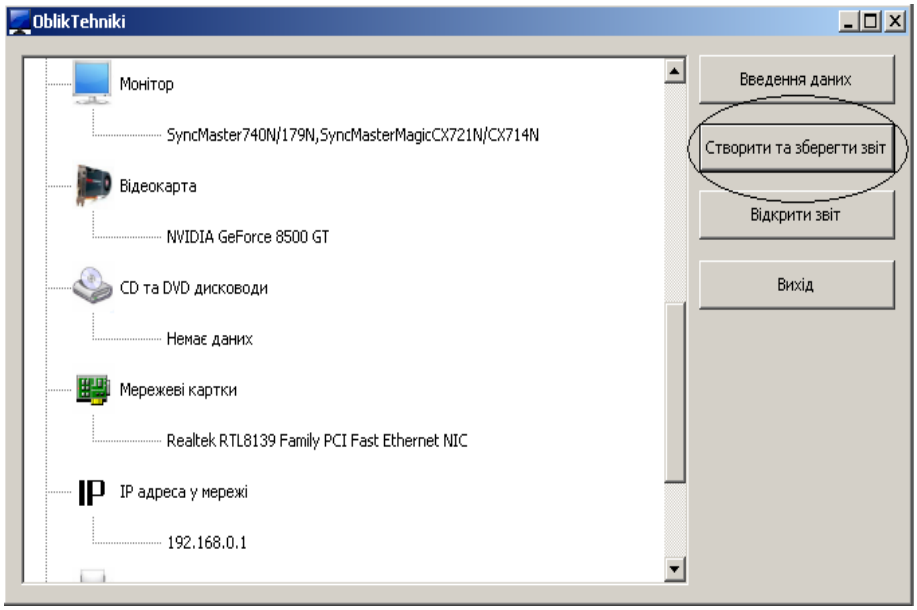


Рисунок 3.9. – Створення та збереження файлу звіту

Відразу після натискання кнопки «Створення та збереження звіту» відкривається стандартне діалогове вікно збереження файлів (рис. 3.10). В даному вікні є можливість вказати місце для збереження файлу звіту. Ім’я звіту являє собою рядок, який генерується автоматично і складається з наступних змінних:

– ім’я комп’ютера;

– поточна дата;

– розширення файлу.

Для збереження файлу звіту у формат електронних книг Excel програма не потребує встановлене програмного забезпечення (наприклад Microsoft Office) для роботи з цим форматом. Як правило в установах Пенсійного фонду України використовується програмне забезпечення Microsoft Office.

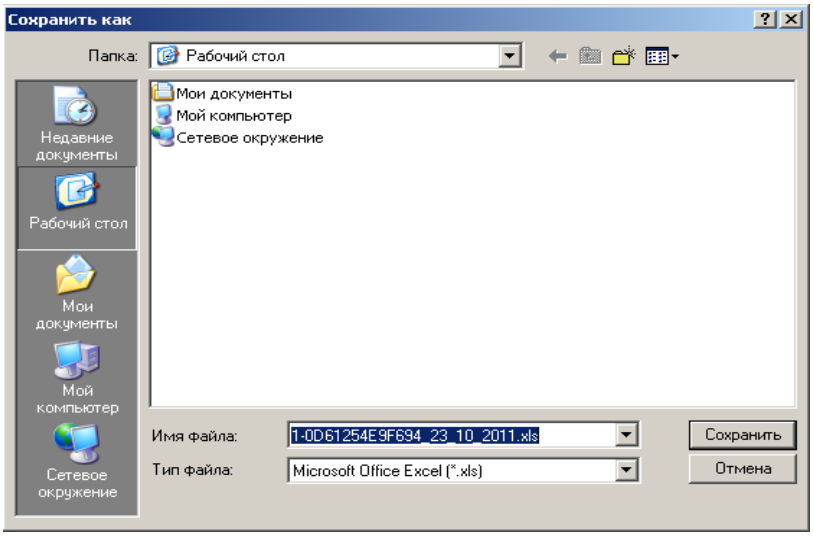


Рисунок 3.10. – Вибір місця для збереження файлу звіту

При натисканні кнопки «Збереження» генерується вікно, яке свідчить про успішність чи помилку цієї операції (рис. 3.11).

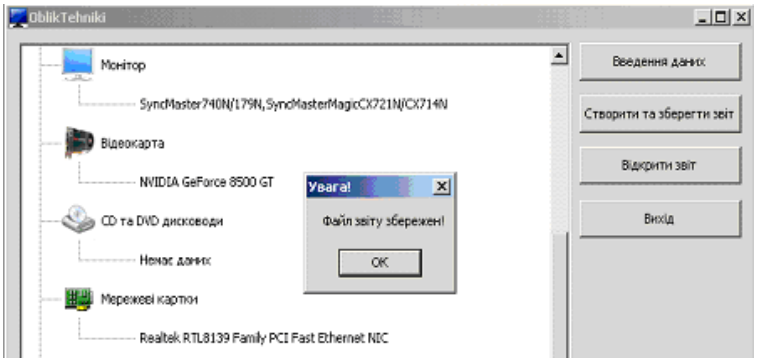


Рисунок 3.11. – Успішне збереження звіту

Якщо користувач програми натисне на кнопку «Скасувати» при збереженні, автоматично згенерується діалогове вікно помилки збереження.

3.2.3. Відкриття звіту

Для зручності, програма облік обчислювальної техніки має в своєму складі функцію відкриття збереження звіту, яка викликається за допомогою кнопки «Відкрити звіт» (рис. 3.12).

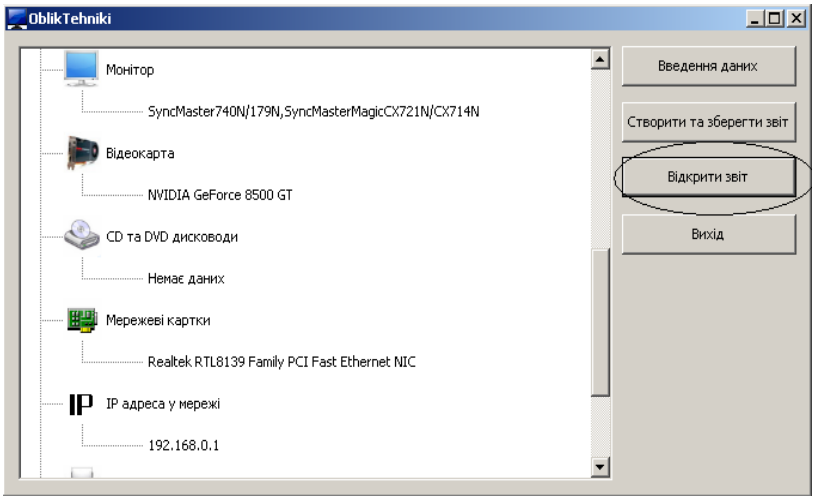


Рисунок 3.12. – Відкривання файла звіту

Після натискання на кнопку «Відкрити звіт» відкривається файлу звіту. Для коректного відкриття файлу необхідно встановлене програмне забезпечення для роботи з Excel формат, наприклад Microsoft Office або інші.

На рисунку 3.13. зображений вихідний файл звіту створений програмою Комп’ютерний облік комп’ютерної техніки. Пусті поля в звіті «Дата ремонту» та «Дата закінчення гарантійного обслуговування» заповнюються за фактом ремонту, спеціалістом відповідальним за збереження матеріальних цінностей. У стовпці «Найменування» крім монітору та системного блоку, які рахуються за одиницю техніки можуть бути інші комплектуючі.

Наприклад, при виході з ладу мережевої карти, в полі «Найменування» пишеться «Мережева карта», а в полі «Дата ремонту» вказується дата заміни або ремонту конкретної комплектуючої.

	A	B	C	D	E	F	G
1				**Облікова картка комп'ютера**			
2							
3	П.І.Б користувача				Каденко О.А		
4	Інвентарний номер системного блоку				1045631		
5	Інвентарний номер монітору				1045373		
6	Конфігурація	Материнська плата			ECSA740GM-M		
7		Процесор			AMD Athlon(tm) 64 X2 Dual Core Processor 3600+ (core 2x2182MHz)		
8		Жорсткі диски			SAMSUNG SP0612N 60Gb		
9					WDC WD800BB-55JKC0 80Gb		
10		Пам'ять			1024 Mb		
11		Відеокарта			NVIDIA GeForce 8500 GT		
12		CD та DVD ROM пристрої			Немає даних		
13		Мережева карта			Realtek RTL8139 Family PCI Fast Ethernet NIC		
14		Монітор			SyncMaster740N/179N,SyncMasterMagicCX721N/CX714N		
15	Периферійні пристрої	Принтер			Немає даних		
16		Сканер			Немає даних		
17		Інвентарний номер UPS			-		
18	Операційна система				MS Windows XP Service Pack 3		
19	Ім'я у мережі				1-0D61254E9F694		
20	Адреса у мережі				192.168.0.1		
21							
22		**Відомості про ремонт та гарантійне обслуговування**					
23							
24	Найменування	Дата випуску	Дата закінчення гарантійного обслуговування		Сервісний центр	Дата закінчення гарантійного обсуговування по ремонту	Дата ремонту
25	Системний блок	01.01.2007	31.12.2007		BMS		
26	Монітор	01.01.2006	01.01.2008		UNICO		
27							
28							
29							

Рисунок 3.13. – Файл звіту сформований програмою

3.2.4. Завершення роботи програми

Після введення даних, створення і збереження файлу звіту, потрібно коректно завершити роботу програми «Облік обчислювальної техніки». Для цього передбачена кнопка «Вихід» (рис. 3.14) натискання якої забезпечує програма успішно завершує свою роботи.

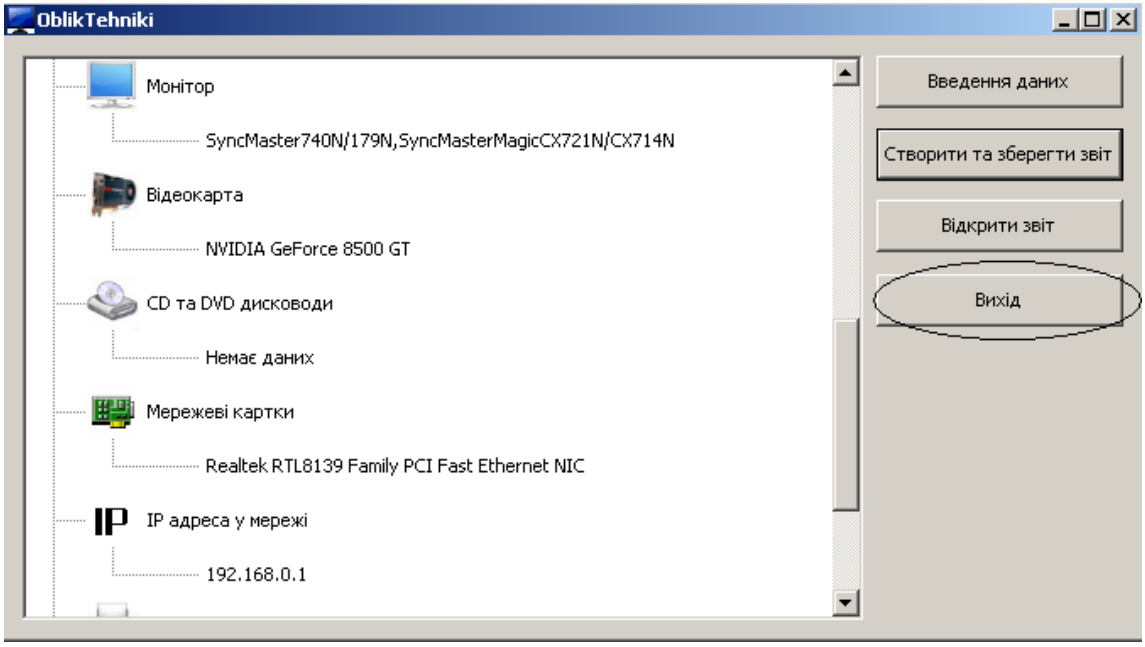


Рисунок 3.14. – Вихід з програми

Програма «Облік обчислювальної техніки» під час своєї роботи не створює тимчасових файлів.

ВИСНОВКИ

Розробка програми для системи обліку несправностей комп’ютерного обладнання підприємства є важливим завданням, яке допомагає забезпечити ефективне управління технічними проблемами і зберегти продуктивність комп’ютерних ресурсів. Така програма дозволяє систематизувати та відстежувати інформацію про виявлені несправності, а також вести контроль над процесом їх вирішення.

Розробка програми системи обліку несправностей комп’ютерного обладнання підприємства по суті виявляється діагностикою системи, тобто це процес ретельного тестування всіх компонентів комп’ютера з метою визначення відповідності їх характеристик заявленим виробником. Цей процес також включає вимірювання реальної продуктивності, такої як швидкість роботи, і порівняння цих показників з еталонними значеннями.

У випускній роботі розглядаються класифікація і задачі діагностування комп’ютерної техніки, здійснюється їх аналітичний огляд. Детально описано процес розробки програми «Система обліку комп’ютерного обладнання».

У даній дипломній роботі мовою програмування обраний C++. В якості середовища розробки найкраще підходить пакет Microsoft Visual Studio 2008, який допомагає оптимізувати та спрощує процес розробки високоефективних програм. Для отримання достовірної інформації про використання застосовуваних технологій і підходів, тестування проводилося на понад 50 комп’ютерах різної конфігурації.

Програма «Облік комп’ютерної техніки» не потребує встановлення та налаштування, що робить її мобільною та простою в експлуатації.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як користуватися Everest. URL : https://uk.soringpcrepair.com/how-to-use-everest/.
2. Діагностика комп'ютера за допомогою програми EVEREST. URL : http://vidpoviday.com/diagnostika-kompyutera-za-dopomogoyu-programi-everest.
3. Програма Everest (AIDA64): що це таке, навіщо потрібна, як користуватися. URL : https://hi-news.pp.ua/kompyuteri/9525-programa-everest-aida64-scho-ce-take-navscho-potrbna-yak-koristuvatisya.html.
4. Як дізнатися продуктивність комп'ютера – все про комп'ютери. URL : http://pro-computer.pp.ua/8671-yak-dznatisya-produktivnst-kompyutera-vse-pro-kompyuteri.html.
5. Програми для тестування комп'ютера. URL : https://uk.soringpcrepair.com/software-for-test-computer/.
6. Список безкоштовних портативних програм для Windows, утиліти, програми. URL : https://uk.begin-it.com/9745-free-portable-windows-tools-utilities-applications.
7. Что такое «winaudit.exe»? URL : https://systemexplorer.net/ru/file-database/file/winaudit-exe.
8. Як користуватися CPU-Z. URL : https://uk.soringpcrepair.com/how-to-use-cpu-z/.
9. CPU-Z: як користуватися, опис програми та можливості. URL : https://hi-news.pp.ua/kompyuteri/14061-cpu-z-yak-koristuvatisya-opis-programi-ta-mozhlivost.html.
10. 18 найкращих програм для комп'ютерних стрес-тестів для тестування процесора, оперативної пам'яті та графічного процесора. URL : https://uk.myservername.com/18-top-computer-stress-test-software-test-cpu.
11. Як користуватися програмою GPU-Z. URL : https://uk.soringpcrepair.com/how-to-use-gpu-z/.

12. Найкращі програми для діагностики комп'ютера. URL : https://www.imena.ua/blog/best-programs-for-computer-diagnostics/.
13. Кращі програми для діагностики комп'ютера на несправності. URL : http://smartandyoung.com.ua/krashhi-programi-dlja-diagnostiki-komp-jutera-na.
14. Програми для діагностики комп'ютера. URL : https://uk.soringpcrepair.com/computer-diagnostic-software/.
15. PC-Wizard: аналіз та порівняння вашого комп'ютера Windows. URL : https://uk.begin-it.com/3393-analyze-and-benchmark-your-windows-computer-system.
16. Моніторинг комп'ютера – кращі програми. URL : https://hi-news.pp.ua/kompyuteri/print:page,1,13298-montoring-kompyutera-krasch-programi.html.
17. Як користуватися програмою HDDScan: Основні функції. URL : http://web-city.org.ua/yak-koristuvatisya-programoyu-hddscan-osnovni-funktsiyi/.
18. Як перевірити жорсткий диск на помилки. URL : https://voll.kiev.ua/uk/blog/yak-pereviriti-zhorstkij-disk-na-pomilki.
19. Microsoft Foundation Classes. URL : https://ru.wikipedia.org/wiki/Microsoft_Foundation_Classes.
20. Бібліотека MFC. URL : http://ni.biz.ua/11/11_8/11_8789_biblioteka-MFC.html.
21. Комп'ютерно-технічна експертиза. URL : https://kndise.gov.ua/kompyuterno-tehnichna/.