Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи контролю та управління доступом на підприємстві”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Продкун А.О.
«____» _____ 2021 р.

Керівник проекту
доктор технічних наук, профсесор
_____ Олексій СМІРНОВ
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Продкун Анастасії Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи контролю та управління доступом на підприємстві*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи контролю та управління доступом на підприємстві*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	
6. Наукова новизна	

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Продкун А.О. Дослідження та програмна реалізація системи контролю та управління доступом на підприємстві. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи контролю та управління доступом на підприємстві.

Метою розробки є дослідження та програмна реалізація системи контролю та управління доступом на підприємстві.

Об'єктом дослідження є процес контролю та управління доступом на підприємстві.

Предметом дослідження є методи контролю та управління доступом на підприємстві.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи контролю та управління доступом на підприємстві.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Embarcadero RAD Studio Delphi 10.4 Sydney.

Ключові слова: комп'ютерна інженерія, управління доступом

ABSTRACT

Prodkun A.O. Research and software implementation of access control and management system at the enterprise. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of higher education the software which is intended for system of control and management of access at the enterprise is developed.

The purpose of development is research and software implementation of access control and management system at the enterprise.

The object of research is the process of access control and management in the enterprise.

The subject of research is the methods of access control and management at the enterprise.

Research methods are based on the methods of information security theory, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the access control and management system at the enterprise.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on a PC IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment Embarcadero RAD Studio Delphi 10.4 Sydney.

Keywords: computer engineering, access control

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи 7
1.2 Область застосування 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 12
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 12
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 21
2.3 Розгорнута постановка завдання 27
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 29
3.1 Опис функціонування системи 29
3.2 Розробка структурної схеми 38
3.3 Розробка функціональної схеми 43
3.4 Розробка діаграми процесів 44
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 46
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 46
4.2 Захист розробленого програмного забезпечення 49
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 51
6 НАУКОВА НОВИЗНА 54

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ 55
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. 55
7.2 Розрахунок трудомісткості розробки програмної продукції 57
7.3 Визначення чисельності виконавців і планового фонду зарплати 59
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника 64
7.5 Визначення собівартості розробки та ціни програмної продукції. 68
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 71
7.7 Визначення експлуатаційних витрат..... 71
7.8 Визначення економічної ефективності програмної продукції..... 73
7.9 Висновок. 75
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 76
8.1 Вступ 76
8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером 77
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста .. 78
8.4 Розробка заходів з умов поліпшення охорони праці..... 81
8.5 Розрахункова частина 82
8.6 Висновки до розділу..... 83
9 ОСНОВНІ ВИСНОВКИ..... 84
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 86

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ – автоматизована система управління

ДСТ – держстандарт

ЕМЗ – електромагнітний замок

ЕОМ – електроно-обчислювальна машина

ІСО – міжнародний стандарт

ПЗ – програмне забезпечення

ПЗП – постійний запам’ятовуючий пристрій

ПК – програмний комплекс

СКУД – система контролю та управління доступом

СУД – система управління доступом

PIN – personal identification cod – персональний ідентифікаційний код

RTE – Request To Exit – кнопка «Вихід»

ВСТУП

Актуальність теми. Сучасні промислові інфраструктури та структури відпочинку, такі як готельні комплекси, офіси, приміщення корпорацій та банків, не обходяться без систем контролю та управління доступом. Завдяки контролю доступу можна запобігти крадіжкам або ушкодження майна, поставити заслін промисловому шпигунству, перекрити доступ на територію або в приміщення зловмисників. Система контролю доступу відповідає вимогам держстандарту: ДСТ 26342-89; ДСТ 50009; ДСТ 12.2.007.0; ДСТ 12.2.004; вимогам ІСО 9000 [1].

Як правило сучасна система контролю та управління побудована з використанням різного виду картрідерів з електронних карточок або таблеток iButton, при цьому пристрої, які організовують зчитування данних з електроних ідентифікаторів називаються картрідерами [1-5].

Нижче перерахуємо можливі варіанти використання картрідерів для забезпечення контролю та управління доступу до приміщень, котрі потребують захисту від фізичного вторгнення зловмисників.

Якщо до об'єкта пред'явлені підвищені вимоги безпеки, то безконтактний картрідер оснащується клавішною панеллю, при наявності якої користувач повинен спочатку ввести картку. У такий спосіб включається кодова клавіатура. Користувач набирає свій особистий код, і двері відкриваються. За допомогою кодової клавіатури вводиться й код тривоги, якщо даному співробітникові загрожує сторонння особа [4].

Контролю й управлінню піддаються як двері з картрідерами, так і без них. Заздалегідь задається час відкриття двері, якщо тривалість відкриття більше заданої, подається тривога. Система може бути запрограмована так, щоб двері відкривалися на певний час (наприклад, на період робочого дня) або тільки тоді, коли в певній зоні перебувають особи, що мають право доступу. Кожні двері можна з'єднати з охоронною й протипожежною системами при подвійному контролі доступу [5].

При відстеженні рухів на виході й вході (подвійний контроль доступу) використовуються шлюзи. Через шлюз може пройти одна людина. А так як в шлюзовій кабіні встановлюються картрідери як на вході, так і на виході, то, якщо людина ввійшла в шлюз, їй необхідно вийти, щоб знову пройти у зворотному напрямку. Тобто одна й та людина контролюється подвійно, як на вході, так й на виході [6].

За допомогою картрідерів можна різними способами управляти ліфтом: наприклад, заблокувати певні поверхи, вхід на які буде здійснюватися при наявності права доступу, або викликати ліфт на «секретні» поверхи за допомогою картки, що також обмежить користування ліфтом.

Потік відвідувачів можна регулювати наданням кожному з них певного часу візиту. Інформація з різними даними відвідувача зберігається в системі.

Контроль в'їзду здійснюється автоматично й на відстані, якщо водій тримає картку збоку у вікна машини. Спеціальні картки монтуються на днище автомобіля й ідентифікуються за допомогою закладеної в полотно дороги дротової петлі. Ця петля повинна бути захищена від як випадкових, так і навмисних ушкоджень.

Для автомобілів разом із системою контролю можна використовувати картрідер , що працює на високих частотах (2,4 Ггц). За допомогою картрідера контролюється в'їзд, транспортування товарів і т.д. картку цей картрідер ідентифікує з відстані до 6 метрів. Система контролю доступу корисна й для обліку робочого часу співробітників.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи контролю та управління доступом на підприємстві.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем контролю та управління доступом на підприємстві.

– Дослідження системи контролю та управління доступом на підприємстві.

– Програмна реалізація системи контролю та управління доступом на підприємстві.

Об'єктом дослідження є процес контролю та управління доступом на підприємстві.

Предметом дослідження є методи контролю та управління доступом на підприємстві.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод контролю та управління доступом на підприємстві.

– Розроблено вітчизняний продукт контролю та управління доступом на підприємстві, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі контролю та управління доступом на підприємстві.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи контролю та управління доступом на підприємстві, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Системи управління доступом

Системи управління доступом – СУД – напрямок, що найбільш активно розвивається у техніці забезпечення безпеки. СУД призначені для підвищення безпеки в робочий час.

Технічні засоби СУД містять у собі механічні, електромеханічні, електричні, електронні конструкції, пристрої й програмні засоби, що забезпечують контроль і управління доступом. Кожний користувач одержує індивідуальний ідентифікатор з кодом (це може бути карта, брелок, iButton та інше). Як ідентифікатор може бути використана біометрія (відбитки пальців, голос, сітківка ока, контур кисті руки й т.д.). Пристрій, що зчитує цю інформацію, передає її в систему, яка, у свою чергу, ситуативно реагує на отримані дані: відкриває або блокує двері, включає сигнал тривоги або просто реєструє співробітника на робочому місці.

Носії ідентифікаційних ознак (ключі-ідентифікатори) – це або різні карти (магнітні, віганд, проксіміті) – або сама людина як носій індивідуальних біометричних ознак. Знімають інформацію й передають її в контролер спеціальні пристрої – картрідери.

Добре зарекомендували себе смарт-карти, що володіють більшим обсягом пам'яті й високим ступенем захищеності від спроб модифікації й дублювання. Але ці плюси обертаються недоліком – високою ціною.

Touch Memory, або «інтелектуальна таблетка iButton» – це мікросхема, одягнена в корпус із нержавіючої сталі. Для ідентифікації таблетка прикладається до картрідера, і протягом 0,1 сек. відбувається процес зчитування інформації, у якості якої можуть виступати такі дані про користувача, як ФІО, підрозділ,

посада й т.п. Переваги «таблетки» – компактність, висока опірність до механічних ушкоджень, корозії, перепадів температур і іншим негативним зовнішнім впливам. Невисока ціна цього пристрою також говорить на його користь. На нашім ринку представлені системи Dallas Lock, Flex, Gars, «Акорд», «Менует», «Полонез» і ін. – усього біля десятка найменувань.

Для ідентифікації застосовуються штриховий код, карти з магнітною смугою, оптичний код, віганд-карти. Надійні й недорогі проксіміті-карти використовуються для дистанційного зчитування кодової інформації. Відстань між картрідером і картою (від 5 см до декількох метрів) визначається потужністю картрідера. Від максимальної дальності зчитування залежить вартість системи. Достоїнством проксіміті є можливість контролю як за людьми, так і за транспортом, а також безконтактне зчитування інформації. Основні виробники – HID Corporation, Indala Corporation, Motorolla, Cotag.

При всіх зручностях і надійності карт кожну з них можна елементарно втратити. Тому назріла необхідність створення біологічних терміналів, які працюють на ідентифікації біометричних даних. Однак, сучасні біометричні системи поки недостатньо ефективні, вартість їх висока, тому масового їхнього впровадження в самому найближчому майбутньому не передбачається.

Однак, крім управління доступом є ще й ряд завдань які пов'язані з контролем доступу.

1.2 Область застосування

Система контролю та управління доступом

Будь-яка система контролю й управління доступом (СКУД) призначена для автоматичного пропуску в приміщення (на територію) людей, що мають право доступу, і перешкоди проходу тим, хто цього доступу не має. СКУД – це не тільки апаратура й програмне забезпечення, але й система управління рухом персоналу.

Пристрій, по якому визначається користувач, називається ідентифікатором користувача. Ідентифікатори – це різні карти (магнітні, безконтактні proximity-карти), брелоки Touch Memory, радіобрелки.

А зчитує інформацію з ідентифікатора й передає її в контролер СКУД пристрій, який називається – картрідер.

Перешкода, обладнана картрідером і виконавчим пристроєм, – це точка проходу. Вона може бути повністю контрольованою й контрольованою на вхід. У першому випадку прохід оснащується двома картрідерами, у другому – тільки одним, на вхід. Вихід вільний або за допомогою кнопки RTE. Кнопка «Вихід» (RTE – Request To Exit) відкриває виконавчий пристрій. При цьому відкривання фіксується в пам'яті контролера без ідентифікації особистості того хто виходить. Кнопки RTE ставляться для забезпечення вільного виходу із приміщень.

Серцевиною СКУД є контролер. Цей пристрій призначений для обробки інформації від картрідера ідентифікаторів, ухвалення рішення й управління виконавчими пристроями. По способу управління контролери СКУД діляться на три класи:

- автономні контролери – повністю закінчені пристрої, що обслуговують, як правило, одну точку проходу. Розраховані на застосування різних типів картрідерів і на обслуговування невеликої кількості користувачів (до 500 чоловік).

- централізовані (мережні) контролери працюють у мережі під управлінням комп'ютера. Тут ухвалення рішення лежить на комп'ютері зі спеціалізованим програмним забезпеченням. Використання мережних контролерів дає величезну кількість додаткових можливостей (звіт про присутність співробітника на робочому місці; автоматичний табель обліку; ведення електронної картотеки й ін.)

- комбіновані контролери сполучають функції мережних і автономних. При наявності зв'язку з керуючим комп'ютером контролери працюють як мережний пристрій, якщо зв'язок відсутня – як автономне.

Для того, щоб утруднити передачу ідентифікатора іншій особі й мати можливість точного визначення місця знаходження власника картки, уводиться функція заборони подвійного проходу (anti passback).

Режим проходів, коли людина, що потрапила в приміщення, обладнане повністю контрольованою точкою проходу, не може пройти в будь-яке інше приміщення, попередньо не вийшовши з контрольованого, називається дисципліною проходу.

Прохідна – точка проходу з особливими функціями. Людина, що не пройшла через прохідну, не зможе потрапити в жодне приміщення об'єкта

Можливість виводу на екран монітора комп'ютера фотографії користувача називається фотоідентифікацією (PhotoID).

На підприємствах можна виділити чотири точки контролю доступу: прохідні, офіси, особливо важливі приміщення, в'їзди/виїзди автотранспорту. Залежно від вартої перед Вами завдання ви можете вибрати відповідну СКУД.

Огляд можливостей СКУД

Сучасні системи контролю й управління доступом (СКУД) діляться на два великі класи: автономні й мережні. Мережні системи (тут всі контролери з'єднані один з одним і підключені до комп'ютерів) використовуються, як правило, на великих підприємствах, автономні – у приватних котеджах або невеликих компаніях.

Найпоширеніший тип точки доступу в приміщення – це двері. На кожний вид дверей установлюється відповідний електричний (електромеханічний або електромагнітний) замок або засувка. Недоліком дверей є те, що слідом за людиною, що має право доступу в охоронюване приміщення, може ввійти сторонній. Тому в організаціях з великою кількістю працівників установлюють турнікети, тому що правильно встановлений турнікет пропускає одну людину (при цьому необхідний ідентифікатор). До плюсів цього типу точок доступу відноситься й висока пропускна здатність (2-3 сек. на одну людину).

Шлагбауми й ворота встановлюються на в'їздах на території підприємств і закриті автостоянки. При бажанні ці точки доступу можна оснастити картрідерами великої дальності, які на відстані в кілька метрів розпізнають мітки на автомобілях.

Об'єкти з підвищеними мірами безпеки використовують шлюзові кабіни, які можуть мати різноманітний алгоритм дії – від простих до дуже складних

Інтелектуальна складова СКУД – це контролери, пристрої ідентифікації (Proximity-картрідери, картрідери смарт-карти, біометричні картрідери), самі ідентифікатори й програмне забезпечення, що управляє всією системою.

Найпростіший спосіб ідентифікації – набір PIN-коду. Мінуси – необхідність запам'ятовувати послідовність цифр і можливість підглянути її сторонніми. Досить стійкими до несанкціонованого доступу є Proximity-карти. Але карта також може бути загублена, украдена або підроблена. Майже неможливо підробити смарт-карту, що йде на зміну Proximity-карті.

Найбільш модний сьогодні напрямок – ідентифікація по біологічних ознаках, наприклад, по відбитку пальця. Мінуси – низька пропускна здатність і високий рівень помилок, що допускаються картрідером. Але незабаром рівень пристроїв з біометричною ідентифікацією повинен піднятися досить високо.

Уже зараз існують пристрої, що сполучають у собі кілька технологій ідентифікації (наприклад, Proximity-картрідер із клавіатурою або сполучення розпізнавання по відбитку пальця й картрідера смарт-карт).

Алгоритми доступу в рамках всієї системи – це й заборона «подвійного проходу» (антипассбек), коли виключається можливість проходу по одній карті кількох людей, і прохід по тимчасовим (гостьовим) картах, і доступ у приміщення по двох картах, і багато що інше. Технології СКУД розвиваються досить динамічно, тому завжди можна знайти систему доступу, що відповідає вимогам замовника до безпеки свого об'єкта.

Типова схема розподіленої системи контролю та управління доступу приведена на рисунку 1.1

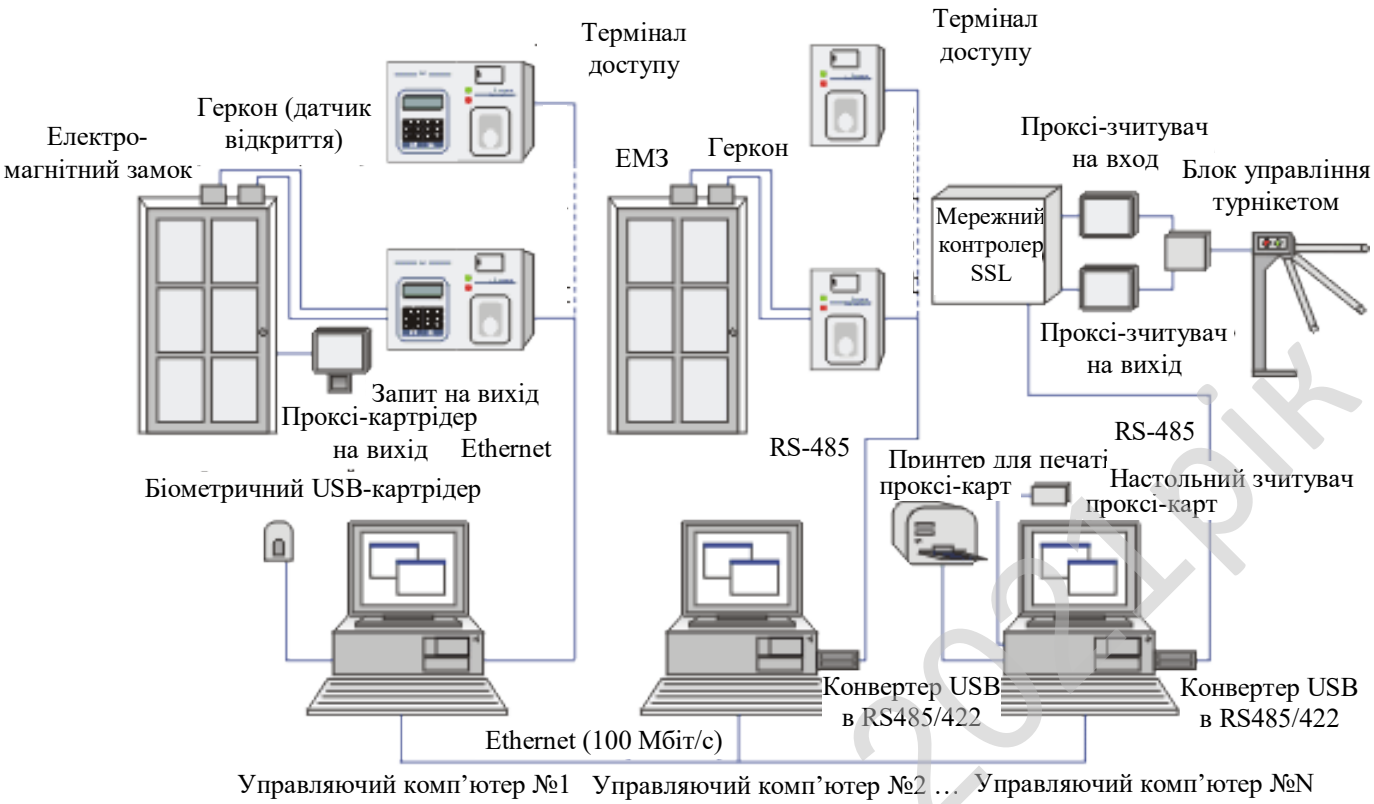

Рисунок 1.1 – Типова схема розподіленої системи контролю та управління доступу

З цієї схеми ми бачимо, що для реалізації сучасної системи контролю та управління доступу необхідно розробляти спеціалізоване програмне забезпечення, яке буде відповідати не тільки з передачу даних по каналам зв'язку мережі, але й буде захищати ці дані.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи контролю та управління доступом на підприємстві, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Вимоги до розробки й впровадження СКУД

У наш час, розробляючи й впроваджуючи охоронні системи, необхідно ставити в основу здатність тієї або іншої системи ефективно протистояти погрозі терористичного акту. Нові умови життя й роботи визначають і нові вимоги до СКУД. Сучасна СКУД, сполучаючи в собі функції затримки, виявлення й реагування, є єдиним інженерно-технічним комплексом, що одночасно вирішує завдання пропуску персоналу, відвідувачів і транспорту.

Один із пріоритетів сьогодні – виявлення зброї, вибухівки або отруйних речовин в автоматичному режимі. Уже зараз існують розробки, що дозволяють проводити пошук заборонених предметів в автоматичному режимі в умовах масового проходу людей.

Зростає актуальність використання сканерів біометричних параметрів людини для автоматичного посвідчення особи. Тому що існуючі нині біометричні апарати не адаптовані до застосування в умовах великого потоку людей, сьогодні є єдиний метод, зміст якого полягає в переносі алгоритмів ухвалення рішення на сам пропускний пристрій і системну частину СКУД.

Для роботи з технічними засобами високого рівня необхідний і відповідний персонал, тому що самі передові технологічні рішення будуть неефективні через негативний вплив людського фактора. Чим вище ступінь автоматизації процесів керування доступом, контролю дій персоналу й прогнозування позаштатних ситуацій, тим вище рівень безпеки об'єкта.

Для диференціації процедур контролю різних категорій осіб СКУД повинна мати можливість як об'єднання різних пропускних пристроїв, так і організації на їхній основі твердих режимів контролю персоналу й відвідувачів.

Щоб відповідати сучасним вимогам, СКУД повинна:

- забезпечувати контроль і керування доступом на різних типах контрольно-пропускних пунктів ;
- виключати пронос/ввіз небезпечних (заборонених) предметів (зброї, вибухових, отрутних, радіоактивних речовин і т.д.);
- затримувати потенційних порушників;
- підтримувати різні способи посвідчення минаючих осіб, у тому числі й біометричні;
- мати відкриту для інтеграції програмно-апаратну платформу;
- мати високий ступінь адаптації;
- забезпечувати автоматизацію процесів керування й координацію діяльності служб безпеки об'єкта;
- функціонувати в умовах поразки її компонентів і в інших надзвичайних ситуаціях.

Розподілені СКУД

Раніше СКУД використовувалися тільки для керування доступом у просторі, обмеженому одним будинком. Сучасні системи контролю доступу справляються із цим завданням повною мірою. Однак, у великих розгалужених організацій з'явилася необхідність у розподілених СКУД. До таких організацій можна віднести промислові підприємства; великі торговельні мережі; компанії, що займаються транспортуванням енергоресурсів; підприємства, технологічний процес яких має значний територіальний охват. Тобто головна їхня особливість – значна розподіленість, будь вона територіальна або географічна.

Розподілені СКУД виконують безліч функцій. Серед них основними є ведення централізованої бази даних про співробітників шляхом синхронізації периферичних баз, централізоване адміністрування всіх локальних СКУД, а

також створення й підтримка глобальних рівні доступу й контролю повторного проходу. Крім основних, виділяються деякі додаткові функції СКУД такого типу (наприклад, глобальний контроль переміщення персоналу й облік його робочого часу).

До розподілених СКУД пред'являється цілий ряд вимог, що найчастіше значно відрізняються від вимог до звичайних СКУД. Тут можна відзначити необхідність у децентралізованності системи на рівні не тільки підрозділу, але й окремого контролера; можливість синхронізувати різні дані на рівні контролерів і ПЗ; наявність підтримки різноманітних стандартів передачі даних для маршрутизації трафіку; підтримка різних матеріалів, по яких здійснюється передача інформації; можливість роботи з різнорідними типами ідентифікаторів; контроль стану системи з метою запобігання або швидкого виправлення порушення її цілісності.

Тепер приведемо основні фактори, на які варто звернути увагу при створенні СКУД такого рівня. По-перше, важливо правильно вибрати стандарти й технології. Найбільш правильний варіант – використання міжнародних відритих стандартів і технологій нижнього й верхнього рівнів (LonWorks і XML\SOAP, наприклад). По-друге, створення цілісної мережної інфраструктури, що повинна бути прозора для всіх стандартизованих протоколів. По-третє, повинне використовуватися встаткування, що відповідає відкритим міжнародним стандартам. Воно може вільно інтегруватися з будь-яким устаткуванням сторонніх виробників на апаратному рівні. Якщо врахувати вищесказане, можна створити безпечну й зручну розподілену СКУД, що буде виконувати всі необхідні функції.

Мережні СКУД

Особливістю мережних систем контролю є, те, що встаткування контролю доступу зв'язано комп'ютерною мережею. Це дозволяє моментально одержувати інформацію, обробляти її, управляти режимами доступу, оперативно реагуючи на зміни, що відбуваються. При цьому прийняття рішень про доступ у кожному

окремому випадку й зберігання списків подій здійснюється контролером – виходить, мережна система не має на увазі перманентного підключення до комп'ютера.

В основі організації мережних систем лежить модульний принцип, завдяки чому можна з мінімальними витратами їх розгорнути, збільшити підконтрольну територію або, приміром, удосконалити для виконання більш складних дій.

Мережні системи контролю доступу вирішують широкий спектр завдань. Вони реєструють системні події, дають можливість прямого керування компонентами системи оператором комп'ютера, ведуть базу даних про співробітників (її також можна внести в систему з іншого електронного носія), призначають їм різні права доступу, оформляють пропуску з фото й необхідними даними, автоматизують табельний облік і самостійно визначають компоненти, що входять у систему.

Від того, які контролери й програмне забезпечення встановлені в СКУД, залежать можливості системи. Розглянемо кілька варіантів конфігурацій систем.

Система PERCO-S-600 є оптимальним варіантом для використання в великих офісних приміщеннях і на промислових підприємствах із числом працівників, що не перевищують 2 тисяч. Система має у своєму розпорядженні збалансований функціональний набір, що дозволяє трохи знизити вартість оснащення прохідної.

Якщо потрібно оснастити системою контролю доступу об'єкт із більшою кількістю працівників, улаштувати особливий контроль доступу на деякі об'єкти, організувати відеоідентифікацію, управляти розкиданими компонентами системи з єдиного центра, то підійде система PERCO-SYSTEM-12000. вона може використовуватися в банках і офісах, навчальних закладах і адміністративних установах, на митницях і промислових підприємствах, а також на багатьох інших об'єктах.

Система PERCO-SYS-15000 являє собою наймогутніше рішення, що увібрало в себе всі сучасні наробітки в області мережних систем контролю доступу. Її програмне забезпечення працює по клієнт-серверному принципу, що значно підвищує надійність системи, що управляє базами даних. Система може гнучко настроюватися під масштаби конкретного об'єкта – від невеликого офісу до великого підприємства.

Нові технології в СКУД

Системи контролю й керування доступам не стоять на місці й постійно розвиваються.

Для початку визначимо основні тенденції розвитку цих систем. У їхнє число (тобто в число тенденцій) входять інтелектуалізація й підвищення надійності деяке апаратне складових, зменшення розмірів картрідерів і керуючих пристроїв при рості функціональності, використання нових шляхів захисту й посилення стійкості до інтелектуального злому, а також збільшення значимості програмного забезпечення в процесі керування СКУД. Останній пункт дуже важливий, адже від ступеня розробленості ПЗ, головного посередника між апаратурою й людиною, залежить те, наскільки ефективно буде працювати вся система.

Широке поширення одержали контактні технології, одна з перших серед яких ґрунтувалася на магнітних картах. По краю карти була нанесена магнітна смужка, де був записаний двійковий код. Для ідентифікації потрібно провести карту через картрідер. У цієї технології два основних переваги (низька вартість і можливість перезапису), а от недоліків набагато більше (недостатній рівень безпеки, нетривалий термін служби й карт, схильність їхнім механічним ушкодженням, необхідність у безпосередньому контакті для зчитування, скромна пропускна здатність).

Штрихова технологія заснована на використанні ідентифікатора з кодом, що складається з паралельно розташованих ліній різної товщини. Технологія знайшла собі широке застосування завдяки низкою вартості, стійкості до

механічних і електромагнітних впливів і донині активно використовується на складах і в магазинах для маркування товару. Поширенню штрих-кодів у СКУД перешкодили, головним чином, легкість створення підробки й відсутність можливості перезапису.

Wiegand-карти схожі на магнітні, але мають смужки з феромагнітних сплавів. Картрідери таких карт позбавлені магнітної головки, зчитування здійснюється через електромагнітне поле при проведенні карти через картрідер. До плюсів магнітних карт додалися перешкодозахищеність, гідний рівень безпеки й довговічність. Єдиним мінусом технології Виганда є необхідність у контакті картрідера й карти.

Наступним етапом стала поява Proximity-технології, головне достоїнство якої полягає в тому, що ідентифікація відбувається дистанційно. Технологія увібрала в себе всі можливі переваги аж до високої пропускної здатності, однак prox-карти не мають можливості перезаписуватися. Цей недолік був усунутий в Smart-картах, причому зробити несанкціонований перезапис на них украй важко. Єдиним мінусом стало підвищення вартості карт.

Найбільш прогресивні біометричні технології ідентифікації, вони відкривають нові обрії в підвищенні ефективності СКУД, адже підробити генетичні, фізіологічні й поведінкові особливості неможливо. Майбутнє систем контролю саме в біометрії.

Аналіз продуктів представлених на ринку СКУД

Сучасні СКУД є складними інтелектуальними продуктами. У ПЗ для СКУД входять наступні підпрограми:

- конфігуратор для настроювання програми під існуючу на об'єкті апаратну частину;
- модуль бюро пропусків для ведення бази даних персоналу;
- модуль робочого місця охоронця;
- модуль обліку робочого часу;

- генератор звітів, що дозволяє аналізувати події з бази даних і робити вибірку по декількох параметрах.

Важливим елементом СКУД є картрідер безконтактних ідентифікаторів. Розглянемо кілька систем.

APOLLO. Залежно від масштабу системи використовуються різні види контролерів – від найпростіших однодверних до мережних багатодверних. Контролер AAN-100 управляє картрідерами в кількості від 16 до 96 і тримає в пам'яті 300 тисяч карт. Номенклатура периферійних модулів достатня широка. Використовуються картрідерери HID і Motorola. Устаткування має сертифікати ДСТ. ПЗ мережне. Базова версія ПЗ APACS Standard підтримує графічні плани об'єктів, блок редагування й печатки карт, обробляє й зберігає всі повідомлення системи.

COTAG. Вироблена компанією Cotag система «Гранта» має в основі контролери на 8 (модель 4101, модульний) або 4 (модель 4010, економічний) картрідерів. Одна з головних переваг «Гранта» – картрідери зі збільшеною дальністю дії. Для автостоянок пропонуються ідентифікатори, що кріпляться на днище автомобілів. Система добре захищена від злому, ПЗ має надійний багаторівневий парольний захист. Модульний принцип побудови «Гранти» дозволяє сконфігурувати систему будь-якої складності. Устаткування має сертифікати ДСТ. ПЗ – потужне мережне багатофункціональне (системний менеджер, програма оператора, тривожна графіка, монітор подій і ін.).

KANTECH. Основа системи – контролер KT-200 на 2 картрідера. ПЗ залежить від запитів замовника. Найбільш потужне ПЗ – KL-8000 – працює з 8-ю мережними комп'ютерами й обслуговує до 32 тисяч користувачів. Kantech підтримує картрідери різних типів ідентифікаторів, мають систему ідентифікації автотранспорту ShadowProx VID. СКУД Kantech має сертифікат ДСТ.

KERI SYSTEMS. Основа системи – контролер на 1 або 2 двері PXL-250 Tiger. У мережу поєднуються до 128 контролерів з обслуговуванням до 256

дверей. ПЗ Doors 32 дозволяє конфігурувати систему. Keri Systems надає 3 -літню гарантію на своє встаткування.

NORTHEN COMPUTERS. В основі системи – контролери серії N-1000 на 1, 2, 3 або 4 двері. Найпоширеніше ПЗ – Win-Pak 1.15. воно русифіковано, дозволяє друкувати карти-перепустки, управляти вилученими об'єктами, системою телеспостереження та ін. новинка Win-Pak Pro інтегрована з Windows, підтримує до 256 одночасних з'єднань для кожної станції. СКУД NC краще всіх інтегруються із системою сигналізації Ademco. Таким чином, в відповідності до своїх можливостей й потреб, кожний споживач може вибрати оптимальну для себе СКУД.

Огляд програмного забезпечення для систем контролю доступу

У переважній більшості випадків процес експлуатації СКУД сполучений з використанням спеціально розробленого для таких систем програмного забезпечення (ПЗ). Залежно від конкретного місця застосування й функцій, необхідних від СКУД, ПЗ має різні характеристики. У кожному разі, програмний пакет керування системою контролю є її координуючим центром, без якого неможлива повноцінна її робота. Розглянемо основні функції й особливості ПЗ, під керуванням яких працюють СКУД різного масштабу.

Так, ПЗ для СКУД підтримують черговий режим – режим, призначений для керування роботою системою в інтерактивному режимі. Тому в ньому необхідний графічний інтерфейс, що відображає всі потрібні дані в зручній для сприйняття формі – як текстовій, так і графічній. При цьому інтерфейс повинен бути доступний для розуміння й освоєння людиною (оператором), що не має спеціалізованої підготовки. Сучасні програмні комплекси (ПК) дозволяють легко одержати детальну інформацію про будь-який об'єкт, що входить у систему, що спрощує контроль загального стану.

Іншою функцією ПЗ систем контролю є ведення й підтримка бази даних карт. Вона служить для перегляду, внесення змін, додавання й видалення відомостей про співробітників об'єкта в базі даних. Найчастіше ПК мають у

своєму активі передвстановленний редактор макетів карт, що дозволяє без проблем створювати макети й роздруковувати їх на пластикових картах.

ПЗ здатно генерувати звіти – автоматично формувати список подій системи для швидкого перегляду на моніторі або печаті. При цьому повідомлення списку діляться на групи:, що сповіщають про події доступу, про дії оператора, про спрацьовування тих або інших датчиків, про стан зв'язку між компонентами, що входять у систему.

Облік робочого часу також посильний сучасному ПЗ: воно може співвідносити графіки робіт з реальною діяльністю співробітника (наприклад, зіставляти потрібний час виходу на робоче місце із сьогоденням).

По масштабах СКУД виділяється кілька основних їхніх видів, кожному з яких відповідає свої ПК. Для малих об'єктів і локальних СКУД всі елементи комплексу встановлюються й запускаються на одному комп'ютері, до якого підключається й все встаткування системи. Централізована система з віддаленим керуванням має на увазі функціонування всіх модулів на одній ЕОМ з можливістю запуску консолі керування з інших комп'ютерів мережі. У великій розподіленій системі (актуальна при великій території) керування базою даних і функціонування ядра відбувається на центральному сервері, а драйвери встаткування й логіки «розкидані» по мережі. При цьому запуск керуючої консолі можливий на будь-якому комп'ютері.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в

чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64 -bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи контролю та управління доступом на підприємстві.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі.

Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис виконуючих пристроїв

Електронні системи контролю доступу забезпечують можливість доступу певних осіб у певні приміщення й обмежують проникнення осіб, що не мають права доступу на ту або іншу територію.

Найпростіші електронні системи доступу – кодонаборні панелі й автономні картрідери карток. Складні системи, що включають у себе комп'ютери, мають більший набір функцій. Їх ПЗ дозволяє одержати різні види звітів про поточні події.

Виконавчі пристрої систем контролю доступу – це різні електрозамки, турнікети, автоматичні двері тощо. Якщо об'єктом доступу є автомобіль, то виконавчі механізми в цьому випадку – шлагбауми й автоматичні приводи воріт.

Всі СКУД будуються на базі ідентифікації чого-небудь. Ідентифікація ґрунтується на самих різних фізичних принципах.

Кодові клавіатури – найпростіші пристрої доступу, що ідентифікують код, набраний користувачем. Сполучені картрідери («карта + код») забезпечують захист у випадку втрати карти, тому що для доступу, крім пред'явлення карти, необхідно набрати код.

Магнітні карти використовуються багатьма системами. Картка проводиться через картрідер, що не завжди зручно. Знаходження карти недалеко від сильних магнітних джерел, а також її забруднення виводять карту з ладу.

Карта Виганда (Wigand) – пластикова картка із запресованими в неї відрізками дроту зі спеціального магнітного сплаву. У кожної карти є

індивідуальний код, що зчитується при проведенні її повз магнітну головку. Ці карти зносостійкі, надійно захищені від підробки, але дорожче магнітних.

Безконтактні (Proximity) карти – зчитуються шляхом піднесення до картрідера, поза залежністю від положення стосовно нього. Ідентифікується на відстані 5-15 см, деякі моделі працюють на відстані 1-2 метри. Картка не зношується, не боїться впливів зовнішнього середовища. На неї можуть наноситися напису й фотографії. Низькочастотна (125 кГц) Proximity-технологія прийнята за промисловий стандарт, оскільки Proximity-карта доступу досить надійна, порівняно недорога, а зчитування її цифрового коду (від 24 до 96 біт) зі швидкістю 2 кбайт відбувається вже на відстані 15 см. По цій же технології виготовляються ідентифікатори у вигляді брелоків, браслетів і ін.

Штрих-код наноситься на паперову або пластикову основу. Картка проводиться через проріз картрідера. Цю карту легко підробити. Якщо штрих-код видн тільки в ІЧ-діапазоні, то таку карту підробити або скопіювати складніше.

Радіоканал використовується для передачі коду картрідеру. Як ідентифікатор виступають радіобрелок або невеликий передавач. Високий ступінь захищеності мають системи з «блукаючим кодом». Мають великий радіус дії. Використовуються на воротах і шлагбаумах.

ІЧ-брелки передають код а ІЧ-діапазоні, добре захищені від перехоплення.

Смарт-карти мають вбудований процесор і контактні площадки для живлення й обміну зі картрідером. Високий ступінь захищеності. Безконтактні смарт-карти володіють крім комірок пам'яті ще й програмним забезпеченням, що перешкоджає розшифровці коду, переданого картрідеру картою, що значно підвищує безпеку. Смарт-карти з великим обсягом пам'яті мають у своєму розпорядженні мікропроцесор, що управляє функціями й розподілом пам'яті.

Електронні ключі – пристрої, що містять код і передають його картрідеру. Нагадують батарейку «таблетка». Зносостійкі, міцні, не бояться впливів зовнішнього середовища.

Біометричні системи розпізнавання засновані на аналізі індивідуальних біометричних характеристик людини (відбитки пальців, тембр голосу, геометрія кисті руки).

Широкий потенціал смарт-карт можливо розкрити, використовуючи їх у зв'язуванні з біометричною технологією. Відразу ж можна вирішити проблему дорожнечі й громіздкості чисто біометричних ідентифікаторів, розмістивши на смарт-карті замість коду, приміром, відбиток пальця. У цьому випадку відпадає необхідність у безпосередньому контакті зі картрідером. Біометричні ідентифікатори в найближчому майбутньому одержать широке поширення, тому що смарт-карти вже зараз недорогі, ціни на біометричні датчики знижуються, а потреба підвищення рівня безпеки назріла давно.

Існує кілька прийнятих конфігурацій підключення систем контролю доступу. Звичайно сигнали до дверей контрольованого об'єкта направляються від однієї панелі; картрідера із дверима об'єднані через протокол Виганда, пульт же може з'єднуватися з базовим комп'ютером по RS-485-інтерфейсу. Інші датчики й кнопки з'єднуються з панеллю прямо. При цьому кабельна обв'язка становить значиму частину вартості установки всієї системи. Заглядаючи в перспективу розвитку конфігурацій підключення, можна відзначити, що в майбутньому кожна система буде складатися із дверних груп, керованих окремими контролерами. Вони, у свою чергу, будуть зв'язані один з одним через RS -485 і через TCP/IP – з ядром системи.

Опис електронних ключів і картридерів

Електронні замки представляються ще занадто новим віянням, щоб бути повністю зрозумілими споживачеві. Розглянемо пристрій електронного замка, щоб вникнути в те, як він функціонує й виявити його переваги й недоліки.

Електронний замок складається із трьох основних частин: привода, блоку керування й ключевини (блоку ідентифікації). Тут варто відзначити, що в добротного замка ці частини повинні існувати окремо, вони не повинні інтегруватися друг у друга. Існує кілька видів замків по типу привода, що

управляє засувами: комбінований електромеханічний замок (звичайний замок з електромагнітним блокуванням); комбінований електромеханічний замок, оснащений моторним приводом; електромагнітна засувка; моторний замок (у випадку грамотної реалізації дуже ефективно протистоїть злому); моторний замок із блокуванням засувів у відповідній частині (найбільш надійні, розкрити при добре реалізованих інших частинах замка попросту неможливо).

Тепер поговоримо про блоки керування. Вони служать для ідентифікації ключа й керування приводом замка. Якщо блок керування й кнопку відкриття двері можна встановити на значній відстані від інших частин замка (наприклад, взагалі не на двері); якщо він має діючий захист від перепадів напруги й працює від резервного джерела живлення, якщо всі провода, що йдуть від його до мотора, мають однаковий колір і товщину і якщо він спроектований таким чином, що має дублюючі системи й при ушкодженні мікропроцесора не запускає двигун, то перед нами – просто ідеальний блок керування.

Переходимо до розгляду «ключевин», або блоків ідентифікації. Їх існує безліч, важливо зробити правильний вибір (втім, як і з іншими складовими замка). Тут головне не перестаратися. Тобто в жодному разі не слід установлювати на двері квартири дактилоскопічні панелі або картрідери карт якого-небудь виду – відкритий тип ідентифікатора сильно залучає до себе увагу деяких особистостей (точніше не до себе, а до того, що може перебувати за дверима з таким ідентифікатором). Отже, ідентифікатори підрозділяються на контактні (звичайний або складальний резісторный ключ, карта з електронним кодом, кодова електронна «таблетка», смарт-карта), псевдобезконтактні (кодова клавіатура, магнітний ключ, магнітна карта, дактилоскопічна панель) і безконтактні (найкраще вибрати саме їх – у цьому випадку відкривається набагато менше можливість для злому).

Моторні замки – технічні пристрої, у яких керування запірним механізмом здійснюється за допомогою електродвигуна. Їхня висока ціна виправдана надійністю пристрою й технічною складністю. Зараз попитом користуються

замки компаній ABLOY (Фінляндія), effeff (Німеччина); із числа новинок необхідно відзначити замок e-VOLUTION, що випускається фірмою CISA (Італія). Особливості даного замка: кілька режимів роботи, простота зі зміною, сигнал стану замка, наявність у комплекті акумулятора. Призначено цю модель для установки в потужні металеві двері.

У замках соленоїдного типу керування потужними запірними ригелями здійснюється за допомогою електромагнітних котушок. Від набору функціональних можливостей залежить ціна на ці вироби. На нашому ринку представлені замки фірм TECNAX (Швейцарія), effeff, YALE CORNI (Італія), PONGEE (Тайвань). Замки соленоїдного типу встановлюються в стандартні, алюмінієві або пластикові двері. Датчик, вбудований у замок даного типу, контролює двері й передають сигнал про її стан на керуючий контролер.

Електромагнітні замки вимагають постійного безперебійного електроживлення. Серед них є й врізні замки, використання яких зменшує площу дверного прорізу й робить екстер'єр дверей більше естетичним.

Електромеханічні замки «взводного» типу відрізняються тим, що тут не потрібна постійна подача електроенергії. Ці замки значно дешевше моторних і соленоїдних. Енергія, використовувана для роботи «взводних» замків, береться від пружини, зведеної при закритих дверях і разблокуємої при подачі керуючого електричного сигналу. При відсутності електроенергії двері залишаються закритими й можуть бути відкриті механічним ключем. Накладні й урізні замки «взводного» типу виробляються компаніями CISA і ISEO (Італія); менш дорогі накладні замки представлені фірмами YUS, MING YANG (Тайвань) і COMMAX (Корея).

Електромеханічні засувки – варіант найбільш простого рішення завдання дистанційного відкривання двері з механічним замком. Лідером виробництва засувок є компанія effeff, а серед інших виробників цих виробів можна відзначити фірми NUOVA FEB (Італія), Openers&Closers (Іспанія), COMMAX (Корея) і ін.

Вибираючи замок, необхідно знати, що у вуличних умовах краще використовувати електромеханічні й електромагнітні замки (при цьому робочу частину потрібно захистити від влучення опадів); електромеханічні засувки, що відрізняються здатністю довгострокового підключення до джерела живлення, установлюються на аварійних виходах і шлюзових проходах; до хитних дверей підійдуть замок, фиксуємий у середньому стані (тобто, або спеціальний замок для хитних дверей, або замок з датчиком положення двері).

Електронний замок із ключами iButton

Замок спроектовано для індивідуальних використань і має гранично просту конструкцію. На вхідних дверях зовні розташовується тільки панелька для iButton і светлодіод відкривання дверей. Відкривання дверей зсередини здійснюється за допомогою кнопки.

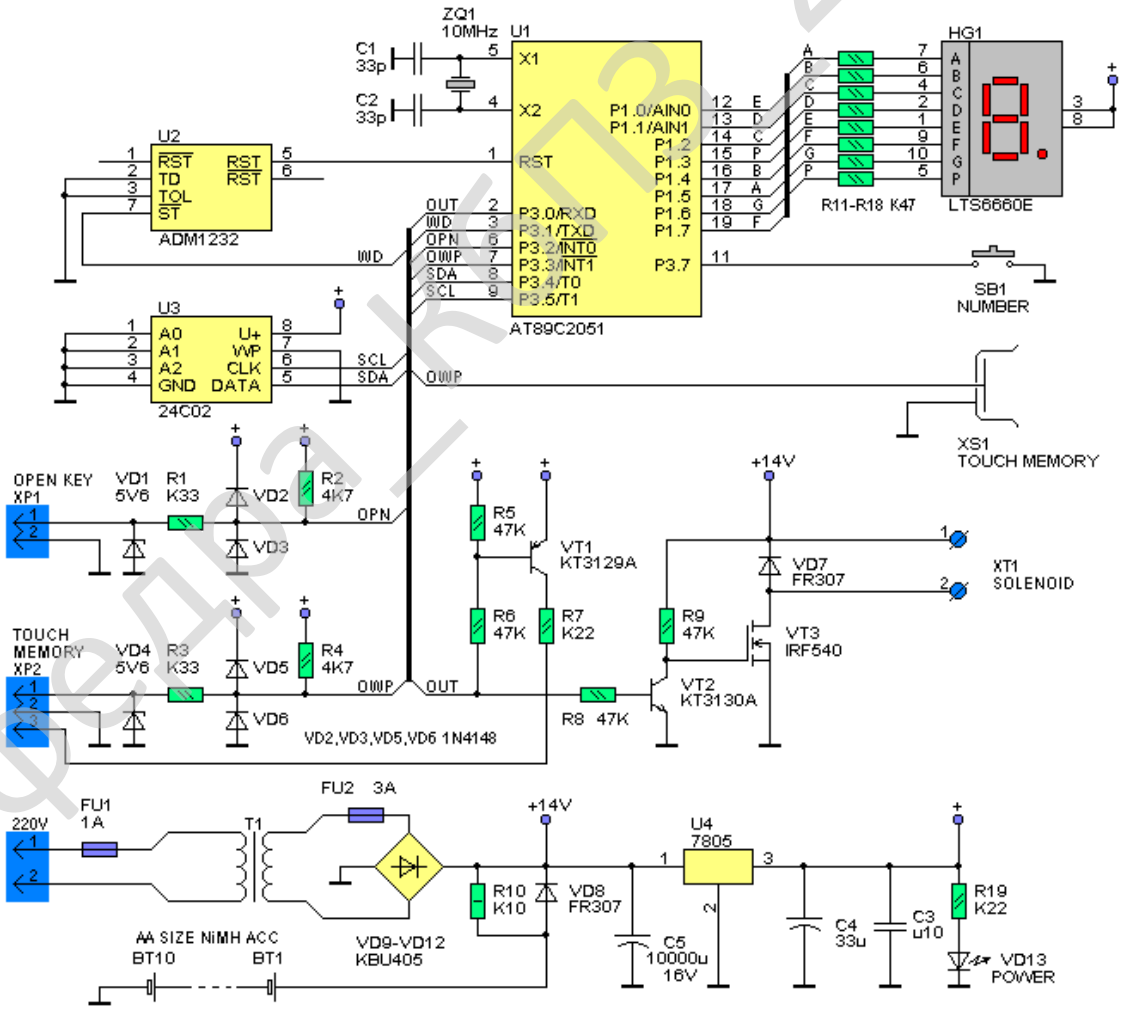

Рисунок 3.1 – Принципова схема замка

Як виконавчий механізм використовується стандартна засувка з електромагнітом, що розрахований на напругу 12В. Коди ключів зберігаються в енергонезалежній пам'яті й можуть стиратися й додаватися користувачем. Для захисту від несанкціонованого перепрограмування замка використовується майстер-ключ. Усього до пам’яті можна записати 9 ключів. Ця кількість продиктована можливостями 1-розрядного індикатора номера програмувального ключа. Якщо задіяти ще й букви, можна збільшити сумарну кількість ключів до 15. Це робиться шляхом заміни значення константи MAXK у програмі. Таким же способом можна й зменшити максимальну кількість ключів.

Принципова схема замка показана на рисунку 3.1. Основою конструкції є мікроконтролер U1 типу AT89C2051 фірми Atmel. До порту P1 підключений 7-сегментний індикатор, що використовується при програмуванні ключів. Для цих же цілей призначена й кнопка SB1, підключена до порту P3.7. Зберігання серійних номерів ключів здійснюється в мікросхемі EEPROM U3 типи 24C02, підключеної до портів P3.4 (SDA) і P3.5 (SCL). Зовнішня панелька для iButton підключається до порту P3.3 через роз’єм XP2 і елементи захисту VD4, R3, VD5 і VD6. резистор, що підтягує, R4 обраний відповідно до специфікації однопровідної шини. Паралельно зовнішній панельці підключена ще й внутрішня панелька XS1, що використовується для програмування ключів. Кнопка відкривання двері підключена до порту P3.2 через роз’єм XP1 і такі ж елементи захисту, як і для iButton. Виконавчим пристроєм замка є електромагніт, підключений через термінал XT1. Електомагнітом управляє ключ VT3, у якості якого використовується потужний МОП-транзистор типу IRF540. Діод VD7 захищає від викидів самоіндукції. Ключем VT3 управляє транзистор VT2, що інвертує сигнал, який надходить із порту P3.0 і забезпечує керуючі рівні 0/12В на затворі VT3. Інверсія потрібна для того, щоб виконавчий пристрій не спрацьовувало під час скидання мікроконтролера, коли на порту присутній рівень логічної одиниці. 12-вольтові керуючі рівні дозволили застосувати звичайний МОП-транзистор замість більш дефіцитного низькопорогового (logick level). Для

індикації відкриття замка використовується світлодіод, який управляється тим же портом, що й електромагніт, але через транзисторний ключ VT1. Світлодіод підключається через той же роз'єм, що й iButton. Оскільки пристрій повинен працювати цілодобово без обслуговування, для підвищення надійності встановлений супервізор U2 типу ADM1232. Він має вбудований сторожовий таймер і монітор живлення. На порту P3.1 мікроконтролер формує періодичні імпульси для скидання сторожового таймера.

Живлення пристрою здійснюється від вбудованого блока живлення, що містить трансформатор T1, випрямний міст VD 9-VD12 і інтегральний стабілізатор U4. Як резервне джерело живлення використовується батарея BT 1-BT10 з 10-ти NiMH-акумуляторів типорозміру AA ємністю 800мА/Ч. При живленні пристрою від мережі батарея акумуляторів заряджається через резистор R10 струмом приблизно 20мА, що становить 0.025С. Режим зарядки малим струмом називають краплинним (trickle charge). У такому режимі акумулятори можуть перебувати як завгодно довго, контроль кінця процесу зарядки не потрібен. Коли акумулятори виявляються повністю зарядженими, енергія, що забирається ними від джерела живлення, перетворюється в тепло. Але оскільки струм зарядки дуже маленький, виділюване тепло розсіюється в навколишній простір без скільки-небудь помітного збільшення температури акумуляторів.

Конструктивно пристрій виконаний у корпусі розміром 150x100x60мм. Більшість елементів, включаючи трансформатор живлення, змонтовано на друкованій платі. Акумулятори розміщаються в стандартних пластмасових тримачах, які закріплені усередині корпуса поруч із платою. У принципі, можна використовувати й інші типи акумуляторів, наприклад 12-вольтову кислотну необслуговувану батарею, що застосовується в охоронних системах. Для підключення виконавчого пристрою на платі є термінали типу ТВ-2, всі інші зовнішні ланцюги підключаються через малогабаритні роз'єми із кроком контактів 2.54мм. Роз'єми розташовані на друкованій платі й зовні корпуса недоступні. Провода виходять із корпуса через гумові ущільнювачі. Оскільки

індикатор HG1, кнопка SB1 і панелька для iButton XS1 використовуються тільки під час програмування, вони розміщені на платі всередині пристрою. Це спрощує конструкцію корпуса й робить його більш захищеним від зовнішніх впливів. На бічній панелі корпуса розміщений тільки світлодіод індикації включення VD13.

При відкриванні дверей на електромагніт подається імпульс тривалістю 3 секунди. Логіка роботи пристрою така, що якщо кнопку відкривання двері втримувати, то весь цей час електромагніт буде під напругою й, відповідно, двері буде відкрито.

Замок може мати максимум 9 ключів, плюс один майстер-ключ. Коди ключів заносяться в енергонезалежну пам'ять під номерами від 1 до 9. Код майстер-ключа занесений у ПЗП мікроконтролера й не може бути змінений. Програмування нових ключів або стирання старих може бути зроблено тільки при наявності майстер-ключа. Як і інші ключі, майстер -ключ може використовуватися для відкривання замка.

Для програмування нового ключа потрібно проробити наступні дії:

1. Нажати кнопку програмування.

2. На індикаторі з'явиться буква «Р», що означає вхід у режим програмування.

3. Торкнутися майстер-ключем панельки.

4. На індикаторі з'явиться цифра «1», що позначає номер програмувального ключа.

5. Кнопкою вибрати потрібний номер.

6. Торкнутися будь-яким ключем панельки.

7. Цифра на індикаторі почне мигати, що говорить про готовність до програмування.

8. Торкнутися панельки тим ключем, код якого потрібно занести до пам’яті.

9. У випадку успішного програмування цифра на індикаторі перестане мигати й почне горіти постійно.

10. Для виходів з режиму програмування потрібно просто почекати 5 секунд, після чого індикатор згасне.

3.2 Розробка структурної схеми

Для перевірки коректності реалізації даної задачі було виконано багато розрахунків та експериментальних матеріалів. Цьому питанню приділялась особлива увага тому, що помилка при розрахунку привела б до ряду негативних наслідків. Відлагодження та перевірка, що підтверджує вірність програмних рішень відбувалась за декількаами етапами:

– математична перевірка окремих модулів;

– математична перевірка всієї системи (з допомогою математичної логіки будується логічна схема всієї системи);

– практична перевірка підпрограм (перевіряється процедурна частина кожної підпрограми окремо);

– практична перевірка всієї системи у дії (перевіряється ситема в цілому за допомогою вводу різних даних у програму, потім на виході з програми перевіряємо отриману інформацію з очікуваною).

Для підтвердження правильності розрахунку програми були використані експериментальні дані різних форматів, були проведені консультації з даного питання зі спеціалістами.

Простота мови проектування та маніпулювання даними, зручність спілкування користувача з системою до мінімума вивчення цієї програми. Користувач програми – це людина, яка повинна володіти азами програмування. При написанні програми я намагалася, щоб програма відповідала наступним параметрам:

– Швидкодія. Програма працює постійно з великою кількістю запитів від катрридерів.

– Захист. Захищений канал передачі пакетів інформації.

– Система, що написана може встановлюватись на будь-якому персональному комп’ютері – використовувати відносно швидкі алгоритми захисту зв'язку.

– Можливість зручно і швидко формувати приклади і теорію для користувача.

– Можливість звертання до системних ресурсів. Користувача системи цікавить її інформаційний та сенсовий зміст. Подробиці організації фізичного зберігання даних його не цікавлять.

Перш за все перед розробкою системи слід одержати уявлення на слідуючи моменти:

– на які частини можна розбити систему;
– одержати уявлення про кожну частину (фрагмент);
– які процеси передачі і обробки даних знаходяться в кожному фрагменті;
– на якому обладнанні планується реалізувати систему;
– технологія функціюваня системи;
– чи необхідна адаптація і настройка системи при змінах деяких умов.

Для розробки програми були попередньо розроблені структурна схема роботи системи, структурна схема взаємодії з картрідерами, функціональна схема роботи системи, діаграма процесів, а також блок-схеми алгоритму програми, розглянемо їх детально.

Розглянемо структурну схему роботи системи, що зображена на рисунку 3.2. У схемі можна чітко побачити два вхідних потоки даних, а саме дані, що надходять із периферії (дані про стан дверних прорізів) і дані оператора.

Розроблена сучасна система контролю й управління доступом до примішень підприємства (СКУД) базується на апаратурі фірми AS-Scan. Система розподілена на мережні вузли, які з’єднуються з базовою станцією, що називається «сервер керування й зв'язку» і із системами архівації, системами контролю доступу.

Рисунок 3.2 – Структурна схема роботи системи

У розробленій системі застосовувалися мережні вузли засновані на зчитували магнітних карт серії Camp Tek MR, застосовувалися наступні моделі: 834 RS/KB, 834 USB, 836 USB, 836 RS/KB (рисунок 3.3).

Рисунок 3.3 – Зчитувач магнітних карт Camp Tek MR серії 843 і 836

Застосування встаткування фірми AS-Scan дозволяє встановлювати сервер контролю й керування на персональних комп'ютерах малої потужності. Так як встаткування бере на себе частину ресурсномістких операцій, прискорюючи процес обробки сигналів з перифірійного встаткування. При застосуванні даних пристроїв автор диплома враховував ці особливості.

Розроблена програма розбита на модульну структуру дозволяючи швидко змінювати необхідні параметри програми без корінної зміни структури.

Розглянемо основні модулі сервера контролю й керування, а також підсистеми архівації даних і контролю доступу.

Сервер контролю й керування:

1. Інтерфейс користувача – основою сервера контролю й керування є розроблене програмне забезпечення яке видається операторові сервера у вигляді інтерфейсу в якому відображаються всі операції, які відбуваються.

2. Модуль взаємодії з операційною системою – модуль контролю системних повідомлень від операційної системи. Так як щомісяця приходять все нові відновлення операційної системи, необхідно убезпечити систему від можливих системних збоїв, помилок ОС, вірусних атак. Цей модуль дозволяє аналізувати зміни в роботі ОС і провести детальний аналіз при виникненні проблем.

3. Модуль аналізу даних – модуль аналізу вхідних даних зі зчи тувачів магнітних карт. Через можливе значно віддалення зчитувачів друг від друга, а також можливості злому системи необхідно убезпечити сервер аналізом переданих даних.

4. Модуль перевірки лінії зв'язку з об'єктами – через значно віддалення також необхідно постійно перевіряти зв'язок між сервером і зчитувачами магнітних карт і при необхідності робити відповідні контрольні дії.

5. Модуль захисту ПЗ – модуль перевірки роботи програми. Для мінімізації й виключення можливості злому сервера був написаний модуль контролю роботи програми який проводить перевірку записів у реєстрі, перевірку

роботи розробленого програмного забезпечения з виділеною пам'яттю, операції читання запису на диск.

6. Модуль виведення статистики роботи – ведення статистики доступу до приміщень, з точним указанням дати й часу доступу. Для чіткого розуміння подій, які відбуваються в підприємстві необхідно мати доступ до зроблених дій.

7. Модуль кодування – модуль настроювання параметрів кодування. Тонке настроювання методу кодування ДСТУ 28147:2009 (довгі ключа, особливості предачі й.т.їнше.).

8. Модуль налагодження ПЗ – модуль загального настроювання розробленої програми.

Система архівації даних:

1. База даних контрольних підсистем – підсистема зберігання резервних копій, статистики роботи й т.д.

2. Таблиці підприємницької справи – таблиці ведення обліку роботи підприємства.

3. Таблиці доступу – таблиці пропускної системи доступу до приміщень підприємства.

4. Таблиці надзвичайних випадків – таблиці ведення статистики надзвичайних ситуацій для аналізу дій, зроблених персоналом.

Система контролю доступу:

1. Охорона правопорядку (система безпеки підприємства) – зв'язок з охороною підприємства при надзвичайних подіях.

2. Автоматизована телефонна система підприємства – організація зв'язку по підприємства, довідник підприємства.

3. Протипожежний контроль – екстрений виклик пожежних і служби газу (протипожежна й газова тривожна кнопка).

4. Інші служби – служби паркування автомобілів, доставки товару.

Коли людина проводить магнітною карткою через щілину картрідера відбувається зчитування 2 магнітних смужок – вихідна інформація з визначаємої карти.

Далі картрідер робить апаратне кодування даних з магнітної карти алгоритмом ДСТУ 28147:2009 і передачу інформації на сервер контролю й керування в зашифрованому виді.

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема роботи системи на основі розподілених картрідерів.

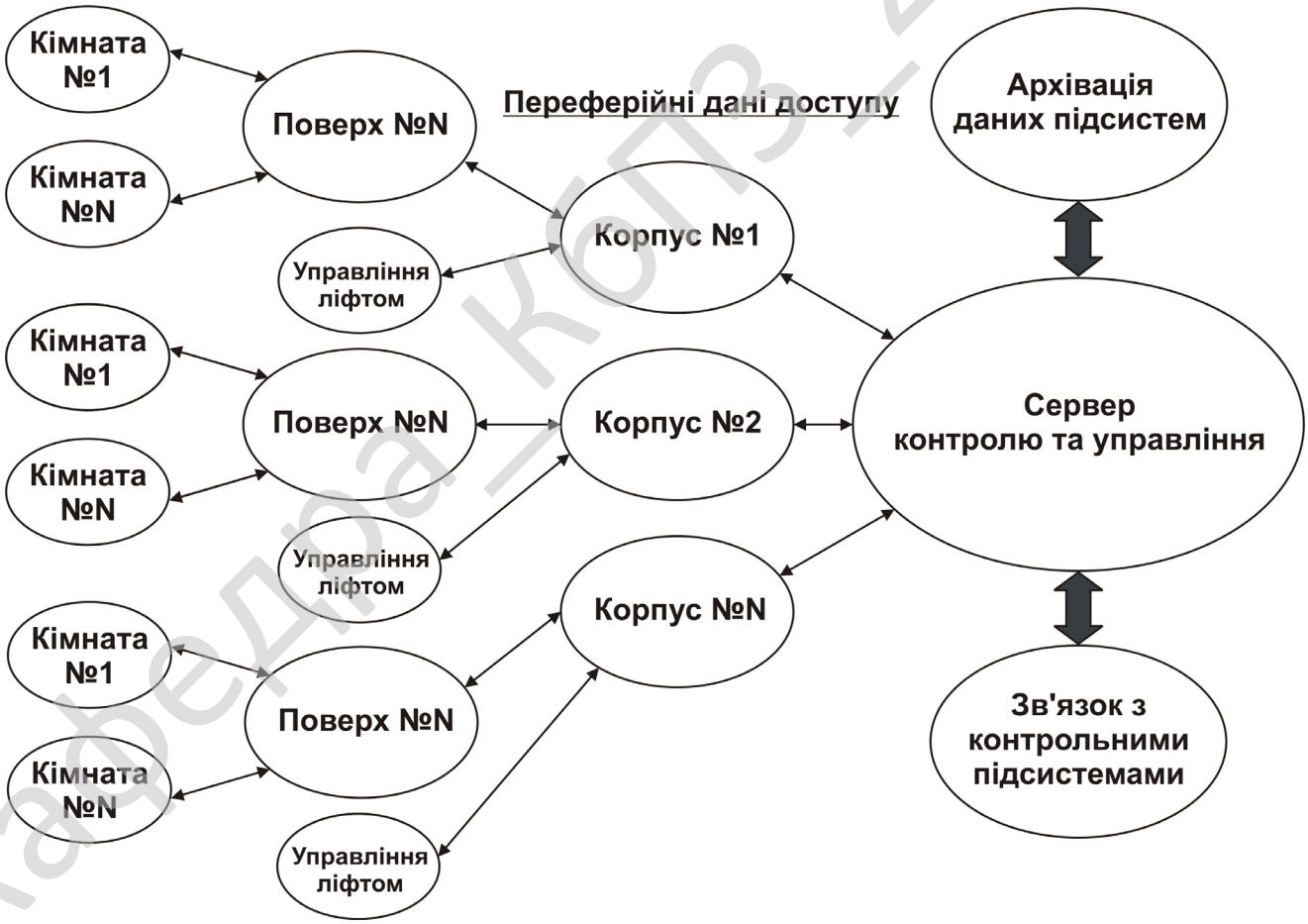

Рисунок 3.4 – Функціональна схема роботи системи

Розроблена програма встановлена на сервері контролю й керування через апаратні вузли фірми AS-Scan, яка робить контроль доступу як до помешкань, поверхів підприємства так і ліфтів (моніторинг, часткове керування).

При розробці диплома й зокрема функціональної схеми велика увага приділялася побудові повнофункціональної схеми деревоподібної структури для забезпечення надійності роботи при обриві лінії зв'язку. За допомогою апаратного забезпечення AS-Scan автор розробив універсальну програму яка дозволяє не тільки контролювати кімнати, поверхи, ліфти але й забезпечити захищений канал передачі даних використовуючи алгоритм кодування ДСТУ 28147:2009.

3.4 Розробка діаграми процесів

Одним з важливих критеріїв при розробці будь-якого програмного забезпечення це грамотна розробка структури роботи системи потоків і процесів рисунок 3.5.

Як видно з рисунка починається й закінчується програма в першому блоці, який є основною точкою звіту в діаграмі (основний блок ПЗ початок/кінець), при переміщенні по стрілках можна побачити загальну схему взаємодії блоків і їхнього входження друг у друга.

Основний блок програми взаємодіє з оброблювачем помилок, модулем зашиті програми, інтерфейсом програми й модулем настроювання ПЗ.

Через інтерфейс програми й оброблювач помилок відбувається взаємодія із блоками перевірки ступеня працездатності й захищеності програми.

Також блок інтерфейс програми через блок аналізу даних взаємодіє із системою архівації даних і службами підприємства.

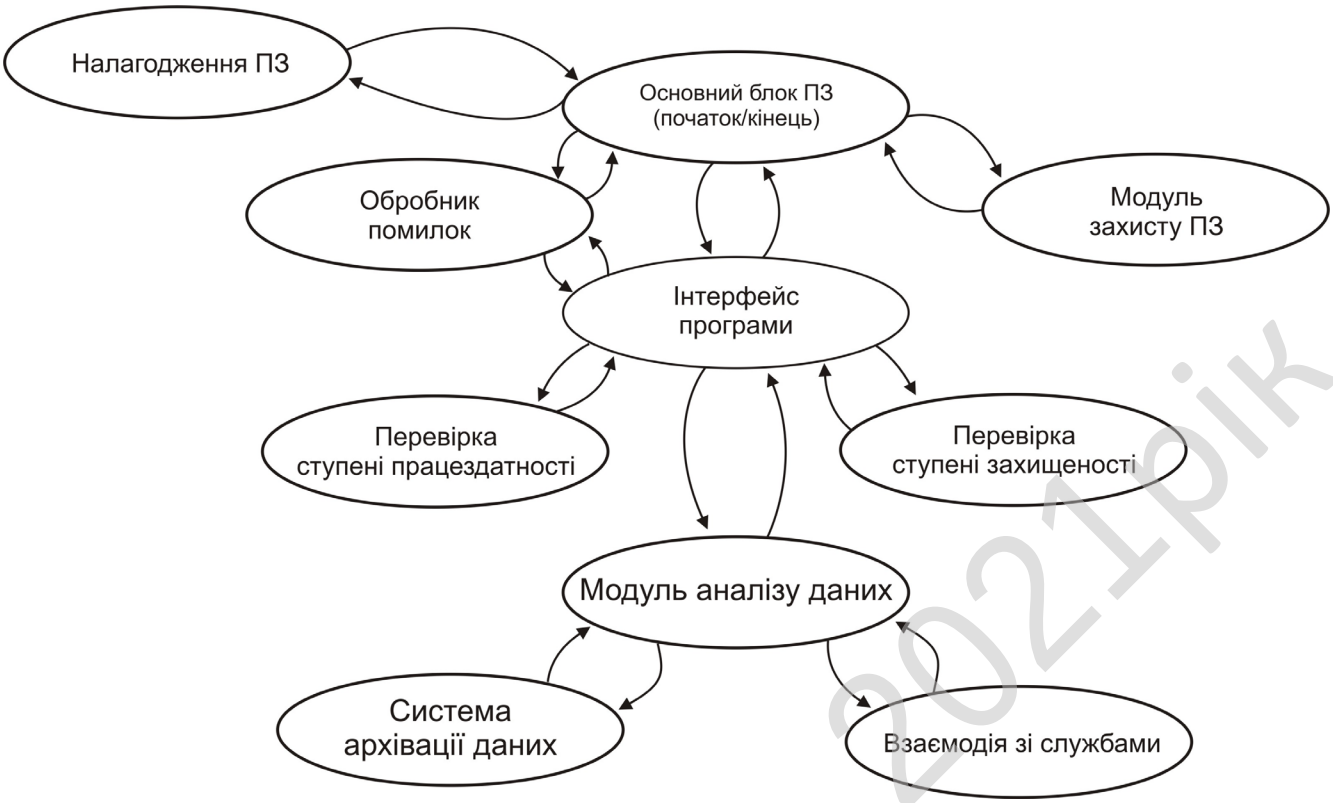

Рисунок 3.5 – Діаграма процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми. Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єкно-модульно-орієнтована, проект, що розробляється, вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулям захисту та опитуванням периферійної апаратури. Від організації програми залежіть багато факторів та в кінцевому результаті веде до створення повноцінного коду програмного продукту з розробленою ціновою політикою і системою знижок покупцям програмної продукції.

На рисунку 4.1 зображена основна блок-схема програми. При детальному розгляді крім стандартних початкових блоків та кінцевих та блоків аварійного завершення роботи, програма розбита на дещо важливих блоків таких як:

– Підключення бібліотек взаємодії з апаратною частиною – бібліотеки необхідні для взаємодії з периферійним апаратним забезпеченням фірми **AS-Scan** (надається й обновляється фірмою);

– Тестування доступу апаратної частини – перевірка доступу до апаратної частини, периферійних пристроїв;

– Перехід у робочий режим – виконання початкового коду роботи програми.

– Виведення головного вікна програми;

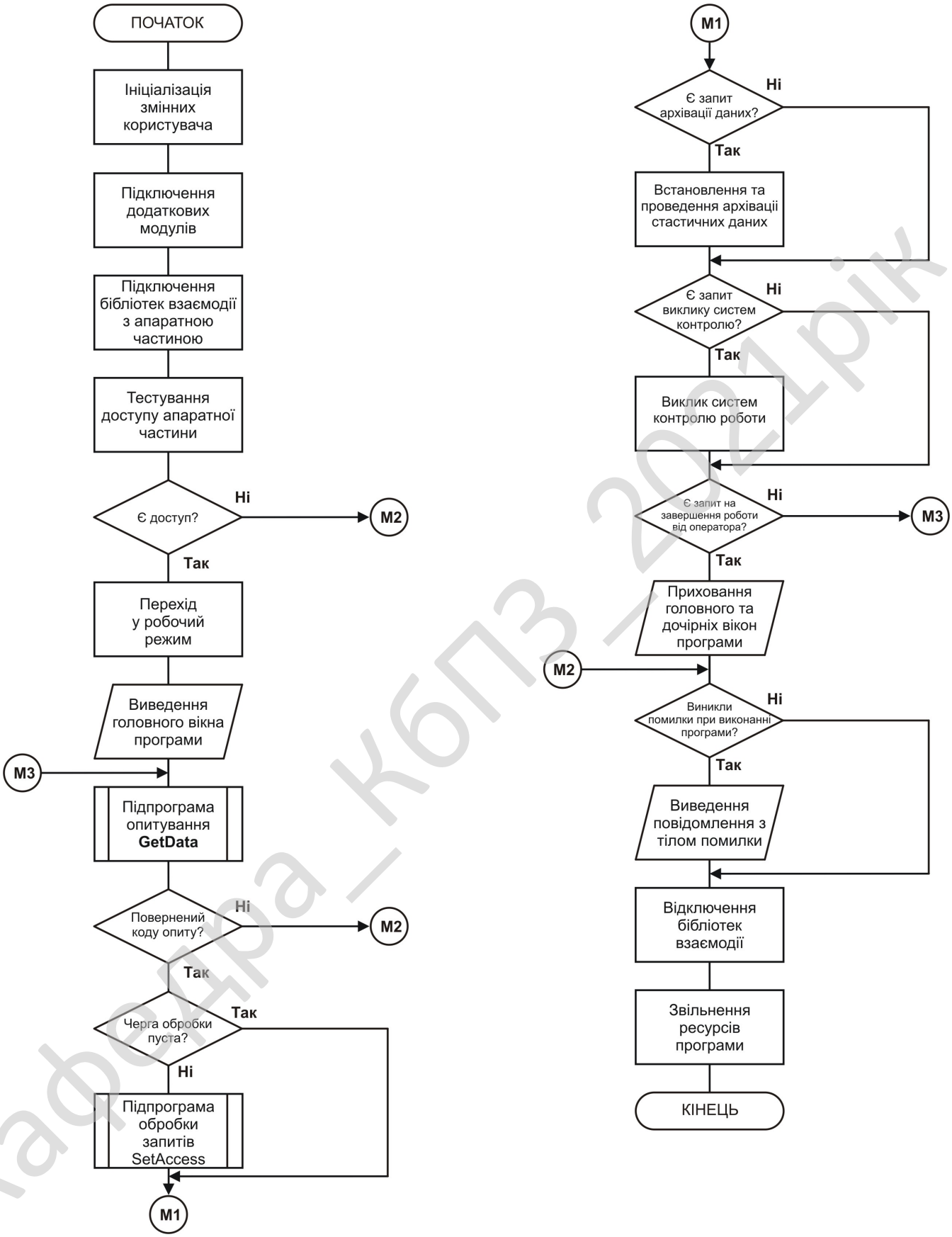

Рисунок 4.1 – Блок схема програми

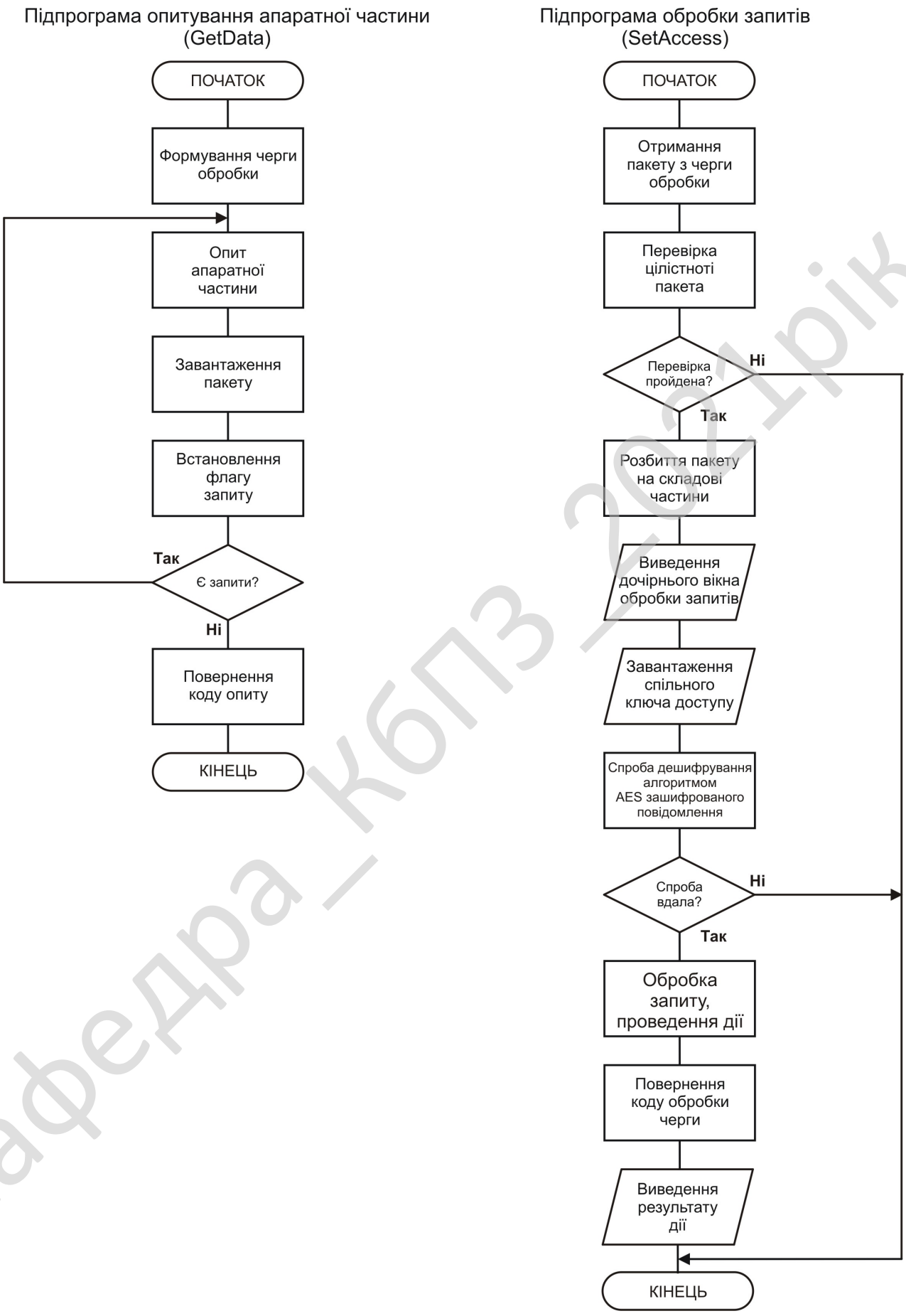

Рисунок 4.2 – Блок схема підпрограми

– Підпрограма опитування GetData – підпрограма опитування пристроїв, виявлення обривів ліній;

– Підпрограма обробки запитів SetAccess – підпрограма обробки дій з переферійних пристроїв;

– Встановлення та проведення архівації стастичних даних – проведення архівації статистичних даних роботи програми.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 28147:2009, що є класичним алгоритмом симетричного шифрування на основі мережі Фейстеля (рисунок 4.3). Даний алгоритм шифрує інформацію блоками по 64 біта (такі алгоритми називаються "блоковими"). Зміст мережі Фейстеля полягає в тому, що блок шифруемої інформації розбивається на два або більше субблоків, частина яких обробляється за певним законом, після чого результат цієї обробки накладається (операцією побітового додавання за модулем 2) на необроблювані субблоки. Потім субблоки міняються місцями, після чого обробляються знову й т.д. певне для кожного алгоритму число раз – раундів.

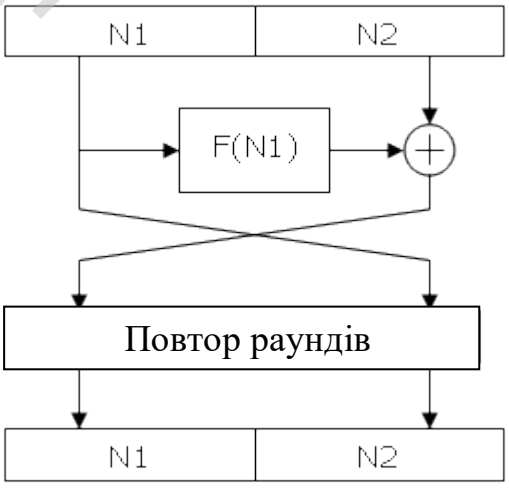

Рисунок 4.3 – Мережа Фейстеля

Основна відмінність алгоритмів симетричного шифрування друг від друга складається саме в різних функціях обробки субблоків. Дана функція часто називається "основним криптографічним перетворенням", оскільки саме вона несе основне навантаження при шифруванні інформації. Основне перетворення алгоритму ДСТУ 28147:2009 є досить простим, що забезпечує високу швидкодію алгоритму; у ньому виконуються наступні операції (рисунок 4.4).

Рисунок 4.4 – Основне перетворення алгоритму ДСТ 28147:2009

1. Додавання субблоку з певним фрагменом ключа шифрування за модулем 2^{32}. Кх – це 32-бітна частина ("підключ") 256-бітного ключа шифрування, якому можна представити як конкатенацію 8 підключів: К = $К_0К_1К_2К_3К_4К_5К_6$ $К_7$. Залежно від номера раунду й режиму роботи алгоритму (про їх – нижче), для даної операції вибирається один з підключів.

2. Таблична заміна. Для її виконання субблок розбивається на 8 4-бітних фрагментів, кожний з яких прогоняється через свою таблицю заміни. Таблиця заміни містить у певній послідовності значення від 0 до 15 (тобто всі варіанти значень 4-бітні фрагменти даних); на вхід таблиці подається блок даних, числове подання якого визначає номер вихідного значення. Наприклад, подається значення 5 на вхід наступної таблиці: "13 0 11 74 91 10 143 5 122 15 8 6". У результаті на виході виходить значення 9 (оскільки 0 заміняється на 13, 1 – на 0, 2 – на 11 і т.д.).

3. Побітове циклічне зрушення даних усередині субблока на 11 біт уліво.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Встановлення та використання програми

Після написання програмного продукту його потрібно впровадити в промислову експлуатацію. При цьому виправляються помилки, які були помічені, система настроюється на відповідний режим роботи.

Головне вікно програми представлене на малюнку 5.1.

Інтерфейс розробленої програми розділяється на чотири частини.

1. Стан ключів доступу – вибір певного картрідеру для перегляду його стану.

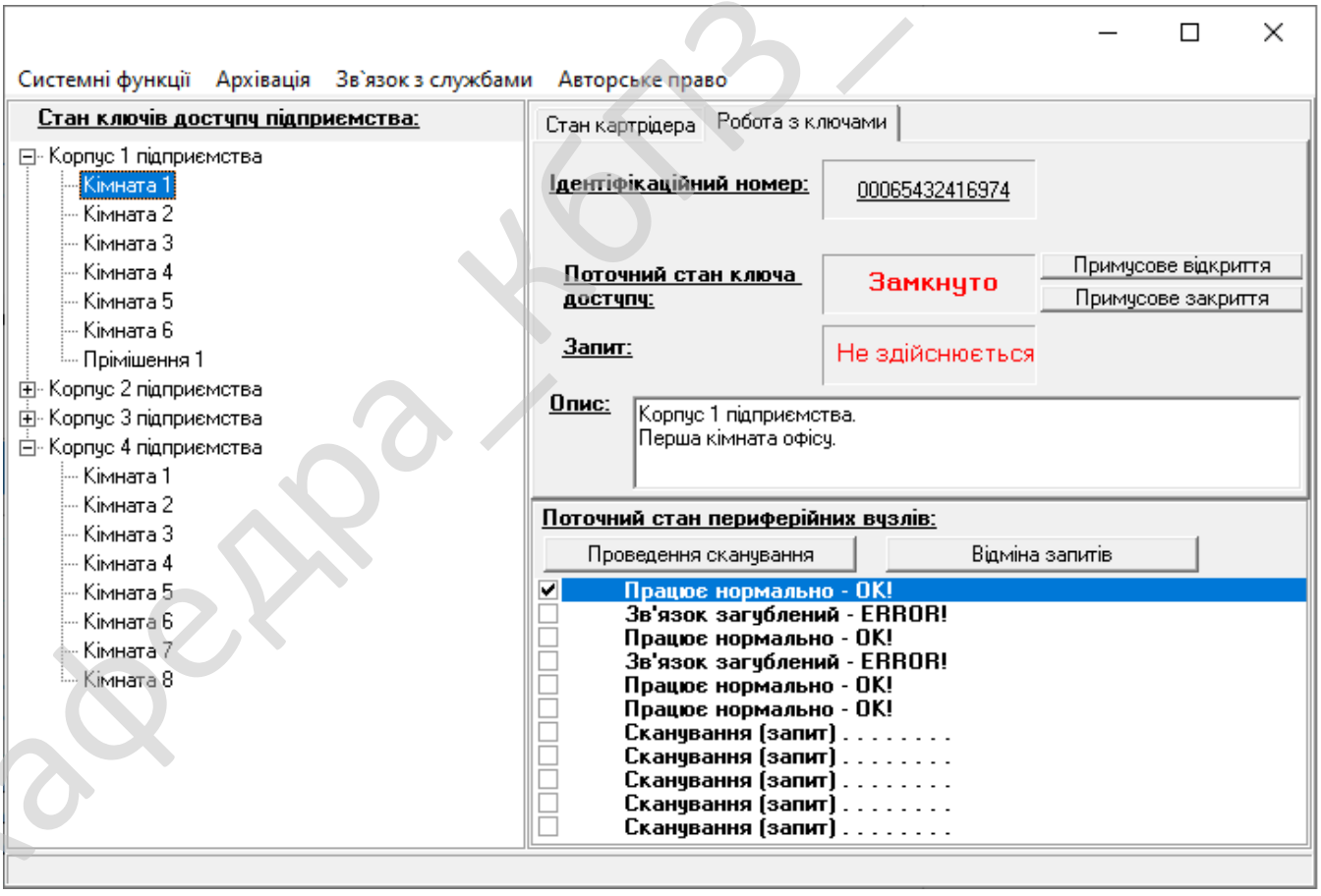

Рисунок 5.1 – Головне вікно програми

2. Стан картрідеру (вкладка робота із ключами) – основні характеристики вилученого картрідеру:

– ідентифікаційний номер;

– стан ключа;

– запит користувача.

3. Поточний стан перевірки периферії – запуск і зупинка сканування апаратної частини для виявлення можливих обривів лінії.

4. Системне меню дій. – стандартне дублююче меню основних дій програми.

На рисунку 5.2 показане збереження текучих даних в архів підсистеми зберігання архівіруваних даних.

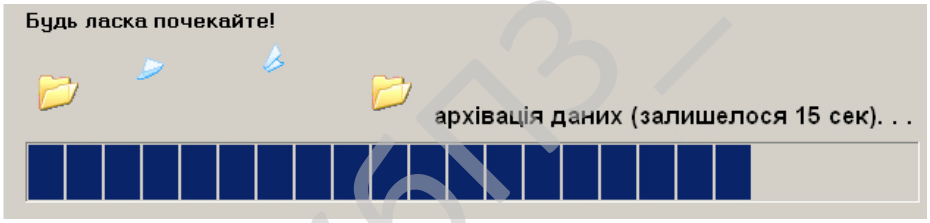

Рисунок 5.2 – Архівація потокових даних

Для збереження авторських прав (рисунок 5.3) розробленого програмного забезпечення створена форма «Про програму». Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс повністю розроблений під дане операційне середовище.

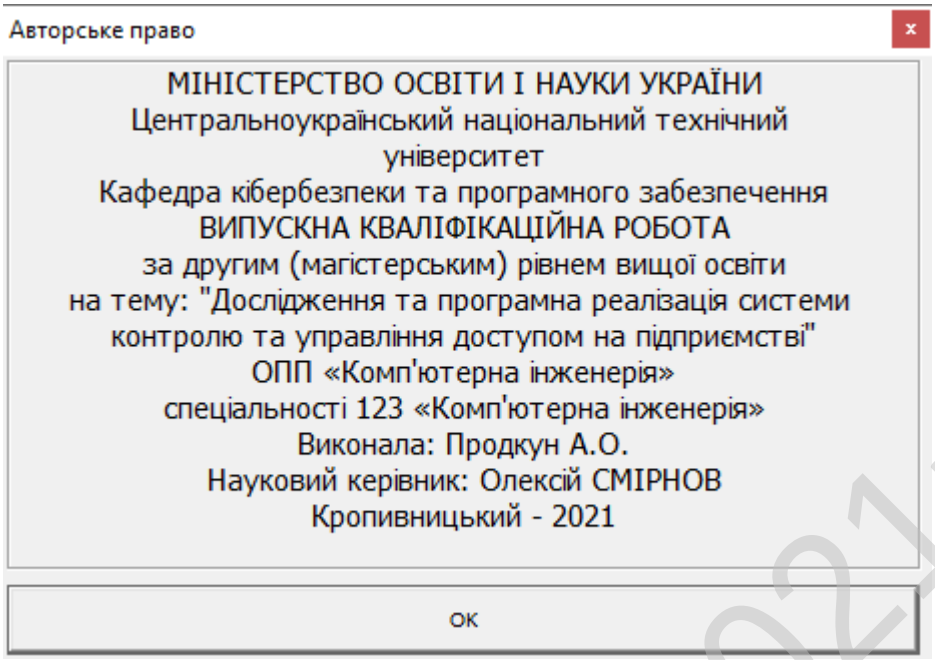

Рисунок 5.3 – Про програму

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи контролю та управління доступом на підприємстві.

Метою розробки є дослідження та програмна реалізація системи контролю та управління доступом на підприємстві.

Об'єктом дослідження є процес контролю та управління доступом на підприємстві.

Предметом дослідження є методи контролю та управління доступом на підприємстві.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод контролю та управління доступом на підприємстві.

– Розроблено вітчизняний продукт контролю та управління доступом на підприємстві, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведено дослідження та виконана програмна реалізація системи контролю та управління доступом на підприємстві.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	150 (2 ост. цифри № зал*10^1)
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	15000 (2 ост. цифр № зал*10^3)
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	60
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, $A = 2{,}45$; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}02\ 7.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПV_j, \qquad (7.3)$$

де: $ПV_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 86 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1- 7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3,8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м.п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{mic}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 567,5/(60·8) = 1,2 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,5	0,25
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	0,5	
Всього		2	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	0,5	0,2
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	15000	45000
Продакт-менеджер	0,25	15000	11250
Інженер-програміст	3,8	15000	171000
Інженер-електронщик	1,2	15000	54000
Адміністратор мережі	0,25	15000	11250
Дизайнер WEB	0,2	15000	9000
Інженер-верстальник	0,2	15000	9000
Бухгалтер-економіст	0,136	15000	6120
Всього за період розробки	$R_{сп}$ = 7,036	-	$Ф_{роб}$ = 316620

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{316620}{7 \cdot 60} = 754 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

S_y – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 200…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \qquad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.6.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину Комбест за 29.11.21. (джерело https://compbest.com.ua/komputer_hp_prodesk_600_g2_tower_intel_core_i3-6100_2_4_yadra_po_3.7_ghz_8_gb_ddr4_500_gb_hdd/.)

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10747
Системний блок	HP ProDesk 600 G2 Tower	7347
Процесор	6th gen Intel® Core™ i3 i3-6100 (3M Cache, (4) ядра по 3.7 GHz)	-
Системна плата	Intel® Q150, PCI Express x16 – 1, PC Express x1 – 3, DDR4-SDRAM, 4x USB 2.0 6x USB 3.0, 4x Audio Ports, RJ-45 (LAN) VGA, 2x DP, COM-Port, 2x PS/2.	-
Відеокарта	Вбудована Intel® HD Graphics 530	-
Жорсткий диск	HDD: 500 Gb 7200 Serial ATA	-
Оперативна пам’ять	8GB (2133 MHz) DDR4-SDRAM (2 x 4 GB)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	-
Корпус	ATX Middle Tower Pro, 3GTLA-489, PSU 450W(FSP Brand: ATX-350PNR, 12cm) black, (front bezel – black+light silver; bod material – 0.6mm), 80mm fan (rear 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR -35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	-
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3400
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10747	8597,6	94573,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	113125,1

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	113125	-	-
Всього по групі	113125	40	45250
Нематеріальні активи			
4. Нематеріальні активи	15000	10	1500
Група 5, 6			
5. Вимірювальні пристрої	5190	25	1297,5
6. Транспортні засоби	0	20	0,0
7. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
Разом	$К_р$ = 1569315		$А_р$ = 125447,5

Примітка: вартість транспортного засобу приймаємо рівним нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 754 \cdot 209/150 = 1050 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_з \cdot 0{,}01, \quad (7.12)$$

де: $Н_з$ – норматив додаткової зарплати, %.

$$З_д = 1050 \cdot 10 \cdot 0{,}01 = 105 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 37\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(1050+105) = 254 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 1050 \cdot 15 \cdot 0{,}01 = 158 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 120$ грн., визначаємо вартість паперу за період розробки $N_м = 3$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 120 \cdot 3 = 360 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 4 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 6 грн./шт.

$$З_{М2} = 15 \cdot 20 + 25 = 325 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з.., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (360 + 325 + 1702)/150 = 16 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п = 15\%$) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 1050 \cdot 15 \cdot 0{,}01 = 158 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е = 150$ прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 125448 \cdot 3/(15 \cdot 12) = 209 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	1050
2. Додаткова зарплата виконавців	$З_д$	105
3. Відрахування на соціальні потреби	$С_{оц}$	254
4. Загальногосподарські витрати	$Г_{осп}$	158
5. Витрати на матеріали	$З_М$	16
6. Освоєння нових операційних систем, мов програмування	$О_п$	158
7. Амортизація основних фондів	$А_м$	209
8. Повна собівартість програмного забезпечення	$С_п$	1950
9. Плановий прибуток	$П_р$	1170
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	3120
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	624
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3744

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 1050+105+254+158+16+158+209 = 1950 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 60%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 60 \cdot 1950 = 1170 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Вартість програмного рішення взятого для порівняння складе 15000 грн. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	4000	3744
Всього капітальних витрат	4000	3744

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	63360	47520
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	1000	936
Всього витрат за рік	$І$	64360	48456

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування сервера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин для обслуговування системи зменшилось з 400 годин на рік до 300 годин на рік, тому витрати складуть:

$$З_{р\,баз} = 400 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 63360\ грн,$$

до:

$$З_{р\,нов} = 300 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 47520\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Визначити різницю споживання електроенергії при впровадженні систем не має можливості, тому витрати на електроенергію в розрахунку приймаємо рівними нулю.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації, %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	4000	3744	1000	936
Всього відрахувань	-	4000	3744	1000	936

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum E_р K_р, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.

$$E_в = (3120 - 1950) \cdot 150 - (0,05 \cdot 1408000 + 0,4 \cdot 113125 + 0,25 \cdot 33190 + 0,1 \cdot 15000) \frac{3}{12} = 144138 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{161315}{(3120 - 1950) \cdot 150 \cdot 12 / 3} = 0,2 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	150
2. Повна собівартість розробленої програми	Грн.	1950
3. Ціна розробленої програми	Грн.	3120
4. Плановий прибуток від реалізації розробленої програми	Грн.	1170
5. Рентабельність програмної продукції	%	60
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1569315
7. Загальний прибуток від реалізації програмної продукції	Грн.	175500
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	144138
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,2
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3744
11.Величина економічного ефекту у користувача програмної продукції	Грн.	15968
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,1

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б + E_н K_б) - (I_н + E_н K_н), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (64360+0,25\cdot4000)-(48456+0,25\cdot3744) = 15968 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{\Delta K}{E_{сп}}, \qquad (7.28)$$

$$T_{сп} = \frac{4000 - 3744}{15968} < 0,1 \text{ року}.$$

Як бачимо з розрахунків запропонований варіант е більш економічно доцільним ніж варіант вибраний для порівняння, який на даний момент є лідером продаж на ринку.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя [3]. Охорона праці є складовою частиною безпеки життя.

Законом України “Про охорону праці” [2] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [4], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [1].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж іс тотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними

терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [1], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

невідповідність ергономічних показників робочого місця діючим вимогам.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	7,3
Довжина	8
Висота	3,1

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,5
Обсяг, V	$м^3$	не менше 20.0	20,1

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працює 9 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [1] та НПАОП 0.00 -1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови Головного державного санітарного лікаря України [5], робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, та к і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	45-60	0,12
Тепла	23-25	50-70	0,1	23-25	52-70	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP Laser 107a, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному

освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору).

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 7,3 м, довжина – 8 м, висота – 3,1 м.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою [6]:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S=7,3 х 8 = 58,4 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5 [6]);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1 [6]);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою [6]:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, S = 58,4 м^2;

h – розрахункова висота підвісу, h = 3,1 м (співпадає з висотою стелі, т.я.

лампи освітлення закріплюються на стелі);

A – ширина приміщення, A = 7,3 м;

B – довжина приміщення, B = 8 м.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$i=1,23.$$

Знаючи індекс приміщення (i), за знаходимо n = 0,48 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам). Підставимо всі значення у формулу, визначемо світловий потік: F=60225 Лм.

Для штучного освітлення приміщення використовуються *LED* світильники *PANEL-B2B-595*, світловий потік яких $F_л$ = 3500 Лм.

Число світильників визначається по формулі [6]:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік одного світильника.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$N= 60225/ 3500=17,2 \text{ шт.}$$

Для забезпечення нормованої мінімальної освітленості приймаємо необхідну кількість світильників 18 шт.

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи контролю та управління доступом на підприємстві.

В межах Укр аїни в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів контролю та управління доступом на підприємстві.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем контролю та управління доступом на підприємстві.

– Досліджена система контролю та управління доступом на підприємстві.

– На основі отриманих результатів досліджень створена програмна реалізація системи контролю та управління доступом на підприємстві.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання контролю та управління доступом на підприємстві.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Embarcadero RAD Studio Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 28147:2009.

В цілому ство рене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 15968 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Продкун А.О. Дослідження та програмна реалізація системи контролю та управління доступом на підприємстві // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe **(Scopus)**.

3. Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus)**.

4. Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/abstract/?doi=ajit.2018.218.230 Doi: ajit.2018.218.230

5. Kovalenko Oleksandr, The mathematical model of the testing technology for DOM XSS vulnerabilities / O. Kovalenko, O. Smirnov, A.Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

6. Коваленко А.В. Технология тестирования DOM XSS

уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

7. Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: монографія / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 305 с.

8. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

9. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

10. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

11. Коваленко А.В. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

12. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

13. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". – Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

14. Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

15. Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

16. Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

17. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

18. Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

19. Коваленко А.В. Масштабирование имитационной модели технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

20. Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

21. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків

розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

22. Коваленко О.В. Управління ризиками розроблення програмного забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. – 2018. – С. 128-140.

23. Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

24. Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

25. Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141

26. Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

27. Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

28. Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

29. Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі

прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2(3). – Харків. – 2018. – С. 41-48.

30. Коваленко О.В. Математичні моделі технології тестування DOM XSS вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія : Технічні науки №4, 2018. – С. 29-36.

31. Коваленко О.В. Математична модель технології тестування вразливості до SQL ін’єкцій / О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

32. Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 173-180, 2019.

33. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць II міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 24-27 лютого 2016 р. – Київ: Європейський університет. – 2016. – С. 138-139.

34. Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

35. Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.В. Коваленко, А.А. Смирнов // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. – С. 264-266.

36. Коваленко А.В. Оценка показателя чистой приведенной стоимости

для количественной оценки рисков проекта разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

37. Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

38. Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). м. Харків. 30 березня – 1 квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

39. Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

40. Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез VIII міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

41. Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез III міжнародної науково-практичної конференції «Інформаційна та економічна

безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХНМІ ДВНЗ «УБС». – 2016. – С. 174-178.

42. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

43. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

44. Коваленко А.В. Метод управления рисками разработки программного обеспечения с использованием псевдобулевых методов бивалентного программирования / А.В. Коваленко, А.А. Смирнов // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

45. Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць III міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

46. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез II науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ

ім. Тараса Шевченко – 2017. – С. 203-205.

47. Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Conferenta internationala (editia a XIII-a). «Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

48. Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез дев'ятнадцятого міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

49. Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). м. Харків.10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

50. Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. Кіровоград: КНТУ. – 2017. – С. 92.

51. Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. –

2017. – С. 61.

52. Kovalenko O.V. Method of testing the dom xss vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // International Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. – 2017. – P. 7.

53. Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

54. Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

55. Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

56. Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез «Securitea informationala 2018». Conferenta internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

57. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез X міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

58. Коваленко О.В. Розробка методу передтестової компіляції й розподілу доступу / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницькй. 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215.

59. Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL -ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

60. Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп’ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування..... 2
2 Підстава для розробки..... 2
3 Мета та призначення розробки..... 2
4 Джерела розробки..... 2
5 Технічні вимоги..... 2
5.1 Вміст проекту..... 2
5.2 Показники призначення..... 3
5.3 Вимоги до функціональних характеристик..... 3
5.4 Вимоги до архітектури..... 3
5.5 Вимоги до надійності..... 3
5.6 Умови експлуатації..... 4
5.7 Вимоги до складу та параметрів технічних засобів..... 4
5.8 Вимоги до інформаційної і програмної сумісності..... 4
5.8.1 Обладнання..... 4
5.8.2 Мова програмування..... 4
5.8.3 Вхідні дані..... 5
5.8.4 Вихідні дані..... 5
6 Вимоги до програмної документації..... 5
7 Економічні вимоги..... 5
8 Вимоги щодо охорони праці..... 5
9 Перелік документів, що розробляються..... 6
10 Етапи розробки..... 6
11 Порядок контролю та приймання..... 6

					ВКРМ-123.21.0015.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Продкун А.О.			Дослідження та програмна реалізація системи контролю та управління доступом на підприємстві	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнов О.А.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи контролю та управління доступом на підприємстві.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи контролю та управління доступом на підприємстві.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи контролю та управління доступом на підприємстві;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Embarcadero RAD Studio Delphi 10.4 Sydney.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута розробка заходів з умов поліпшення охорони праці.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 95 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнов О.А.

Дослідження та програмна реалізація

системи контролю та управління доступом на підприємстві

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 52

Літера: РП

Кропивницький – 2021 року

Головний файл проекту

```
program KARTRIDER_PROJECT;

{%File 'PR.inc'}
uses
  Forms,
  Main in 'M1.pas' { Form1},
  AB in 'AB.pas' { Form2},
  Splash in 'Splash.pas' {wnd_Splash};

{$R *.res}
procedure StartPointProc;
begin
 asm
  DB 000h, 034h, 0F4h, 02Ah, 000h, 03Dh, 0FFh, 0CAh
 end;
end
procedure CheckTrial;
begin
 asm
  DB 09Dh, 03Dh, 04Ah, 061h, 025h, 0CDh, 0C7h, 0DFh
  DB 0DAh, 0DAh, 0E6h, 025h, 0DAh, 0DAh, 0DAh, 0DAh
  DB 03Ch, 025h, 025h, 025h, 071h, 057h, 04Ch, 044h
  DB 049h, 005h, 055h, 040h, 057h, 04Ch, 04Ah, 041h
  DB 005h, 04Dh, 044h, 056h, 005h, 040h, 05Dh, 055h
  DB 04Ch, 057h, 040h, 041h, 00Bh, 025h, 025h, 025h
 end;
end;
begin
  Application.Initialize;
  Application.Title := 'Kartrider';
      wnd_Splash:=Twnd_Splash.Create(Application);
      wnd_Splash.Show;//Показання
      wnd_Splash.Update;//Обновлення

  Application.CreateForm(Twnd_msi_Main, wnd_msi_Main);
  Application.CreateForm(Twnd_msi_Main, wnd_msi_Main);
  Application.CreateForm(Twnd_msi_Main, wnd_msi_Main);


  Application.Run;
  wnd_Splash.Free;
end.
```

Головне вікно програми Unit1

```
unit Unit1;

interface
  {Інтерфейсна частина програми}
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ShellApi, Registry, ExtCtrls, unit3;

  type
TForm1 = class(TForm)
Edit1: TEdit;
Label1: TLabel; Label2: TLabel; Label3: TLabel; Label4: TLabel;
Timer1: TTimer; Timer2: TTimer;
Panel1: TPanel;
gb1: TGroupBox; gb2: TGroupBox;
en_code: TSpeedButton; savet: TSpeedButton;
clear2: TSpeedButton; clear1: TSpeedButton;
loadt: TSpeedButton; GroupBox1: TGroupBox;
rb1: TRadioButton; rb2: TRadioButton;
GroupBox2: TGroupBox;
i1: TSpeedButton;
opend: TOpenDialog; saved: TSaveDialog;
ApplicationEvents1: TApplicationEvents;
RxTrayIcon1: TRxTrayIcon;
PopupMenu1: TPopupMenu;
N1: TMenuItem; N2: TMenuItem; N3: TMenuItem;
procedure rb1Click(Sender: TObject);
procedure rb2Click(Sender: TObject);
procedure en_codeClick(Sender: TObject);
procedure clear1Click(Sender: TObject);
procedure clear2Click(Sender: TObject);
procedure i1Click(Sender: TObject);
procedure loadtClick(Sender: TObject);
procedure savetClick(Sender: TObject);
procedure ApplicationEvents1ShowHint(var HintStr: String;
var CanShow: Boolean; var HintInfo: THintInfo);
procedure N3Click(Sender: TObject);
procedure RxTrayIcon1DblClick(Sender: TObject);
procedure N2Click(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormCreate(Sender: TObject);
procedure Label2Click(Sender: TObject);
procedure Label2MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure Label3Click(Sender: TObject);
procedure Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure Timer1Timer(Sender: TObject);
procedure Timer2Timer(Sender: TObject);
procedure FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
procedure Panel1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure FormResize(Sender: TObject);
private
      { Private declarations }
    public
      { Public declarations }
    end;
  var
    hMenuHandle:HMENU;
    hTaskBar : THandle;
    Form1: TForm1;
    Mask: Boolean;
```

```
    Guest: Boolean;
    ShowButton: Boolean;
    AccessEnabled: Boolean;
    Pass: String;
    GPass: String;
    F2:textfile;
    Reg: Tregistry;

  implementation
  {Реаізаційна частина програми}
  {$R *.DFM}

procedure TForm1.FormClose(Sender:TObject; var Action:TCloseAction);
  begin
    chdir(extractfilepath(application.exename));
    AssignFile(F2,'users.log');
    append(F2);
    Writeln(F2,timetostr(time) + ':'+edit1.text );
    CloseFile(F2);
    if showbutton and AccessEnabled then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      Action:=caNone;
      timer2.Interval:= 0;
      frmshield.show;
    end;
    if pass = edit1.Text then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      if not (csDesigning in ComponentState) then
      RegisterServiceProcess(GetCurrentProcessID, 1);
      timer2.Interval := 0;
      SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, nil, 0);
      hTaskbar := FindWindow('Shell_TrayWnd', Nil);
      ShowWindow(hTaskBar, SW_SHOWNORMAL);
      frmshield.show;
    end;
    if (GPass = Edit1.Text) and guest then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      Action:=caNone;
      timer2.Interval := 0;
      frmshield.show;
    end;
    if (pass<>edit1.Text) and ((gpass<>edit1.Text)or not guest) then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      label4.Visible := true;
      timer1.Interval := 2000;
      Action:=caNone;
    end;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    form1.Top := 0;
    form1.Left := 0;
    form1.Width := screen.Width;
    form1.Height := screen.Height;
    hTaskbar := FindWindow('Shell_TrayWnd', Nil);
    ShowWindow(hTaskBar, SW_HIDE);
```

```
  if not (csDesigning in ComponentState) then
  RegisterServiceProcess(GetCurrentProcessID, 0);
  hMenuHandle := GetSystemMenu(Handle, FALSE);
  IF (hMenuHandle <> 0) THEN
  DeleteMenu(hMenuHandle, SC_CLOSE, MF_BYCOMMAND);
  SystemParametersInfo(SPI_SCREENSAVERRUNNING, 1, nil, 0);
  chdir(extractfilepath(application.exename));
  AssignFile(F2,'users.log');
  append(F2);
  Writeln(F2,'');
  CloseFile(F2);
  Reg := TRegistry.Create;
  Reg.RootKey := HKEY_CURRENT_USER;
  if not Reg.OpenKey('\Software\',True) then
  try
    Reg.RootKey := HKEY_CURRENT_USER;
    if Reg.OpenKey('\Software\',True)
    then
    begin
      Guest := reg.ReadBool ('Guest');
      Mask := reg.ReadBool ('Mask');
      ShowButton := reg.ReadBool ('ShowButton');
      Pass := Reg.ReadString('Pass');
      GPass := Reg.ReadString('GPass');
    end
  finally
    Reg.CloseKey;
    Reg.Free;
    inherited;
  end;
  edit1.PasswordChar := #0;
  if mask then edit1.PasswordChar := '*';
  if showbutton then label3.Visible := true;
end;

procedure TForm1.Label2Click(Sender: TObject);
begin
  close;
end;

procedure TForm1.Label2MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  label2.Font.Color := clYellow;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  label2.Font.Color := clNavy;
  label3.Font.Color := clNavy;
end;

procedure TForm1.Label3Click(Sender: TObject);
begin
  AccessEnabled:= True;
  Form1.Close ;
end;

procedure TForm1.Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  label3.Font.Color := clYellow;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  label4.Visible := false;
  timer1.Interval := 0;
```

```
  end;

  procedure TForm1.Timer2Timer(Sender: TObject);
  begin
    if (mouse.CursorPos.x < panel1.Width) and (mouse.CursorPos.y >
(screen.height-40)) then
    begin
      setcursorpos(mouse.CursorPos.x ,screen.height-41);
    end;
    hTaskbar := FindWindow('Shell_TrayWnd', Nil);//Hide taskbar
    ShowWindow(hTaskBar, SW_HIDE);
  end;

  procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
  begin
    if ((key>=112) and (key<=123)) or (key=16) or (key=17) or (key=18)  then
exit;
  end;

  procedure TForm1.Panel1MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  begin
    setcursorpos(x,screen.height-41);
  end;

  procedure TForm1.FormResize(Sender: TObject);
  begin
    form1.Top := 0;
    form1.Left := 0;

    form1.Width := screen.Width;
    form1.Height := screen.Height;
  end;

procedure TForm1.rb1Click(Sender: TObject);
begin
 if rb1.Checked = true then
k1.Enabled:=true;
 k1.Color:=clWindow;
end;

procedure TForm1.rb2Click(Sender: TObject);
begin
 if rb2.Checked = true then
  k1.Enabled:=true;
  k1.Color:=clWindow;
end;

function Crypt(Text,Key: String; Encode: boolean): String;
var
  i, KeyLength: integer;
  Sign: ShortInt;
begin
 KeyLength:=Length(Key);
 if Encode then Sign :=-1 else Sign:=1;
 for i:=1 to Length(Text) do
  Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
  Result:=Text;
end;

function CheckRb: boolean;
begin
 if (Form1.rb1.Checked = false) and (Form1.rb2.Checked = false) then
  begin
   Result:=false;
  end
 else
  Result:= true;
```

```
end;

function CheckK(K: String): boolean;
begin
 if Length(K) = 0 then Result:=false else Result:=true;
end;

procedure TForm1.en_codeClick(Sender: TObject);
begin
 if Checkrb = false then Exit else
  if rb1.Checked then
   begin
    if CheckK(k1.Text) = false then
     begin
      Exit;
     end
    else
     Memo2.Text:=Crypt(Memo1.Text, k1.Text, false);
   end;
  if rb2.Checked = true then
   begin
    if CheckK(k1.Text) = false then
     begin
      Exit;
     end
    else
     Memo2.Text:=Crypt(Memo1.Text, k1.Text, true);
   end;
end;

procedure TForm1.clear1Click(Sender: TObject);
begin
 Memo1.Clear;
end;

procedure TForm1.clear2Click(Sender: TObject);
begin
 Memo2.Clear;
end;

procedure TForm1.loadtClick(Sender: TObject);
var
 F: TextFile;
 T: String;
begin
 if opend.Execute = true then
  begin
   AssignFile(F, opend.FileName);
   Reset(F);
    while not EoF(F) do
     begin
      ReadLn(F, T);
      Memo1.Lines.Add(T);
     end;
    CloseFile(F);
  end
 else Exit;
end;

procedure TForm1.savetClick(Sender: TObject);
var
 F: TextFile;
 T: String;
begin
 if saved.Execute = true then
  begin
   AssignFile(F, saved.FileName);
   ReWrite(F);
   Write(F, Memo2.Text);
```

```
   CloseFile(F);
  end
 else Exit;
end;

procedure TForm1.ApplicationEvents1ShowHint(var HintStr: String;
  var CanShow: Boolean; var HintInfo: THintInfo);
begin
 if (HintInfo.HintControl.ClassName = 'TEdit') then
  HintStr:=(HintInfo.HintControl as TEdit).Text;
end;

procedure TForm1.ApplicationMinimize(Sender: TObject);
begin
ShowWindow(Application.Handle, SW_HIDE);
end;

procedure TForm1.ApplicationRestore(Sender: TObject);
begin
ShowWindow(Application.Handle, SW_HIDE);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
Application.OnMinimize := ApplicationMinimize;
Application.OnRestore := ApplicationRestore;
end;

procedure TForm1.N3Click(Sender: TObject);
begin
RxTrayIcon1DblClick(Sender);
end;

procedure TForm1.RxTrayIcon1DblClick(Sender: TObject);
begin

if (IsWindowVisible(Form1.Handle)=false) then
begin
  Form1.show;
  Form1.BringToFront;
end
 else
  begin
    Form1.hide;
  end;
end;

procedure TForm1.N2Click(Sender: TObject);
begin
Form1.Close;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
if IEQ = mrYes then
  begin
     Action:=caFree;
  end
  else begin
        Form1.Repaint;
        Action:=caNone;
        Form1.Hide;
        end;
end;
  end.
```

```
unit AB;

interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
 Buttons, ExtCtrls;

type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

implementation
uses Unit1;

{$R *.dfm}

procedure TAboutBox.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Action:=caNone;
end;

procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
AboutBox.hide;
Form1.show;
end;

end.
```

Вікно заставки

```
Unit splash;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, GlPrgrs, jpeg, ExtCtrls, StdCtrls;
const
  KOL_FORMS=28;//21
type
  TU_Form_Splash = class(TForm)
    Panel1: TPanel;
    Image1: TImage;
    glProgress1: TglProgress;
  private
    { Private declarations }
  public
    { Public declarations }
    procedure MS(var A:Tmessage);message WM_USER+342;
  end;

var
  U_Form_Splash: TU_Form_Splash;

implementation

{$R *.dfm}

{ TU_Form_Splash }

procedure TU_Form_Splash.MS(var A: Tmessage);
var
G:Extended;
s:string;
begin
s:=pchar(A.LParam);
glProgress1.Caption:='progress...[%d%%]'+' Завантаження програми '+s;
G:=100/KOL_FORMS;
if glProgress1.Percent<100 then
glProgress1.Percent:=glProgress1.Percent+round(G);
end;

end.
```

Основний модуль даних

```
unit KardriderData;

interface

uses Messages, Windows, SysUtils, Classes, Graphics, Menus, Controls, Imm,
ActnList, MultiMon, HelpIntfs;

type

TKartRiderApp = class(TComponent)
private
  FOnModalBegin: TNotifyEvent;
  FOnModalEnd: TNotifyEvent;
  FOnHelp: THelpEvent;
  FOnHint: TNotifyEvent;
  FOnIdle: TIdleEvent;
  FOnDeactivate: TNotifyEvent;
  FOnActivate: TNotifyEvent;
  FOnMinimize: TNotifyEvent;
  FOnRestore: TNotifyEvent;
  FOnShortCut: TShortCutEvent;
  FOnShowHint: TShowHintEvent;
  FOnSettingChange: TSettingChangeEvent;
  function CheckIniChange(var Message: TMessage): Boolean;
  function DispatchAction(Msg: Longint; Action: TBasicAction): Boolean;
  procedure DoActionIdle;
  function DoMouseIdle: TControl;
  procedure DoNormalizeTopMosts(IncludeMain: Boolean);
  function GetCurrentHelpFile: string;
  function GetDialogHandle: HWND;
  function GetExeName: string;
  function GetIconHandle: HICON;
  function GetTitle: string;
  procedure HintTimerExpired;
  procedure IconChanged(Sender: TObject);
  function InvokeHelp(Command: Word; Data: Longint): Boolean;
  procedure NotifyForms(Msg: Word);
  function ProcessMessage(var Msg: TMsg): Boolean;
  procedure SetBiDiMode(Value: TBiDiMode);
  procedure SetDialogHandle(Value: HWnd);
  procedure SetHandle(Value: HWnd);
  procedure SetHint(const Value: string);
  procedure SetHintColor(Value: TColor);
  procedure SetIcon(Value: TIcon);
  procedure SetShowHint(Value: Boolean);
  procedure SetTitle(const Value: string);
  procedure SettingChange(var Message: TWMSettingChange);
  procedure StartHintTimer(Value: Integer; TimerMode: TTimerMode);
  procedure StopHintTimer;
  procedure WndProc(var Message: TMessage);
  procedure UpdateVisible;
  function  ValidateHelpSystem: Boolean;
  procedure WakeMainThread(Sender: TObject);
protected
  procedure Idle(const Msg: TMsg);
  function IsDlgMsg(var Msg: TMsg): Boolean;
  function IsHintMsg(var Msg: TMsg): Boolean;
  function IsKeyMsg(var Msg: TMsg): Boolean;
  function IsMDIMsg(var Msg: TMsg): Boolean;
  function IsShortCut(var Message: TWMKey): Boolean;
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  procedure ActivateHint(CursorPos: TPoint);
  procedure BringToFront;
  procedure ControlDestroyed(Control: TControl);
```

```
  procedure CancelHint;
  procedure CreateForm(InstanceClass: TComponentClass; var Reference);
  procedure CreateHandle;
  function ExecuteAction(Action: TBasicAction): Boolean; reintroduce;
  procedure HandleException(Sender: TObject);
  procedure HandleMessage;
  function HelpCommand(Command: Integer; Data: Longint): Boolean;
  function HelpContext(Context: THelpContext): Boolean;
  function HelpJump(const JumpID: string): Boolean;
  function HelpKeyword(const Keyword: String): Boolean;
  procedure HideHint;
  procedure HintMouseMessage(Control: TControl; var Message: TMessage);
  procedure HookMainWindow(Hook: TWindowHook);
  procedure HookSynchronizeWakeup;
  procedure NormalizeTopMosts;
  procedure ProcessMessages;
  procedure Restore;
  procedure RestoreTopMosts;
  procedure Run;
  procedure ShowException(E: Exception);
  procedure Terminate;
  property Handle: HWnd read FHandle write SetHandle;
  property HelpFile: string read FHelpFile write FHelpFile;
  property Hint: string read FHint write SetHint;
end;


var
Application: TKartRiderApp;
Screen: TScreen;
Ctl3DBtnWndProc: Pointer = nil;
HintWindowClass: THintWindowClass = THintWindow;

function DisableTaskWindows(ActiveWindow: HWnd): Pointer;
procedure EnableTaskWindows(WindowList: Pointer);
function KeysToShiftState(Keys: Word): TShiftState;
function KeyDataToShiftState(KeyData: Longint): TShiftState;
function KeyboardStateToShiftState(const KeyboardState: TKeyboardState):
TShiftState; overload;
function KeyboardStateToShiftState: TShiftState; overload;

procedure RestoreFocusState(FocusState: TFocusState);
begin
FocusCount:= Integer(FocusState);
end;

procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
var
Style: Longint;
begin
if ClientHandle <> 0 then
begin
  Style:= GetWindowLong(ClientHandle, GWL_EXSTYLE);
  if ShowEdge then
    if Style and WS_EX_CLIENTEDGE = 0 then
      Style:= Style or WS_EX_CLIENTEDGE
    else
      Exit
  else if Style and WS_EX_CLIENTEDGE <> 0 then
    Style:= Style and not WS_EX_CLIENTEDGE
  else
    Exit;
  SetWindowLong(ClientHandle, GWL_EXSTYLE, Style);
  SetWindowPos(ClientHandle, 0, 0,0,0,0, SWP_FRAMECHANGED or SWP_NOACTIVATE or
    SWP_NOMOVE or SWP_NOSIZE or SWP_NOZORDER);
end;
end;

type
```

```
PTaskWindow = ^TTaskWindow;
TTaskWindow = record
  Next: PTaskWindow;
  Window: HWnd;
end;

var
TaskActiveWindow: HWnd = 0;
TaskFirstWindow: HWnd = 0;
TaskFirstTopMost: HWnd = 0;
TaskWindowList: PTaskWindow = nil;

procedure DoneApplication;
begin
with Application do
begin
  if Handle <> 0 then ShowOwnedPopups(Handle, False);
  ShowHint:= False;
  Destroying;
  DestroyComponents;
end;
end;

function DoDisableWindow(Window: HWnd; Data: Longint): Bool; stdcall;
var
P: PTaskWindow;
begin
if (Window <> TaskActiveWindow) and IsWindowVisible(Window) and
  IsWindowEnabled(Window) then
begin
  New(P);
  P^.Next:= TaskWindowList;
  P^.Window:= Window;
  TaskWindowList:= P;
  EnableWindow(Window, False);
end;
Result:= True;
end;

function DisableTaskWindows(ActiveWindow: HWnd): Pointer;
var
SaveActiveWindow: HWND;
SaveWindowList: Pointer;
begin
Result:= nil;
SaveActiveWindow:= TaskActiveWindow;
SaveWindowList:= TaskWindowList;
TaskActiveWindow:= ActiveWindow;
TaskWindowList:= nil;
try
  try
    EnumThreadWindows(GetCurrentThreadID, @DoDisableWindow, 0);
    Result:= TaskWindowList;
  except
    EnableTaskWindows(TaskWindowList);
    raise;
  end;
finally
  TaskWindowList:= SaveWindowList;
  TaskActiveWindow:= SaveActiveWindow;
end;
end;

procedure EnableTaskWindows(WindowList: Pointer);
var
P: PTaskWindow;
begin
while WindowList <> nil do
begin
```

```
  P:= WindowList;
  if IsWindow(P^.Window) then EnableWindow(P^.Window, True);
  WindowList:= P^.Next;
  Dispose(P);
end;
end;

function DoFindWindow(Window: HWnd; Param: Longint): Bool; stdcall;
begin
if (Window <> TaskActiveWindow) and (Window <> Application.FHandle) and
  IsWindowVisible(Window) and IsWindowEnabled(Window) then
  if GetWindowLong(Window, GWL_EXSTYLE) and WS_EX_TOPMOST = 0 then
  begin
    if TaskFirstWindow = 0 then TaskFirstWindow:= Window;
  end else
  begin
    if TaskFirstTopMost = 0 then TaskFirstTopMost:= Window;
  end;
Result:= True;
end;

function FindTopMostWindow(ActiveWindow: HWnd): HWnd;
begin
TaskActiveWindow:= ActiveWindow;
TaskFirstWindow:= 0;
TaskFirstTopMost:= 0;
EnumThreadWindows(GetCurrentThreadID, @DoFindWindow, 0);
if TaskFirstWindow <> 0 then
  Result:= TaskFirstWindow else
  Result:= TaskFirstTopMost;
end;

function SendFocusMessage(Window: HWnd; Msg: Word): Boolean;
var
Count: Integer;
begin
Count:= FocusCount;
SendMessage(Window, Msg, 0, 0);
Result:= FocusCount = Count;
end;

type
PCheckTaskInfo = ^TCheckTaskInfo;
TCheckTaskInfo = record
  FocusWnd: HWnd;
  Found: Boolean;
end;

function CheckTaskWindow(Window: HWnd; Data: Longint): Bool; stdcall;
begin
Result:= True;
if PCheckTaskInfo(Data)^.FocusWnd = Window then
begin
  Result:= False;
  PCheckTaskInfo(Data)^.Found:= True;
end;
end;

function ForegroundTask: Boolean;
var
Info: TCheckTaskInfo;
begin
Info.FocusWnd:= GetActiveWindow;
Info.Found:= False;
EnumThreadWindows(GetCurrentThreadID, @CheckTaskWindow, Longint(@Info));
Result:= Info.Found;
end;

function FindGlobalComponent(const Name: string): TComponent;
```

```
var
I: Integer;
begin
for I:= 0 to Screen.FormCount - 1 do
begin
  Result:= Screen.Forms[I];
  if not (csInline in Result.ComponentState) and
    (CompareText(Name, Result.Name) = 0) then Exit;
end;
for I:= 0 to Screen.DataModuleCount - 1 do
begin
  Result:= Screen.DataModules[I];
  if CompareText(Name, Result.Name) = 0 then Exit;
end;
Result:= nil;
end;

function Subclass3DWnd(Wnd: HWnd): Boolean;
begin
Result:= False;
end;

procedure Subclass3DDlg(Wnd: HWnd; Flags: Word);
begin
end;

procedure SetAutoSubClass(Enable: Boolean);
begin
end;

function MakeObjectInstance(Method: TWndMethod): Pointer;
begin
Result:= WinUtils.MakeObjectInstance(Method);
Result:= Classes.MakeObjectInstance(Method);
end;

procedure FreeObjectInstance(ObjectInstance: Pointer);
begin
WinUtils.FreeObjectInstance(ObjectInstance);
Classes.FreeObjectInstance(ObjectInstance);
end;

function AllocateHWnd(Method: TWndMethod): HWND;
begin
Result:= WinUtils.AllocateHWnd(Method);
Result:= Classes.AllocateHWnd(Method);
end;

procedure DeallocateHWnd(Wnd: HWND);
begin
WinUtils.DeallocateHWnd(Wnd);
Classes.DeallocateHWnd(Wnd);
end;

function KeysToShiftState(Keys: Word): TShiftState;
begin
Result:= [];
if Keys and MK_SHIFT <> 0 then Include(Result, ssShift);
if Keys and MK_CONTROL <> 0 then Include(Result, ssCtrl);
if Keys and MK_LBUTTON <> 0 then Include(Result, ssLeft);
if Keys and MK_RBUTTON <> 0 then Include(Result, ssRight);
if Keys and MK_MBUTTON <> 0 then Include(Result, ssMiddle);
if GetKeyState(VK_MENU) < 0 then Include(Result, ssAlt);
end;

function KeyDataToShiftState(KeyData: Longint): TShiftState;
const
AltMask = $20000000;
CtrlMask = $10000000;
```

```
ShiftMask = $08000000;

begin
Result:= [];
if GetKeyState(VK_SHIFT) < 0 then Include(Result, ssShift);
if GetKeyState(VK_CONTROL) < 0 then Include(Result, ssCtrl);
if KeyData and AltMask <> 0 then Include(Result, ssAlt);
if KeyData and CtrlMask <> 0 then Include(Result, ssCtrl);
if KeyData and ShiftMask <> 0 then Include(Result, ssShift);

end;

function KeyboardStateToShiftState(const KeyboardState: TKeyboardState):
TShiftState;
begin
Result:= [];
if KeyboardState[VK_SHIFT] and $80 <> 0 then Include(Result, ssShift);
if KeyboardState[VK_CONTROL] and $80 <> 0 then Include(Result, ssCtrl);
if KeyboardState[VK_MENU] and $80 <> 0 then Include(Result, ssAlt);
if KeyboardState[VK_LBUTTON] and $80 <> 0 then Include(Result, ssLeft);
if KeyboardState[VK_RBUTTON] and $80 <> 0 then Include(Result, ssRight);
if KeyboardState[VK_MBUTTON] and $80 <> 0 then Include(Result, ssMiddle);
end;

function KeyboardStateToShiftState: TShiftState; overload;
var
KeyState: TKeyBoardState;
begin
GetKeyboardState(KeyState);
Result:= KeyboardStateToShiftState(KeyState);
end;

function IsAccel(VK: Word; const Str: string): Boolean;
begin
Result:= CompareText(Char(VK), GetHotKey(Str)) = 0;
end;

function GetParentForm(Control: TControl): TCustomForm;
begin
while Control.Parent <> nil do Control:= Control.Parent;
if Control is TCustomForm then
  Result:= TCustomForm(Control) else
  Result:= nil;
end;

function ValidParentForm(Control: TControl): TCustomForm;
begin
Result:= GetParentForm(Control);
if Result = nil then
  raise EInvalidOperation.CreateFmt(SParentRequired, [Control.Name]);
end;

constructor TControlCartRider.Create(AControl: TScrollingWinControl;
AKind: TScrollBarKind);
begin
inherited Create;
FControl:= AControl;
FKind:= AKind;
FPageIncrement:= 80;
FIncrement:= FPageIncrement div 10;
FVisible:= True;
FDelay:= 10;
FLineDiv:= 4;
FPageDiv:= 12;
FColor:= clBtnHighlight;
FParentColor:= True;
FUpdateNeeded:= True;
end;
```

```
function TControlCartRider.IsIncrementStored: Boolean;
begin
Result:= not Smooth;
end;

procedure TControlCartRider.Assign(Source: TPersistent);
begin
if Source is TControlScrollBar then
begin
  Visible:= TControlScrollBar(Source).Visible;
  Range:= TControlScrollBar(Source).Range;
  Position:= TControlScrollBar(Source).Position;
  Increment:= TControlScrollBar(Source).Increment;
  Exit;
end;
inherited Assign(Source);
end;

procedure TControlCartRider.ChangeBiDiPosition;
begin
if Kind = sbHorizontal then
  if IsScrollBarVisible then
    if not FControl.UseRightToLeftScrollBar then
      Position:= 0
    else
      Position:= Range;
end;

procedure TControlCartRider.CalcAutoRange;
var
I: Integer;
NewRange, AlignMargin: Integer;

procedure ProcessHorz(Control: TControl);
begin
  if Control.Visible then
    case Control.Align of
      alLeft, alNone:
        if (Control.Align = alLeft) or (Control.Anchors * [akLeft, akRight] =
[akLeft]) then
          NewRange:= Max(NewRange, Position + Control.Left + Control.Width);
      alRight: Inc(AlignMargin, Control.Width);
    end;
end;

procedure ProcessVert(Control: TControl);
begin
  if Control.Visible then
    case Control.Align of
      alTop, alNone:
        if (Control.Align = alTop) or (Control.Anchors * [akTop, akBottom] =
[akTop]) then
          NewRange:= Max(NewRange, Position + Control.Top + Control.Height);
      alBottom: Inc(AlignMargin, Control.Height);
    end;
end;

begin
if FControl.FAutoScroll then
begin
  if FControl.AutoScrollEnabled then
  begin
    NewRange:= 0;
    AlignMargin:= 0;
    for I:= 0 to FControl.ControlCount - 1 do
      if Kind = sbHorizontal then
        ProcessHorz(FControl.Controls[I]) else
        ProcessVert(FControl.Controls[I]);
    DoSetRange(NewRange + AlignMargin + Margin);
```

```
  end
  else DoSetRange(0);
end;
end;

function TControlCartRider.IsScrollBarVisible: Boolean;
var
Style: Longint;
begin
Style:= WS_HSCROLL;
if Kind = sbVertical then Style:= WS_VSCROLL;
Result:= (Visible) and
          (GetWindowLong(FControl.Handle, GWL_STYLE) and Style <> 0);
end;

function TControlCartRider.ControlSize(ControlSB, AssumeSB: Boolean): Integer;
var
BorderAdjust: Integer;

function ScrollBarVisible(Code: Word): Boolean;
var
  Style: Longint;
begin
  Style:= WS_HSCROLL;
  if Code = SB_VERT then Style:= WS_VSCROLL;
  Result:= GetWindowLong(FControl.Handle, GWL_STYLE) and Style <> 0;
end;

function Adjustment(Code, Metric: Word): Integer;
begin
  Result:= 0;
  if not ControlSB then
    if AssumeSB and not ScrollBarVisible(Code) then
      Result:= -(GetSystemMetrics(Metric) - BorderAdjust)
    else if not AssumeSB and ScrollBarVisible(Code) then
      Result:= GetSystemMetrics(Metric) - BorderAdjust;
end;

begin
BorderAdjust:= Integer(GetWindowLong(FControl.Handle, GWL_STYLE) and
  (WS_BORDER or WS_THICKFRAME) <> 0);
if Kind = sbVertical then
  Result:= FControl.ClientHeight + Adjustment(SB_HORZ, SM_CXHSCROLL) else
  Result:= FControl.ClientWidth + Adjustment(SB_VERT, SM_CYVSCROLL);
end;

function TControlCartRider.GetScrollPos: Integer;
begin
Result:= 0;
if Visible then Result:= Position;
end;

function TControlCartRider.NeedsScrollBarVisible: Boolean;
begin
Result:= FRange > ControlSize(False, False);
end;

procedure TControlCartRider.ScrollMessage(var Msg: TWMScroll);
var
Incr, FinalIncr, Count: Integer;
CurrentTime, StartTime, ElapsedTime: Longint;

function GetRealScrollPosition: Integer;
var
  SI: TScrollInfo;
  Code: Integer;
begin
  SI.cbSize:= SizeOf(TScrollInfo);
  SI.fMask:= SIF_TRACKPOS;
```

```
  Code:= SB_HORZ;
  if FKind = sbVertical then Code:= SB_VERT;
  Result:= Msg.Pos;
  if FlatSB_GetScrollInfo(FControl.Handle, Code, SI) then
    Result:= SI.nTrackPos;
end;

begin
with Msg do
begin
  if FSmooth and (ScrollCode in [SB_LINEUP, SB_LINEDOWN, SB_PAGEUP,
SB_PAGEDOWN]) then
  begin
    case ScrollCode of
      SB_LINEUP, SB_LINEDOWN:
        begin
          Incr:= FIncrement div FLineDiv;
          FinalIncr:= FIncrement mod FLineDiv;
          Count:= FLineDiv;
        end;
      SB_PAGEUP, SB_PAGEDOWN:
        begin
          Incr:= FPageIncrement;
          FinalIncr:= Incr mod FPageDiv;
          Incr:= Incr div FPageDiv;
          Count:= FPageDiv;
        end;
    else
      Count:= 0;
      Incr:= 0;
      FinalIncr:= 0;
    end;
    CurrentTime:= 0;
    while Count > 0 do
    begin
      StartTime:= GetCurrentTime;
      ElapsedTime:= StartTime - CurrentTime;
      if ElapsedTime < FDelay then Sleep(FDelay - ElapsedTime);
      CurrentTime:= StartTime;
      case ScrollCode of
        SB_LINEUP: SetPosition(FPosition - Incr);
        SB_LINEDOWN: SetPosition(FPosition + Incr);
        SB_PAGEUP: SetPosition(FPosition - Incr);
        SB_PAGEDOWN: SetPosition(FPosition + Incr);
      end;
      FControl.Update;
      Dec(Count);
    end;
    if FinalIncr > 0 then
    begin
      case ScrollCode of
        SB_LINEUP: SetPosition(FPosition - FinalIncr);
        SB_LINEDOWN: SetPosition(FPosition + FinalIncr);
        SB_PAGEUP: SetPosition(FPosition - FinalIncr);
        SB_PAGEDOWN: SetPosition(FPosition + FinalIncr);
      end;
    end;
  end
  else
    case ScrollCode of
      SB_LINEUP: SetPosition(FPosition - FIncrement);
      SB_LINEDOWN: SetPosition(FPosition + FIncrement);
      SB_PAGEUP: SetPosition(FPosition - ControlSize(True, False));
      SB_PAGEDOWN: SetPosition(FPosition + ControlSize(True, False));
      SB_THUMBPOSITION:
          if FCalcRange > 32767 then
            SetPosition(GetRealScrollPosition) else
            SetPosition(Pos);
      SB_THUMBTRACK:
```

```
        if Tracking then
          if FCalcRange > 32767 then
            SetPosition(GetRealScrollPosition) else
            SetPosition(Pos);
      SB_TOP: SetPosition(0);
      SB_BOTTOM: SetPosition(FCalcRange);
      SB_ENDSCROLL: begin end;
    end;
end;
end;

procedure TControlCartRider.SetButtonSize(Value: Integer);
const
SysConsts: array[TScrollBarKind] of Integer = (SM_CXHSCROLL, SM_CXVSCROLL);
var
NewValue: Integer;
begin
if Value <> ButtonSize then
begin
  NewValue:= Value;
  if NewValue = 0 then
    Value:= GetSystemMetrics(SysConsts[Kind]);
  FButtonSize:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
  if NewValue = 0 then
    FButtonSize:= 0;
end;
end;

procedure TControlCartRider.SetColor(Value: TColor);
begin
if Value <> Color then
begin
  FColor:= Value;
  FParentColor:= False;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
end;
end;

procedure TControlCartRider.SetParentColor(Value: Boolean);
begin
if ParentColor <> Value then
begin
  FParentColor:= Value;
  if Value then Color:= clBtnHighlight;
end;
end;

procedure TControlCartRider.SetPosition(Value: Integer);
var
Code: Word;
Form: TCustomForm;
OldPos: Integer;
begin
if csReading in FControl.ComponentState then
  FPosition:= Value
else
begin
  if Value > FCalcRange then Value:= FCalcRange
  else if Value < 0 then Value:= 0;
  if Kind = sbHorizontal then
    Code:= SB_HORZ else
    Code:= SB_VERT;
  if Value <> FPosition then
  begin
    OldPos:= FPosition;
    FPosition:= Value;
```

```
    if Kind = sbHorizontal then
      FControl.ScrollBy(OldPos - Value, 0) else
      FControl.ScrollBy(0, OldPos - Value);
    if csDesigning in FControl.ComponentState then
    begin
      Form:= GetParentForm(FControl);
      if (Form <> nil) and (Form.Designer <> nil) then Form.Designer.Modified;
    end;
  end;
  if FlatSB_GetScrollPos(FControl.Handle, Code) <> FPosition then
    FlatSB_SetScrollPos(FControl.Handle, Code, FPosition, True);
end;
end;

procedure TControlCartRider.SetSize(Value: Integer);
const
SysConsts: array[TScrollBarKind] of Integer = (SM_CYHSCROLL, SM_CYVSCROLL);
var
NewValue: Integer;
begin
if Value <> Size then
begin
  NewValue:= Value;
  if NewValue = 0 then
    Value:= GetSystemMetrics(SysConsts[Kind]);
  FSize:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
  if NewValue = 0 then
    FSize:= 0;
end;
end;

procedure TControlCartRider.SetStyle(Value: TScrollBarStyle);
begin
if Style <> Value then
begin
  FStyle:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
end;
end;

procedure TControlCartRider.SetThumbSize(Value: Integer);
begin
if Value <> ThumbSize then
begin
  FThumbSize:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
end;
end;

procedure TControlCartRider.DoSetRange(Value: Integer);
begin
FRange:= Value;
if FRange < 0 then FRange:= 0;
FControl.UpdateScrollBars;
end;

procedure TControlCartRider.SetRange(Value: Integer);
begin
FControl.FAutoScroll:= False;
FScaled:= True;
DoSetRange(Value);
end;

function TControlCartRider.IsRangeStored: Boolean;
begin
```

```
Result:= not FControl.AutoScroll;
end;

procedure TControlCartRider.SetVisible(Value: Boolean);
begin
FVisible:= Value;
FControl.UpdateScrollBars;
end;

procedure TControlCartRider.Update(ControlSB, AssumeSB: Boolean);
type
TPropKind = (pkStyle, pkButtonSize, pkThumbSize, pkSize, pkBkColor);
var
Code: Word;
ScrollInfo: TScrollInfo;
begin
FCalcRange:= 0;
Code:= SB_HORZ;
if Kind = sbVertical then Code:= SB_VERT;
if Visible then
begin
  FCalcRange:= Range - ControlSize(ControlSB, AssumeSB);
  if FCalcRange < 0 then FCalcRange:= 0;
end;
ScrollInfo.cbSize:= SizeOf(ScrollInfo);
ScrollInfo.fMask:= SIF_ALL;
ScrollInfo.nMin:= 0;
if FCalcRange > 0 then
  ScrollInfo.nMax:= Range else
  ScrollInfo.nMax:= 0;
ScrollInfo.nPage:= ControlSize(ControlSB, AssumeSB) + 1;
ScrollInfo.nPos:= FPosition;
ScrollInfo.nTrackPos:= FPosition;
UpdateScrollProperties(FUpdateNeeded);
FUpdateNeeded:= False;
FlatSB_SetScrollInfo(FControl.Handle, Code, ScrollInfo, True);
SetPosition(FPosition);
FPageIncrement:= (ControlSize(True, False) * 9) div 10;
if Smooth then FIncrement:= FPageIncrement div 10;
end;

constructor TScrollingKartRider.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
ControlStyle:= ControlStyle + [csNeedsBorderPaint];
FHorzScrollBar:= TControlCartRider.Create(Self, sbHorizontal);
FVertScrollBar:= TControlCartRider.Create(Self, sbVertical);
FAutoScroll:= True;
end;

destructor TScrollingKartRider.Destroy;
begin
FHorzScrollBar.Free;
FVertScrollBar.Free;
inherited Destroy;
end;

procedure TScrollingKartRider.CreateParams(var Params: TCreateParams);
begin
inherited CreateParams(Params);
with Params.WindowClass do
  style:= style and not (CS_HREDRAW or CS_VREDRAW);
end;

procedure TScrollingKartRider.CreateWnd;
begin
inherited CreateWnd;
if not SysLocale.MiddleEast and
   not CheckWin32Version(5, 1) then
```

```
  InitializeFlatSB(Handle);
UpdateScrollBars;
end;

procedure TScrollingKartRider.AlignControls(AControl: TControl; var ARect:
TRect);
begin
CalcAutoRange;
inherited AlignControls(AControl, ARect);
end;

function TScrollingKartRider.AutoScrollEnabled: Boolean;
begin
Result:= not AutoSize and not (DockSite and UseDockManager);
end;

procedure TScrollingKartRider.DoFlipChildren;
var
Loop: Integer;
TheWidth: Integer;
ScrollBarActive: Boolean;
FlippedList: TList;
begin
FlippedList:= TList.Create;
try
  TheWidth:= ClientWidth;
  with HorzScrollBar do begin
    ScrollBarActive:= (IsScrollBarVisible) and (TheWidth < Range);
    if ScrollBarActive then
    begin
      TheWidth:= Range;
      Position:= 0;
    end;
  end;

  for Loop:= 0 to ControlCount - 1 do with Controls[Loop] do
  begin
    FlippedList.Add(Controls[Loop]);
    Left:= TheWidth - Width - Left;
  end;
  for Loop:= 0 to FlippedList.Count - 1 do
    TControl(FlippedList[Loop]).Perform(CM_ALLCHILDRENFLIPPED, 0, 0);
  if ScrollBarActive then
    HorzScrollBar.ChangeBiDiPosition;
finally
   FlippedList.Free;
end;
end;

procedure TScrollingKartRider.CalcAutoRange;
begin
if FAutoRangeCount <= 0 then
begin
  HorzScrollBar.CalcAutoRange;
  VertScrollBar.CalcAutoRange;
end;
end;

procedure TScrollingKartRider.SetAutoScroll(Value: Boolean);
begin
if FAutoScroll <> Value then
begin
  FAutoScroll:= Value;
  if Value then CalcAutoRange else
  begin
    HorzScrollBar.Range:= 0;
    VertScrollBar.Range:= 0;
  end;
end;
```

```
end;

procedure TScrollingKartRider.SetHorzScrollBar(Value: TControlScrollBar);
begin
FHorzScrollBar.Assign(Value);
end;

procedure TScrollingKartRider.SetVertScrollBar(Value: TControlScrollBar);
begin
FVertScrollBar.Assign(Value);
end;

procedure TScrollingKartRider.UpdateScrollBars;
begin
if not FUpdatingScrollBars and HandleAllocated then
  try
    FUpdatingScrollBars:= True;
    if FVertScrollBar.NeedsScrollBarVisible then
    begin
      FHorzScrollBar.Update(False, True);
      FVertScrollBar.Update(True, False);
    end
    else if FHorzScrollBar.NeedsScrollBarVisible then
    begin
      FVertScrollBar.Update(False, True);
      FHorzScrollBar.Update(True, False);
    end
    else
    begin
      FVertScrollBar.Update(False, False);
      FHorzScrollBar.Update(True, False);
    end;
  finally
    FUpdatingScrollBars:= False;
  end;
end;

procedure TScrollingKartRider.AutoScrollInView(AControl: TControl);
begin
if (AControl <> nil) and not (csLoading in AControl.ComponentState) and
  not (csLoading in ComponentState) then
  ScrollInView(AControl);
end;

procedure TScrollingKartRider.DisableAutoRange;
begin
Inc(FAutoRangeCount);
end;

procedure TScrollingKartRider.EnableAutoRange;
begin
if FAutoRangeCount > 0 then
begin
  Dec(FAutoRangeCount);
  if (FAutoRangeCount = 0) and (FHorzScrollBar.Visible or
    FVertScrollBar.Visible) then CalcAutoRange;
end;
end;

procedure TScrollingKartRider.ScrollInView(AControl: TControl);
var
Rect: TRect;
begin
if AControl = nil then Exit;
Rect:= AControl.ClientRect;
Dec(Rect.Left, HorzScrollBar.Margin);
Inc(Rect.Right, HorzScrollBar.Margin);
Dec(Rect.Top, VertScrollBar.Margin);
Inc(Rect.Bottom, VertScrollBar.Margin);
```

```
Rect.TopLeft:= ScreenToClient(AControl.ClientToScreen(Rect.TopLeft));
Rect.BottomRight:= ScreenToClient(AControl.ClientToScreen(Rect.BottomRight));
if Rect.Left < 0 then
  with HorzScrollBar do Position:= Position + Rect.Left
else if Rect.Right > ClientWidth then
begin
  if Rect.Right - Rect.Left > ClientWidth then
    Rect.Right:= Rect.Left + ClientWidth;
  with HorzScrollBar do Position:= Position + Rect.Right - ClientWidth;
end;
if Rect.Top < 0 then
  with VertScrollBar do Position:= Position + Rect.Top
else if Rect.Bottom > ClientHeight then
begin
  if Rect.Bottom - Rect.Top > ClientHeight then
    Rect.Bottom:= Rect.Top + ClientHeight;
  with VertScrollBar do Position:= Position + Rect.Bottom - ClientHeight;
end;
end;

procedure TScrollingKartRider.ScaleScrollBars(M, D: Integer);
begin
if M <> D then
begin
  if not (csLoading in ComponentState) then
  begin
    HorzScrollBar.FScaled:= True;
    VertScrollBar.FScaled:= True;
  end;
  HorzScrollBar.Position:= 0;
  VertScrollBar.Position:= 0;
  if not FAutoScroll then
  begin
    with HorzScrollBar do if FScaled then Range:= MulDiv(Range, M, D);
    with VertScrollBar do if FScaled then Range:= MulDiv(Range, M, D);
  end;
end;
HorzScrollBar.FScaled:= False;
VertScrollBar.FScaled:= False;
end;

procedure TScrollingKartRider.ChangeScale(M, D: Integer);
begin
ScaleScrollBars(M, D);
inherited ChangeScale(M, D);
end;

procedure TScrollingKartRider.WMSize(var Message: TWMSize);
var
NewState: TWindowState;
begin
Inc(FAutoRangeCount);
try
  inherited;
  NewState:= wsNormal;
  case Message.SizeType of
    SIZENORMAL: NewState:= wsNormal;
    SIZEICONIC: NewState:= wsMinimized;
    SIZEFULLSCREEN: NewState:= wsMaximized;
  end;
  Resizing(NewState);
finally
  Dec(FAutoRangeCount);
end;
FUpdatingScrollBars:= True;
try
  CalcAutoRange;
finally
  FUpdatingScrollBars:= False;
```

```
end;
if FHorzScrollBar.Visible or FVertScrollBar.Visible then
  UpdateScrollBars;
end;

procedure TScrollingKartRider.WMHScroll(var Message: TWMHScroll);
begin
if (Message.ScrollBar = 0) and FHorzScrollBar.Visible then
  FHorzScrollBar.ScrollMessage(Message) else
  inherited;
end;

procedure TScrollingKartRider.WMVScroll(var Message: TWMVScroll);
begin
if (Message.ScrollBar = 0) and FVertScrollBar.Visible then
  FVertScrollBar.ScrollMessage(Message) else
  inherited;
end;

procedure TScrollingKartRider.AdjustClientRect(var Rect: TRect);
begin
Rect:= Bounds(-HorzScrollBar.Position, -VertScrollBar.Position,
  Max(HorzScrollBar.Range, ClientWidth), Max(ClientHeight,
  VertScrollBar.Range));
inherited AdjustClientRect(Rect);
end;

procedure TScrollingKartRider.CMBiDiModeChanged(var Message: TMessage);
var
Save: Integer;
begin
Save:= Message.WParam;
try
  if not (Self is TScrollBox) then Message.wParam:= 1;
  inherited;
finally
  Message.wParam:= Save;
end;
if HandleAllocated then
begin
  HorzScrollBar.ChangeBiDiPosition;
  UpdateScrollBars;
end;
end;

constructor TBox.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
ControlStyle:= [csAcceptsControls, csCaptureMouse, csClickEvents,
  csSetCaption, csDoubleClicks];
Width:= 185;
Height:= 41;
FBorderStyle:= bsSingle;
end;

procedure TBox.CreateParams(var Params: TCreateParams);
const
BorderStyles: array[TBorderStyle] of DWORD = (0, WS_BORDER);
begin
inherited CreateParams(Params);
with Params do
begin
  Style:= Style or BorderStyles[FBorderStyle];
  if NewStyleControls and Ctl3D and (FBorderStyle = bsSingle) then
  begin
    Style:= Style and not WS_BORDER;
    ExStyle:= ExStyle or WS_EX_CLIENTEDGE;
  end;
end;
```

```
end;

procedure TBox.SetBorderStyle(Value: TBorderStyle);
begin
if Value <> FBorderStyle then
begin
  FBorderStyle:= Value;
  RecreateWnd;
end;
end;

procedure TBox.WMNCHitTest(var Message: TMessage);
begin
DefaultHandler(Message);
end;

procedure TBox.CMCtl3DChanged(var Message: TMessage);
begin
if NewStyleControls and (FBorderStyle = bsSingle) then RecreateWnd;
inherited;
end;

constructor TCustom.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
ControlStyle:= [csAcceptsControls, csCaptureMouse, csClickEvents,
  csSetCaption, csDoubleClicks, csParentBackground];
if (ClassType <> TFrame) and not (csDesignInstance in ComponentState) then
begin
  if not InitInheritedComponent(Self, TFrame) then
    raise EResNotFound.CreateFmt(SResNotFound, [ClassName]);
end
else
begin
  Width:= 320;
  Height:= 240;
end;
end;

procedure TCustom.CreateParams(var Params: TCreateParams);
begin
inherited;
if Parent = nil then
  Params.WndParent:= Application.Handle;
end;

procedure TCustom.GetChildren(Proc: TGetChildProc; Root: TComponent);
var
I: Integer;
OwnedComponent: TComponent;
begin
inherited GetChildren(Proc, Root);
if Root = Self then
  for I:= 0 to ComponentCount - 1 do
  begin
    OwnedComponent:= Components[I];
    if not OwnedComponent.HasParent then Proc(OwnedComponent);
  end;
end;

procedure TCustom.AddActionList(ActionList: TCustomActionList);
var
Form: TCustomForm;
begin
Form:= GetParentForm(Self);
if Form <> nil then
begin
  if Form.FActionLists = nil then Form.FActionLists:= TList.Create;
  Form.FActionLists.Add(ActionList);
```

```
end;
end;

procedure TCustom.RemoveActionList(ActionList: TCustomActionList);
var
Form: TCustomForm;
begin
Form:= GetParentForm(Self);
if (Form <> nil) and (Form.FActionLists <> nil) then
  Form.FActionLists.Remove(ActionList);
end;

procedure TCustom.Notification(AComponent: TComponent;
Operation: TOperation);
begin
inherited;
case Operation of
  opInsert:
    if AComponent is TCustomActionList then
      AddActionList(TCustomActionList(AComponent));
  opRemove:
    if AComponent is TCustomActionList then
      RemoveActionList(TCustomActionList(AComponent));
end;
end;

procedure TCustom.SetParent(AParent: TWinControl);

procedure UpdateActionLists(Operation: TOperation);
var
  I: Integer;
  Component: TComponent;
begin
  for I:= 0 to ComponentCount - 1 do
  begin
    Component:= Components[I];
    if Component is TCustomActionList then
      case Operation of
        opInsert: AddActionList(TCustomActionList(Component));
        opRemove: RemoveActionList(TCustomActionList(Component));
      end;
  end;
end;

begin
if Parent <> nil then UpdateActionLists(opRemove);
if (Parent = nil) and HandleAllocated then
  DestroyHandle;
inherited;
if Parent <> nil then UpdateActionLists(opInsert);
end;

constructor TCustomActiveRider.Create(AOwner: TComponent);
begin
FAxBorderStyle:= afbSingle;
inherited Create(AOwner);
BorderStyle:= bsNone;
BorderIcons:= [];
TabStop:= True;
end;

procedure TCustomActiveRider.SetAxBorderStyle(Value: TActiveFormBorderStyle);
begin
if FAxBorderStyle <> Value then
begin
  FAxBorderStyle:= Value;
  if not (csDesigning in ComponentState) then RecreateWnd;
end;
end;
```

```
procedure TCustomActiveRider.CreateParams(var Params: TCreateParams);
begin
inherited CreateParams(Params);
if not (csDesigning in ComponentState) then
  with Params do
  begin
    Style:= Style and not WS_CAPTION;
    case FAxBorderStyle of
      afbNone: ;
      afbSingle: Style:= Style or WS_BORDER;
      afbSunken: ExStyle:= ExStyle or WS_EX_CLIENTEDGE;
      afbRaised:
        begin
          Style:= Style or WS_DLGFRAME;
          ExStyle:= ExStyle or WS_EX_WINDOWEDGE;
        end;
    end;
  end;
end;

function TCustomActiveRider.WantChildKey(Child: TControl; var Message:
TMessage): Boolean;
begin
Result:= ((Message.Msg = WM_CHAR) and (Message.WParam = VK_TAB)) or
  (Child.Perform(CN_BASE + Message.Msg, Message.WParam,
    Message.LParam) <> 0);
end;

constructor TCustomForm.Create(AOwner: TComponent);
begin
GlobalNameSpace.BeginWrite;
try
  CreateNew(AOwner);
  if (ClassType <> TForm) and not (csDesigning in ComponentState) then
  begin
    Include(FFormState, fsCreating);
    try
      if not InitInheritedComponent(Self, TForm) then
        raise EResNotFound.CreateFmt(SResNotFound, [ClassName]);
    finally
      Exclude(FFormState, fsCreating);
    end;
    if OldCreateOrder then DoCreate;
  end;
finally
  GlobalNameSpace.EndWrite;
end;
end;

procedure TCustomForm.AfterConstruction;
begin
if not OldCreateOrder then DoCreate;
if fsActivated in FFormState then
begin
  Activate;
  Exclude(FFormState, fsActivated);
end;
end;

constructor TCustomForm.CreateNew(AOwner: TComponent; Dummy: Integer);
begin
inherited Create(AOwner);
ControlStyle:= [csAcceptsControls, csCaptureMouse, csClickEvents,
  csSetCaption, csDoubleClicks];
Left:= 0;
Top:= 0;
Width:= 320;
Height:= 240;
```

```
FIcon:= TIcon.Create;
FIcon.Width:= GetSystemMetrics(SM_CXSMICON);
FIcon.Height:= GetSystemMetrics(SM_CYSMICON);
FIcon.OnChange:= IconChanged;
FCanvas:= TControlCanvas.Create;
FCanvas.Control:= Self;
FBorderIcons:= [biSystemMenu, biMinimize, biMaximize];
FBorderStyle:= bsSizeable;
FWindowState:= wsNormal;
FDefaultMonitor:= dmActiveForm;
FInCMParentBiDiModeChanged:= False;
FPixelsPerInch:= Screen.PixelsPerInch;
FPrintScale:= poProportional;
FloatingDockSiteClass:= TWinControlClass(ClassType);
FAlphaBlendValue:= 255;
FTransparentColorValue:= 0;
Visible:= False;
ParentColor:= False;
ParentFont:= False;
Ctl3D:= True;
Screen.AddForm(Self);
FSnapBuffer:= 10;
end;

function TScrKartRider.GetDefaultIme: String;
begin
GetImes;
Result:= FDefaultIme;
end;

procedure TScrKartRider.IconFontChanged(Sender: TObject);
begin
Application.NotifyForms(CM_SYSFONTCHANGED);
if (Sender = FHintFont) and Assigned(Application) and Application.ShowHint then
begin
  Application.ShowHint:= False;
  Application.ShowHint:= True;
end;
end;

function TScrKartRider.GetDataModule(Index: Integer): TDataModule;
begin
Result:= FDataModules[Index];
end;

function TScrKartRider.GetDataModuleCount: Integer;
begin
Result:= FDataModules.Count;
end;

function TScrKartRider.GetCursors(Index: Integer): HCURSOR;
var
P: PCursorRec;
begin
Result:= 0;
if Index <> crNone then
begin
  P:= FCursorList;
  while (P <> nil) and (P^.Index <> Index) do P:= P^.Next;
  if P = nil then Result:= FDefaultCursor else Result:= P^.Handle;
end;
end;

procedure TScrKartRider.SetCursor(Value: TCursor);
var
P: TPoint;
Handle: HWND;
Code: Longint;
begin
```

```
if Value <> Cursor then
begin
  FCursor:= Value;
  if Value = crDefault then
  begin
    GetCursorPos(P);
    Handle:= WindowFromPoint(P);
    if (Handle <> 0) and
      (GetWindowThreadProcessId(Handle, nil) = GetCurrentThreadId) then
    begin
      Code:= SendMessage(Handle, WM_NCHITTEST, 0,
LongInt(PointToSmallPoint(P)));
      SendMessage(Handle, WM_SETCURSOR, Handle, MakeLong(Code, WM_MOUSEMOVE));
      Exit;
    end;
  end;
  Windows.SetCursor(Cursors[Value]);
end;
Inc(FCursorCount);
end;

procedure TScrKartRider.SetCursors(Index: Integer; Handle: HCURSOR);
begin
if Index = crDefault then
  if Handle = 0 then
    FDefaultCursor:= LoadCursor(0, IDC_ARROW)
  else
    FDefaultCursor:= Handle
else if Index <> crNone then
begin
  DeleteCursor(Index);
  if Handle <> 0 then InsertCursor(Index, Handle);
end;
end;

procedure TScrKartRider.SetHintFont(Value: TFont);
begin
FHintFont.Assign(Value);
end;

procedure TScrKartRider.SetIconFont(Value: TFont);
begin
FIconFont.Assign(Value);
end;

procedure TScrKartRider.SetMenuFont(Value: TFont);
begin
FMenuFont.Assign(Value);
end;

procedure TScrKartRider.GetMetricSettings;
var
LogFont: TLogFont;
NonClientMetrics: TNonClientMetrics;
SaveShowHint: Boolean;
begin
SaveShowHint:= False;
if Assigned(Application) then SaveShowHint:= Application.ShowHint;
try
  if Assigned(Application) then Application.ShowHint:= False;
  if SystemParametersInfo(SPI_GETICONTITLELOGFONT, SizeOf(LogFont), @LogFont, 0)
then
    FIconFont.Handle:= CreateFontIndirect(LogFont)
  else
    FIconFont.Handle:= GetStockObject(SYSTEM_FONT);
  NonClientMetrics.cbSize:= SizeOf(NonClientMetrics);
  if SystemParametersInfo(SPI_GETNONCLIENTMETRICS, 0, @NonClientMetrics, 0) then
  begin
    FHintFont.Handle:= CreateFontIndirect(NonClientMetrics.lfStatusFont);
```

```
    FMenuFont.Handle:= CreateFontIndirect(NonClientMetrics.lfMenuFont);
  end else
  begin
    FHintFont.Size:= 8;
    FMenuFont.Handle:= GetStockObject(SYSTEM_FONT);
  end;
  FHintFont.Color:= clInfoText;
  FMenuFont.Color:= clMenuText;
finally
  if Assigned(Application) then Application.ShowHint:= SaveShowHint;
end;
end;

procedure TScrKartRider.DisableAlign;
begin
Inc(FAlignLevel);
end;

procedure TScrKartRider.EnableAlign;
begin
Dec(FAlignLevel);
if (FAlignLevel = 0) and (csAlignmentNeeded in FControlState) then Realign;
end;

procedure TScrKartRider.Realign;
begin
AlignForm(nil);
end;

procedure TScrKartRider.AlignForms(AForm: TCustomForm; var Rect: TRect);
var
AlignList: TList;

function InsertBefore(C1, C2: TCustomForm; AAlign: TAlign): Boolean;
begin
  Result:= False;
  case AAlign of
    alTop: Result:= C1.Top < C2.Top;
    alBottom: Result:= (C1.Top + C1.Height) > (C2.Top + C2.Height);
    alLeft: Result:= C1.Left < C2.Left;
    alRight: Result:= (C1.Left + C1.Width) > (C2.Left + C2.Width);
  end;
end;

procedure DoPosition(Form: TCustomForm; AAlign: TAlign);
var
  NewLeft, NewTop, NewWidth, NewHeight: Integer;
begin
  with Rect do
  begin
    NewWidth:= Right - Left;
    if (NewWidth < 0) or (AAlign in [alLeft, alRight]) then
      NewWidth:= Form.Width;
    NewHeight:= Bottom - Top;
    if (NewHeight < 0) or (AAlign in [alTop, alBottom]) then
      NewHeight:= Form.Height;
    if (AAlign = alTop) and (Form.WindowState = wsMaximized) then
    begin
      NewLeft:= Form.Left;
      NewTop:= Form.Top;
      NewWidth:= GetSystemMetrics(SM_CXMAXIMIZED);
    end
    else
    begin
      NewLeft:= Left;
      NewTop:= Top;
    end;
    case AAlign of
      alTop: Inc(Top, NewHeight);
```

```
      alBottom:
        begin
          Dec(Bottom, NewHeight);
          NewTop:= Bottom;
        end;
      alLeft: Inc(Left, NewWidth);
      alRight:
        begin
          Dec(Right, NewWidth);
          NewLeft:= Right;
        end;
    end;
  end;
  Form.SetBounds(NewLeft, NewTop, NewWidth, NewHeight);
  if Form.WindowState = wsMaximized then
  begin
    Dec(NewWidth, NewLeft);
    Dec(NewHeight, NewTop);
  end;
  if (Form.Width <> NewWidth) or (Form.Height <> NewHeight) then
    with Rect do
      case AAlign of
        alTop: Dec(Top, NewHeight - Form.Height);
        alBottom: Inc(Bottom, NewHeight - Form.Height);
        alLeft: Dec(Left, NewWidth - Form.Width);
        alRight: Inc(Right, NewWidth - Form.Width);
        alClient:
          begin
            Inc(Right, NewWidth - Form.Width);
            Inc(Bottom, NewHeight - Form.Height);
          end;
      end;
end;

procedure DoAlign(AAlign: TAlign);
var
  I, J: Integer;
  Form: TCustomForm;
begin
  AlignList.Clear;
  if (AForm <> nil) and (AForm.Parent = nil) and
    not (csDesigning in AForm.ComponentState) and
    AForm.Visible and (AForm.Align = AAlign) and
    (AForm.WindowState <> wsMinimized) then
    AlignList.Add(AForm);
  for I:= 0 to CustomFormCount - 1 do
  begin
    Form:= TCustomForm(CustomForms[I]);
    if (Form.Parent = nil) and (Form.Align = AAlign) and
      not (csDesigning in Form.ComponentState) and
      Form.Visible and (Form.WindowState <> wsMinimized) then
    begin
      if Form = AForm then Continue;
      J:= 0;
      while (J < AlignList.Count) and not InsertBefore(Form,
        TCustomForm(AlignList[J]), AAlign) do Inc(J);
      AlignList.Insert(J, Form);
    end;
  end;
  for I:= 0 to AlignList.Count - 1 do
    DoPosition(TCustomForm(AlignList[I]), AAlign);
end;

function AlignWork: Boolean;
var
  I: Integer;
begin
  Result:= True;
  for I:= CustomFormCount - 1 downto 0 do
```

```
    with TCustomForm(CustomForms[I]) do
      if (Parent = nil) and not (csDesigning in ComponentState) and
        (Align <> alNone) and Visible and (WindowState <> wsMinimized) then
Exit;
  Result:= False;
end;
begin
if AlignWork then
begin
  AlignList:= TList.Create;
  try
    DoAlign(alTop);
    DoAlign(alBottom);
    DoAlign(alLeft);
    DoAlign(alRight);
    DoAlign(alClient);
  finally
    AlignList.Free;
  end;
end;
end;

procedure TScrKartRider.AlignForm(AForm: TCustomForm);
var
Rect: TRect;
begin
if FAlignLevel <> 0 then
  Include(FControlState, csAlignmentNeeded)
else
begin
  DisableAlign;
  try
    SystemParametersInfo(SPI_GETWORKAREA, 0, @Rect, 0);
    AlignForms(AForm, Rect);
  finally
    Exclude(FControlState, csAlignmentNeeded);
    EnableAlign;
  end;
end;
end;

function TScrKartRider.GetFonts: TStrings;
var
DC: HDC;
LFont: TLogFont;
begin
if FFonts = nil then
begin
  FFonts:= TStringList.Create;
  DC:= GetDC(0);
  try
    FFonts.Add('Default');
    FillChar(LFont, sizeof(LFont), 0);
    LFont.lfCharset:= DEFAULT_CHARSET;
    EnumFontFamiliesEx(DC, LFont, @EnumFontsProc, LongInt(FFonts), 0);
    TStringList(FFonts).Sorted:= TRUE;
  finally
    ReleaseDC(0, DC);
  end;
end;
Result:= FFonts;
end;

procedure TScrKartRider.ResetFonts;
begin
FreeAndNil(FFonts);
end;

function GetHint(Control: TControl): string;
```

```
begin
while Control <> nil do
  if Control.Hint = '' then
    Control:= Control.Parent
  else
  begin
    Result:= Control.Hint;
    Exit;
  end;
Result:= '';
end;

function GetHintControl(Control: TControl): TControl;
begin
Result:= Control;
while (Result <> nil) and not Result.ShowHint do Result:= Result.Parent;
if (Result <> nil) and (csDesigning in Result.ComponentState) then Result:= nil;
end;

procedure HintTimerProc(Wnd: HWnd; Msg, TimerID, SysTime: Longint); stdcall;
begin
if Application <> nil then
try
  Application.HintTimerExpired;
except
  Application.HandleException(Application);
end;
end;

var
HintThreadID: DWORD;
HintDoneEvent: THandle;

procedure HintMouseThread(Param: Integer); stdcall;
var
P: TPoint;
begin
HintThreadID:= GetCurrentThreadID;
while WaitForSingleObject(HintDoneEvent, 100) = WAIT_TIMEOUT do
begin
  if (Application <> nil) and (Application.FHintControl <> nil) then
  begin
    GetCursorPos(P);
    if FindVCLWindow(P) = nil then
      Application.CancelHint;
  end;
end;
end;

var
HintHook: HHOOK;
HintThread: THandle;

function HintGetMsgHook(nCode: Integer; wParam: Longint; var Msg: TMsg):
Longint; stdcall;
begin
Result:= CallNextHookEx(HintHook, nCode, wParam, Longint(@Msg));
if (nCode >= 0) and (Application <> nil) then Application.IsHintMsg(Msg);
end;

procedure HookHintHooks;
var
ThreadID: DWORD;
begin
if not Application.FRunning then
begin
  if HintHook = 0 then
    HintHook:= SetWindowsHookEx(WH_GETMESSAGE, @HintGetMsgHook, 0,
GetCurrentThreadID);
```

```
  if HintDoneEvent = 0 then
    HintDoneEvent:= CreateEvent(nil, False, False, nil);
  if HintThread = 0 then
    HintThread:= CreateThread(nil, 1000, @HintMouseThread, nil, 0, ThreadID);
end;
end;

procedure UnhookHintHooks;
begin
if HintHook <> 0 then UnhookWindowsHookEx(HintHook);
HintHook:= 0;
if HintThread <> 0 then
begin
  SetEvent(HintDoneEvent);
  if GetCurrentThreadId <> HintThreadID then
    WaitForSingleObject(HintThread, INFINITE);
  CloseHandle(HintThread);
  HintThread:= 0;
end;
end;

function GetAnimation: Boolean;
var
Info: TAnimationInfo;
begin
Info.cbSize:= SizeOf(TAnimationInfo);
if SystemParametersInfo(SPI_GETANIMATION, SizeOf(Info), @Info, 0) then
  Result:= Info.iMinAnimate <> 0 else
  Result:= False;
end;

procedure SetAnimation(Value: Boolean);
var
Info: TAnimationInfo;
begin
Info.cbSize:= SizeOf(TAnimationInfo);
BOOL(Info.iMinAnimate):= Value;
SystemParametersInfo(SPI_SETANIMATION, SizeOf(Info), @Info, 0);
end;

procedure ShowWinNoAnimate(Handle: HWnd; CmdShow: Integer);
var
Animation: Boolean;
begin
Animation:= GetAnimation;
if Animation then SetAnimation(False);
ShowWindow(Handle, CmdShow);
if Animation then SetAnimation(True);
end;

function TScrKartRider.GetDesktopRect: TRect;
begin
Result:= Bounds(DesktopLeft, DesktopTop, DesktopWidth, DesktopHeight);
end;

function TScrKartRider.GetWorkAreaHeight: Integer;
begin
with WorkAreaRect do
  Result:= Bottom - Top;
end;

function TScrKartRider.GetWorkAreaLeft: Integer;
begin
Result:= WorkAreaRect.Left;
end;

function TScrKartRider.GetWorkAreaRect: TRect;
begin
SystemParametersInfo(SPI_GETWORKAREA, 0, @Result, 0);
```

```
end;

function TScrKartRider.GetWorkAreaTop: Integer;
begin
Result:= WorkAreaRect.Top;
end;

function TScrKartRider.GetWorkAreaWidth: Integer;
begin
with WorkAreaRect do
  Result:= Right - Left;
end;

const
MonitorDefaultFlags: array[TMonitorDefaultTo] of DWORD =
(MONITOR_DEFAULTTONEAREST,
  MONITOR_DEFAULTTONULL, MONITOR_DEFAULTTOPRIMARY);

function TScrKartRider.MonitorFromPoint(const Point: TPoint;
MonitorDefault: TMonitorDefaultTo): TMonitor;
begin
Result:= FindMonitor(MultiMon.MonitorFromPoint(Point,
  MonitorDefaultFlags[MonitorDefault]));
end;

function TScrKartRider.MonitorFromRect(const Rect: TRect;
MonitorDefault: TMonitorDefaultTo): TMonitor;
begin
Result:= FindMonitor(MultiMon.MonitorFromRect(@Rect,
  MonitorDefaultFlags[MonitorDefault]));
end;

function TScrKartRider.MonitorFromWindow(const Handle: THandle;
MonitorDefault: TMonitorDefaultTo): TMonitor;
begin
Result:= FindMonitor(MultiMon.MonitorFromWindow(Handle,
  MonitorDefaultFlags[MonitorDefault]));
end;

var
WindowClass: TWndClass = (
  style: 0;
  lpfnWndProc: @DefWindowProc;
  cbClsExtra: 0;
  cbWndExtra: 0;
  hInstance: 0;
  hIcon: 0;
  hCursor: 0;
  hbrBackground: 0;
  lpszMenuName: nil;
  lpszClassName: 'TKartRiderApp');

constructor TKartRiderApp.Create(AOwner: TComponent);
var
P: PChar;
ModuleName: array[0..255] of Char;
begin
FTopMostList:= TList.Create;
FWindowHooks:= TList.Create;
FHintControl:= nil;
FHintWindow:= nil;
FHintColor:= DefHintColor;
FHintPause:= DefHintPause;
FHintShortCuts:= True;
FHintShortPause:= DefHintShortPause;
FHintHidePause:= DefHintHidePause;
FShowHint:= False;
FActive:= True;
FAutoDragDocking:= True;
```

```
FIcon:= TIcon.Create;
FIcon.Handle:= LoadIcon(MainInstance, 'MAINICON');
FIcon.OnChange:= IconChanged;
GetModuleFileName(MainInstance, ModuleName, SizeOf(ModuleName));
OemToAnsi(ModuleName, ModuleName);
P:= AnsiStrRScan(ModuleName, '\');
if P <> nil then StrCopy(ModuleName, P + 1);
P:= AnsiStrScan(ModuleName, '.');
if P <> nil then P^:= #0;
AnsiLower(ModuleName + 1);
FTitle:= ModuleName;
if not IsLibrary then CreateHandle;
UpdateFormatSettings:= True;
UpdateMetricSettings:= True;
FShowMainForm:= True;
FAllowTesting:= True;
FTestLib:= 0;
ValidateHelpSystem;
HookSynchronizeWakeup;
end;

destructor TKartRiderApp.Destroy;
type
TExceptionEvent = procedure (E: Exception) of object;
var
P: TNotifyEvent;
E: TExceptionEvent;
begin
UnhookSynchronizeWakeup;
P:= HandleException;
if @P = @Classes.ApplicationHandleException then
  Classes.ApplicationHandleException:= nil;
E:= ShowException;
if @E = @Classes.ApplicationShowException then
  Classes.ApplicationShowException:= nil;
if FTestLib <> 0 then
  FreeLibrary(FTestLib);
FActive:= False;
CancelHint;
ShowHint:= False;
inherited Destroy;
UnhookMainWindow(CheckIniChange);
if (FHandle <> 0) and FHandleCreated then
begin
  if NewStyleControls then SendMessage(FHandle, WM_SETICON, 1, 0);
  DestroyWindow(FHandle);
end;
if FHelpSystem <> nil then FHelpSystem:= nil;
if FObjectInstance <> nil then WinUtils.FreeObjectInstance(FObjectInstance);
if FObjectInstance <> nil then Classes.FreeObjectInstance(FObjectInstance);
FWindowHooks.Free;
FTopMostList.Free;
FIcon.Free;
end;

procedure TKartRiderApp.CreateHandle;
var
TempClass: TWndClass;
SysMenu: HMenu;
begin
if not FHandleCreated
  and not IsConsole then
  then
begin
  FObjectInstance:= WinUtils.MakeObjectInstance(WndProc);
  FObjectInstance:= Classes.MakeObjectInstance(WndProc);
  WindowClass.lpfnWndProc:= @DefWindowProc;
  if not GetClassInfo(HInstance, WindowClass.lpszClassName, TempClass) then
  begin
```

```
    WindowClass.hInstance:= HInstance;
    if Windows.RegisterClass(WindowClass) = 0 then
      raise EOutOfResources.Create(SWindowClass);
  end;
  FHandle:= CreateWindow(WindowClass.lpszClassName, PChar(FTitle),
    WS_POPUP or WS_CAPTION or WS_CLIPSIBLINGS or WS_SYSMENU
    or WS_MINIMIZEBOX,
    GetSystemMetrics(SM_CXSCREEN) div 2,
    GetSystemMetrics(SM_CYSCREEN) div 2,
    0, 0, 0, 0, HInstance, nil);
  FTitle:= '';
  FHandleCreated:= True;
  SetWindowLong(FHandle, GWL_WNDPROC, Longint(FObjectInstance));
  if NewStyleControls then
  begin
    SendMessage(FHandle, WM_SETICON, 1, GetIconHandle);
    SetClassLong(FHandle, GCL_HICON, GetIconHandle);
  end;
  SysMenu:= GetSystemMenu(FHandle, False);
  DeleteMenu(SysMenu, SC_MAXIMIZE, MF_BYCOMMAND);
  DeleteMenu(SysMenu, SC_SIZE, MF_BYCOMMAND);
  if NewStyleControls then DeleteMenu(SysMenu, SC_MOVE, MF_BYCOMMAND);
end;
end;

procedure TKartRiderApp.ControlDestroyed(Control: TControl);
begin
if FMainForm = Control then FMainForm:= nil;
if FMouseControl = Control then FMouseControl:= nil;
if Screen.FActiveControl = Control then Screen.FActiveControl:= nil;
if Screen.FActiveCustomForm = Control then
begin
  Screen.FActiveCustomForm:= nil;
  Screen.FActiveForm:= nil;
end;
if Screen.FFocusedForm = Control then Screen.FFocusedForm:= nil;
if FHintControl = Control then FHintControl:= nil;
Screen.UpdateLastActive;
end;

type
PTopMostEnumInfo = ^TTopMostEnumInfo;
TTopMostEnumInfo = record
  TopWindow: HWND;
  IncludeMain: Boolean;
end;

function GetTopMostWindows(Handle: HWND; Info: Pointer): BOOL; stdcall;
begin
Result:= True;
if GetWindow(Handle, GW_OWNER) = Application.Handle then
  if (GetWindowLong(Handle, GWL_EXSTYLE) and WS_EX_TOPMOST <> 0) and
    ((Application.MainForm = nil) or PTopMostEnumInfo(Info)^.IncludeMain or
    (Handle <> Application.MainForm.Handle)) then
    Application.FTopMostList.Add(Pointer(Handle))
  else
  begin
    PTopMostEnumInfo(Info)^.TopWindow:= Handle;
    Result:= False;
  end;
end;

procedure TKartRiderApp.DoNormalizeTopMosts(IncludeMain: Boolean);
var
I: Integer;
Info: TTopMostEnumInfo;
begin
if Application.Handle <> 0 then
begin
```

```
  if FTopMostLevel = 0 then
  begin
    Info.TopWindow:= Handle;
    Info.IncludeMain:= IncludeMain;
    EnumWindows(@GetTopMostWindows, Longint(@Info));
    if FTopMostList.Count <> 0 then
    begin
      Info.TopWindow:= GetWindow(Info.TopWindow, GW_HWNDPREV);
      if GetWindowLong(Info.TopWindow, GWL_EXSTYLE) and WS_EX_TOPMOST <> 0 then
        Info.TopWindow:= HWND_NOTOPMOST;
      for I:= FTopMostList.Count - 1 downto 0 do
        SetWindowPos(HWND(FTopMostList[I]), Info.TopWindow, 0, 0, 0, 0,
          SWP_NOMOVE or SWP_NOSIZE or SWP_NOACTIVATE or SWP_NOOWNERZORDER);
    end;
  end;
  Inc(FTopMostLevel);
end;
end;

procedure TKartRiderApp.ModalStarted;
begin
Inc(FModalLevel);
if (FModalLevel = 1) and Assigned(FOnModalBegin) then
  FOnModalBegin(Self);
end;

procedure TKartRiderApp.ModalFinished;
begin
Dec(FModalLevel);
if (FModalLevel = 0) and Assigned(FOnModalEnd) then
  FOnModalEnd(Self);
end;

procedure TKartRiderApp.NormalizeTopMosts;
begin
DoNormalizeTopMosts(False);
end;

procedure TKartRiderApp.NormalizeAllTopMosts;
begin
DoNormalizeTopMosts(True);
end;

procedure TKartRiderApp.RestoreTopMosts;
var
I: Integer;
begin
if (Application.Handle <> 0) and (FTopMostLevel > 0) then
begin
  Dec(FTopMostLevel);
  if FTopMostLevel = 0 then
  begin
    for I:= FTopMostList.Count - 1 downto 0 do
      SetWindowPos(HWND(FTopMostList[I]), HWND_TOPMOST, 0, 0, 0, 0,
        SWP_NOMOVE or SWP_NOSIZE or SWP_NOACTIVATE or SWP_NOOWNERZORDER);
    FTopMostList.Clear;
  end;
end;
end;

function TKartRiderApp.IsRightToLeft: Boolean;
begin
Result:= SysLocale.MiddleEast and (FBiDiMode <> bdLeftToRight);
end;

function TKartRiderApp.UseRightToLeftReading: Boolean;
begin
Result:= SysLocale.MiddleEast and (FBiDiMode <> bdLeftToRight);
end;
```

```
function TKartRiderApp.UseRightToLeftAlignment: Boolean;
begin
Result:= SysLocale.MiddleEast and (FBiDiMode = bdRightToLeft);
end;

function TKartRiderApp.UseRightToLeftScrollBar: Boolean;
begin
Result:= SysLocale.MiddleEast and
          (FBiDiMode in [bdRightToLeft, bdRightToLeftNoAlign]);
end;

function TKartRiderApp.CheckIniChange(var Message: TMessage): Boolean;
begin
Result:= False;
if (Message.Msg = RM_TaskbarCreated) or
   (Message.Msg = WM_WININICHANGE) then
begin
  if UpdateFormatSettings then
  begin
    SetThreadLocale(LOCALE_USER_DEFAULT);
    GetFormatSettings;
  end;
  if UpdateMetricSettings then
    Screen.GetMetricSettings;

  if Message.Msg = RM_TaskbarCreated then
  begin
    Screen.ResetFonts;
  end;
end;
end;

procedure TKartRiderApp.SettingChange(var Message: TWMSettingChange);
begin
if Assigned(FOnSettingChange) then
  with Message do
    FOnSettingChange(Self, Flag, Section, Result);
end;


procedure TKartRiderApp.WndProc(var Message: TMessage);
type
TInitTestLibrary = function(Size: DWord; PAutoClassInfo: Pointer): Boolean;
stdcall;

var
I: Integer;
SaveFocus, TopWindow: HWnd;
InitTestLibrary: TInitTestLibrary;

procedure Default;
begin
  with Message do
    Result:= DefWindowProc(FHandle, Msg, WParam, LParam);
end;

procedure DrawAppIcon;
var
  DC: HDC;
  PS: TPaintStruct;
begin
  with Message do
  begin
    DC:= BeginPaint(FHandle, PS);
    DrawIcon(DC, 0, 0, GetIconHandle);
    EndPaint(FHandle, PS);
  end;
end;
```

```
begin
try
  Message.Result:= 0;
  for I:= 0 to FWindowHooks.Count - 1 do
    if TWindowHook(FWindowHooks[I]^)(Message) then Exit;
  CheckIniChange(Message);
  with Message do
    case Msg of
      WM_SYSCOMMAND:
        case WParam and $FFF0 of
          SC_MINIMIZE: Minimize;
          SC_RESTORE: Restore;
        else
          Default;
        end;
      WM_CLOSE:
        if MainForm <> nil then MainForm.Close;
      WM_PAINT:
        if IsIconic(FHandle) then DrawAppIcon else Default;
      WM_ERASEBKGND:
        begin
          Message.Msg:= WM_ICONERASEBKGND;
          Default;
        end;
      WM_QUERYDRAGICON:
        Result:= GetIconHandle;
      WM_SETFOCUS:
        begin
          PostMessage(FHandle, CM_ENTER, 0, 0);
          Default;
        end;
      WM_ACTIVATEAPP:
        begin
          Default;
          FActive:= TWMActivateApp(Message).Active;
          if TWMActivateApp(Message).Active then
          begin
            RestoreTopMosts;
            PostMessage(FHandle, CM_ACTIVATE, 0, 0)
          end
          else
          begin
            NormalizeTopMosts;
            PostMessage(FHandle, CM_DEACTIVATE, 0, 0);
          end;
        end;
      WM_ENABLE:
        if TWMEnable(Message).Enabled then
        begin
          RestoreTopMosts;
          if FWindowList <> nil then
          begin
            EnableTaskWindows(FWindowList);
            FWindowList:= nil;
          end;
          Default;
        end else
        begin
          Default;
          if FWindowList = nil then
            FWindowList:= DisableTaskWindows(Handle);
          NormalizeAllTopMosts;
        end;
      WM_CTLCOLORMSGBOX..WM_CTLCOLORSTATIC:
        Result:= SendMessage(LParam, CN_BASE + Msg, WParam, LParam);
      WM_ENDSESSION: if TWMEndSession(Message).EndSession then FTerminate:=
True;
      WM_COPYDATA:
```

```
          if (PCopyDataStruct(Message.lParam)^.dwData = DWORD($DE534454)) and
            (FAllowTesting) then
            if FTestLib = 0 then
            begin
              if FTestLib <> 0 then
              begin
                Result:= 0;
                @InitTestLibrary:= GetProcAddress(FTestLib, 'RegisterAutomation');
                if @InitTestLibrary <> nil then
                  InitTestLibrary(PCopyDataStruct(Message.lParam)^.cbData,
                    PCopyDataStruct(Message.lParam)^.lpData);
              end
              else
              begin
                Result:= GetLastError;
                FTestLib:= 0;
              end;
            end
            else
              Result:= 0;
        CM_ACTIONEXECUTE, CM_ACTIONUPDATE:
          Message.Result:= Ord(DispatchAction(Message.Msg,
TBasicAction(Message.LParam)));
        CM_APPKEYDOWN:
          if IsShortCut(TWMKey(Message)) then Result:= 1;
        CM_APPSYSCOMMAND:
          if MainForm <> nil then
            with MainForm do
              if (Handle <> 0) and IsWindowEnabled(Handle) and
                IsWindowVisible(Handle) then
              begin
                FocusMessages:= False;
                SaveFocus:= GetFocus;
                Windows.SetFocus(Handle);
                Perform(WM_SYSCOMMAND, WParam, LParam);
                Windows.SetFocus(SaveFocus);
                FocusMessages:= True;
                Result:= 1;
              end;
        CM_ACTIVATE:
          if Assigned(FOnActivate) then FOnActivate(Self);
        CM_DEACTIVATE:
          if Assigned(FOnDeactivate) then FOnDeactivate(Self);
        CM_ENTER:
          if not IsIconic(FHandle) and (GetFocus = FHandle) then
          begin
            TopWindow:= FindTopMostWindow(0);
            if TopWindow <> 0 then Windows.SetFocus(TopWindow);
          end;
        WM_HELP,
        CM_INVOKEHELP: InvokeHelp(WParam, LParam);
        CM_WINDOWHOOK:
          if wParam = 0 then
            HookMainWindow(TWindowHook(Pointer(LParam)^)) else
            UnhookMainWindow(TWindowHook(Pointer(LParam)^));
        CM_DIALOGHANDLE:
          if wParam = 1 then
            Result:= FDialogHandle
          else
            FDialogHandle:= lParam;
        WM_SETTINGCHANGE:
          begin
            Mouse.SettingChanged(wParam);
            SettingChange(TWMSettingChange(Message));
            Default;
          end;
        WM_FONTCHANGE:
          begin
            Screen.ResetFonts;
```

```
          Default;
        end;
      WM_THEMECHANGED:
        if ThemeServices.ThemesEnabled then
          ThemeServices.ApplyThemeChange;
      WM_NULL:
        CheckSynchronize;
    else
      Default;
    end;
except
  HandleException(Self);
end;
end;

function TKartRiderApp.GetIconHandle: HICON;
begin
Result:= FIcon.Handle;
if Result = 0 then Result:= LoadIcon(0, IDI_APPLICATION);
end;

procedure TKartRiderApp.Minimize;
begin
if not IsIconic(FHandle) then
begin
  NormalizeTopMosts;
  SetActiveWindow(FHandle);
  if (MainForm <> nil) and (ShowMainForm or MainForm.Visible)
    and IsWindowEnabled(MainForm.Handle) then
  begin
    SetWindowPos(FHandle, MainForm.Handle, MainForm.Left, MainForm.Top,
      MainForm.Width, 0, SWP_SHOWWINDOW);
    DefWindowProc(FHandle, WM_SYSCOMMAND, SC_MINIMIZE, 0);
  end else
    ShowWinNoAnimate(FHandle, SW_MINIMIZE);
  if Assigned(FOnMinimize) then FOnMinimize(Self);
end;
end;

procedure TKartRiderApp.Restore;
begin
if IsIconic(FHandle) then
begin
  SetActiveWindow(FHandle);
  if (MainForm <> nil) and (ShowMainForm or MainForm.Visible)
    and IsWindowEnabled(MainForm.Handle) then
    DefWindowProc(FHandle, WM_SYSCOMMAND, SC_RESTORE, 0)
  else ShowWinNoAnimate(FHandle, SW_RESTORE);
  SetWindowPos(FHandle, 0, GetSystemMetrics(SM_CXSCREEN) div 2,
    GetSystemMetrics(SM_CYSCREEN) div 2, 0, 0, SWP_SHOWWINDOW);
  if (FMainForm <> nil) and (FMainForm.FWindowState = wsMinimized) and
    not FMainForm.Visible then
  begin
    FMainForm.WindowState:= wsNormal;
    FMainForm.Show;
  end;
  RestoreTopMosts;
  if Screen.ActiveControl <> nil then
    Windows.SetFocus(Screen.ActiveControl.Handle);
  if Assigned(FOnRestore) then FOnRestore(Self);
end;
end;

procedure TKartRiderApp.BringToFront;
var
TopWindow: HWnd;
begin
if Handle <> 0 then
begin
```

```
  TopWindow:= GetLastActivePopup(Handle);
  if (TopWindow <> 0) and (TopWindow <> Handle) and
    IsWindowVisible(TopWindow) and IsWindowEnabled(TopWindow) then
    SetForegroundWindow(TopWindow);
end;
end;

function TKartRiderApp.GetTitle: string;
var
Buffer: array[0..255] of Char;
begin
if FHandleCreated then
  SetString(Result, Buffer, GetWindowText(FHandle, Buffer,
    SizeOf(Buffer))) else
  Result:= FTitle;
end;

procedure TKartRiderApp.SetIcon(Value: TIcon);
begin
FIcon.Assign(Value);
end;

procedure TKartRiderApp.SetBiDiMode(Value: TBiDiMode);
var
Loop: Integer;
begin
if FBiDiMode <> Value then
begin
  FBiDiMode:= Value;
  with Screen do
    for Loop:= 0 to FormCount-1 do
      Forms[Loop].Perform(CM_PARENTBIDIMODECHANGED, 0, 0);
end;
end;

procedure TKartRiderApp.SetTitle(const Value: string);
begin
if FHandleCreated then
begin
  if (GetTitle <> Value) or (FTitle <> '') then
  begin
    SetWindowText(FHandle, PChar(Value));
    FTitle:= '';
  end;
end
else
  FTitle:= Value;
end;

procedure TKartRiderApp.SetHandle(Value: HWnd);
begin
if not FHandleCreated and (Value <> FHandle) then
begin
  if FHandle <> 0 then UnhookMainWindow(CheckIniChange);
  FHandle:= Value;
  if FHandle <> 0 then HookMainWindow(CheckIniChange);
end;
end;

function TKartRiderApp.IsDlgMsg(var Msg: TMsg): Boolean;
begin
Result:= False;
if FDialogHandle <> 0 then
  Result:= IsDialogMessage(FDialogHandle, Msg);
end;

function TKartRiderApp.IsMDIMsg(var Msg: TMsg): Boolean;
begin
Result:= False;
```

```
if (MainForm <> nil) and (MainForm.FormStyle = fsMDIForm) and
   (Screen.ActiveForm <> nil) and (Screen.ActiveForm.FormStyle = fsMDIChild)
then
  Result:= TranslateMDISysAccel(MainForm.ClientHandle, Msg);
end;

function TKartRiderApp.IsKeyMsg(var Msg: TMsg): Boolean;
var
Wnd: HWND;
begin
Result:= False;
with Msg do
  if (Message >= WM_KEYFIRST) and (Message <= WM_KEYLAST) then
  begin
    Wnd:= GetCapture;
    if Wnd = 0 then
    begin
      Wnd:= HWnd;
      if (MainForm <> nil) and (Wnd = MainForm.ClientHandle) then
        Wnd:= MainForm.Handle
      else
      begin
        while (FindControl(Wnd) = nil) and (Wnd <> 0) do
          Wnd:= GetParent(Wnd);
        if Wnd = 0 then Wnd:= HWnd;
      end;
      if SendMessage(Wnd, CN_BASE + Message, WParam, LParam) <> 0 then
        Result:= True;
    end
    else if (LongWord(GetWindowLong(Wnd, GWL_HINSTANCE)) = HInstance) then
    begin
      if SendMessage(Wnd, CN_BASE + Message, WParam, LParam) <> 0 then
        Result:= True;
    end;
  end;
end;

function TKartRiderApp.IsHintMsg(var Msg: TMsg): Boolean;
begin
Result:= False;
if (FHintWindow <> nil) and FHintWindow.IsHintMsg(Msg) then
  CancelHint;
end;

function TKartRiderApp.IsShortCut(var Message: TWMKey): Boolean;
begin
Result:= False;
if Assigned(FOnShortCut) then FOnShortCut(Message, Result);
Result:= Result or (MainForm <> nil) and IsWindowEnabled(MainForm.Handle) and
  MainForm.IsShortCut(TWMKey(Message))
end;

function TKartRiderApp.ProcessMessage(var Msg: TMsg): Boolean;
var
Handled: Boolean;
begin
Result:= False;
if PeekMessage(Msg, 0, 0, 0, PM_REMOVE) then
begin
  Result:= True;
  if Msg.Message <> WM_QUIT then
  begin
    Handled:= False;
    if Assigned(FOnMessage) then FOnMessage(Msg, Handled);
    if not IsHintMsg(Msg) and not Handled and not IsMDIMsg(Msg) and
      not IsKeyMsg(Msg) and not IsDlgMsg(Msg) then
    begin
      TranslateMessage(Msg);
      DispatchMessage(Msg);
```

```
    end;
  end
  else
    FTerminate:= True;
end;
end;

procedure TKartRiderApp.HandleMessage;
var
Msg: TMsg;
begin
if not ProcessMessage(Msg) then Idle(Msg);
end;

procedure TKartRiderApp.HookMainWindow(Hook: TWindowHook);
var
WindowHook: ^TWindowHook;
begin
if not FHandleCreated then
begin
  if FHandle <> 0 then
    SendMessage(FHandle, CM_WINDOWHOOK, 0, Longint(@@Hook));
end else
begin
  FWindowHooks.Expand;
  New(WindowHook);
  WindowHook^:= Hook;
  FWindowHooks.Add(WindowHook);
end;
end;

procedure TKartRiderApp.UnhookMainWindow(Hook: TWindowHook);
var
I: Integer;
WindowHook: ^TWindowHook;
begin
if not FHandleCreated then
begin
  if FHandle <> 0 then
    SendMessage(FHandle, CM_WINDOWHOOK, 1, Longint(@@Hook));
end else
  for I:= 0 to FWindowHooks.Count - 1 do
  begin
    WindowHook:= FWindowHooks[I];
    if (TMethod(WindowHook^).Code = TMethod(Hook).Code) and
      (TMethod(WindowHook^).Data = TMethod(Hook).Data) then
    begin
      Dispose(WindowHook);
      FWindowHooks.Delete(I);
      Break;
    end;
  end;
end;

procedure TKartRiderApp.Initialize;
begin
if InitProc <> nil then TProcedure(InitProc);
end;

procedure TKartRiderApp.CreateForm(InstanceClass: TComponentClass; var
Reference);
var
Instance: TComponent;
begin
Instance:= TComponent(InstanceClass.NewInstance);
TComponent(Reference):= Instance;
try
  Instance.Create(Self);
except
```

```
  TComponent(Reference):= nil;
  raise;
end;
if (FMainForm = nil) and (Instance is TForm) then
begin
  TForm(Instance).HandleNeeded;
  FMainForm:= TForm(Instance);
end;
end;

procedure TKartRiderApp.Run;
begin
FRunning:= True;
try
  AddExitProc(DoneApplication);
  if FMainForm <> nil then
  begin
    case CmdShow of
      SW_SHOWMINNOACTIVE: FMainForm.FWindowState:= wsMinimized;
      SW_SHOWMAXIMIZED: MainForm.WindowState:= wsMaximized;
    end;
    if FShowMainForm then
      if FMainForm.FWindowState = wsMinimized then
        Minimize else
        FMainForm.Visible:= True;
    repeat
      try
        HandleMessage;
      except
        HandleException(Self);
      end;
    until Terminated;
  end;
finally
  FRunning:= False;
end;
end;

procedure TKartRiderApp.Terminate;
begin
if CallTerminateProcs then PostQuitMessage(0);
end;

procedure TKartRiderApp.HandleException(Sender: TObject);
begin
if GetCapture <> 0 then SendMessage(GetCapture, WM_CANCELMODE, 0, 0);
if ExceptObject is Exception then
begin
  if not (ExceptObject is EAbort) then
    if Assigned(FOnException) then
      FOnException(Sender, Exception(ExceptObject))
    else
      ShowException(Exception(ExceptObject));
end else
  SysUtils.ShowException(ExceptObject, ExceptAddr);
end;

function TKartRiderApp.MessageBox(const Text, Caption: PChar; Flags: Longint):
Integer;
var
ActiveWindow: HWnd;
WindowList: Pointer;
MBMonitor, AppMonitor: HMonitor;
MonInfo: TMonitorInfo;
Rect: TRect;
FocusState: TFocusState;
begin
ActiveWindow:= GetActiveWindow;
MBMonitor:= MonitorFromWindow(ActiveWindow, MONITOR_DEFAULTTONEAREST);
```

```
AppMonitor:= MonitorFromWindow(Handle, MONITOR_DEFAULTTONEAREST);
if MBMonitor <> AppMonitor then
begin
  MonInfo.cbSize:= Sizeof(TMonitorInfo);
  GetMonitorInfo(MBMonitor, @MonInfo);
  GetWindowRect(Handle, Rect);
  SetWindowPos(Handle, 0,
MonInfo.rcMonitor.Left+((MonInfo.rcMonitor.Right - MonInfo.rcMonitor.Left) div
2),
MonInfo.rcMonitor.Top+((MonInfo.rcMonitor.Bottom - MonInfo.rcMonitor.Top) div
2),
    0, 0, SWP_NOACTIVATE or SWP_NOREDRAW or SWP_NOSIZE or SWP_NOZORDER);
end;
WindowList:= DisableTaskWindows(0);
FocusState:= SaveFocusState;
if UseRightToLeftReading then Flags:= Flags or MB_RTLREADING;
try
  Result:= Windows.MessageBox(Handle, Text, Caption, Flags);
finally
  if MBMonitor <> AppMonitor then
    SetWindowPos(Handle, 0,
      Rect.Left + ((Rect.Right - Rect.Left) div 2),
      Rect.Top + ((Rect.Bottom - Rect.Top) div 2),
      0, 0, SWP_NOACTIVATE or SWP_NOREDRAW or SWP_NOSIZE or SWP_NOZORDER);
  EnableTaskWindows(WindowList);
  SetActiveWindow(ActiveWindow);
  RestoreFocusState(FocusState);
end;
end;

procedure TKartRiderApp.ShowException(E: Exception);
var
Msg: string;
begin
Msg:= E.Message;
if (Msg <> '') and (AnsiLastChar(Msg) > '.') then Msg:= Msg + '.';
MessageBox(PChar(Msg), PChar(GetTitle), MB_OK + MB_ICONSTOP);
end;

function TKartRiderApp.InvokeHelp(Command: Word; Data: Longint): Boolean;
var
CallHelp: Boolean;
HelpHandle: HWND;
ActiveForm: TCustomForm;
begin
Result:= False;
CallHelp:= True;
ActiveForm:= Screen.ActiveCustomForm;

if Assigned(ActiveForm) and Assigned(ActiveForm.FOnHelp) then
  Result:= ActiveForm.FOnHelp(Command, Data, CallHelp)
else if Assigned(FOnHelp) then
  Result:= FOnHelp(Command, Data, CallHelp);

if Assigned(ActiveForm) then
begin
  if csDesigning in ActiveForm.ComponentState then CallHelp:= False;
  if (ActiveForm.TabOrder = -1) and (ActiveForm.Visible = false) and
     (not Assigned(ActiveForm.ActiveControl)) then CallHelp:= False;
end;

if CallHelp and (not Result) then
  if Assigned(ActiveForm) and ActiveForm.HandleAllocated and
(ActiveForm.FHelpFile <> '') then
  begin
    HelpHandle:= ActiveForm.Handle;
    if ValidateHelpSystem then
      Result:= HelpSystem.Hook(Longint(HelpHandle), ActiveForm.FHelpFile,
Command, Data);
```

```
  end
  else
  if FHelpFile <> '' then
  begin
    HelpHandle:= Handle;
    if FMainForm <> nil then HelpHandle:= FMainForm.Handle;
    if ValidateHelpSystem then Result:= HelpSystem.Hook(Longint(HelpHandle),
FHelpFile, Command, Data);
  end else
    if not FHandleCreated then
      PostMessage(FHandle, CM_INVOKEHELP, Command, Data);
end;

function TKartRiderApp.HelpKeyword(const Keyword: String): Boolean;
begin
Result:= true;
if ValidateHelpSystem then
  HelpSystem.ShowHelp(Keyword, GetCurrentHelpFile)
else Result:= false;
end;

function TKartRiderApp.HelpContext(Context: THelpContext): Boolean;
begin
Result:= true;
if ValidateHelpSystem then
  HelpSystem.ShowContextHelp(Context, GetCurrentHelpFile)
else Result:= false;
end;

function TKartRiderApp.HelpCommand(Command: Integer; Data: Longint): Boolean;
begin
Result:= InvokeHelp(Command, Data);
end;

function TKartRiderApp.HelpJump(const JumpID: string): Boolean;
begin
Result:= true;
if ValidateHelpSystem then
  HelpSystem.ShowTopicHelp(JumpID, GetCurrentHelpFile)
else Result:= false;
end;

function TKartRiderApp.GetExeName: string;
begin
Result:= ParamStr(0);
end;

procedure TKartRiderApp.SetShowHint(Value: Boolean);
begin
if FShowHint <> Value then
begin
  FShowHint:= Value;
  if FShowHint then
  begin
    FHintWindow:= HintWindowClass.Create(Self);
    FHintWindow.Color:= FHintColor;
  end else
  begin
    FHintWindow.Free;
    FHintWindow:= nil;
  end;
end;
end;

procedure TKartRiderApp.SetHintColor(Value: TColor);
begin
if FHintColor <> Value then
begin
  FHintColor:= Value;
```

```
  if FHintWindow <> nil then
    FHintWindow.Color:= FHintColor;
end;
end;

procedure TKartRiderApp.DoActionIdle;
var
I: Integer;
begin
for I:= 0 to Screen.CustomFormCount - 1 do
  with Screen.CustomForms[I] do
    if HandleAllocated and IsWindowVisible(Handle) and
      IsWindowEnabled(Handle) then
      UpdateActions;
end;

function TKartRiderApp.DoMouseIdle: TControl;
var
CaptureControl: TControl;
P: TPoint;
begin
GetCursorPos(P);
Result:= FindDragTarget(P, True);
CaptureControl:= GetCaptureControl;
if FMouseControl <> Result then
begin
  if ((FMouseControl <> nil) and (CaptureControl = nil)) or
    ((CaptureControl <> nil) and (FMouseControl = CaptureControl)) then
    FMouseControl.Perform(CM_MOUSELEAVE, 0, 0);
  FMouseControl:= Result;
  if ((FMouseControl <> nil) and (CaptureControl = nil)) or
    ((CaptureControl <> nil) and (FMouseControl = CaptureControl)) then
    FMouseControl.Perform(CM_MOUSEENTER, 0, 0);
end;
end;

procedure TKartRiderApp.Idle(const Msg: TMsg);
var
Control: TControl;
Done: Boolean;
begin
Control:= DoMouseIdle;
if FShowHint and (FMouseControl = nil) then
  CancelHint;
Application.Hint:= GetLongHint(GetHint(Control));
Done:= True;
try
  if Assigned(FOnIdle) then FOnIdle(Self, Done);
  if Done then DoActionIdle;
except
  HandleException(Self);
end;
if (GetCurrentThreadID = MainThreadID) and CheckSynchronize then
if (Libc.GetCurrentThreadID = MainThreadID) and CheckSynchronize then
  Done:= False;
if Done then WaitMessage;
end;

function TKartRiderApp.ExecuteAction(Action: TBasicAction): Boolean;
begin
Result:= False;
if Assigned(FOnActionExecute) then FOnActionExecute(Action, Result);
end;

function TKartRiderApp.UpdateAction(Action: TBasicAction): Boolean;
begin
Result:= False;
if Assigned(FOnActionUpdate) then FOnActionUpdate(Action, Result);
end;
```

```
procedure TKartRiderApp.WakeMainThread(Sender: TObject);
begin
PostMessage(Handle, WM_NULL, 0, 0);
end;

procedure TKartRiderApp.HookSynchronizeWakeup;
begin
Classes.WakeMainThread:= WakeMainThread;
end;

procedure TKartRiderApp.UnhookSynchronizeWakeup;
begin
Classes.WakeMainThread:= nil;
end;

end.
```