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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Полярус Є.В.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Полярусу Євгенію Вячеславовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 132-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи Cloud-телефонії на базі віртуальних АТМ*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Полярус Є.В.
(прізвище та ініціали)

АНОТАЦІЯ

Полярус Є.В. Програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Cloud-телефонії на базі віртуальних АТМ.

Метою розробки є програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ.

Результат роботи – програмна реалізація системи Cloud-телефонії на базі віртуальних АТМ.

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 з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: комп'ютерна інженерія, Cloud-телефонія

ABSTRACT

Polarus E.V. Cloud-telephony system software based on virtual ATMs. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the Cloud-telephony system based on virtual ATMs.

The purpose of the development is the software of the Cloud-telephony system based on virtual ATMs.

The result of the work is the software implementation of the Cloud-telephony system based on virtual ATMs.

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.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Delphi 10.4 environment.

Keywords: computer engineering, Cloud-telephony

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АТМУ – автоматизована телефона мережа установи
БД – база даних
КМЗ – корпоративна мережа зв’язку
ЛОМ – локальна обчислювальна мережа
ОЗП – оперативно-запам’ятовувальний пристрій
ПАТМ – пристрій автоматизованої телефонної мережі
ПЗ – програмне забезпечення
ПП – програмний продукт
СПД – система передачі даних
СУБД – система управління БД
ТфОП – телефонний оператор
AVVID – архітектурна модель фирмы Cisco Systems
CNG – Comfort Noise Generator. Генератор комфортного шуму
DSP – Digital Signal Processor. Процесор цифрової обробки сигналів
DTMF – Dual Tone Multi-Freqency. Багаточастотна система кодування цифр номера
GK – Gatekeeper. Гейткіпер . Виконує функції керування зоною мережі H.323
GW – Gateway. Шлюз. Апаратно-програмний комплекс, що забезпечує обмін даними між мережами різних типів
H.323 – Рекомендація ITU-T, що визначає системи мультимедійного зв'язку в мережах з пакетною комутацією
H.248 – Протокол керування транспортним шлюзом
IP – Internet Protocol. Протокол міжмережної взаємодії
LPC – Linear Prediction Coding. Кодування з лінійним пророкуванням
MCU – Multipoint Control Unit. Пристрій керування конференцією
MG – Media Gateway. Транспортний шлюз
MGCP – Media Gateway Control Protocol. Протокол керування шлюзами

MP – Multipoint processor. Процесор для обробки інформації користувачів при централізованих конференціях

OSI – Open System Interconnection. Взаємодія відкритих систем

PPP – Point-to-Point Protocol. Протокол двостороннього зв'язку

RADIUS – Remote Authentication Dial-In User Service. Протокол автентифікації й авторизації абонентів, а також обліку обсягу наданих їм послуг

RAS – Registration Admission and Status. Протокол взаємодії термінального встаткування з gatekeeper. Входить у сімейство протоколів H.323

RSVP – Resource Reservation Protocol. Протокол резервування ресурсів

RTCP – Real-time Transport Control Protocol. Протокол контролю транспортування інформації в реальному часі

SIP – Session Initiation Protocol. Протокол ініціювання сеансів зв'язку

TAPI – Telephony Applications Programming Interface. Інтерфейс для програмування телефонних додатків

TCP – Transmission Control Protocol. Протокол керування передачею (даних) Основний транспортний протокол у стеці протоколів TCP/IP.

TCP/IP – Transmission Control Protocol/Internet Protocol. Стек протоколів, що забезпечують організацію зв'язку між комп'ютерами в мережі Інтернет

UDP – User Datagram Protocol. Протокол передачі дейтаграмм користувача. Подібно TCP, використовує для доставки даних протокол IP. На відміну від TCP/IP, передбачає обмін дейтаграммами без підтвердження

VoIP – Voice over Internet Protocol. Технологія, що дозволяє використовувати IP-мережу для передачі мовної інформації

ВСТУП

Актуальність теми. В Україні швидко набирають популярність послуги корпоративної IP-телефонії, включаючи послуги віртуальних АТМ. Завдяки широкому поширенню високошвидкісного Інтернету підвищується попит на IP-рішення в області зв'язку й стимулюється розвиток хмарних сервісів, основними споживачами яких є компанії малого й середнього бізнесу. Телекомунікаційні компанії розробляють і впроваджують хмарні сервіси бізнес-рівня, у числі яких помітне місце займає так звана хмарна телефонія на базі віртуальних АТМ (ВАТМ). Вона має всі хмарні властивості, включаючи наявність можливості для самостійного адміністрування й конфігурування користувачами, масштабованість (нарощування ресурсів, номерній ємності й функціональності), еластичність (здатність справлятися з піковими навантаженнями) і оплату по фактичному споживанню послуг. Для ВАТМ характерний також обов'язковий для телефонії високий рівень готовності сервісу, а основною відмінністю від традиційних операторських послуг є тісна інтеграція з бізнес-додатками. За прогнозами Infonetics Research, складеним для світового ринку, з 2022 по 2026 рік число робочих місць, що обслуговуються корпоративними операторськими сервісами VoIP і уніфікованих комунікацій (Unified Communications, UC), зросте більш ніж удвічі. Середньорічний ріст ринку ВАТМ і хмарних UC складає 13%. До 2028 року цей сегмент може збільшитися до 9 млрд євро, тоді як ринок традиційних офісних АТМ от уже кілька років перебуває в стані стагнації. За даними опитування Easter Management Group, існують три основні причини придбання послуг ВАТМ компаніями із сегмента SMB – це наявність засобів для продуктивної роботи, економія витрат, а також простота переїзду в інше приміщення або швидке відкриття нового офісу.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ.

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

– Огляд існуючих систем Cloud-телефонії на базі віртуальних АТМ.

– Дослідження системи Cloud-телефонії на базі віртуальних АТМ.

– Програмна реалізація системи Cloud-телефонії на базі віртуальних АТМ.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі Cloud-телефонії на базі віртуальних АТМ.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Три чверті споживачів ВАТМ – малі й середні компанії, які всі частіше замість покупки встаткування звертаються до хмарних рішень, що майже не вимагають первісних вкладень. Причому найбільшим попитом вони користуються в тих, хто займається роздрібними продажами, у телекомунікаційних і IT-компаній, промислових підприємств. Сумарний обсяг ринку послуг для організацій з даних секторів становить біля половини від загального ринку послуг ВАТМ для SMB.

У сегменті великого бізнесу послуга ВАТМ найбільш затребувана з боку компаній з державного, телекомунікаційного й IT-сектора, а також промислових підприємств. Якщо середній і малий бізнес залучають простота розгортання сервісів і відносна доступність ВАТМ, то великі компанії мають потребу в комплексних рішеннях, що включають у себе не тільки системи зв'язку, але й засобу безпеки, документообігу й т.д.

Попит на віртуальні АТМ росте, але як і раніше спостерігається тільки в сегменті SMB і SOHO. Для таких організацій дуже важливе швидке розгортання системи й низька вартість володіння. Головна перевага хмари в цьому випадку – гнучкість і невисокі первісні витрати (у порівнянні з устаткуванням, що здобувається у власність). Адже купувати потрібно тільки IP-термінали, а вони сьогодні досить дешеві.

Віртуальні АТМ дійсно вигідні, але головним чином для тих компаній, які створюються для реалізації коротких проектів. У середньо- і довгостроковій перспективі вартість володіння в підсумку виявляється вище, однак у поточних швидко мінливих економічних умовах зниження початкових витрат виходить на перший план.

1.2 Область застосування

Відсутність капітальних витрат на інфраструктурне встаткування – одна з основних переваг хмарного сервісу, як і швидкий старт: вся необхідна інфраструктура рішення вже розгорнута на стороні провайдеру. У підсумку строк впровадження сервісів скорочується в кілька разів. Крім того, провайдери хмарних сервісів самостійно забезпечують відновлення ПЗ. Вартість же хмарного сервісу телефонії звичайно розраховується по числу користувачів.

ВАТМ – надзвичайно вигідне в цьому плані рішення, тому що не тільки дозволяє уникнути капітальних вкладень у створення інфраструктури, але й рятує від витрат на обслуговування й модернізацію встаткування. Крім того, у випадку ВАТМ компанія платить тільки за реально використовувані функції, що теж сприяє оптимізації витрат. До того ж ВАТМ і інтегровані з нею додатки пропонують безліч інструментів для оптимізації бізнес-процесів, що передбачають спілкування із клієнтами. Це дає можливість не просто скоротити витрати, а домогтися зростання прибутку за рахунок збільшення продажів і підвищення якості обслуговування.

Важливе достоїнство віртуальних АТМ – масштабованість і гнучкість, настільки необхідні в несприятливих економічних умовах. При наявності таких можливостей можна швидко адаптувати систему телефонії до будь-яких організаційних змін і тим самим уникнути їхнього негативного впливу на роботу компанії.

Що стосується змін попиту, то останнім часом відзначається збільшення глибини використання ВАТМ. Раніше ВАТМ цікавила покупців в основному як інструмент для керування віртуальними номерами, а тепер наші клієнти використовують усе більше можливостей віртуальної телефонії, широко застосовуючи функції запису розмов, моніторингу, автоінформування в період очікування відповіді оператора. Багато хто підключають центр обробки викликів і хмарну CRM.

Важливо, що найчастіше віртуальні АТМ споконвічно створюються розроблювачами не у вигляді розширення традиційної УАТМ, а як новий продукт із коштовними для замовника функціями. Наприклад, вони дозволяють одержувати детальну статистику по дзвінках, доповнюються іншими додатками SaaS для бізнесу.

В умовах підривного росту числа використовуваних мобільних пристроїв, масового поширення мереж 3G і 4G, а також збільшення обсягів переданих даних мобільний напрямок стає для багатьох провайдерів VoIP одним із пріоритетних. Істотним драйвером для розвитку мобільної IP-телефонії є активне розширення LTE, послуги «голос поверх LTE» (VoLTE).

По даним Infonetics Research, світовий ринок мобільної IP-телефонії продовжить рости за рахунок нарощування продажів смартфонів і збільшення проникнення мобільного інтернету. За прогнозами Infonetics Research, в 2027 році кількість користувачів mVoIP (mobile VoIP) досягне майже 410 млн чоловік (за останні чотири роки збільшиться в 4,5 рази).

Як показує практика, ВАТМ підходять в основному невеликій і середній компаніям, у яких немає можливості здобувати апаратно-програмні комплекси IP-ВАТМ. Хмарне рішення дозволяє ефективно управляти викликами. Такий варіант стане оптимальним вибором для тимчасових колективів, туристичних, страхових і транспортних компаній, підприємств, що займаються готельним бізнесом, для медичних клінік і цілого ряду інших організацій з різних галузей. IP-телефонія добре інтегрується з додатками, що дозволяє перейти на новий рівень бізнес-комунікацій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Мегаплан

У сервісі для спільної роботи й керування бізнесом Мегаплан на повну силу заробила убудована телефонія. Тепер користувачам сервісу не потрібно купувати окрему АТМ і інтегрувати її.

Подзвонити потрібному клієнтові або співробітникові можна одним кліком прямо з Мегаплана. Дзвонити за кордон можна через IP-телефонію по низьких тарифах. Дзвінки усередині компанії – безкоштовні. Для прийому вхідних дзвінків можна прив'язати до сервісу свій номер або орендувати новий (багатоканальний з необмеженою кількістю додаткових). У налаштуваннях можна задати сценарії обробки вхідних викликів. Віджет телефонії не заважає під час розмови робити замітки, дізнаватися про минулі домовленості й призначати наступний контакт. Запис розмов прив’язується до клієнтів і їх завжди можна прослухати.

Система А2Б

Система керування бізнесом А2Б інтегрувала свою CRM із сервісом телефонії СКОРОЗВОН. Тепер можна дзвонити клієнтам одним кліком усього від 40 копійок у хвилину.

Зручні дзвінки й швидке фіксування результатів дозволять менеджерам не витрачати час на пошук номера контактної особи і його набір у телефоні. Сервіс СКОРОЗВОН надає можливість запису телефонних розмов і зберігає їх до 1 року (залежно від тарифу). Система А2Б також дозволяє управляти проектами, контролювати доручення, звістки планування.

FreshOffice CRM

Всі провідні CRM системи інтегруються із провідними віртуальними АТМ. Можна дзвонити клієнтами прямо з CRM, вести облік дзвінків по кожному клієнті, бачити картку клієнта при вхідному дзвінку, контролювати менеджерів, прослуховуючи їхні розмови із клієнтами.

Розроблювачі FreshOffice теж спочатку інтегрувалися зі сторонніми АТМками, а потім вирішили створити власну хмарну АТМ для максимальної (рідний) інтеграції з CRM системою. Тим більше, що комунікаційні можливості сервісу вже містять у собі електронну пошту, розсилання й внутрішній месенджер. На додаток ще додали власну послуг дешевих дзвінків через IP-телефонію й можливість підключення багатоканального номера. Виходить уже не CRM, а повноцінний віртуальний офіс.

Clientobox

В онлайн CRM системі Clientobox з'явилася можливість робити клієнтам дешеві дзвінки через IP-телефонію. Щоб подзвонити за допомогою Clientobox потрібно кликнути по будь-якому телефонному номері в системі.

Всі дзвінки зроблені в рамках угоди фіксуються усередині угоди. І усередині контакту, якому дзвонили ваші співробітники. Детальний звіт про дзвінки, їхньої тривалості, про тім хто їх робив, можуть побачити адміністратори аккаунту в розділі «Дзвінки» у налаштуваннях. Включити телефонію може адміністратор аккаунту Clientobox. Телефонія реалізована за допомогою платформи voximplant.com. Вартість телефонії не входить в оплату сервісу Кліентобокс, дзвінки тарифікуються окремо.

Гравітел

Інтеграція сервісів Гравітел і amoCRM дозволяє істотно підвищити ефективність діяльності відділу продажів і надає можливість автоматичного розподілу вхідних дзвінків по менеджерах, автоматичного відкриття під час дзвінка картка клієнта (менеджер може називати клієнта по ім'ю й оперативно одержати всю інформацію про нього), можливість систематизувати й

упорядкувати дані по клієнтах, можливість відслідковувати кількість продажів і якість роботи співробітників у реальному часі.

FreshOffice CRM

В CRM системі FreshOffice був повністю перероблений модуль телефонії, що дозволяє робити дзвінки одним кліком з картки контрагента. Розроблювачі говорять, що завдяки новій платформі, модуль телефонії тепер працює стабільно й забезпечує висока якість зв'язку. Оновлений модуль не зобов'язує працювати з певним провайдером телефонії, ви самі в праві вибрати будь-якого постачальника послуг, підключитися до нього й без яких-небудь додаткових інтеграцій користуватися "Телефонією". Також обіцяють, що система інтегрується з будь-якою офісною АТМ. Крім того, з'явилися нові функції: можливість переводити вхідні дзвінки на інших абонентів, функція "зайнято", що придасться, якщо в цей момент немає можливості відповісти на вхідний дзвінок.

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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи Cloud-телефонії на базі віртуальних АТМ.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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

Важлива відмінність хмарної телефонії (віртуальних АТМ) від традиційних операторських сервісів Hosted PBX/IPCentrex полягає в тому, що клієнтові не потрібно розбиратися в тонкостях – він може запустити сервіс буквально «однією кнопкою» і настроїти віртуальну АТМ за кілька кроків. Це максимально простий і доступний спосіб одержати саме послугу, а не інструмент для чергового етапу робіт. Крім стандартних функцій, віртуальна АТМ завжди надає можливість віддаленого налаштування й моніторингу.

Статистика, що збирається нею, і історія дзвінків становлять основу для аналітики й реалізації інтелектуальних функцій. Наприклад, можна визначити, з яким співробітником клієнт спілкувався останній раз, або ідентифікувати нового клієнта. На підставі наявних даних легко здійснювати маршрутизацію викликів у самої АТМ. До розширених функцій ставляться інтеграція з іншими хмарними продуктами, наприклад з CRM, і програмні інтерфейси (API) для сполучення ВАТМ із власними додатками клієнта.

Завдяки застосуванню протоколу SIP прийом викликів може бути настроєний на всіх пристроях, будь те телефон, комп'ютер або планшет. Найбільший ефект ВАТМ дає в тому випадку, якщо використовується як інструмент продажів або підтримки. При продажах система дозволяє збирати статистику по всіх дзвінках, контролювати пропущені, записувати розмови для детального аналізу, надавати клієнтові можливість оцінювати роботу співробітника.

Для служби підтримки важливо, щоб сервіс був максимально зручним – швидко відповісти на дзвінок, «довідатися» клієнта, що дзвонить в офіс, запропонувати йому просте самообслуговування в голосовому меню. Запис

розмов і статистику можна одержати в будь-який час, причому для цього не потрібно звертатися до системних адміністраторів. Співробітники бізнес-підрозділів, що працюють із показниками ефективності, обов'язково будуть користуватися IP-телефонією, і хмарної АТМ зокрема.

Віртуальні АТМ перевершують УАТМ у можливостях і зручності. Перша й очевидна перевага – можливість швидко розгорнути телефонію без складних налаштувань. При цьому вона буде доступна в будь-якому місці через Інтернет, тоді як УАТМ звичайно має прив'язку до офісу. Важлива й фінансова складова рішення. У віртуальну АТМ компанії не потрібно інвестувати як у випадку телефонної станції, а платити прийде тільки за обрану функціональність і за активних співробітників, при цьому функціонал і номерна ємність легко розширюються.

Переваги хмарної АТМ перед власною офісною УАТМ полягають у самій концепції хмарних сервісів. З погляду бізнесу це насамперед перетворення капітальних витрат в операційні (CAPEX в OPEX). Замість того щоб здобувати дороге встаткування, компанія обмежується щомісячною оплатою оренди даного сервісу (послуги). Класичні УАТМ найчастіше не надають ніякій мобільності, у той час як для роботи з будь-якого місця за підтримкою хмарної АТМ потрібний усього лише доступ в Інтернет. Та й офіс компанії не прив'язаний фізично до конкретного місця розташування. Звичайно, у хмарних АТМ є й недоліки. Наприклад, іноді виникають проблеми з безпекою, глибокої кастомізацією сервісу під конкретні завдання бізнесу і його залежністю від доступності Інтернету, але для більшості компаній всі вони нівелюються значною економією коштів.

ВАТМ заощаджує час, гроші й дає гнучкість. Хмарні АТМ радикально зменшують капітальні витрати й відчутна знижують операційні, змінюючи структуру інвестицій і роблячи їх менш ризикованими. Крім того, найбільш розвинені хмарні АТМ дозволяють не просто заощаджувати на зв'язку, а перетворювати колись слабкі крапки ключових бізнес-процесів (продажів, роботи

із клієнтами та ін.) у сильні, тобто генерувати додаткові доходи. Вони заощаджують час керівників, дозволяючи ним не займатися рішенням непрофільних завдань. Більше того, хмарна телефонія щільно вбудовується в бізнес-процеси підприємств, тому проектування, конфігурування, накладення її інструментів на бізнес-ситуації, їхнє ефективне застосування до цілям компанії стають робочими завданнями. Нарешті, бізнес одержує небачену раніше гнучкість, здатність чітко, швидко й без додаткових інвестицій проводити розширення/скорочення штату, відкривати/закривати нові філії, напрямки або проекти, без проблем міняти офіси, правильно вишиковувати бізнес-процеси».

Крім названих переваг, він виділяє простоту й невисоку вартість об'єднання в єдиний робочий простір всіх філій, торговельних крапок, складів і інших важливих структур територіально розподілених підприємств. Причому в цей простір можна легко включати віддалених і надомних співробітників.

У цей час основні споживачі віртуальної телефонії – підприємства середнього й малого бізнесу, для яких покупка й підтримка офісної АТМ і організація роботи ЦОВ можуть виявитися занадто складним і ресурсномістким завданням. У цьому випадку перенос інфраструктури в хмару – розумна альтернатива. «Особистий кабінет» надає користувачеві можливість налаштовувати й масштабувати ВАТМ у міру росту компанії.

Тенденція розвитку віртуальних АТМ, побудованих на основі хмарних технологій, буде наростати. Особливою популярністю цей вид зв'язку користується в сегменті малого й середнього бізнесу, тому що ВАТМ не вимагає більших первісних фінансових вкладень, зручна у використанні й має набір функцій, необхідних невеликий компанії. Вона досить проста в налаштуванні й масштабується залежно від цілей і завдань компанії. ВАТМ можна налаштовувати через «особистий кабінет» в Інтернеті, причому набір параметрів зведений до адекватного мінімуму, а більша частина тонких налаштувань виконується технічною службою оператора.

Крім того, хмарні технології вигідні й економічно: оплата послуг віртуальної АТМ здійснюється по факті (дзвінки тарифікуються похвилино), причому тільки за успішне з'єднання, факсимільні повідомлення оплачуються по кількості переданих сторінок, витрати на міжміський зв'язок також мінімальні.

Сучасні віртуальні АТМ, крім основних функцій офісних АТМ, можуть запропонувати замовникові можливості створення ЦОВ, запису розмов, CRM. Функціональність сучасних рішень розвивається в напрямку забезпечення незалежності персоналу від робочого місця, тобто переносу більшості оперативних даних для роботи й взаємодії в хмару. Рішення на базі хмарної АТМ добре доповнюють функціональність календаря співробітників компанії, проведення конференцій і вебинарів, відеоконсультації й відеодзвінки.

Найбільшу користь хмарні АТМ здатні принести при створенні невеликих ЦОВ для оптимізації процесів обробки великої кількості дзвінків, а також обдзвону й SMS-оповіщення. Такий ЦОВ можна організувати через Інтернет у будь-якій крапці країни або миру. Збільшення попиту на термінальне встаткування SIP говорить про зростаючий інтерес українських користувачів до нових технологій і, зокрема, до хмарних АТМ.

ВАТМ – один з важливих способів оптимізації комунікацій із клієнтами. Вона забезпечує основні комунікаційні потреби будь-якої компанії, дозволяє краще організувати телефонний зв'язок, гнучко управляти внутрішніми комунікаціями й вхідними дзвінками, аналізувати якість роботи персоналу, підвищити ефективність планування, оптимізувати роботу відділів продажів і служб техпідтримки.

Цінність пропозиції збільшується за рахунок інтеграції сервісів, наприклад телефонії з CRM і ЦОВ. Замовник може самостійно, без залучення зовнішніх консультантів і навіть власних IT-фахівців, впроваджувати, налаштовувати й швидко розгортати нові рішення відповідно до вимог бізнесу.

У порівнянні із класичними віртуальні АТМ більше націлені на зовнішні комунікації й підтримку бізнесу на рівні ключових бізнес-процесів (продажів,

обслуговування клієнтів і ін.). Тому в розвинених хмарних АТМ акцент, як правило, робиться на наданні послуг детальної статистики й моніторингу викликів, на оптимальному розподілі дзвінків клієнтів, запису розмов і інструментах, що допомагають оцінювати роботу співробітників і приймати управлінські рішення. Розроблювачі не забувають і про засоби керування персоналом у реальному часі, про інструменти, що дозволяють вибудувати точну, прозору й зрозумілу систему KPI. Всі ці сервіси й інструменти дозволяють відшліфувати процеси взаємини із клієнтами й максимально розширити вирву продажів фахівця, відділу й усього підприємства.

Найчастіше з ВАТМ інтегрується CRM. Її завдання – керування різними сферами взаємодії із замовниками: автоматизація й планування продажів, керування маркетинговими акціями, ведення й облік інформації про клієнтів, аналітика й звітність, обробка дзвінків. Можливості хмарної системи CRM з комплексною підтримкою IP-телефонії, надаваної оператором зв'язку, можуть містити в собі спливаючі картки компаній при вхідних дзвінках, автоматичне розпізнавання назв організацій по номерах телефонів, шаблони для різних типів продажів, контроль статусів угод, хронологію телефонних контактів із клієнтом, що налаштовуються повідомлення про зміни в продажах, коментарі й обговорення. CRM дозволяє контролювати й правильно організовувати внутрішні робочі процеси компанії з розмежуванням прав доступу й відповідальності, контролем строків і статусів завдань.

CRM надає цілий ряд корисних даних, включаючи вирву продажів, динаміку нових продажів у часі, активність співробітників (по числу продажів і завдань), різні звіти, аналітику по дзвінках і іншій інформації. Інтеграція з ВАТМ дозволяє також з вигодою використовувати в CRM такі можливості, як сценарії обробки дзвінків, черги очікування, автоматична переадресація дзвінка клієнта на співробітника. Інтеграція статистики й аналітики ВАТМ із CRM допомагає оцінити реальну ефективність менеджерів по роботі із клієнтами.

Дуже корисна інтеграція хмарних АТМ із іншими бізнес-додатками – наприклад, із системою керування бізнес-процесами, системою документообігу, менеджерами завдань: «Найбільший ефект дає інтеграція із суміжними системами, що ідеально доповнюють один одного. У першу чергу – з контакт-центрами й додатками для спільної роботи.

З ВАТМ можуть інтегруватися сервіси HelpDesk і LiveChat. В Україні ці системи тільки здобувають популярність, але з кожним днем актуальність подібної інтеграції буде зростати, упевнені в ITooLabs.

Пріоритетним завданням є підвищення продуктивності праці, тому усе більше затребувані не просто АТМ, а засоби об'єднаних комунікацій,. Користувач одержує функціонала миттєвих повідомлень, аудіо- і відеоконференцій, електронної й голосової пошти, інтеграцію з усіма офісними додатками, ERP, системами електронного документообігу.

Розвиток ВАТМ орієнтований у першу чергу на розширення функціональних можливостей в області інтеграції й роботи з різними мобільними пристроями. Щоб підвищити ефективність віртуальних АТМ, має сенс розширити можливості інтеграції з різними додатками ERP – наприклад, системами автоматизації документообігу, керування ресурсами, логістичними, виробничими й збутовими процесами.

ВАТМ – сучасне економічне рішення для комплексної телефонізації компанії – допомагає підвищити ефективність бізнес-процесів, створити клієнтоорієнтовану організацію й розширити географію продажів. Корисно скласти перелік вимог бізнесу до даного сервісу, спрогнозувавши їх хоча б на парі років уперед і з'ясувавши, наскільки відповідає їм функціонал кожного постачальника. Треба впевнитися й у наявності можливостей підключення іншого телефонного оператора й інтеграції з офісною АТМ. Безкоштовний тестовий період дозволить зрозуміти, чи підходить підприємству дана послуга.

Застосовуване користувачами встаткування SIP повинне підтримувати всі функції віртуальної АТМ, у тому числі переадресацію й переклад виклику,

конференц-зв'язок, «чорний список» і інші. ВАТМ проявляють свої кращі якості насамперед у процесах продажів і обслуговування клієнтів. Відповідно до наявної статистики, застосування кращих віртуальних АТМ дозволяє розширити вирву продажів на 40-70%.

Одна з найважливіших характеристик, яку потрібно враховувати при виборі прикінцевого пристрою для віртуальної АТМ, – якість звуку при дзвінках через Інтернет, а також простота налаштування. Наприклад, у всіх моделях SIP-телефонів Panasonic використовується технологія High Definition Sound Performance (HDSP), що дозволяє передавати звук при дзвінках через Інтернет без перекручувань. У провідних SIP-телефонах Panasonic (KX-UT) для забезпечення HD-якості звуку в динаміку трубки встановлений потужний магніт, а в корпус убудована акустична камера, що поліпшує якість звуку на низьких частотах (близько 300 Гц) і який зменшує луну й перекручування звуку під час розмови по голосному зв'язку.

Гарний результат дає взаємодія віртуальної АТМ, ЦОВ і CRM.

Традиційно споживачами хмарних сервісів уважаються компанії сегмента SMB, однак розвиток функціональності хмарних сервісів повинне відповідати потребам і більшим замовникам, і вказують на стійкий приплив, що почався, великих компаній-абонентів. При цьому зростає важливість інтеграції сервісів з іншими бізнес-додатками й з елементами інфраструктури IT. Від успішного рішення цих завдань буде залежати, наскільки швидко сформується новий сегмент українського ринку хмарних сервісів, де пропонуються бізнес-додатка з інтегрованою телефонією. Крім того, зараз далеко не кожне підприємство повністю представляє можливості хмарної телефонії й використовує їх для підвищення ефективності бізнесу. Головна помилка – ставитися до сучасної хмарного АТМ не як до бізнес-додатку, а як до аналога «залізної» офісної АТМ, завдання якої зводиться лише до забезпечення абонентських з'єднань.

Кілька тижнів назад проект Zadarma опублікував власний API. Він дозволяє інтегрувати основні функції IP-телефонії й безкоштовної хмарної АТМ.

Можливо створити безкоштовну хмарну АТМ і підключити до неї інтерфейс API, а через нього вже інтегрувати АТМ зі своїми додатками.

Нижче приведемо список і опис методів, посилання на класи й приклади, які вже є для роботи.

Основні функції, доступні в API:

– відображення й зміна налаштувань SIP-аккаунтів (у тому числі установка/зміна переадресації номерів, можливість прив'язки до SIM-карти);

– відображення балансу, вибірка статистики з різною деталізацією (як SIP, так і безкоштовної АТМ);

– здійснення вихідних дзвінків (функція CallBack на зовнішні й внутрішні номери);

– відправлення SMS-Повідомлень;

– повідомлення вашого сервера про кожний вхідний дзвінок в АТМ, а також маршрутизація цих дзвінків вашим додатком.

Поточних функцій вистачить для легкої інтеграції систем CRM, Call tracking і подібних.

Кожний запит, що має потребу в авторизації, супроводжується додатковим заголовком, виду:

"Authorization: ключ_користувача: підпис"

Підпис складається за наступним алгоритмом:

– масив з переданих параметрів (GET, POST, PUT, DELETE) сортується за назвою ключа за алфавітом;

– з отриманого масиву формується рядок запиту (наприклад, функція http_build_query в PHP), приклад “from=DATEFROM&to=DATETO...”;

– і далі – з'єднується за формулою:

– рядок = ім'я_методу + рядок_запиту + md5(рядок_запиту);

– де “ім'я_методу” – рядок запиту, починаючи від домену (із вказівкою версії API), до початку перерахування параметрів, наприклад – /v1/sip/";

– отриманий рядок кешується по алгоритму sha1 із секретним ключем користувача:

– хеш = hash(рядок, секретний_ключ);

– і далі хеш кодується в base64;

– підпис = base64_encode(хеш).

Формати відповіді: json (по-умовчанню) і xml.

Щоб одержати відповідь від API у форматі xml, у рядок запиту додається параметр «format=xml».

У випадку блокування, метод віддає відповідь із 429 заголовком «You exceeded the rate limit».

Відповідь складається з обов'язкового ключа «status» (success або error). Далі, залежно від методу, віддаються свої ключі із запитуваною інформацією.

У випадку помилки, віддається додатковий ключ «message» з описом помилки.

Вихідні дзвінки виробляються за допомогою CallBack (опис роботи Callback тут).

Схема роботи:

– Від вас надходить API запит на сервер.

– Ви одержуєте вхідний дзвінок і прослуховуєте повідомлення «Дочекайтеся дзвінка».

– Сервер посилає другий виклик на телефон потрібного вам абонента.

– Коли абонент відповів на дзвінок – можна спілкуватися.

Зверніть увагу: як на місці «вашого телефону», так і на місці «телефону, на який дзвоните» може бути ваш SIP-логін або внутрішній номер АТМ. Це дозволяє робити й вихідні дзвінки з офісних IP-телефонів/шлюзів/програм SIP а також дзвінки усередині мережі.

Назва методу: **/v1/request/callback/**

Параметри:

– from – ваш номер телефону або SIP, що викликається callback;

– to – номер телефону або SIP, якому дзвонять;

– sip (опціонально) – номер SIP-користувача, якщо він хоче використовувати певний CallerID, закріплений за цим SIP-ом;

– predicted (опціонально) – якщо зазначено цей прапор, то запит є предикативним (система споконвічно дзвонить на номер “to” і тільки якщо йому додзвонюється, з'єднує з вашим SIP або телефонним номером);

Спеціально для коллцентрів є можливість предикативного набору – система споконвічно дзвонить на номер “to” і тільки якщо йому додзвонюється, з'єднує з вашим SIP або телефонним номером.

Наприклад: у вас є великий список номерів куди дзвонити й кілька співробітників коллцентру, із предикативним набором вам не потрібно щоб співробітники по черзі набирали номери й чекали результат (втрачаючи робочий час).

Для зручного обдзвона номерів з коллцентру:

– передаєте список номерів через виклики функції callback в API;

– у параметрі from указуєте номер вашої безкоштовної АТМ, до якої й підключені співробітники коллцентру, також указуєте прапор predicted;

– готово! Система сама обдзвоне ваших клієнтів, і тільки якщо клієнт візьме трубку до ваших співробітників надійде вхідний дзвінок і почнеться розмова. Помітна економія часу співробітників.

Зверніть увагу: Система зробить такий обдзвон тільки якщо в ньому більше чверті дійсних номерів.

Система може відправляти інформацію про кожний вхідний дзвінок у віртуальної АТМ і направляти дзвінок на потрібний внутрішній номер залежно від відповіді. Для цього необхідно створити відкриту для загального доступу посилання, що буде приймати POST-Запити з інформацією із системи.

Параметри, які відправляються на callback-посилання:

– caller_id – номер що дзвонить;

– called_did – номер, на який подзвонили;

– callstart – час початку дзвінка

– redirect – id сценарію редиректа або внутрішній номер АТМ, або «blacklist» – у цьому випадку дзвінок буде відхилений із сигналом зайнято;

– caller_name – можна дати ім'я вхідному номеру.

Повний список методів API:

– **/v1/info/balance/ –** поточний баланс користувача;

– **/v1/info/price/** – вартість дзвінка з урахуванням поточного тарифу користувача;

– **/v1/request/callback/ –** запит на callback (описаний вище);

– **/v1/sip/** – список SIP-номерів користувача;

– **/v1/sip/callerid/** – зміна CallerID (із придбаних або підтверджених номерів);

– **/v1/sip/redirection/** – відображення поточних переадресацій по SIP-номерах користувача, включення/вимикання переадресації, зміна параметрів переадресації;

– **/v1/sms/send/** – відправлення SMS;

– **/v1/statistics/** – одержання загальної статистики;

– **/v1/statistics/pbx/** – статистика по безкоштовної АТМ.

3.2 Розробка структурної схеми

Віртуальна АТМ здатна замінити офісну АТМ, із самим більшим набором можливостей, а інтеграція функцій телефонії з бізнес-додатками перетворює її в коштовне комунікаційне рішення, завдяки якому робота компаній різного профілю і їхня взаємодія із клієнтами стають набагато ефективніше.

Так звана віртуальна телефонія, або «телефонія із хмари», насправді далеко не нове телекомунікаційне рішення. Створити ефективну систему обробки вхідних і вихідних дзвінків без придбання дорогого встаткування можна було й раніше. У даній роботі пропонується реалізація повнофункціональної цифрової

телефонної станції, коли офісна АТМ перебуває в ЦОД оператора, управляється й обслуговується їм.

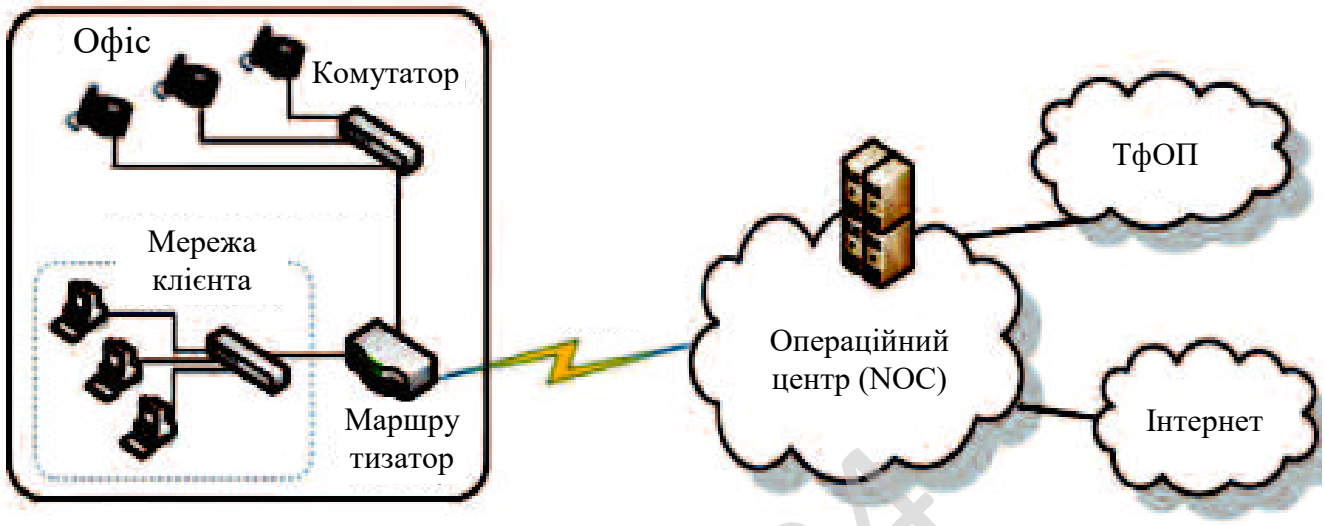

Рисунок 3.1 – Структурна схема системи

Однак сучасна хмарна телефонія ґрунтується на віртуальних АТМ (ВАТМ), сервіси яких мають всі хмарні властивості й переваги, включаючи можливість самостійного адміністрування й конфігурування, масштабованість (нарощування ресурс, номерній ємності й функціональності), еластичність (здатність справлятися з піковими навантаженнями) і оплату по фактичному споживанню послуг. Для ВАТМ характерна також висока готовність сервісу, властивої телефонії, а від традиційних операторських послуг вони відрізняються інтеграцією з бізнес-додатками – як хмарними, так і локальними.

Сервіс надається клієнту за допомогою встаткування й ПЗ, що працює в провайдеру, причому найчастіше через Інтернет, хоча можливі й інші варіанти: виділений IP-канал або канали ТфОП. Слід зазначити, що в хмарної телефонії сильніше виражені саме хмарні властивості. Наприклад, сервіси, як правило, налаштовувані краще й масштабовані.

Віртуальні АТМ мають більше високу надійність. Наприклад, у випадку збоїти одного із серверів оператори хмарної телефонії можуть змінити маршрут трафіку й послуги будуть як і раніше надаватися.

Хмарна телефонія стає досить популярноюв операторів: на українському ринку спостерігається справжній бум. Однак поки навіть у великих містах число абонентів віртуальних АТМ становить усього кілька відсотків від загальної кількості абонентів офісних АТМ, та й основний дохід провайдери одержують не за рахунок сервісів SaaS (до якимсь ставиться ВАТМ), а від надання послуг зв'язку. Можливо, у найближчі роки ринок досягне обсягів, цікавих для великих компаній, і вони почнуть просувати в Україні свої продукти. Поки ж на ньому працюють в основному вітчизняні розроблювачі ПЗ й телекомунікаційні оператори.

ВАТМ – це послуга, надавана по моделі SaaS (Software as a Service). Фізично ВАТМ може реалізовуватися провайдером по-різному – наприклад, як виділений сервер із програмним забезпеченням Asterisk і Web-інтерфейсом FreePBX для керування АТМ або як розподілений міжрегіональний програмний комутатор. У кожному разі мова йде про ПЗ, що функціонує на серверах у власному або орендованому ЦОД. Як рішення операторського класу ВАТМ передбачає резервування всіх компонентів. Оператор телефонії займається керуванням, відновленням і розвитком структури хмарної АТМ: її сервери розподілені по декількох площадках і перебувають під контролем фахівців.

«Відкриті» АТМ привертають увагу все більшого числа клієнтів низькими цінами, а хмарна телефонія – відсутністю витрат на розгортання спеціальної інфраструктури. Клієнт бере під оренду прямий віртуальний номер і одержує функціонала відповідно до обраного тарифного плану. Йому не потрібно здобувати, установлювати й налаштовувати дороге телекомунікаційне встаткування, а також наймати фахівців з налаштування й підтримки.

Дзвінки можуть переводитися на будь-які телефони (звичайні, мобільні й IP) у будь-яку крапку світу. Їх можна приймати на комп'ютері, а голосова пошта

й факси будуть надходити на задану адресу електронної пошти. Віртуальну АТМ можна підключити на наявний міський телефонний номер або виділити багатоканальний віртуальний телефонний номер для прийому вхідних дзвінків із ТфОП.

Компанії малого й середнього бізнесу воліють телефонізувати свої офіси за рахунок сучасних хмарних засобів, оскільки вони представляють більше гнучкі й прості в налаштуванні рішення, Крім високої якості зв'язку й стандартного – як у традиційної аналогової телефонної станції – набору можливостей, віртуальна АТМ надає розширені функції. Компанії можуть створювати індивідуальні сценарії переадресації вхідних дзвінків, вишиковувати їх у чергу, записувати голосові повідомлення й телефонні переговори й одержувати по них статистику, заносити номера в чорний список, відправляти й приймати факси в електронному виді, а при необхідності підключатися до розмов співробітників. Деякі ВАТМ навіть «розуміють» голосові запити й переводять дзвінки в автоматичному режимі.

Віртуальна АТМ розміщається на надійному кластері в ЦОД оператора зв'язку й поставляється клієнтові як послуга – в останнього перебувають тільки телефони. Хмарна АТМ – це вже промислове масштабоване рішення. Сервіси, раніше доступні тільки великим компаніям і потребуючому довгому розгортанню, тепер легко впроваджуються в малих або середніх організаціях: їхні співробітники одержують можливість привітати клієнта при кожному дзвінку по ім'ю й відразу відтворювати на екрані історію взаємин з ним. Крім того, функціонал ВАТМ істотно доповнюється послугами інформування по SMS і електронній пошті, сервісами спільної роботи над проектами.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

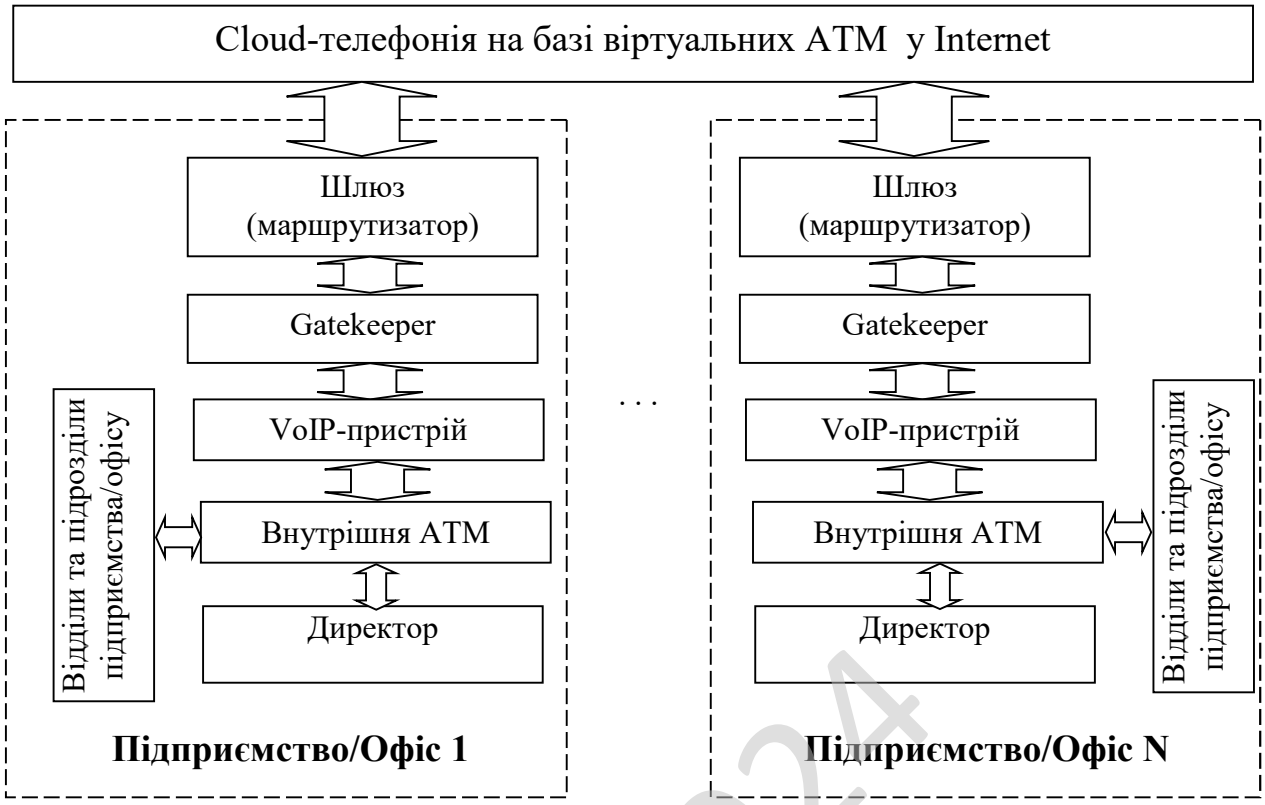

Рисунок 3.2 – Функціональна схема системи

Завдяки функціональній схемі можна чітко побачити основні функціональні блоки системи та взаємозв'язки між ними. При розробці функціональної схеми основний упор робився на існуючи розробки ПЗ і їх модулі допомоги. Аналіз рисунка 3.2 дозволяє чітко прослідити як працює програма. Розглянемо схему зверху вниз, в напрямку від пристрою до кінцевої програми – за допомогою внутрішнього пула доступу що забезпечує закриті канали зв'язку абоненти генеральний директор, відділи бухгалтерії, логістики, складу, збуту, економічної безпеки – можуть взаємодіяти з дочірніми виробниками з використанням внутрішньої АТМ. і чітко налагодженої технічної системи взаємодії. Технічна частина взаємодії забезпечується VoIP та гейткіпером з використанням маршрутизатора.

Розглянемо системний взаємозв'язок програмних модулів і інформаційних файлів. Технологія внутримашиної організації задається послідовністю

реалізованих процедур – схем взаємозв'язку програмних модулів і інформаційних масивів. Така схема являє собою декомпозицію загального процесу рішення задачі на окремі процедури перетворення масивів, іменованими модулями (це – виклик IP-адресату, завершення сеансу зв’язку, редагування легенди IP-номерів, редагування бази даних IP-адрес, моніторинг бази даних IP-адрес та легенд або організація селективного зв’язку і т.і.).

Основне призначення створюваної Cloud-телефонії на базі віртуальних АТМ – це автоматизація процесу керування. Отже, структуру програм можна описати наступними основними блоками.

Робота з програмою починається з введення інформаційного вікна й активізації системи меню. Робота програми здійснюється по діалоговому і по-дійному режиму, при цьому по діалогом розуміється надання користувачу декількох альтернатив і обробка його вибору. У діалогову систему входять головне меню з відповідними спливаючими підменю а також діалогові вікна. Під подіями розуміються процеси, що активізуються користувачем (наприклад – натискання функціональних клавіш), а також програмні події – одержання з’єднання з абонентом або закінчення з’єднання з абонентом. На підставі даних подій активізуються процедури контролю допустимості даних.

Модуль “Головне меню” призначений для запуску основних процедур програми і завершення роботи з програмою.

Модуль роботи з довідниками містить у собі два довідники:

– Довідник – Довідкова система.

– Довідник – Інструкція користувача.

Модуль роботи з базою даних абонентів корпоративної мережі включає в себе наступні підмодулі:

– Редагування легенди IP-номерів.

– Редагування бази даних IP-адрес.

– Моніторинг бази даних IP-адрес та легенд.

Призначення даного модуля є пошук і перегляд інформації з телефонних

даних адрес абонентів корпорації, а також їх легенд.

Інформаційною базою даного модуля є таблиці: Значення IP-номерів та Легенда IP-номерів. Дані в інформаційну базу заносяться за допомогою спеціальних форм, що викликаються з головного меню програми.

Модуль «Формування вхідної інформації» призначений для введення первинних даних і перегляду раніше занесених. Даний модуль реалізує задачі обліку телефонних номерів, забезпечуючи введення номерів та їх легенд, ранжування за важливістю та їх знищенню.

У комп'ютерних системах користувачі для введення, перегляду та редагування інформації бази даних (IP-адрес та відповідних легенд) можуть застосовувати форми. Основні переваги використання форм наступні:

– При введенні даних у поля-форми, додаток може зчитувати словник даних сервера й автоматично перевірити допустимість даних відповідно до правил цілісності.

– Поле введення у формі може представляти список допустимих значень, з яких користувачі можуть легко вибрати потрібне.

– Область форми може виводити шаблон, що відповідає поточної виведеної у формі запису.

– Командні кнопки у формі можуть виконувати дії, зв'язані з виведеної у формі поточною записом.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврської дипломної роботи, наведена на рисунку 3.3.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне

представлення «потоків» даних в інформаційній системі.

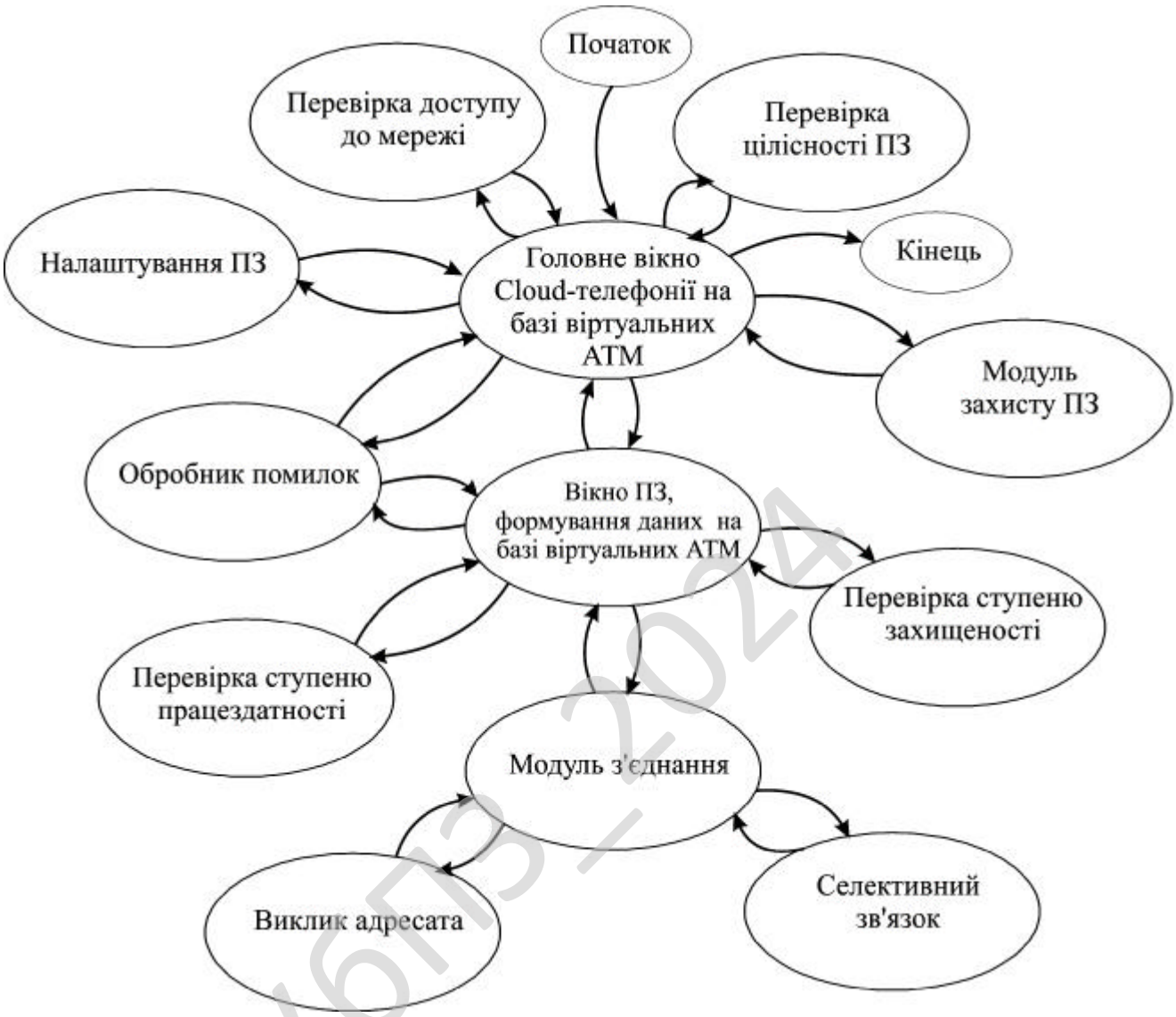

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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 ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

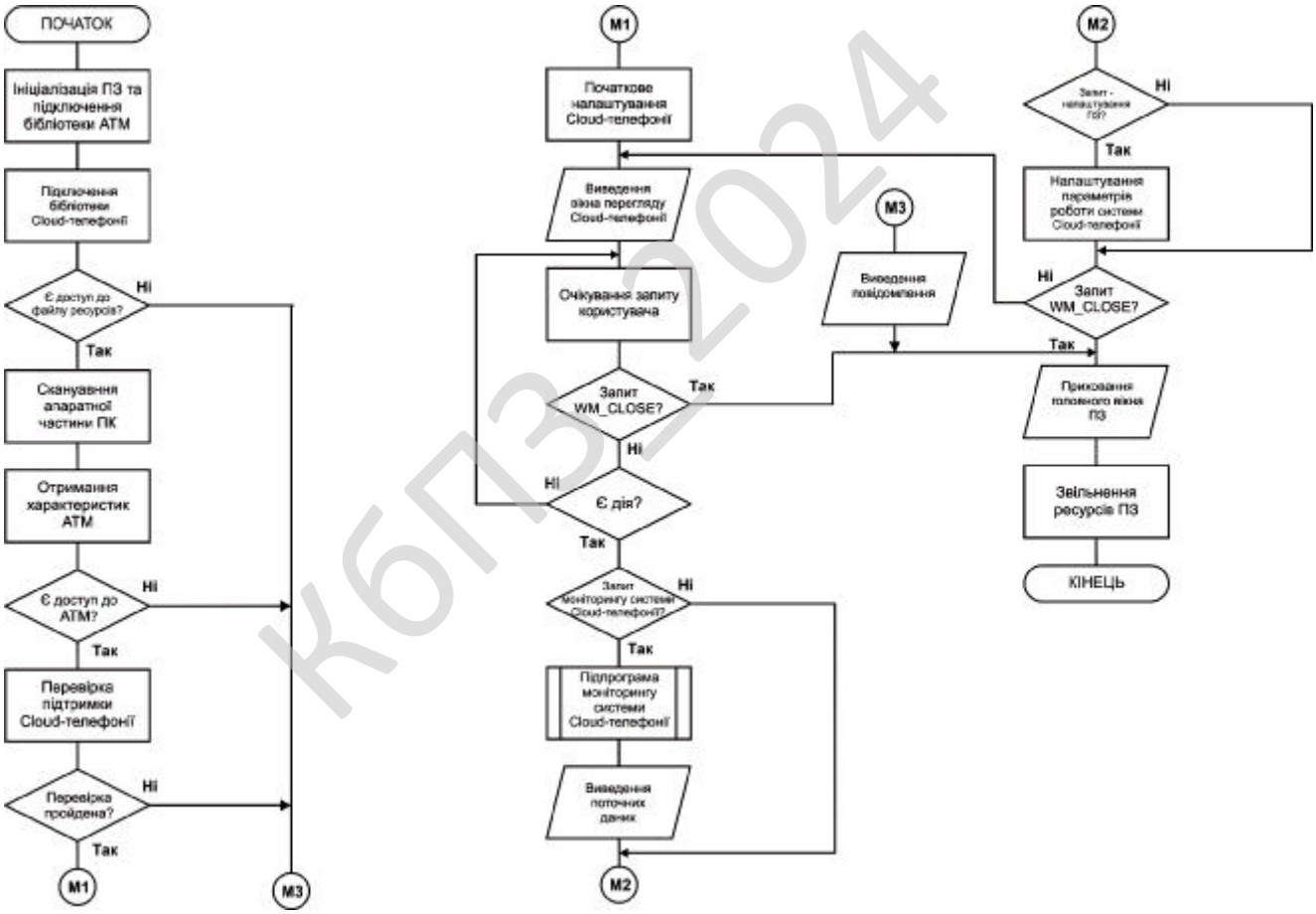

Рисунок 4.1 – Блок-схема основної програми

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми. При виборі початкової

точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю Cloud-телефонії на базі віртуальних АТМ.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

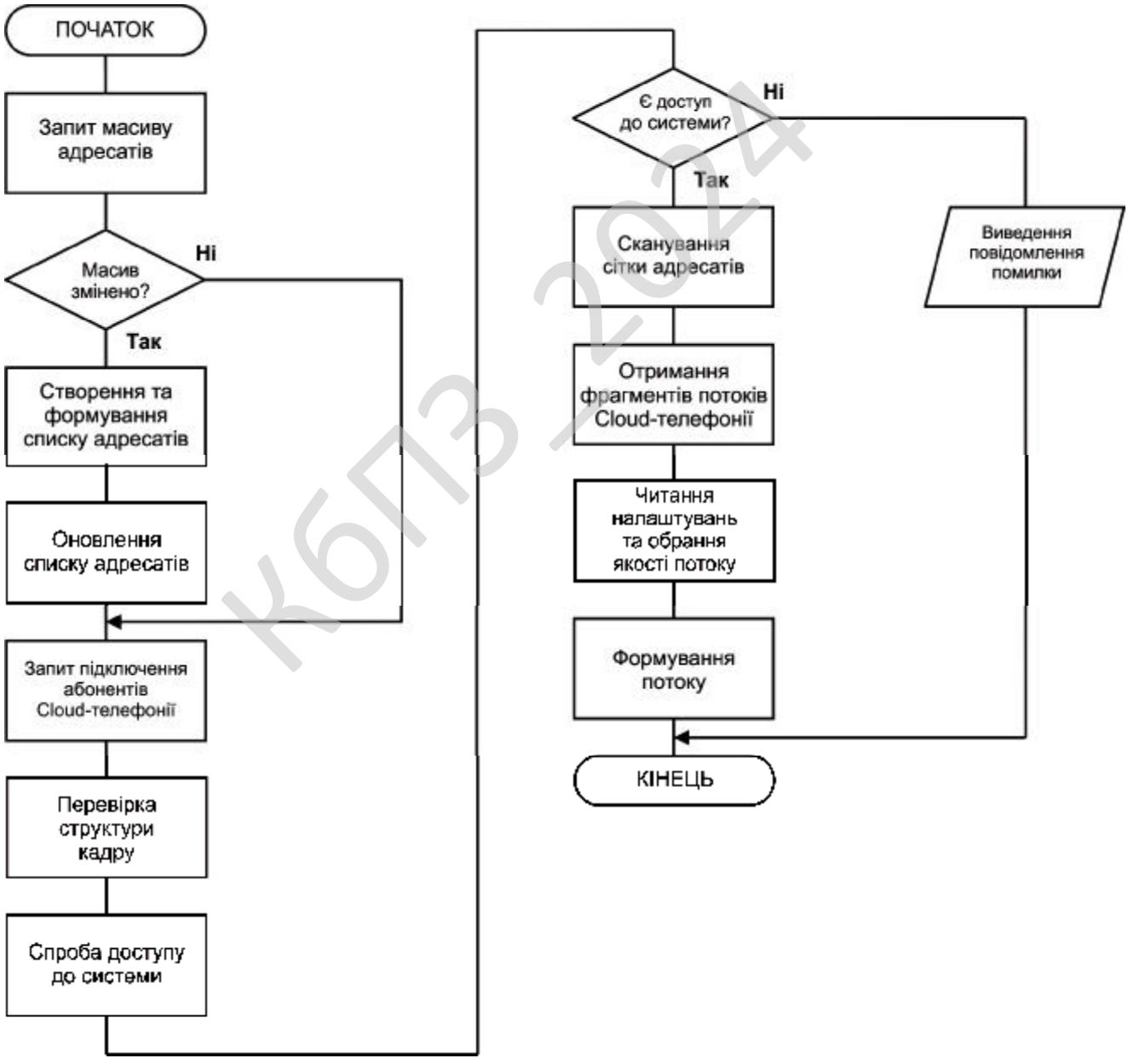

Рисунок 4.2 – Блок-схема роботи підпрограми

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу);.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною

одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Технологічний процес складається з двох основних етапів – збір і облік даних по IP-адресам й формування відповідної легенди та власне сама робота програми у режимі Cloud-телефонії та селективного зв’язку. Вони можуть виконуватися в будь-який календарний момент часу і включають операції введення, з’єднання і ін. Операції мають програмне виконання, підлегле єдиній алгоритмічній схемі. Програма реалізована в середовищі Delphi. Робота з програмою починається з виведення інформаційного вікна й активізації системи меню й здійснюється в діалоговому і по-дійному режиму. При цьому під діалогом розуміється надання користувачу декількох альтернатив і обробка його вибору. У

діалогову систему входять головне меню з відповідними спливаючими підменю а також діалогові вікна. Під подіями розуміються процеси, що активізуються користувачем (наприклад – натискання функціональних клавіш), а також програмні події – з'єднання по IP-технології.

Програма складається з наступних основних модулів.

Основна процедура – конфігурація середовища оточення, формування основного екрана програми, створення системи головного меню і відповідних підміню, активізація меню.

Процедура обробки головного меню – запуск відповідної процедури. Процедура введення даних – забезпечення введення інформації у бази даних IP-адрес та легенд до них, контроль за допустимістю значень, забезпечення введення даних шляхом вибору зі списку.

Допоміжні процедури і функції – реалізація запитів, повідомлень, формування списків вибору IP-адрес та легенд, а також контроль за даними, що вводяться.

Усі модулі в програмі зв'язані між собою за даними, що аналізуються на вході і виробляються на виході. Дані в модулі надходять через діалог з користувачем, параметри і документи інформаційної бази. Передача даних від одного модуля до іншого здійснюється тільки через збережені документи.

Дані через діалог можуть бути отримані прямим і непрямим способом. Прямий спосіб реалізується шляхом їхнього введення за шаблоном чи по запиту конкретних значень реквізитів. Непрямий спосіб – шляхом чи меню логічних (альтернативних) запитів – «так», «ні». При непрямому способі дані, що надходять у модуль, заздалегідь передбачені алгоритмом, але зовні виглядають в обліку відомими фразами.

Параметри (мова) – вхідні дані, отримані у виді конкретних значень, переданих в оперативній пам'яті суміжним модулям (функціям).

Обґрунтування інформаційного забезпечення

Перелік вихідних даних. У ході розробки корпоративної системи зв'язку з використанням IP-технологій визначено, що вихідною інформацією є мова або відеозображення на пункті отримання інформації. Як може здатися на перший погляд, вузькополосне кодування мови, що вимагає обчислювальної потужності, є самим складним завданням, виконуваної устаткуванням IP-телефонії. Однак це не так: алгоритми кодування мови стандартизовані й відмінно документовані, більше того, на ринку доступні досить ефективні їхні реалізації для всіх популярних DSP-платформ. З іншого боку, в устаткуванні IP-телефонії повинні бути реалізовані багато інших функцій, спосіб реалізації яких не є об'єктом стандартизації.

На передавальній стороні устаткування IP-телефонії працює за принципом «закодував, передав і забув». На прийомній стороні все набагато складніше. Пакети приходять із мережі із затримкою, що міняється за випадковим законом. Більше того, пакети можуть прийти не в тій послідовності, у якій були передані, а деякі пакети можуть взагалі бути загублені. Приймач повинен справлятися з усіма цими труднощами, забезпечуючи на виході нормальний звуковий потік з тактовою синхронізацією, або генерованим на основі прийнятого потоку даних, або одержуваним із ТфОП по каналах E1. Прив'язка мовних потоків до місцевого тактового сінхросигналу здійснюється шляхом непомітної на слух деформації періодів мовчання у відтвореному сигналі. До цього залишається додати необхідність передачі факсимільної інформації в реальному часі з автоматичним розпізнаванням сигналів факсимільних апаратів і передачу DTMF-сигналів з коректним їхнім відновленням у приймачі.

Сигнали багаточастотного набору номера (DTMF) – просто звукові сигнали, передані по телефонному каналі. При передачі їх по цифровій телефонній мережі не виникає ніяких проблем, тому що кодування за допомогою алгоритму G.711 не накладає ніяких обмежень на вид звукових сигналів – це може бути мова, сигнали модему, або тональні сигнали – всі будуть успішно відтворені на приймачі [17-25].

Вузькополосні кодеки, щоб досягти низьких швидкостей передачі, використовують той факт, що сигнал, який вони кодують, представляє саме мову. Сигнали DTMF при проходженні через такі кодеки спотворюються й не можуть бути успішно розпізнані приймачем на прийомній стороні [27].

Коли користувачеві ТфОП потрібно ввести якусь додаткову інформацію у віддалену систему при вже встановленому з'єднанні, необхідно забезпечити можливість надійної передачі DTMF-сигналів через мережу IP-телефонії. У випадках, коли система, взаємодіюча з користувачем, просто ставить запитання й чекає введення, тривалість і момент передачі сигналу не важливі. В інших випадках система видає користувачеві список і просить його нажати, наприклад, кнопку «#», як тільки він почує потрібну інформацію; тут ситуація більше складна, і необхідна більше точна прив'язка вчасно.

Існуючі методи передачі сигналів DTMF по мережах IP-телефонії [24,26].

– Обов'язковий метод. Спеціальне повідомлення протоколу H.245 може містити символи цифр і «*», «#». У цьому випадку використовується надійне TCP-з'єднання, так що інформація не може бути загублена. Однак через особливості TCP можуть мати місце значні затримки [24];

– Нестандартний метод. Він може бути застосований у терміналах H.323v2 при використанні процедури fastStart і відсутності каналу H.245. Для передачі сигналів DTMF відкривається спеціальна RTP-сесія, у якій передаються кодовані значення прийнятих цифр, а також дані про амплітуду й тривалість сигналів. Може бути використана та ж сесія, що й для мови, але зі спеціальним типом корисного навантаження. Використання RTP дозволяє прив'язати DTMF– сигнали до реального часу, що є важливою перевагою даного методу [26].

У принципі, перший метод може бути більше кращим, однак у випадку міжнародних викликів і при використанні віддалених систем, що вимагають твердої прив'язки введення користувача до часу, може виявитися необхідним застосувати другий метод. Шлюзи IP-телефонії повинні обов'язково придушувати перекручені сигнали DTMF, що пройшли через основний мовний канал. У

противному випадку, при відновленні сигналів, про які була прийнята інформація, можуть виникнути неприємні ефекти накладення й розмноження сигналів. На основі даного огляду функцій устаткування IP-телефонії можна зробити вивід, про тім що, незважаючи на існування стандартних алгоритмів кодування мови, у розроблювачів є величезний простір для діяльності, спрямованої на подальше вдосконалювання технології IP-телефонії.

Перелік вхідних даних. Під вхідною інформацією розуміється вся інформація, необхідна для вирішення задачі і розташована на різних носіях: первинних документах, машинних носіях, у пам'яті персонального комп'ютера. Вхідною інформацією для розроблювальної в дипломному проекті корпоративної системи зв’язку з використанням IP-технологій є мова або відеозображення.

Щоб передати мову через телефонну мережу, мовну інформацію потрібно перетворити в аналоговий електричний сигнал. При переході до цифрових мереж зв'язку виникла необхідність перетворити аналоговий електричний сигнал у цифровий формат на передавальній стороні, тобто закодувати, і перевести назад в аналогову форму, тобто декодувати, на прийомній стороні [5-8].

Процес перетворення аналогового мовного сигналу в цифрову форму називають аналізом або цифровим кодуванням мови, а зворотний процес відновлення аналогової форми мовного сигналу – синтезом або декодуванням мови.

Ціль будь-якої схеми кодування – одержати таку цифрову послідовність, що вимагає мінімальної швидкості передачі й з якої декодер може відновити вихідний мовний сигнал з мінімальними змінами.

При перетворенні мовного сигналу в цифрову форму, мають місце два процеси – дискретизація, тобто формування дискретних у часі відрахунків амплітуди сигналу, і квантування, тобто дискретизація отриманих значень за амплітудою. Ці дві функції виконуються т.зв. аналого-цифровими перетворювачами (АЦП), які розміщаються в сучасних АТС на платі абонентських комплектів, а у випадку передачі мови по IP-мережах – у терміналі

користувача (комп'ютері або IP-телефоні).

Процес аналого-цифрового перетворення одержав, стосовно до систем зв'язку, назву імпульсно-кодової модуляції (ІКМ).

Щоб знизити необхідну швидкість передачі біт, застосовують нелінійний (логарифмічний) закон квантування, тобто квантуванню піддається не амплітуда сигналу, а її логарифм. У цьому випадку має місце процес «стиску» динамічного діапазону сигналу, а при відновленні сигналу відбувається зворотний процес.

Сьогодні застосовуються два основні різновиди ІКМ: з кодуванням по m-закону й по А-закону. У результаті стиску сигнал з амплітудою, що кодується 12-13 бітами, описується всього вісьма бітами. Розрізняються ці різновиди ІКМ деталями процесу стиску (m-закон кодування переважніше використовувати при малій амплітуді сигналу й при малому відношенні сигнал/шум). У Північній Америці використовується кодування по m-закону, а в Європі – по А-закону. Тому при міжнародному зв'язку в багатьох випадках потрібне перетворення m-закону в А-закон, відповідальність за яке несе країна, у якій використовується m-закон кодування. В обох випадках кожний відлік кодується 8 бітами, або одним байтом, який можна вважати звуковим фрагментом. Для передачі послідовності таких фрагментів необхідна пропускна здатність каналу, рівна 64 Кбіт/с. Оскільки ІКМ була першою стандартною технологією, що одержала широке застосування в цифрових системах передачі, пропускна здатність каналу, рівна 64 Кбіт/с, стала всесвітнім стандартом для цифрових мереж всіх видів, причому – стандартом, що забезпечує передачу мови з дуже гарною якістю. Однак така висока якість передачі мовного сигналу (що є еталоном при оцінці якості інших схем кодування) досягнута в системах ІКМ за рахунок явно надлишкова, при сучасному рівні технології, швидкості передачі інформації [3].

Щоб зменшити властиву ІКМ надмірність і знизити вимоги до смуги пропущення, послідовність чисел, отримана в результаті перетворення мовного аналогового сигналу в цифрову форму, піддається математичним перетворенням, що дозволяють зменшити необхідну швидкість передачі. Ці перетворення

«сирого» цифрового потоку в потік меншої швидкості називають «стиском» (а часто – кодуванням, розглядаючи ІКМ як якусь відправну точку для подальшої обробки інформації). Існує безліч підходів до «стиску» мовної інформації; всі їх можна розділити на три категорії: кодування форми сигналу (waveform coding), кодування вихідної інформації (source coding) і гібридне кодування, що представляє собою сполучення двох підходів.

Кодування форми сигналу. Імпульсно-кодова модуляція, по суті, і являє собою схему кодування форми сигналу. Однак цікавлять більш складні алгоритми, що дозволяють знизити вимоги до смуги пропущення. Розглянуті методи кодування форми сигналу використовують ту обставину, що між випадковими значеннями декількох послідовних обчислень існує деяка залежність. Це дозволяє з досить високою точністю пророчити значення будь-якого відліку на основі значень декількох попередніх йому обчислень.

При побудові алгоритмів кодування названа закономірність використовується двома способами. По-перше, є можливість змінювати параметри квантування залежно від характеру сигналу. У цьому випадку крок квантування може змінюватися, що дозволяє певною мірою згладити протиріччя між зменшенням числа біт, необхідних для кодування величини відліку при збільшенні кроку квантування, і звуженням динамічного діапазону кодера, неминучим без адаптації. Деякі алгоритми передбачають зміну параметрів квантування приблизно в рамках вимовних складів, а деякі змінюють крок квантування на основі аналізу статистичних даних про амплітуду сигналу, отриманих за відносно короткий проміжок часу. По-друге, існує підхід, називаний диференціальним кодуванням або лінійним пророкуванням. Замість того, щоб кодувати вхідний сигнал безпосередньо, кодують різницю між вхідним сигналом і «передвіщеною» величиною, обчисленою на основі декількох попередніх значень сигналу. Описаний метод називається лінійним пророкуванням, тому що він використовує тільки лінійні функції попередніх обчислень. Найпростішою реалізацією останнього підходу є так звана дельта-модуляція (ДМ), алгоритм якої

передбачає кодування різниці між сусідніми обчисленнями сигналу тільки одним інформаційним бітом, забезпечуючи передачу, по суті, тільки знака різниці.

Алгоритмом, побудованим на описані вище принципах, є алгоритм адаптивної диференціальної імпульсно-кодової модуляції (АДІКМ) (G.726). Алгоритм передбачає формування сигналу помилки пророкування і його наступне адаптивне квантування. При досить гарних характеристиках алгоритму, АДІКМ практично не застосовується для передачі мови по мережах з комутацією пакетів, тому що цей алгоритм дуже чутливий до втрат цілих блоків відліку, що відбуваються при втратах пакетів у мережі. У таких випадках порушується синхронізація кодера й декодера, що приводить до катастрофічного погіршення якості відтворення мови навіть при малій імовірності втрат [11-27].

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму SHA-3 (Keccak) – алгоритм гешування змінної розрядності, розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing. 2 жовтня 2022 року Keccak став переможцем конкурсу криптографічних алгоритмів, проведеним Національним інститутом стандартів і технологій США. 5 серпня 2025 року алгоритм затверджено та опубліковано в якості стандарту FIPS 202[1]. У програмній реалізації автори заявляють про 12,5 циклах на байт при виконанні на ПК з процесором Intel Core 2. Проте в апаратних реалізаціях Keccak виявився набагато швидшим, ніж всі інші фіналісти. Конструкція функції губки, використана в геш-функції. P_i – вхідні блоки, Z_j – вихід алгоритму. Невикористаний для виведення набір бітів c («capacity») повинен мати значний розмір для досягнення стійкості до атак. Алгоритм SHA-3 побудований за принципом криптографічної губки (дана структура криптографічних алгоритмів була запропонована авторами алгоритму Keccak раніше).

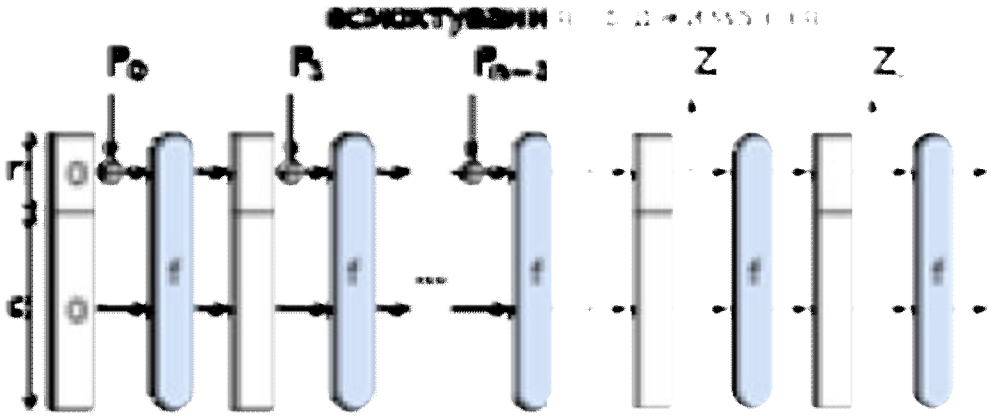

Рисунок 4.3 – Конструкція функції губки, використана в геш-функції

Геш-функції сімейства SHA-3 побудовані на основі конструкції криптографічної губки, в якій дані спочатку «вбираються» в губку, при якому початкове повідомлення M піддається багатораундовим перестановкам f, потім результат Z «віджимається» з губки. На етапі «вбирання» блоки повідомлення додаються за модулем 2 з підмножиною стану, який потім перетвориться з допомогою функції перестановки f. На етапі «віджимання» вихідні блоки зчитуються з одного і того ж підмножинного стану, зміненого функцією перестановок f. Розмір частини стану, який записується і зчитується, називається «швидкістю» (англ. rate) і позначається r, а розмір частки, яка незаймана введенням / виведенням, називається «ємністю» (англ. capacity) і позначається c.

Алгоритм отримання значення хеш-функції можна розділити на кілька етапів:

– Вихідне повідомлення M додається до рядка P довжини, кратній r, за допомогою функції доповнення (pad-функції).

– Рядок P ділиться на n блоків довжини r: P_0, P_1,..., P_{n-1}

– «Всмоктування»: кожен блок P_i доповнюється нулями до рядка довжини b біт і підсумовується по модулю 2 з рядком стану S, де S – рядок довжини b біт (b = r + c). Перед використанням цієї функції всі елементи S дорівнюють нулю.

Для кожного наступного блоку стан – рядок, отриманий застосуванням функції перестановок f до результату попереднього кроку.

– «Віджимання»: поки довжина Z менша d (d – кількість біт в результаті геш-функції), до Z додається r перших біт стану S, після кожного додавання до S, застосовується функція перестановок f. Потім S обрізається до довжини d біт

– Рядок Z довжини d біт повертається в якості результату

Завдяки тому, що стан містить c додаткових біт, алгоритм стійкий до атаки подовженням повідомлення, до якої прийняті алгоритми SHA-1 і SHA-2.

У SHA-3 стан S – це масив 5×5 слів довжиною w = 64 біта, всього $5 \times 5 \times 64 = 1600$ біт. Також в Keccak можуть використовуватися довжини w, рівні меншим ступенями 2 (від w = 1 до w = 32).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення Cloud-телефонії на базі віртуальних АТМ складається з наступних функціональних блоків:

– Навігаційне меню: Сеанс зв’язку; Налаштування.

– Функції представлені у графічному вигляді (іконки).

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок системи.

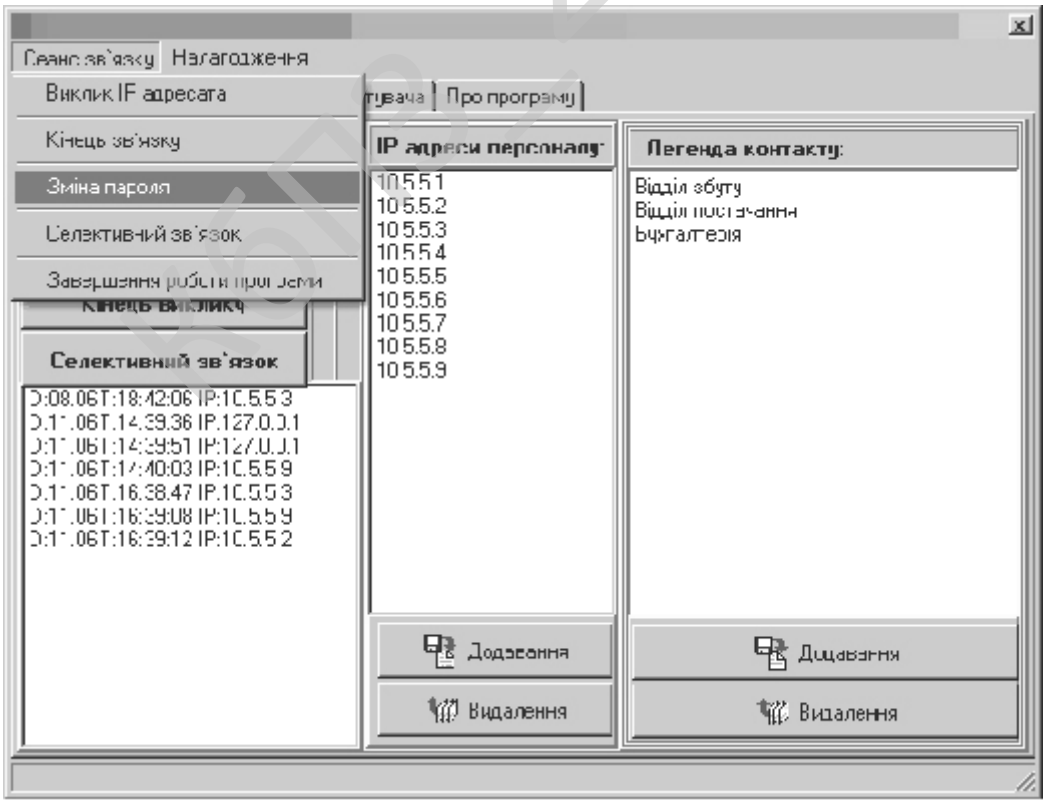

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

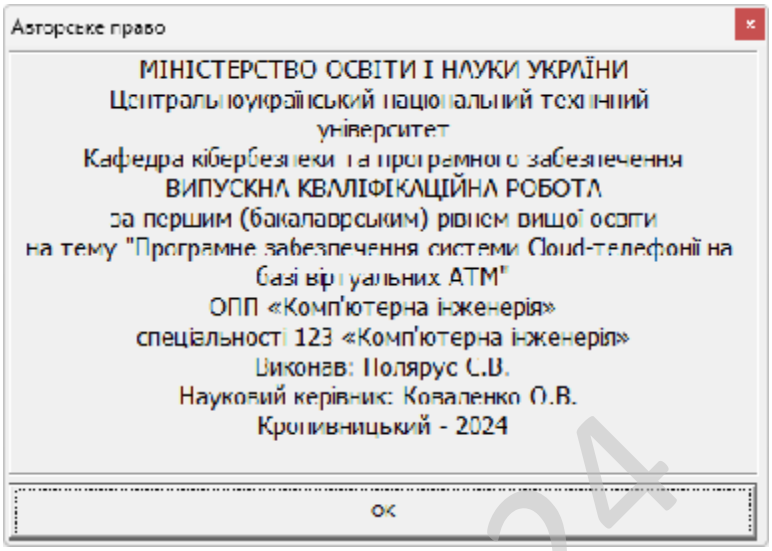

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи Cloud-телефонії на базі віртуальних АТМ.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем Cloud-телефонії на базі віртуальних АТМ.

– Досліджена система Cloud-телефонії на базі віртуальних АТМ.

– На основі отриманих результатів досліджень створена програмна реалізація системи Cloud-телефонії на базі віртуальних АТМ.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання Cloud-телефонії на базі віртуальних АТМ.

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 Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи Cloud-телефонії на базі віртуальних АТМ. 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 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHA-3.

В цілому створене 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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ITU-T Recommendation G.723.1. Dual Rate speech coder for multimedia communication transmitting at 5.3 and 6.3 kit / sec. – 1996.
2. ITU-T Recommendation G.729. Speech codec for multimedia telecommunications transmitting at 8 / 13 kbit / s. – 1996.
3. ITU-T Recommendation H.225.0. Call signaling protocols and media stream packetization for packet-based multimedia communication systems. -Geneva, 1998.
4. ITU-T Recommendation H.245. Control protocol for multimedia communication. -Geneva, 1998
5. ITU-T Recommendation H.248. Gateway control protocol. – Geneva, 2000.
6. ITU-T Recommendation H.320. Narrow-band Visual Telephone Systems and Terminal Equipment. – 1996.
7. ITU-T Recommendation H.321. Adaptation of H.320 Visual Telephone Terminals to B-ISDN Environments. – 1996.
8. ITU-T Recommendation H.322. Visual Telephone Systems and Terminal Equipment for Local Area Networks which Provide a Guaranteed Quality of Service. – 1996.
9. ITU-T Recommendation H.323. Packet based multimedia communication systems. – Geneva, 1998.
10. ITU-T Recommendation H.324. Terminal for Low Bit Rate Multimedia Communications. -1996.
11. ITU-T Recommendation Q.931. ISDN User-Network Interface Layer 3 Specification for Basic Call Control. – 1993.
12. RFC 2705. Media Gateway Control Protocol (MGCP) Version 1.0. M. Arango, A. Dugan, I. Elliott, C. Huitema, S. Pickett. October 1999.
13. RFC 2865. Remote Authentication Dial In User Service (RADIUS).

C.Rigney, S. Willens, A. Rubens, W. Simpson. June 2000.

14. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

15. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

16. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

17. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

18. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

19. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

20. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

21. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

22. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

23. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

24. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

25. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

26. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

27. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

28. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-

телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

31. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

32. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

33. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

34. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

35. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

36. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

37. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

38. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

39. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

40. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

41. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

42. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

43. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

44. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

45. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

46. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

47. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

48. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

49. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

50. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

52. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

53. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

54. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник. – Кіровоград: КНТУ 2012. – 250 с.

55. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник. – Кіровоград: КНТУ 2012. – 454 с

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи Cloud-телефонії на базі віртуальних АТМ.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 132-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи Cloud-телефонії на базі віртуальних АТМ.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи Cloud-телефонії на базі віртуальних АТМ;

– 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, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

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 10/11.

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 Delphi 10.4.

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 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 65 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 7.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко О.В.

Програмне забезпечення системи Cloud-телефонії на базі віртуальних АТМ

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 34

Літера: РП

Кропивницький – 2024 року

Основна програма – Програма організації Cloud-телефонії на базі віртуальних АТМ

```
program iptel;

uses
  Forms,
  Dialogs,
  Windows,
  Sysutils,
  Messages,
  usimple in 'usimple.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2},
  Unit3 in 'Unit3.pas' {Form3},
  Unit1 in 'Unit1.pas' {Form0},
  Unit4 in 'Unit4.pas' {Form4};
{$R simple.RES}
var
  i:integer;
{         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;}
begin
Application.Initialize;
{
if FileExists('main.dat')=false then
 begin
    MessageDlg('Файл       main.dat       не       знайдено!       завершення
програми',mtInformation,[mbOK],0);
    Application.Terminate;
 end else
 begin
   AssignFile(F, 'main.dat');
   Reset(F);
   Readln(F, S);
   CloseFile(F);
      if Crypt(InputBox('Увага!','Введіть пароль',Y),KEY,false)=S then
      begin
           MessageDlg('Дякую, пароль вірний!', mtInformation,[mbOK],0);}
            Try
                Form2:=TForm2.Create(Application);
                Form2.Show;
                Form2.Update;
                for i:=1 to 10 do
                begin
                  sleep(200);
                  Form2.ProgressBar1.Position:=i*10;
                  Form2.Update;
                end;
               Application.Title := 'IpTel';
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm0, Form0);
  Application.CreateForm(TForm4, Form4);
  Finally
                Form2.Free;
   End;
     Application.Run;
{      end
      else
```

```
      begin
      MessageDlg('Невірний пароль!',mtInformation,[mbOK],0);
      Application.Terminate;
      end;
  end;}
end.
```

Підпрограма завантаження загрузочного вікна

```
unit U_SPLASH;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, jpeg, ExtCtrls, StdCtrls, Gauges;

type
  TForm_SPLASH = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Gauge1: TGauge;
    Timer1: TTimer;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_SPLASH: TForm_SPLASH;

implementation

{$R *.dfm}

end.
```

Головне вікно програми

```
unit usimple;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  telefoon, StdCtrls, ComCtrls, ExtCtrls, jpeg, Menus, Buttons;

type
  TForm1 = class(TForm)
    Telefoon1: TTelefoon;
    StatusBar1: TStatusBar;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    Panel1: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel2: TPanel;
    Label1: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Button2: TButton;
    Button1: TButton;
    Panel5: TPanel;
    TabSheet2: TTabSheet;
    Panel6: TPanel;
    Label2: TLabel;
    Panel7: TPanel;
    MainMenu1: TMainMenu;
    Panel8: TPanel;
    ListBox1: TListBox;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    ListBox2: TListBox;
    Panel9: TPanel;
    Label3: TLabel;
    Panel10: TPanel;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    N1: TMenuItem;
    IP1: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    BitBtn5: TBitBtn;
    Memo1: TMemo;
    TabSheet3: TTabSheet;
    Panel11: TPanel;
    Image3: TImage;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    N11: TMenuItem;
    N12: TMenuItem;
    ListBox3: TListBox;
    Label9: TLabel;
    Label10: TLabel;
    N2: TMenuItem;
    PB1: TProgressBar;
    PB2: TProgressBar;
    T1: TTimer;
    Image1: TImage;
    Timer1: TTimer;
```

```
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure ListBox2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure BitBtn5Click(Sender: TObject);
    procedure N12Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure N3Click(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure T1Timer(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit3, Unit2, Unit4, Unit1;
var
  KEY:string='2%#$T%&(*qwrdas@@@#45';
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
begin
 Telefoon1.placecall(Edit1.text);
 ListBox3.items.add('D:'+DateToStr(now)+'                T:'+TimeToStr(now)+'
IP:'+Edit1.text);
  T1.Enabled:=true;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
 Telefoon1.calling:=false;
 T1.Enabled:=false;
 PB1.Position:=0;
 PB2.Position:=0;
end;
procedure TForm1.ListBox1Click(Sender: TObject);
begin
 Edit1.text:=ListBox1.Items.Strings[ListBox1.ItemIndex];
end;
procedure TForm1.ListBox2Click(Sender: TObject);
begin
 Label4.Caption:=ListBox2.Items.Strings[ListBox2.ItemIndex];
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  InputString: string;
begin
  InputString:=  InputBox('Додавання  ip  адреси',  'Прошу',  'Введення  IP  не
здійснено');
  if (InputString<>'Введення IP не здійснено') then
  begin
  Listbox1.Items.add(InputString);
  end;
end;
procedure TForm1.BitBtn3Click(Sender: TObject);
var
```

```
  InputString: string;
begin
 InputString:=  InputBox('Додавання легенди ip адреси', 'Прошу', 'Введення
легенди не здійснено');
 if (InputString<>'Введення легенди не здійснено') then
 begin
 Listbox2.Items.add(InputString);
 end;
end;
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 ListBox1.Items.Strings[ListBox1.ItemIndex]:='';
 ListBox1.Items.Delete(ListBox1.ItemIndex);
end;

procedure TForm1.BitBtn4Click(Sender: TObject);
begin
 ListBox2.Items.Strings[ListBox1.ItemIndex]:='';
 ListBox2.Items.Delete(ListBox1.ItemIndex);
end;
procedure TForm1.N10Click(Sender: TObject);
begin
 Application.Terminate;
end;
procedure TForm1.BitBtn5Click(Sender: TObject);
var
 i:integer;
 G:boolean;
begin
 Form3.ListBox1.Clear;
 g:=false;
 for i:=0 to (ListBox1.Items.Count - 1) do
 begin
  if ListBox1.Selected[i] then
    begin
        Form3.ListBox1.Items.add(ListBox1.Items.Strings[i]);
        g:=true;
    end;
    if g then Form3.show;
 end;
end;
procedure TForm1.N12Click(Sender: TObject);
var
  F: TextFile;
  H,S:string;
  OLD:string;
        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;
begin
if FileExists('main.dat') then
begin
    AssignFile(F, 'main.dat');
    Reset(F);
    Readln(F, S);
     CloseFile(F);
  if Crypt(InputBox('Увага!','Введіть старий пароль',''),KEY,false)=S then
    begin
        H:=Crypt(InputBox('Увага!','Введіть новий пароль',''),KEY,false);
        if H<>'' then
        begin
```

```
          MessageDlg('Дякую пароль змінено',mtInformation,[mbOK],0);
          AssignFile(F, 'main.dat');
          Rewrite(F);
          Writeln(F,H);
          CloseFile(F);
        end else  MessageDlg('Введіть значення!',mtInformation,[mbOK],0);
    end
    else
     begin
        MessageDlg('Файл main.dat не знайдено чи пароль невірний! завершення
програми',mtInformation,[mbOK],0);
        Application.Terminate;
      end;
end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
randomize;
PB1.Position:=0;
PB2.Position:=0;
T1.Enabled:=false;
Timer1.Enabled:=true;
if FileExists('MainLegend.dat')=false then
 begin
    MessageDlg('Файл MainLegend.dat не знайдено!',mtInformation,[mbOK],0);
 end else
 begin
    ListBox2.Items.LoadFromFile('MainLegend.dat');
  end;
if FileExists('MainIP.dat')=false then
 begin
    MessageDlg('Файл MainIP.dat не знайдено!',mtInformation,[mbOK],0);
 end else
 begin
    ListBox1.Items.LoadFromFile('MainIP.dat');
  end;
if  FileExists('MainHISTORY.dat')=false  then  MessageDlg('Файл  MainHISTORY.dat
не знайдено!',mtInformation,[mbOK],0)
else ListBox3.Items.LoadFromFile('MainHISTORY.dat');
if  FileExists('mainHelp.dat')=false  then  MessageDlg('Файл  mainHelp.dat  не
знайдено!',mtInformation,[mbOK],0)
else Memo1.Lines.LoadFromFile('mainHelp.dat');
end;
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
if FileExists('MainLegend.dat')=false then MessageDlg('Файл MainLegend.dat не
знайдено!',mtInformation,[mbOK],0)
else ListBox2.Items.SaveToFile('MainLegend.dat');
if   FileExists('MainIP.dat')=false   then   MessageDlg('Файл   MainIP.dat   не
знайдено!',mtInformation,[mbOK],0)
else ListBox1.Items.SaveToFile('MainIP.dat');
if  FileExists('MainHISTORY.dat')=false  then  MessageDlg('Файл  MainHISTORY.dat
не знайдено!',mtInformation,[mbOK],0)
else ListBox3.Items.SaveToFile('MainHISTORY.dat');
end;
procedure TForm1.N3Click(Sender: TObject);
begin
TabSheet2.Visible:=true;
end;
procedure TForm1.N6Click(Sender: TObject);
begin
 Telefoon1.calling:=false
end;
procedure TForm1.N2Click(Sender: TObject);
begin
Form0.show;
end;
procedure TForm1.FormShow(Sender: TObject);
begin
```

```
PB1.Position:=0;
PB2.Position:=0;
end;
procedure TForm1.T1Timer(Sender: TObject);
var
i1,i2:integer;
begin
PB1.Position:=random(100);
PB2.Position:=PB1.Position;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
Form4.show;
Form1.hide;
Timer1.Enabled:=false;
end;
end.
```

Підпрограма налагодження звукової карти AudioSettings

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, StdCtrls, AMixer, MMSystem;
type
  TForm0 = class(TForm)
    ComboBox1: TComboBox;
    ComboBox2: TComboBox;
    TrackBar: TTrackBar;
    CheckBox: TCheckBox;
    Label1: TLabel;
    Label2: TLabel;
    Mixer: TAudioMixer;
    Label3: TLabel;
    Label4: TLabel;
    ComboBox3: TComboBox;
    LabelStereo: TLabel;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);
    procedure MixerControlChange(Sender: TObject; MixerH, ID: Integer);
    procedure TrackBarChange(Sender: TObject);
    procedure CheckBoxClick(Sender: TObject);
    procedure ComboBox3Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    Setting:Boolean;
  public
    { Public declarations }
  end;
var
  Form0: TForm0;
implementation
uses usimple;
{$R *.DFM}
procedure TForm0.FormCreate(Sender: TObject);
var A:Integer;
begin
  For A := 0 to Mixer.MixerCount - 1 do
    ComboBox3.Items.Add ('Mixer '+IntToStr(A));
  If (ComboBox3.Items.Count > 0) then
    ComboBox3.ItemIndex := 0;
  ComboBox3Change (Sender);
end;
procedure TForm0.ComboBox1Change(Sender: TObject);
var A:Integer;
begin
  ComboBox2.Items.Clear;                                    ComboBox2.Items.Add
(Mixer.Destinations[ComboBox1.ItemIndex].Data.szName);
  For A:=0 to Mixer.Destinations[ComboBox1.ItemIndex].Connections.Count-1 do
ComboBox2.Items.Add(Mixer.Destinations[ComboBox1.ItemIndex].Connections[A].Dat
a.szName);
  If ComboBox2.Items.Count>0 then
  begin
    ComboBox2.ItemIndex:=0;     ComboBox2Change (Self);
  end;
end;
procedure TForm0.ComboBox2Change(Sender: TObject);
var L,R,M:Integer;
    VD,MD:Boolean;
    Stereo:Boolean;
    IsSelect:Boolean;
begin
```

```
  Mixer.GetVolume                    (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,L,R,M,Stereo,VD,MD,IsSelect);
  Setting:=True;
  TrackBar.Visible:=not VD;
  Label1.Visible:=not VD;
  Label3.Visible:=VD;
  If TrackBar.Visible then
    TrackBar.Position:=L;
  CheckBox.Visible:=not MD;
  Label2.Visible:=not MD;
  Label4.Visible:=MD;
  If CheckBox.Visible then
    CheckBox.Checked:=M<>0;
  If (Stereo) then
    LabelStereo.Caption := '- stereo -'
  else
    LabelStereo.Caption := '- mono -';
  Setting:=False;
end;
procedure TForm0.MixerControlChange(Sender: TObject; MixerH, ID: Integer);
begin
  ComboBox2Change (Self);
end;
procedure TForm0.TrackBarChange(Sender: TObject);
begin
  If (not Setting) then
  begin
    Setting:=True;
    Mixer.SetVolume                  (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;
procedure TForm0.CheckBoxClick(Sender: TObject);
begin
  If not Setting then
  begin
    Setting:=True;
    Mixer.SetVolume                  (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;
procedure TForm0.ComboBox3Change(Sender: TObject);
var A:Integer;
begin
  If (ComboBox3.ItemIndex >= 0) AND (ComboBox3.ItemIndex < Mixer.MixerCount)
then
    Mixer.MixerId := ComboBox3.ItemIndex;
  ComboBox1.Items.Clear;
  If Mixer.MixerCount>0 then
  begin
    For A:=0 to Mixer.Destinations.Count-1 do
      ComboBox1.Items.Add (Mixer.Destinations[A].Data.szName);
    If ComboBox1.Items.Count>0 then
    begin
      ComboBox1.ItemIndex:=0;    ComboBox1Change (Self);
    end;
  end
  else
   begin
    ComboBox1.OnChange:=nil;     ComboBox2.OnChange:=nil;
    TrackBar.OnChange:=nil;       CheckBox.OnClick:=nil;
    MessageDlg ('No mixer present in the system !',mtError,[mbOK],0);
  end;
  Setting:=False;
end;
procedure TForm0.Button1Click(Sender: TObject);
begin
```

```
Form0.hide;  Form1.show;
end;
procedure TForm0.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form0.hide; Form1.show;
end;
end.
```

Підпрограма виклику вікна налагодження звукової карти AudioSettings

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  jpeg, ExtCtrls, StdCtrls, ComCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    RichEdit1: TRichEdit;
    ProgressBar1: TProgressBar;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.DFM}

end.
```

Підпрограма селективного зв`язку

```
unit Unit3;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, jpeg, ComCtrls;

type
  TForm3 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    ListBox1: TListBox;
    Animate1: TAnimate;
    Panel4: TPanel;
    BitBtn2: TBitBtn;
    BitBtn1: TBitBtn;
    BitBtn3: TBitBtn;
    Image1: TImage;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

uses usimple;

{$R *.DFM}

procedure TForm3.BitBtn2Click(Sender: TObject);
begin
form3.hide;
end;

procedure TForm3.BitBtn1Click(Sender: TObject);
begin
Animate1.Active:=true;
form1.telefoon1.placecall(form1.edit1.text);
end;

procedure TForm3.BitBtn3Click(Sender: TObject);
begin
Form1.Telefoon1.calling:=false;
Animate1.Active:=false;
end;

end.
```

Підпрограма парольного захисту

```
unit Unit4;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, Mask, ExtCtrls;
type
  TForm4 = class(TForm)
    Panel1: TPanel;
    M: TMaskEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form4: TForm4;
implementation
uses usimple;
VAR
  KEY:string='2%#$T%&(*qwrdas@@@#45';
  DAT:string='VUW';
{$R *.DFM}
        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;
procedure TForm4.BitBtn1Click(Sender: TObject);
var
  F: TextFile;
  i:integer;
  s:string;
  Y:string;
  UUU:string;
begin
if FileExists('main.dat')=TRUE then
 begin
   AssignFile(F, 'main.dat');
   Reset(F);
   Readln(F, S);
   CloseFile(F);
UUU:=Crypt(M.text,KEY,false);
if UUU=S then
      begin
           Form4.hide;
           Form1.show;
           MessageDlg('Дякую, пароль вірний!',mtInformation,[mbOK],0);
      end
      else MessageDlg('Введіть пароль!',mtInformation,[mbOK],0);
 end
      else       begin
    MessageDlg('Файл       main.dat       не       знайдено!       завершення
програми',mtInformation,[mbOK],0);
    Application.Terminate;
      end;
end;
end.
```

```
unit AMixer;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  MMSystem;

type
  TAudioMixer=class;

  TPListFreeItemNotify=procedure (Pntr:Pointer) of object;
  TMixerChange=procedure (Sender:TObject;MixerH:HMixer;ID:Integer) of object;
    {MixerH is handle of mixer, which sent this message.
     ID is ID of changed item (line or control).}

  TPointerList=class(TObject)
  private
    FOnFreeItem:TPListFreeItemNotify;
    Items:Tlist;
  protected
    function GetPointer (Ind:Integer):Pointer;
    function GetCount :integer;
  public
    constructor Create;
    destructor Destroy; override;
    procedure Clear;
    procedure Add (Pntr:Pointer);
    property Count:Integer read GetCount;
    property Pointer[Ind:Integer]:Pointer read GetPointer; default;
    property    OnFreeItem:TPListFreeItemNotify    read    FOnFreeItem    write
FOnFreeItem;
  end;

  TMixerControls=class(TObject)
  private
    heap:pointer;
    FControls:TPointerList;
  protected
    function GetControl (Ind:Integer):PMixerControl;
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Control[Ind:Integer]:PMixerControl read GetControl; default;
    property Count:Integer read GetCount;
  end;

  TMixerConnection=class(TObject)
  private
    XMixer:TAudioMixer;
    FData:TMixerLine;
    FControls:TMixerControls;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Controls:TMixerControls read FControls;
    property Data:TMixerLine read FData;
  end;

  TMixerConnections=class(TObject)
  private
    XMixer:TAudioMixer;
    FConnections:TPointerList;
  protected
    procedure DoFreeItem (Pntr:Pointer);
    function GetConnection (Ind:Integer):TMixerConnection;
```

```
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property  Connection[Ind:Integer]:TMixerConnection  read  GetConnection;
default;
    property Count:Integer read GetCount;
  end;

  TMixerDestination=class(TObject)
  private
    XMixer:TAudioMixer;
    FData:TMixerLine;
    FControls:TMixerControls;
    FConnections:TMixerConnections;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Connections:TMixerConnections read FConnections;
    property Controls:TMixerControls read FControls;
    property Data:TMixerLine read FData;
  end;

  TMixerDestinations=class(TObject)
  private
    FDestinations:TPointerList;
  protected
    function GetDestination (Ind:Integer):TMixerDestination;
    procedure DoFreeItem (Pntr:Pointer);
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer);
    destructor Destroy; override;
    property Count:Integer read GetCount;
    property  Destination[Ind:Integer]:TMixerDestination  read  GetDestination;
default;
  end;

  TAudioMixer = class(TComponent)
  private
    XWndHandle:HWnd;

    FDestinations:TMixerDestinations;
    FMixersCount:Integer;
    FMixerHandle:HMixer;
    FMixerId:Integer;
    FMixerCaps:TMixerCaps;
    FDriverVersion: MMVERSION;
    FManufacturer: String;
    FProductId: Word;
    FNumberOfLine: Integer;
    FProductName: String;
    FOnLineChange:TMixerChange;
    FOnControlChange:TMixerChange;
  protected
    procedure SetMixerId (Value:Integer);
    procedure MixerCallBack (var Msg:TMessage);
    procedure CloseMixer;
  published
    constructor Create (AOwner:TComponent); override;
    destructor Destroy; override;
    property DriverVersion: MMVERSION read FDriverVersion;
    property ProductId: WORD read FProductId;
    property NumberOfLine: Integer read FNumberOfLine;
    property Manufacturer: string read FManufacturer;
    property ProductName: string read FProductName;
    property MixerId:Integer read FMixerId write SetMixerId;
      {Opened mixer - value must be in range 0..MixersCount-1
       If no mixer is opened this value is  -1}
```

```
    property OnLineChange:TMixerChange read FOnLineChange write FOnLineChange;
    property OnControlChange:TMixerChange read FOnControlChange write
FOnControlChange;
  public
    function GetVolume (ADestination, AConnection:Integer; var LeftVol,
RightVol, Mute:Integer; var Stereo, VolDisabled, MuteDisabled,
MuteIsSelect:Boolean):Boolean;
      {This function return volume of selected Destination and Connection.
       ADestination must be from range 0..Destinations.Count-1
       AConnection must be in range
0..Destinations[ADestination].Connections.Count-1
       If you want to read master volume of some Destination, you have to
         set AConnection to -1.
       If LeftVol, RightVol or Mute is not supported by queried connection,
         it's return value will be -1.

       LeftVol and RightVol are in range 0..65536

       If Mute is non-zero then the connection is silent (or vice-versa - see
MuteIsSelect parameter)
       If specified line is recording source then Mute specifies if programs
will
         record from this connection (it is copy of "Select" Checkbox in
         standard Windows Volume Control program)
       Stereo is true, then this control is stereo.
       VolDisabled or MuteDisabled is True when you cannot apply settings to
this
         control (but can read it).
       MuteIsSelect returns True is "mute" work here as select - opposite of
mute.

       Return value of the function is True if no error has occured,
         otherwise it returns False.}
    function SetVolume (ADestination, AConnection:Integer; LeftVol, RightVol,
Mute:Integer):Boolean;
      {This function sets volume.
       If you set RightVol to -1 and connection is stereo then LeftVol will be
         copied to RightVol.
       If LeftVol or Mute is -1 then this value will not be set.
       Note that "Mute" can be "select" (which is reversed mute) - see
function
         GetVolume, parameter MuteIsSelect.

       Return value is True if ADestination and AConnection are correct,
otherwise False.}

    function GetPeak(ADestination, AConnection:Integer; var LeftPeak,
RightPeak:Integer):Boolean;
    function GetMute(ADestination, AConnection:Integer; var
Mute:Boolean):Boolean;
    function SetMute(ADestination, AConnection:Integer; Mute:Boolean):Boolean;

    property Destinations:TMixerDestinations read FDestinations;
      {Ind must be in range 0..DestinationsCount-1}
    property MixerCaps:TMixerCaps read FMixerCaps;
    property MixerCount:Integer read FMixersCount;
      {Number of mixers present in system; mostly 1}
    property MixerHandle:HMixer read FMixerHandle;
      {Handle of opened mixer}
  end;

procedure Register;

implementation

{------------}
{TPointerList}
{------------}
```

```
constructor TPointerList.Create;
begin
  Items := TList.Create;
end;

destructor TPointerList.Destroy;
begin
  Clear;
  Items.Free;
end;

procedure TPointerList.Add (Pntr:Pointer);
begin
  Items.Add (Pntr);
end;

function TPointerList.GetPointer (Ind:Integer):Pointer;
begin
  Result := nil;
  If (Ind < Count) then
    Result := Items[Ind];
end;

procedure TPointerList.Clear;
var I:Integer;
begin
  for I := 0 to Items.Count-1 do begin
    If Assigned (FOnFreeItem) then
      FOnFreeItem (Items[I])
  end;
  Items.Clear;
end;

function TPointerList.GetCount:Integer;
begin
  Result := Items.Count;
end;

{--------------}
{TMixerControls}
{--------------}
constructor TMixerControls.Create (AMixer:TAudioMixer; AData:TMixerLine);
var MLC:TMixerLineControls;
    A,B:Integer;
    P:PMixerControl;
begin
  FControls := TPointerList.Create;
  GetMem (P, SizeOf(TMixerControl)*AData.cControls);
  heap := P;
  MLC.cbStruct := SizeOf(MLC);
  MLC.dwLineID := AData.dwLineID;
  MLC.cbmxctrl := SizeOf(TMixerControl);
  MLC.cControls := AData.cControls;
  MLC.pamxctrl := P;
  A          :=          MixerGetLineControls(AMixer.MixerHandle,          @MLC,
MIXER_GETLINECONTROLSF_ALL);
  If A = MMSYSERR_NOERROR then
  begin
    For B := 0 to AData.cControls-1 do
    begin
      FControls.Add (P);
      P := PMixerControl (DWORD(P) + sizeof (TMixerControl));
    end;
  end;
end;

destructor TMixerControls.Destroy;
begin
  FControls.free;
```

```
  freemem(heap);
  inherited;
end;

function TMixerControls.GetControl (Ind:Integer):PMixerControl;
begin
  Result := FControls.Pointer[Ind];
end;

function TMixerControls.GetCount:Integer;
begin
  Result := FControls.Count;
end;

{----------------}
{TMixerConnection}
{----------------}

constructor TMixerConnection.Create (AMixer:TAudioMixer; AData:TMixerLine);
begin
  FData := AData;
  XMixer := AMixer;
  FControls := TMixerControls.Create (AMixer, AData);
end;

destructor TMixerConnection.Destroy;
begin
  FControls.Free;
  inherited;
end;

{-----------------}
{TMixerConnections}
{-----------------}

constructor TMixerConnections.Create (AMixer:TAudioMixer; AData:TMixerLine);
var A,B:Integer;
    ML:TMixerLine;
begin
  XMixer := AMixer;
  FConnections := TPointerList.Create;
  FConnections.OnFreeItem := Dofreeitem;
  ML.cbStruct := SizeOf(TMixerLine);
  ML.dwDestination := AData.dwDestination;
  For A := 0 to AData.cConnections-1 do
  begin
    ML.dwSource := A;
    B       :=       MixerGetLineInfo       (AMixer.MixerHandle,       @ML,
MIXER_GETLINEINFOF_SOURCE);
    If B = MMSYSERR_NOERROR then
      FConnections.Add (Pointer(TMixerConnection.Create (XMixer, ML)));
  end;
end;

destructor TMixerConnections.Destroy;
begin
  FConnections.Free;
  inherited;
end;

procedure TMixerConnections.DoFreeItem (Pntr:Pointer);
begin
  TMixerConnection(Pntr).Free;
end;

function TMixerConnections.GetConnection (Ind:Integer):TMixerConnection;
begin
  Result := FConnections.Pointer[Ind];
end;
```

```
function TMixerConnections.GetCount:Integer;
begin
  Result := FConnections.Count;
end;

{-----------------}
{TMixerDestination}
{-----------------}

constructor TMixerDestination.Create (AMixer:TAudioMixer; AData:TMixerLine);
begin
  FData := AData;
  XMixer := AMixer;
  FConnections := TMixerConnections.Create (XMixer, FData);
  FControls := TMixerControls.Create (XMixer, AData);
end;

destructor TMixerDestination.Destroy;
begin
  Fcontrols.Free;
  FConnections.Free;
  inherited;
end;

{------------------}
{TMixerDestinations}
{------------------}

constructor TMixerDestinations.Create (AMixer:TAudioMixer);
var A,B:Integer;
    ML:TMixerLine;
begin
  FDestinations := TPointerList.Create;
  FDestinations.OnFreeItem := DoFreeItem;
  if (AMixer = nil) then
    Exit;
  For A := 0 to AMixer.MixerCaps.cDestinations-1 do
  begin
    ML.cbStruct := SizeOf(TMixerLine);
    ML.dwDestination := A;
    B       :=      MixerGetLineInfo       (AMixer.MixerHandle,       @ML,
MIXER_GETLINEINFOF_DESTINATION);
    If B = MMSYSERR_NOERROR then
      FDestinations.Add (Pointer(TMixerDestination.Create (AMixer, ML)));
  end;
end;

procedure TMixerDestinations.DoFreeItem (Pntr:Pointer);
begin
  TMixerDestination(Pntr).Free;
end;

destructor TMixerDestinations.Destroy;
begin
  FDestinations.Free;
  inherited;
end;

function TMixerDestinations.GetDestination (Ind:Integer):TMixerDestination;
begin
  Result := nil;
  If (Assigned (FDestinations)) then
    Result := FDestinations.Pointer[Ind];
end;

function TMixerDestinations.GetCount:Integer;
begin
  Result := FDestinations.Count;
```

```
end;

{-----------}
{TAudioMixer}
{-----------}

constructor TAudioMixer.Create (AOwner:TComponent);
begin
  inherited Create (AOwner);
  XWndHandle := AllocateHWnd (MixerCallBack);
  FMixersCount := mixerGetNumDevs;
  FMixerId := -1;
  if (FMixersCount = 0) then
    FDestinations := TMixerDestinations.Create (nil)
  else
  begin
    FDestinations := nil;
    SetMixerId (0);
  end;
end;

destructor TAudioMixer.Destroy;
begin
  CloseMixer;
  if XWndHandle <> 0 then
    DeAllocateHwnd (XWndHandle);
  inherited;
end;

procedure TAudioMixer.CloseMixer;
begin
  If FMixerId >= 0 then
  begin
    mixerClose (FMixerHandle);
    FMixerId := -1;
  end;
  FDestinations.Free;
  FDestinations := nil;
end;

procedure TAudioMixer.SetMixerId (Value:Integer);
label AllOK;
begin
  If (Value < 0) OR (Value >= FMixersCount) then
    Exit;
  CloseMixer;

  If mixerOpen (@FMixerHandle, Value, XWndHandle, 0, CALLBACK_WINDOW OR
MIXER_OBJECTF_MIXER) = MMSYSERR_NOERROR then
    goto AllOK;

  // we will go here very rarely, but sometimes it could help
  If mixerOpen (@FMixerHandle, Value, XWndHandle, 0, CALLBACK_WINDOW) =
MMSYSERR_NOERROR then
    goto AllOK;
  If mixerOpen (@FMixerHandle, Value, 0, 0, 0) = MMSYSERR_NOERROR then
    goto AllOK;

  // an error has occured
  FMixerId := -1;
  FDestinations := TMixerDestinations.Create (nil);

  Exit;

AllOK:
  FMixerId := Value;
  mixerGetDevCaps (MixerId, @FMixerCaps, SizeOf (TMixerCaps));

  if FMixerCaps.wMid = MM_MICROSOFT then
```

```
    FManufacturer := 'Microsoft'
  else
    FManufacturer := IntToStr(FMixerCaps.wMid) + ' = Unknown';
  FDriverVersion := FMixerCaps.vDriverVersion;
  FProductId := FMixerCaps.wPid;
  FProductName := StrPas(FMixerCaps.szPName);
  FNumberOfLine := FMixerCaps.cDestinations;

  FDestinations := TMixerDestinations.Create (Self);
end;

procedure TAudioMixer.MixerCallBack (var Msg:TMessage);
begin
  case Msg.Msg of
    MM_MIXM_LINE_CHANGE:
        If Assigned (OnLineChange) then
          OnLineChange (Self, Msg.wParam, Msg.lParam);
    MM_MIXM_CONTROL_CHANGE:
        If Assigned (OnControlChange) then
          OnControlChange (Self, Msg.wParam, Msg.lParam);
    else
      Msg.Result   :=   DefWindowProc   (XWndHandle,   Msg.Msg,   Msg.WParam,
Msg.LParam);
  end;
end;

const MIXER_LONG_NAME_CHARS = 64;

type MIXERCONTROLDETAILS_LISTTEXT = record
       dwParam1:DWORD;
       dwParam2:DWORD;
       szName:Array [0..MIXER_LONG_NAME_CHARS-1] of Char;
     end;

type ListTextArray = array [0..1000] of MIXERCONTROLDETAILS_LISTTEXT;

function  TAudioMixer.GetVolume  (ADestination,AConnection:Integer;var  LeftVol,
RightVol,     Mute:Integer;var     Stereo,     VolDisabled,     MuteDisabled,
MuteIsSelect:Boolean):Boolean;
var MD:TMixerDestination;
    MC:TMixerConnection;
    Cntrls:TMixerControls;
    MCD:TMixerControlDetails;
    Cntrl:PMixerControl;
    A,B:Integer;
    ML:TMixerLine;
    details:array [0..100] of Integer;
    ltext:^ListTextArray;
    MCDText:TMixerControlDetails;

begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  Stereo := False;
  MuteDisabled := True;
  VolDisabled := True;
  LeftVol := -1;
  RightVol := -1;
  Mute := -1;
  MuteIsSelect := False;
  MD := Destinations[ADestination];
  MC := nil;
  If AConnection <> -1 then
    MC := MD.Connections[AConnection];

  If MD <> nil then
  begin
```

```
    Result := True;

//      If Mute = -1 then
    begin
      If AConnection <> -1 then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Mute = -1) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
//                If (Cntrl.dwControlType AND MIXERCONTROL_CONTROLTYPE_MIXER =
MIXERCONTROL_CONTROLTYPE_MIXER) OR
//                     (Cntrl.dwControlType AND MIXERCONTROL_CONTROLTYPE_MUX =
MIXERCONTROL_CONTROLTYPE_MUX) then

//                   If (Cntrl.dwControlType AND MIXERCONTROL_CT_CLASS_MASK =
MIXERCONTROL_CT_CLASS_LIST) then

            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
               // Mux is similar to mixer, but only one line can be selected
at a time
            begin
              MCD.cbStruct := SizeOf(TMixerControlDetails);
              MCD.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCD.cChannels := 1
              else
                MCD.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCD.cMultipleItems := Cntrl.cMultipleItems
              else
                MCD.cMultipleItems := 0;
              MCD.cbDetails := 4;
              MCD.paDetails := @Details;
              B                    :=                    mixerGetControlDetails
(FMixerHandle,@MCD,MIXER_GETCONTROLDETAILSF_VALUE);
              If B <> MMSYSERR_NOERROR then
              begin
                Inc (A);
                continue;
              end;

              MCDText.cbStruct := sizeof (MCDText);
              MCDText.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCDText.cChannels := 1
              else
                MCDText.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCDText.cMultipleItems := Cntrl.cMultipleItems
              else
                MCDText.cMultipleItems := 0;
              GetMem  (ltext,  MCDText.cChannels  *  MCDText.cMultipleItems  *
sizeof (MIXERCONTROLDETAILS_LISTTEXT));
              MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
              MCDText.paDetails := ltext;
              B    :=    mixerGetControlDetails    (FMixerHandle,    @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
              If B <> MMSYSERR_NOERROR then
              begin
                FreeMem (ltext);
                Inc (A);
                continue;
```

```
              end;
              B := MCD.cChannels - 1;
              while (B < integer(MCD.cMultipleItems)) do
              begin
                if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                  break;
                Inc (B, MCD.cChannels);
              end;
              FreeMem (ltext);

              If (B < integer (MCD.cMultipleItems)) then
              begin
                Mute := Details[B];
                MuteDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                MuteIsSelect := True;
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;
    end;

    If AConnection = -1 then
    begin
      Cntrls := MD.Controls;
      ML := MD.Data;
    end
    else
    begin
      If MC <> nil then
      begin
        Cntrls := MC.Controls;
        ML := MC.Data;
      end
      else
        Cntrls := nil;
    end;
    If Cntrls <> nil then
    begin
      A := 0;
      while ((LeftVol = -1) OR (Mute = -1)) AND (A < Cntrls.Count) do
      begin
        Cntrl := Cntrls[A];
        If Cntrl <> nil then
        begin
          If ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) OR
              ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) AND (Mute
= -1))) AND
             (Cntrl.fdwControl     AND     MIXERCONTROL_CONTROLF_MULTIPLE     <>
MIXERCONTROL_CONTROLF_MULTIPLE)
            then
          begin
            if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) then
              MCD.cbStruct := SizeOf(TMixerControlDetails)
            else
              MCD.cbStruct := SizeOf(TMixerControlDetails);
            MCD.dwControlID := Cntrl.dwControlID;
            If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
              MCD.cChannels := 1
            else
              MCD.cChannels := ML.cChannels;
            MCD.cMultipleItems := 0;
            MCD.cbDetails := SizeOf(Integer);
            MCD.paDetails := @details;
            B      :=      mixerGetControlDetails      (FMixerHandle,      @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
```

```
            If B = MMSYSERR_NOERROR then
            begin
              If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) AND
(LeftVol = -1) then
              begin
                VolDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                LeftVol := details[0];
                If MCD.cChannels > 1 then
                begin
                  RightVol := Details[1];
                  Stereo := True;
                end
                else
                  RightVol := LeftVol;
              end
              else If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE)
AND (Mute = -1) then
              begin
                MuteDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                If Details[0] <> 0 then
                  Mute := 1
                else
                  Mute := 0;
              end
// NEW ->
(*              else If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF)
AND (Mute = -1) then
              begin
                MuteDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                If Details[0] <> 0 then
                  Mute := 1
                else
                  Mute := 0;
                MuteIsSelect := True;
              end;*)
// <- NEW
            end;
          end;
        end;
        Inc (A);
      end;

{      If LeftVol = -1 then
        VolDisabled := True;
      If Mute = -1 then
        MuteDisabled := True;}
    end;
  end;
end;

function TAudioMixer.SetVolume (ADestination, AConnection:Integer; LeftVol,
RightVol, Mute:Integer):Boolean;
var MD:TMixerDestination;
    MC:TMixerConnection;
    Cntrls:TMixerControls;
    MCD:TMixerControlDetails;
    Cntrl:PMixerControl;
    A,B:Integer;
    ML:TMixerLine;
    details:array [0..100] of Integer;
    VolSet,MuteSet:Boolean;
    ltext:^ListTextArray;
    MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
```

```
    Exit;
  MC := nil;
  MD := Destinations[ADestination];
  If MD <> nil then
  begin
    If AConnection <> -1 then
      MC := MD.Connections[AConnection];

    VolSet := LeftVol = -1;
    MuteSet := Mute = -1;
    Result := True;

    If not MuteSet then
    begin
      If AConnection <> -1 then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while not MuteSet AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (*(Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR*)
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              MCD.cbStruct := SizeOf(TMixerControlDetails);
              MCD.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCD.cChannels := 1
              else
                MCD.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCD.cMultipleItems := Cntrl.cMultipleItems
              else
                MCD.cMultipleItems := 0;
              MCD.cbDetails := 4;
              MCD.paDetails := @Details;
              MuteSet := True;
              mixerGetControlDetails            (FMixerHandle,            @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
              if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
                For B := 0 to Cntrl.cMultipleItems-1 do
                  Details[B] := 0;

              GetMem  (ltext,  MCD.cChannels  *  MCD.cMultipleItems  *  sizeof
(MIXERCONTROLDETAILS_LISTTEXT));
              MCDText.cbStruct := sizeof (MCDText);
              MCDText.dwControlID := Cntrl.dwControlID;
              MCDText.cChannels := MCD.cChannels;
              MCDText.cMultipleItems := MCD.cMultipleItems;
              MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
              MCDText.paDetails := ltext;
              mixerGetControlDetails          (FMixerHandle,          @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
              B := MCD.cChannels - 1;
              while (B < integer (MCD.cMultipleItems)) do
              begin
                if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                  break;
                Inc (B, MCD.cChannels);
              end;
              FreeMem (ltext);

              If (B < integer (MCD.cMultipleItems)) then
              begin
                Details[B] := Mute;
```

```
                mixerSetControlDetails          (FMixerHandle,          @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;
    end;



    If AConnection = -1 then
    begin
      Cntrls := MD.Controls;
      ML := MD.Data;
    end
    else
    begin
      If MC <> nil then
      begin
        Cntrls := MC.Controls;
        ML := MC.Data;
      end
      else
        Cntrls := nil;
    end;
    If Cntrls <> nil then
    begin

      A := 0;
      while (not VolSet OR not MuteSet) AND (A < Cntrls.Count) do
      begin
        Cntrl := Cntrls[A];
        If Cntrl <> nil then
        begin
          If (((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) AND not
VolSet) OR
              ((Cntrl.dwControlType  =  MIXERCONTROL_CONTROLTYPE_MUTE)  AND  not
MuteSet) (* NEW -> *) (*OR
              ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF) AND not
MuteSet)*) (* <- NEW *)) AND
             (Cntrl.fdwControl    AND    MIXERCONTROL_CONTROLF_MULTIPLE    <>
MIXERCONTROL_CONTROLF_MULTIPLE)
              then
          begin
            MCD.cbStruct := SizeOf(TMixerControlDetails);
            MCD.dwControlID := Cntrl.dwControlID;
            If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
              MCD.cChannels := 1
            else
              MCD.cChannels := ML.cChannels;
            MCD.cMultipleItems := 0;
            MCD.cbDetails := SizeOf(Integer);
            MCD.paDetails := @Details;
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) then
            begin
              Details[0] := LeftVol;
              If RightVol = -1 then
                Details[1] := LeftVol
              else
                Details[1] := RightVol;
              VolSet := True;
            end
            else if ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) (*
NEW -> *) (* OR
                     (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF) *)
(* <- NEW *)) then
```

```
            begin
              For B := 0 to MCD.cChannels - 1 do
                Details[B] := Mute;
              MuteSet := True;
            end;
            mixerSetControlDetails             (FMixerHandle,             @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
          end;
        end;
        Inc (A);
      end;

    end;
  end;
end;

function  TAudioMixer.GetMute(ADestination,  AConnection:  Integer;  var  Mute:
Boolean):Boolean;
var
  MD : TMixerDestination;
  MC : TMixerConnection;
  mlcMixerLineControlsMute : TMIXERLINECONTROLS;
  mcdMixerDataMute : TMIXERCONTROLDETAILS;
  pmcMixerControlMute : PMIXERCONTROL;
  pmcdsMixerDataUnsignedMute : PMIXERCONTROLDETAILSBOOLEAN;
  mlMixerLine : TMixerLine;
  Cntrl:PMixerControl;
  Cntrls:TMixerControls;
  ML:TMixerLine;
  A,B:Integer;
  details:array [0..100] of Integer;
  ltext:^ListTextArray;
  MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  MC := nil;
  Mute := False;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
      mlMixerLine := MD.Data
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
        mlMixerLine := MC.Data
      else
        Exit;
    end;

    GetMem(pmcMixerControlMute, SizeOf(TMIXERCONTROL));
    GetMem(pmcdsMixerDataUnsignedMute, SizeOf(TMIXERCONTROLDETAILSBOOLEAN));

    with mlcMixerLineControlsMute do
    begin
      cbStruct := SizeOf(TMIXERLINECONTROLS);
      dwLineID := mlMixerLine.dwLineID;
      dwControlType := MIXERCONTROL_CONTROLTYPE_MUTE;
      cControls := 1;
      cbmxctrl := SizeOf(TMIXERCONTROL);
      pamxctrl := pmcMixerControlMute;
    end;

    if      (mixerGetLineControls(FMixerHandle,      @mlcMixerLineControlsMute,
MIXER_GETLINECONTROLSF_ONEBYTYPE) = MMSYSERR_NOERROR) then
    begin
```

```
      with mcdMixerDataMute do
      begin
        cbStruct := SizeOf(TMIXERCONTROLDETAILS);
        dwControlID := pmcMixerControlMute^.dwControlID;
        cChannels := 1;
        cMultipleItems := 0;
        cbDetails := SizeOf(TMIXERCONTROLDETAILSBOOLEAN);
        paDetails := pmcdsMixerDataUnsignedMute;
      end;

      if        mixerGetControlDetails(FMixerHandle,        @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) = MMSYSERR_NOERROR then
      begin
        Mute := pmcdsMixerDataUnsignedMute^.fValue = 1;
        Result := True;
      end;
    end
    else
    begin
      If (AConnection <> -1) then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Result = False) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              mcdMixerDataMute.cbStruct := SizeOf(TMixerControlDetails);
              mcdMixerDataMute.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                mcdMixerDataMute.cChannels := 1
              else
                mcdMixerDataMute.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                mcdMixerDataMute.cMultipleItems := Cntrl.cMultipleItems
              else
                mcdMixerDataMute.cMultipleItems := 0;
              mcdMixerDataMute.cbDetails := 4;
              mcdMixerDataMute.paDetails := @Details;
              mixerGetControlDetails     (FMixerHandle,     @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE);

              GetMem        (ltext,        mcdMixerDataMute.cChannels        *
mcdMixerDataMute.cMultipleItems * sizeof (MIXERCONTROLDETAILS_LISTTEXT));
              MCDText.cbStruct := sizeof (MCDText);
              MCDText.dwControlID := Cntrl.dwControlID;
              MCDText.cChannels := mcdMixerDataMute.cChannels;
              MCDText.cMultipleItems := mcdMixerDataMute.cMultipleItems;
              MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
              MCDText.paDetails := ltext;
              mixerGetControlDetails          (FMixerHandle,          @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
              B := mcdMixerDataMute.cChannels - 1;
              while (B < integer (mcdMixerDataMute.cMultipleItems)) do
              begin
                if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                  break;
                Inc (B, mcdMixerDataMute.cChannels);
              end;
              FreeMem (ltext);

              If (B < integer (mcdMixerDataMute.cMultipleItems)) then
              begin
```

```
                Result := True;
                Mute := Details[B] <> 0;
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;
    end;

    FreeMem(pmcdsMixerDataUnsignedMute);
    FreeMem(pmcMixerControlMute);
  end;
end;

function   TAudioMixer.SetMute(ADestination,   AConnection:   Integer;   Mute:
Boolean):Boolean;
var
  MD : TMixerDestination;
  MC : TMixerConnection;
  mlcMixerLineControlsMute : TMIXERLINECONTROLS;
  mcdMixerDataMute : TMIXERCONTROLDETAILS;
  pmcMixerControlMute : PMIXERCONTROL;
  pmcdsMixerDataUnsignedMute : PMIXERCONTROLDETAILSBOOLEAN;
  mlMixerLine : TMixerLine;
  Cntrl:PMixerControl;
  Cntrls:TMixerControls;
  ML:TMixerLine;
  A,B:Integer;
  details:array [0..100] of Integer;
  ltext:^ListTextArray;
  MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  MC := nil;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
      mlMixerLine := MD.Data
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
        mlMixerLine := MC.Data
      else
        Exit;
    end;

    GetMem(pmcMixerControlMute, SizeOf(TMIXERCONTROL));
    GetMem(pmcdsMixerDataUnsignedMute, SizeOf(TMIXERCONTROLDETAILSBOOLEAN));

    with mlcMixerLineControlsMute do
    begin
      cbStruct := SizeOf(TMIXERLINECONTROLS);
      dwLineID := mlMixerLine.dwLineID;
      dwControlType := MIXERCONTROL_CONTROLTYPE_MUTE;
      cControls := 0;
      cbmxctrl := SizeOf(TMIXERCONTROL);
      pamxctrl := pmcMixerControlMute;
    end;

    if      (mixerGetLineControls(FMixerHandle,      @mlcMixerLineControlsMute,
MIXER_GETLINECONTROLSF_ONEBYTYPE) = MMSYSERR_NOERROR) then
    begin
      with mcdMixerDataMute do
```

```
      begin
        cbStruct := SizeOf(TMixerControlDetails);
        dwControlID := pmcMixerControlMute^.dwControlID;
        cChannels := 1;
        cMultipleItems := 0;
        cbDetails := SizeOf(TMIXERCONTROLDETAILSBOOLEAN);
        paDetails := pmcdsMixerDataUnsignedMute;
      end;

      if Mute then
        pmcdsMixerDataUnsignedMute^.fValue := 1
      else
        pmcdsMixerDataUnsignedMute^.fValue := 0;

      if
(mixerSetControlDetails(FMixerHandle,@mcdMixerDataMute,MIXER_SETCONTROLDETAILS
F_VALUE) = MMSYSERR_NOERROR) then
        Result := True;
    end
    else
    begin
      If (AConnection <> -1) then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Result = False) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              mcdMixerDataMute.cbStruct := SizeOf(TMixerControlDetails);
              mcdMixerDataMute.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                mcdMixerDataMute.cChannels := 1
              else
                mcdMixerDataMute.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                mcdMixerDataMute.cMultipleItems := Cntrl.cMultipleItems
              else
                mcdMixerDataMute.cMultipleItems := 0;
              mcdMixerDataMute.cbDetails := 4;
              mcdMixerDataMute.paDetails := @Details;
              if   (mixerGetControlDetails   (FMixerHandle,   @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) <> MMSYSERR_NOERROR) then
              begin
                Inc (A);
                continue;
              end;
              if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
                For B := 0 to Cntrl.cMultipleItems-1 do
                  Details[B] := 0;
              If Mute then
              begin
                GetMem       (ltext,       mcdMixerDataMute.cChannels       *
mcdMixerDataMute.cMultipleItems * sizeof (MIXERCONTROLDETAILS_LISTTEXT));
                MCDText.cbStruct := sizeof (MCDText);
                MCDText.dwControlID := Cntrl.dwControlID;
                MCDText.cChannels := mcdMixerDataMute.cChannels;
                MCDText.cMultipleItems := mcdMixerDataMute.cMultipleItems;
                MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
                MCDText.paDetails := ltext;
                mixerGetControlDetails        (FMixerHandle,        @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
                B := mcdMixerDataMute.cChannels - 1;
```

```
                while (B < integer (mcdMixerDataMute.cMultipleItems)) do
                begin
                  if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                    break;
                  Inc (B, mcdMixerDataMute.cChannels);
                end;
                FreeMem (ltext);

                If (B < integer (mcdMixerDataMute.cMultipleItems)) then
                  Details[B] := 1;
              end;
              if   (mixerSetControlDetails   (FMixerHandle,   @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) = MMSYSERR_NOERROR) then
              begin
                Result := True;
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;
    end;

    FreeMem(pmcdsMixerDataUnsignedMute);
    FreeMem(pmcMixerControlMute);
  end;
end;

function TAudioMixer.GetPeak (ADestination, AConnection:Integer; var LeftPeak,
RightPeak:Integer): Boolean;
var
  MD : TMixerDestination;
  MC : TMixerConnection;
  mcdMixerDataPeak : TMIXERCONTROLDETAILS;
  pmcMixerControlPeak : PMIXERCONTROL;
{  pmcdsMixerDataSignedPeak : PMIXERCONTROLDETAILSSIGNED;}
  mlMixerLine : TMixerLine;
  A:Integer;
  Cntrls:TMixerControls;
  Details:Array [1..100] of Integer;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  LeftPeak := 0;
  RightPeak := 0;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
    begin
      mlMixerLine := MD.Data;
      Cntrls := MD.Controls;
    end
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
      begin
        mlMixerLine := MC.Data;
        Cntrls := MC.Controls;
      end
      else
        Exit;
    end;
    GetMem(pmcMixerControlPeak, SizeOf(TMIXERCONTROL));

    A := 0;
```

```
    while (A < Cntrls.Count) do
    begin
      If    (Cntrls[A].dwControlType    AND    MIXERCONTROL_CT_CLASS_MASK)    =
MIXERCONTROL_CT_CLASS_METER then
        break;
      Inc (A);
    end;
    If A = Cntrls.Count then
    begin
      FreeMem(pmcMixerControlPeak);
      Exit;
    end;

    with mcdMixerDataPeak do
    begin
      cbStruct := SizeOf(TMIXERCONTROLDETAILS);
      dwControlID := Cntrls[A].dwControlID;
      cChannels := mlMixerLine.cChannels;
      If    (Cntrls[A].fdwControl    AND    MIXERCONTROL_CONTROLF_MULTIPLE)    =
MIXERCONTROL_CONTROLF_MULTIPLE then
        cMultipleItems:=Cntrls[A].cMultipleItems
      else
        cMultipleItems:=0;
      cbDetails := SizeOf(TMIXERCONTROLDETAILSSIGNED);
      paDetails := @Details;
    end;
    if
(mixerGetControlDetails(FMixerHandle,@mcdMixerDataPeak,MIXER_GETCONTROLDETAILS
F_VALUE) = MMSYSERR_NOERROR) then
    begin
      LeftPeak := Details[1];
      if mlMixerLine.cChannels = 2 then
        RightPeak := Details[2]
      else
        RightPeak := LeftPeak;
      Result := True;
    end;
    FreeMem(pmcMixerControlPeak);
   end;
end;
procedure Register;
begin
  RegisterComponents('Samples', [TAudioMixer]);
end;
end.
```