Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти

на тему

“Програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях”

Виконав здобувач вищої освіти
IV курсу, групи КІ-19ПЗ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Балюк В.С.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Кафедра_КбПЗ_2023рік

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
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 2023 року

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

Балюк Вероніці Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях*

2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 15-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях*

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*

Кафедра_КБПЗ_2023рік

7. Дата видачі завдання « 17 » січня 2023 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	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Дрєєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Балюк В.С.
(прізвище та ініціали)

Кафедра_КБПЗ_2023рік

АНОТАЦІЯ

Балюк В.С. Програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

Метою розробки є програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

Результат роботи – програмна реалізація системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

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

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

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, паралельний порт

Кафедра_КОПЗ_2023рік

ABSTRACT

Baliuk V.S. Software of the visualization system of the parallel port information exchange process for educational purposes. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software was developed, which is intended for the visualization system of the parallel port information exchange process for educational purposes.

The purpose of the development is the software of the visualization system of the parallel port information exchange process for educational purposes.

The result of the work is the software implementation of the visualization system of the parallel port information exchange process for educational purposes.

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 PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Builder C++ environment.

Keywords: computer engineering, parallel port

Кафедра_КбПЗ_2023рік

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи 5
1.2 Область застосування 11
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 13
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 13
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 18
2.3 Розгорнута постановка завдання 20
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 21
3.1 Опис функціонування системи 21
3.2 Розробка структурної схеми 35
3.3 Розробка функціональної схеми 40
3.4 Розробка діаграми процесів 42
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 44
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 44
4.2 Захист розробленого програмного забезпечення 52
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 59
6 ОСНОВНІ ВИСНОВКИ 61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63

Кафедра_КБПЗ_2023рік

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІТ	–	інформаційні технології
ОТ	–	обчислювальна техніка
ПЗ	–	програмне забезпечення
ПІТ	–	перспективні інформаційні технології

Кафедра_К6ПЗ_2023рік

ВСТУП

Актуальність теми. Сучасний навчальний процес неможливо представити без повсюдного впровадження комп’ютерної техніки та інформаційних технологій. Оцінюючи ефективність впровадження комп'ютера в навчальний процес, можна виділити дві категорії.

Перша – підвищення успішності по окремим предметам, тобто забезпечення орієнтованого на результат підходу. Застосування ЕОМ у цьому випадку в основному пов'язане з діючими навчальними програмами й покликано полегшити, прискорити й зробити більше досконалим процес навчання.

Друга – розвиток загальних здатностей: вирішувати поставлені завдання, самостійно мислити, володіти комунікативними навичками (збір, аналіз, синтез інформації), тобто упор на процеси, що лежать в основі формування тієї або іншої навички.

Перша персоніфікує традицію, обумовлену концепцією навчання як системи стимульно-реактивних відносин. На цій основі розроблені програми, що вимагають відповідей, від тих хто вчаться, які забезпечують поступове їхнє просування до поставленим цілям навчання. У цьому випадку навчання визначається як піддається кількісному виміру зміна в навченості.

Цілі другої категорії втілюють переважно такий підхід до навчання, при якому ті, кого навчають, сприймаються як активні учасники навчального процесу, що конструюють власні розумові схеми, а не просто як пасивні одержувачі інформації.

У першому випадку передбачається:

а) високоструктуроване навчальне середовище, у якій програма контролює характер і напрямок (або напрямки) навчання, при цьому можливості й форми участі самого того, якого навчають, обмежені;

б) докладний аналіз завдання;

в) послідовне наближення до поставленої кінцевої мети, звичайно обумовленої як поведінкове завдання;

Кафедра_КОПЗ_2023рік

г) упор на зовнішні підкріплювальні фактори, які можуть бути не пов'язані з характером завдання [5].

Для другого характерні:

а) високий ступінь контролю того, якого навчають, за ходом навчання – комп'ютер лише створює операційне середовище;

б) акцент на процес, а не на результат; вважається, що навчання є органічною похідною структури взаємодії;

в) пропозиція, що така воля взаємодії внутрішньо мотивована й тому зовнішніх підкріплень не потрібно.

Особливо актуально впровадження інформаційних технологій у навчальний процес при навчанні системного програміста. Окрім наявності сучасної обчислювальної техніки, потрібна наявність сучасних навчаючих програм та емуляторів роботи складових ЕОМ.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

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

– Огляд існуючих систем візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

– Дослідження системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

– Програмна реалізація системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КБПЗ_2023рік

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації програмного забезпечення візуалізації процесу обміну інформації паралельного порту у навчальних цілях. Паралельний порт (порт принтера, IEEE 1284, LPT) – міжнародний стандарт паралельного інтерфейсу для підключення периферійних пристроїв персонального комп'ютера.

В основному використовується для підключення до комп'ютера принтера, сканера й інших зовнішніх пристроїв (часто використовувався для підключення зовнішніх пристроїв зберігання даних), однак може застосовуватися й для інших цілей (організація зв'язку між двома комп'ютерами, підключення яких-небудь механізмів телесигналізації й телекерування).

В основі даного стандарту лежить інтерфейс Centronics і його розширені версії (ECP, EPP).

Назва «LPT» утворено від найменування стандартного пристрою принтера «LPT1» (Line Printer Terminal або Line PrinTer) в операційних системах сімейства MS-DOS.

Інтерфейс Centronics і стандарт IEEE 1284

Паралельний порт Centronics – порт, використовуваний з 1981 року в персональних комп'ютерах фірми IBM для підключення друкувальних пристроїв, розроблений фірмою Centronics Data Computer Corporation; уже давно став стандартом де-факто, хоча в дійсності офіційно на даний момент він не стандартизований.

Цей порт був розроблений тільки для симплексної (односпрямованої) передачі даних, тому що передбачалося, що порт Centronics повинен використовуватися тільки для роботи із принтером. Згодом різними фірмами були розроблені дуплексні розширення інтерфейсу (byte mode, EPP,

Кафедра_КБПЗ_2023рік

ECP). Потім був прийнятий міжнародний стандарт IEEE 1284, що описує як базовий інтерфейс Centronics, так і всього його розширення.

Роз’єми

Порт на стороні керуючого пристрою (комп'ютера) має 25-контактний 2-рядний роз’єм DB-25-female ("мама") (IEEE 1284-A). Не плутати з аналогічним male-роз’ємом ("тато"), що встановлювався на старих комп'ютерах і являє собою 25-піновий COM-порт. На периферійних пристроях звичайно використовується 36-контактний роз’ємиCentronics (IEEE 1284-B), тому кабелі для підключення периферійних пристроїв до комп'ютера по паралельному порту звичайно виконуються з 25-контактним роз’ємом на одній стороні й 36 IEEE 1284-B на іншій (AB-кабель). Зрідка застосовується AC-кабель із 36-контактним роз’ємом MiniCentronics (IEEE 1284-C).

Рисунок 1.1 – Кабельний 36-контактний роз’єм Centronics для підключення зовнішнього пристрою (IEEE 1284-B)

Рисунок 1.2 – 25-контактний роз’єм DB-25, використовуваний як LPT-порт на персональних комп'ютерах (IEEE 1284-A)

Кафедра_ЮБТВ_2023рік

Існують також СС-кабелі з роз’ємами MiniCentronics на обох кінцях, призначені для підключення приладів у стандарті IEEE 1284-II, що застосовується рідко.

Довжина сполучного кабелю не повинна перевищувати 3 метрів. Конструкція кабелю: кручені пари в загальному екрані, або кручені пари в індивідуальних екранах. Зрідка використовуються стрічкові кабелі.

Для підключення сканера, і деяких інших пристроїв використовується кабель, у якого замість роз’єму (IEEE 1284-B) установлені роз’єми DB-25-male. Звичайно сканер оснащується другим інтерфейсом з роз’ємом DB-25-female (IEEE 1284-A) для підключення принтера (оскільки звичайно комп'ютер оснащується тільки одним інтерфейсом IEEE 1284). Схемотехніка сканера побудована таким чином, щоб при роботі із принтером сканер прозоро передавав дані з одного інтерфейсу на іншій.

Фізичний інтерфейс

Базовий інтерфейс Centronics є односпрямованим паралельним інтерфейсом, містить характерні для такого інтерфейсу сигнальні лінії (8 для передачі даних, строб, лінії стану пристрою).

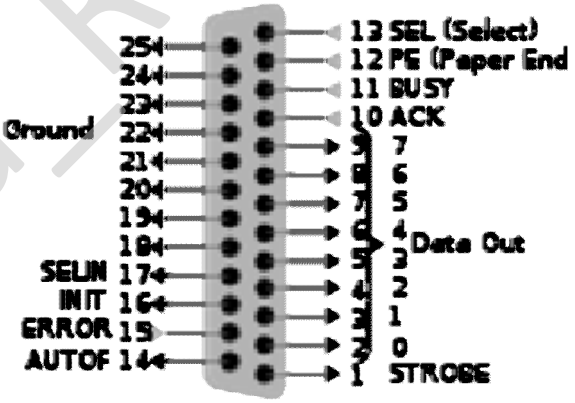

Рисунок 1.3 – Інтерфейс роз’єму

Дані передаються в одну сторону: від комп'ютера до зовнішнього пристрою. Але повністю односпрямованим його назвати не можна. Так, 4 зворотні лінії використовуються для контролю за станом пристрою. Centronics

Кафедра_КБПЗ_2023рік

дозволяє підключати один пристрій, тому для спільного чергового використання декількох пристроїв потрібно додатково застосовувати селектор.

Швидкість передачі даних може варіюватися й досягати 1,2 Мбіт/с.

Таблиця 1.1 – Спрощена таблиця сигналів інтерфейсу Centronics

Контакти DB-25 IEEE 1284-A	Контакти Centronics IEEE 1284-B	Позначення	Примітка	Функція
1	1	Strobe	Маркер циклу передачі (вихід)	Керування
2	2	Data 1	Сигнал 1 (вихід)	Дані
3	3	Data 2	Сигнал 2 (вихід)	Дані
4	4	Data 3	Сигнал 3 (вихід)	Дані
5	5	Data 4	Сигнал 4 (вихід)	Дані
6	6	Data 5	Сигнал 5 (вихід)	Дані
7	7	Data 6	Сигнал 6 (вихід)	Дані
8	8	Data 7	Сигнал 7 (вихід)	Дані
9	9	Data 8	Сигнал 8 (вихід)	Дані
10	10	Acknowledge	Готовність прийняти (вхід)	Стан
11	11	Busy	Зайнятий (вхід)	Стан
12	12	Paper End	Ні паперу (вхід)	Стан
13	13	Select	Вибір (вхід)	Стан
14	14	Auto Feed	Автоподача (вихід)	Керування
15	64	Error	Помилка (вхід)	Стан
16	31	Init	Ініціалізація (вихід)	Керування
17	36	Select In	Керування друком (вихід)	Керування
18-25	16-17, 19-30	GND	Загальний	Земля

Кафедра_КБПЗ_2023рік

Фірмові розширення

Більшість фірмових розширень первісного інтерфейсу згодом було стандартизовано індустрією, процес завершився прийняттям серії стандартів IEEE-1284.

Однак, слід зазначити, що повної відповідності між цим стандартом і попередніми йому фірмовими розширеннями немає. Найбільш відомими фірмовими розширеннями є від фірми Hewlett-Packard. Це Bitronics, що забезпечує двосторонню передачу інформації (застосовується насамперед для зняття розширеної інформації про стан принтера) і протокол мультиплексування шини від HP, призначений для того, щоб підключати до одному LPT-порту безліч пристроїв за схемою "ланцюжок". На основі цього протоколу були розроблені стандарти 1284.3-2000 і 1284.4-2000, але повної сумісності досягнуто не було.

У результаті деякі дуже старі пристрої від HP можуть не цілком коректно працювати з портами, зконфігурованими в режим IEEE-1284. Переважна більшість сучасної техніки не має такої проблеми.

Режими роботи

Стандарт дозволяє використовувати інтерфейс у декількох режимах:

– SPP (Standart Paralell Port) – односпрямований порт, повністю сполучимий з інтерфейсом Centronics.

– Nibble Mode – дозволяє організувати двонаправлений обмін даними в режимі SPP шляхом використання керуючих ліній (4 біт) для передачі даних від периферійного пристрою до контролера. Історично це був єдиний спосіб використовувати Centronics для двостороннього обміну даними.

– Byte Mode – рідко використовуваний режим двостороннього обміну даними. Використовувався в деяких старих контролерах до прийняття стандарту IEEE 1284.

– EPP (Enhanced Parallel Port) – розроблений компаніями Intel, Xircom і Zenith Data Systems – двонаправлений порт, зі швидкістю передачі даних до 2Мб/сек.(1991).

Кафедра_КБПЗ_2023рік

– ECP (Extended Capabilities Port) – розроблений компаніями Hewlett-Packard і Microsoft – на додаток з'явилися такі можливості, як наявність апаратного стиску даних, наявність буфера й можливість роботи в режимі DMA.

Стандарт IEEE-1284

Став результатом тривалої боротьби за забезпечення сумісності. Він містить у собі формальний опис всіх перерахованих вище режимів роботи (до його прийняття не було ніякого формального документа, що дозволяє при дотриманні його розраховувати на коректну роботу пристрою у всіх можливих конфігураціях).

Стандарт містить у собі наступні документи:

– IEEE 1284-1994: Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers – двонаправлена передача даних.

– IEEE 1284.1-1997: Transport Independent Printer/System Interface– a protocol for returning printer configuration and status – стандартизований метод одержання інформації про стан принтера.

– IEEE 1284.2: Standard for Test, Measurement and Conformance to IEEE 1284 (not approved) – тестування на відповідність стандарту (так і не був прийнятий).

– IEEE 1284.3-2000: Interface and Protocol Extensions to IEEE 1284-Compliant Peripherals and Host Adapters – a protocol to allow sharing of the parallel port by multiple peripherals (daisy chaining) – протокол і схема до одного порту багатьох пристроїв, об'єднаних в «ланцюжок» (дозволяє вибрати потрібний пристрій і працювати з ним).

– IEEE 1284.4-2000: Data Delivery and Logical Channels for IEEE 1284 Interfaces – allows a device to carry on multiple, concurrent exchanges of data – протокол одночасної передачі інформації багатьом пристроям (дозволяє працювати одночасно з декількома пристроями в ланцюжку).

У цей час стандарт IEEE-1284 не розвивається. Остаточна стандартизація паралельного порту збіглася з початком впровадження інтерфейсу USB, що дозволяє підключати також і комбіновані апарати (сканер-принтер-копір) і

Кафедра_КБПЗ_2023рік

забезпечує більше високу швидкість друку й надійну роботу принтера. Також, альтернативою паралельному інтерфейсу є мережний інтерфейс Ethernet.

1.2 Область застосування

Областю застосування розроблювального, у результаті виконання бакалаврського проектування, системного програмного забезпечення є системи навчання та підготовка фахівців у галузі інформаційних технологій та програмування, з використання комп'ютерної техніки.

В умовах складності процесів становлення обновлюваної системи вищого утворення варіантом, що гарантує успіх, є зв'язок випускаючої кафедри вузу з фірмами за схемою взаємної зацікавленості:

– кафедра для фірм – підготовка фахівців, індивідуальне навчання по замовленнях і планам фірм або їхніх підрозділів, розробки, перепідготовка персоналу;

– фірми для кафедри – допомога в придбанні засобів обчислювальної техніки (ОТ) і іншого встаткування, стажування, практика, виконання реальних курсових і бакалаврських проектів.

При цьому вирішальним фактором такого співробітництва служить рівень (ціна) фахівця, підготовленого в умовах вузу. Тому, щоб існувати, кафедра повинна мати можливості й засоби підготовки фахівців найвищого рівня [2]. Людство вступило в нову фазу інформаційної революції, коли інформація стала ключовим і необхідним елементом існування людства, інформатика – однією з визначальних наукових дисциплін, а інформаційні технології – рушійною силою розвитку суспільства. Тому рівень підготовки сучасного фахівця, крім професійного володіння основами своєї спеціальності, загальнотехнічних, соціальних і гуманітарних наук, значною мірою визначається його включеністю у світовий інформаційний простір:

– умінням ефективно організувати й підтримувати професійні й довільні інформаційні процеси;

Кафедра_КБПЗ_2023рік

– умінням грамотно оперувати інформаційними ресурсами (накопичувати, зберігати) і використовувати для цього всілякі технічні засоби;

– умінням ефективно працювати з інформацією (знаходити, перетворювати, представляти, оформляти у вигляді, зручному для інших, і т.п.).

Формування стійкого “багажу” відзначених знань і навичок є наслідком знання й освоєння цілого ряду перспективних інформаційних технологій. Інформаційна технологія – система методів, засобів і прийомів (способів) збору, наповнення, зберігання, пошуку, обробки й видачі інформації.

Нові й перспективні інформаційні технології (ПІТ) зв'язують із використанням сучасної й перспективної електронної техніки для обробки інформації (ОТ, техніка зв'язку, побутова електроніка, теле– і радіомовлення) [3].

Основні напрямки ПІТ [5-10]:

– телекомунікації, локальні, корпоративні, глобальні й комбіновані обчислювальні мережі;

– відкриті системи й підтримка розподілених обчислень на основі об'єктної орієнтації й технології “ клієнт-сервер”;

– мультимедіа, у тому числі підтримка звуку, зображень (графіка, слайди, анімація), відео, гіпермедіа, експертмедіа;

– підтримка складних середовищ (віртуальна реальність, фільми або ігри з альтернативними або гіпер-сценаріями).

Показово, що кожна ПІТ будь-якого напрямку складається з постійно вдосконалюємого й розширюємого набору фізичних явищ, технологічних і інформаційних процесів, технічних прийомів, засобів ОТ і іншого (часто спеціального) устаткування, програмних систем і середовищ.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КБПЗ_2023рік

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

У цьому розділі проведемо огляд існуючих емуляторів роботи та програмування складових ЕОМ.

До таких систем відносяться наступні:

– Bochs.

– VMware Workstation.

Bochs

Останнім часом емуляторами стали користуватися все частіше й частіше. Це стало можливим завдяки збільшенню загальної продуктивність ПК. В основному, такі програми емулюють роботу інших систем, написаних під іншу платформу. Bochs – це емулятор IBM PC для IBM PC, тобто система емулює сама себе.

Bochs "зовні" дуже схожий на VMware. У ньому також можна емулювати кілька систем, і перемикатися між ними. Bochs підтримує велику кількість "Гостьових ОС", починаючи с Minix, закінчуючи Windows. Правда у відмінності від тої ж VMware, BOCHS просто інтерпретує інструкції процесора, що змушує програму працювати більш природно. Зрозуміло, це позначається на продуктивності, тому Bochs працює повільніше. У цей час існують різні версії Bochs, під більшість ОС. Це як би сполучна ланка, між різними системами.

"Завдяки" низькій продуктивності використовувати його як повноцінний емулятор ОС, такий як Windows, Linux або MacOS, не має змісту. Але от для "маленьких" систем, він відмінно підходить. Дуже часто Bochs використовують в

Кафедра_КБПЗ_2023рік

*nix системах, для емуляції DOS, і запуску додатків під цю ОС. Також емулятор використовують для вивчення шкідливих програм (вірусів, хробаків, троянів).

Таким чином, ми прийшли до виводу, що Bochs – дійсно корисна програма, і навчитися їй користуватися буде не зайвим, але перш ніж доступися до безпосередньо установки й настроювання емулятора, давайте розглянемо ще декілька його можливостей.

Можливості

Bochs емулює IBM-сумісний комп'ютер до якого можна “підключити”:

– Відеоадаптер VGA/SVGA. розрішення – до 1024x768x64bpp.

– Жорсткий диск обсягом до 64Гб – у формі файлового образа. Bochs підтримує flat, sparse і growing образи, а також уміє робити відкат (rollback) зроблених змін.

– Привод ATAPI-4/IDE CDROM – так само як і у випадку з дисководом можна використовувати звичайний ISO-образ.

– NE 2000-сумісну мережну карту, за допомогою якої "гостьова ОС" може зв'язуватися з іншими машинами у своїй локальній мережі або навіть виходити в Інтернет.

– Звукову карту, сумісну з SoundBlaster.

– Паралельний і послідовний порти.

Також Bochs підтримує плагіни, що робить його ще більш універсальним.

Ще однією чудовою особливістю Bochs є убудований відладник, що дозволяє зупиняти "Гостьову ОС", вивчати стан пам'яті, регістрів, установлювати точки останова "breakpoint". Ця функція буде дуже корисна системним програмістам.

От невеликий опис настроювань необхідних для роботи:

– config_interface – формат конфігураційного файлу;

– megs – кількість мегабайт оперативної пам'яті;

– romimage – шлях до драйвера CD-ROM;

– vgaromimage – шлях до драйвера відеокарти;

Кафедра_КБПЗ_2023рік

– boot – носій з якого буде запущена ОС;

– floppy_bootsig_check – перевірка дисководу на наявність завантажувальних дискет;

– vga – режим роботи відеокарти;

– cpu – тактова частота процесора.

Інші настроювання необхідні для конфігурування заліза, або для настроювання інтерфейсу.

VMWare Workstation

VMware Workstation – Програма дозволяє запускати на комп'ютері одночасно кілька операційних систем. VMware Workstation емулює самостійний комп'ютер і дозволяє працювати з ним як зі звичайною системою, ставити операційні системи й додатки. Корисність пакета полягає в можливості тестування різного програмного забезпечення під різними операційними системами, при цьому, не шкодячи поточної ОС, установленої на комп'ютері. Істотно спрощує життя розроблювачам крос-платформних додатків і аматорам попрацювати з новими операційними системами. Дозволяє організовувати мережу між віртуальними машинами, а так само взаємодіяти із зовнішніми пристроями.

В VMware Workstation реалізована підтримка 64-бітних гостьових операційних систем FreeBSD, Linux і Windows, а також Solaris x86. Таким чином, тестери й розроблювачі зможуть одночасно запускати 64-бітні й 64-бітні операційні системи на тому самому встаткуванні.

Продукт компанії VMware Workstation оптимізований для професійного використання й забезпечує повномасштабну й одночасну експлуатацію декількох мережних операційних систем (включаючи Linux, Novell NetWare, NetWare, Solaris x86 і Windows) на одному комп'ютері.

VMWare Workstation

Нові можливості VMware Workstation:

Кафедра_КБПЗ_2023рік

– Можливість вибору в якості гостьової ОС віртуальної машини VMware vSphere 4.

– Підтримка Windows 10/11 як у якості гостьової, так і хостової ОС.

– Драйвер WDDM для гостьовий ОС Windows 10/11.

– OpenGL і Shader Model для гостьових ОС Windows.

– Динамічне завантаження пакета VMware Tools.

– Багатоядерні й чотирипроцесорні віртуальні машини (4 vCPU).

– Можливість друку з віртуальної машини без установки драйверів за рахунок засобів ThinPrin.

– Створення снапшотов через задані інтервали часу (AutoProtect).

– Захист віртуальних машин 128-бітним шифруванням.

– Підтримка бібліотеки ALSA для гостьових ОС Linux, що дозволить грати музику з декількох віртуальних машин паралельно

– Поліпшений Linux Virtual Network Editor.

– VMware Workstation – програма, що призначена для робочих станцій, дозволяє емулювати самостійний комп'ютер у середовищі робочої операційної системи й працювати з ним як зі звичайною системою.

VMware Workstation – продукт для віртуалізації робочих місць на сучасний момент. Платформа пропонує розширені можливості для розроблювачів додатків, інженерів по контролі якості, фахівців із продажів технологічних рішень і IT-адміністраторів.

За допомогою такого методу можна встановлювати різні ОС і додатка для них, виходити в інтернет і займатися повсякденними завданнями, щоправда, із трохи вповільненою швидкістю. Користь від такого способу полягає в можливості тестування різного програмного забезпечення під різними операційними системами, не наносячи при цьому шкоди поточній системі, установленої на основному комп'ютері, а також випробування деяких апаратних рішень.

Кафедра_КБПЗ_2023рік

Може істотно полегшити життя розроблювачам крос-платформних додатків і аматорам попрацювати з новими ОС.

Одним із самих помітних нововведень пакета VMware Workstation стала розширена підтримка 64-бітних версій Windows.

Для відображення стану віртуальних машин тепер повноцінно використовуються функції Flip 3D і Aero Peek, реалізовані в інтерфейсі Windows 10/11. Також пакет VMware Workstation став першим продуктом з повною підтримкою графічної оболонки Aero у системах Windows 10/11. Істотно розширений перелік тривимірних додатків на базі технологій Direct 9.0c Shader Model 3 і OpenGL 2.1, які можна запускати у віртуальних машинах Windows. Для зручності розроблювачів пакет VMware Workstation пропонує нові засоби інтеграції з популярними середовищами розробки Spring STS (Source Tools Suite) і Eclipse. Тепер платформа віртуалізації повністю інтегрується в процес створення додатків на мовах Java і C/C++, а поліпшена технологія налагодження із записом і відтворенням роботи додатка (Record Replay Debugging) допомагає швидше знаходити недетерміновані помилки в коді.

Підвищено потужність віртуальних машин, які можна створювати в VMware Workstation – тепер кожна віртуальна машина 64-бітною операційною системою може містити до 4 віртуальних процесорів і 64 Гбайт оперативної пам'яті. Убезпечити віртуальні машини допомагає шифрування образів системи алгоритмом AES з довжиною ключа 256 біт. Перелік підтримуваних операційних систем у новій версії перевищив 200, з яких уперше представлена підтримка таких систем, як Windows, Ubuntu, а також VMware vSphere і VMware ESXi.

Крім того, слід зазначити можливість підключення образів віртуальних машин, створених за допомогою технології Windows Mode, представленої в промислових версіях Windows 10/11 для запуску успадкованих Windows-додатків в ізольованому віртуальному середовищі. Також підтримується перенос ліцензійних ключів між платформами – тепер розроблювач може використовувати один ключ у двох операційних системах: Windows і Linux.

Кафедра_КОПЗ_2023рік

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об'єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об'єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об'єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 64-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам'яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об'єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Кафедра_КОПЗ_2023рік

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції застосунку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

Кафедра_КБПЗ_2023рік

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 візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

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.

Кафедра_КбПЗ_2023рік

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис роботи паралельного порту

BIOS може працювати із трьома паралельними принтерними портами. У процесі тестування й ініціалізації системи BIOS знаходить працездатні принтерні порти й записує їхні базові адреси в таблицю. Таблиця адрес розташовується в області даних BIOS за адресою 0000:0408h. Можливі наступні значення базових адрес:

– 378h – принтерний порт LPT1;

– 278h – принтерний порт LPT2;

– 3BCh – принтерний порт на платі адаптера монохромного дисплея.

Принтерні порти можуть виробляти запити на переривання:

– LPT1 – IRQ7, INT 0Fh;

– LPT2 – IRQ5, INT 0Dh.

Кожний принтерний порт (принтерний адаптер) обслуговують кілька портів вводу/виводу.

Базова адреса порту (378h, 278h, 3BCh) використовується для передачі байта даних з комп'ютера у принтер (біт 0 – D0; біт 1 – D1; біт 2 – D2; біт 3 – D3; біт 4 – D4; біт 5 – D5; біт 6 – D6; біт 7 – D7).

Базова адреса порту + 1 (379h, 279h, 3BDh) використовується для передачі із принтера в комп'ютер бітів стану:

– біт 3 – ERROR;

– біт 4 – SLCT;

– біт 5 – PE;

– біт 6 – ACK;

– біт 7 – BUSY.

Кафедра_КБПЗ_2023рік

Базова адреса порту + 2 (37Ah, 27Ah, 3BEh) використовується для передачі з комп'ютера в принтер бітів керування:

– біт 0 – STROBE;

– біт 1 – AutoFeed;

– біт 2 -INIT;

– біт 3 – SelectIn;

– біт 4: коли дорівнює одиниці, дозволене переривання від принтера.

Засоби BIOS для роботи із принтером

BIOS використовує для роботи із принтером функції 0, 1, 2 переривання INT 17h.

Функція 00h призначена для друку одного символу:

На вході: AH = 00h;

AL = ASCII-код символу для друку;

DX = номер принтера: 0, 1 або 2.

На виході: AH = слово стану принтера (див. нижче).

Ця функція виводить на принтер один символ, заданий у регістрі AL. У регістр DX необхідно записати номер використовуваного принтера, для LPT1 це 0, для LPT2 – 1 і т.д.

Після виконання переривання регістр AH буде містити слово стану, що має формат наведений в таблиці 3.1.

Викликавши функцію 0 переривання INT 17h, програма повинна перевірити окремі біти слова стану й переконатися в тім, що вивід байта відбувся без помилок. Найбільше часто оператор забуває перевести принтер у стан ONLINE, або вставити папір, або взагалі включити принтер. У цьому випадку доцільно нагадати операторові про необхідність виконання цих дій і потім повторити печатка символу.

Якщо принтер несправний, програма повинна надати операторові можливість скасувати друк тексту. Нижче ми приведемо приклад програми, що

Кафедра_КБПЗ_2023рік

виконує друк тексту й помилки, що аналізує, які можуть виникнути в процесі друку.

Таблиця 3.1 – Формат слова стану

Номер біта	Значення біта
0	таймаут, занадто велика затримка при виконанні операції друку, можливо, що принтер несправний
1-2	не використовуються
3	помилка вводу/виводу
4	1 – принтер обраний для роботи 0 – принтер у стані offline
5	кінець паперу
6	підтвердження
7	1 – принтер готовий 0 – принтер зайнятий

Зверніть увагу на біт 1 байта стану – таймаут. Якщо принтер перебуває в стані OFFLINE, функція 0 переривання INT 17h очікує якийсь час готовності принтера, після чого якщо принтер так і не перейшов у стан готовності, установлює біт 1 у байті стану. Область даних BIOS за адресою 0000h:0478h містить чотири байти, які використовуються як лічильники часу при очікуванні готовності принтера.

Переривання INT 17h має ще дві функції, що виконують ініціалізацію принтера й отримуючу поточний стан принтера.

Функція 01h ініціалізує принтер:

На вході: AH = 01h;

DX = номер принтера: 0, 1 або 2.

На виході: AH = слово стану принтера.

Ця функція виконує апаратне скидання принтера. Якщо ви завантажили в принтер який-небудь шрифт (наприклад, кирилицю), після скидання

Кафедра_КБПЗ_2023рік

завантаження шрифту прийдеться виконувати заново. Тому не слід виконувати скидання принтера, якщо це дійсно не потрібно. Звичайно принтер доводиться скидати або перед настроюванням його на заданий режим роботи, що виконується один раз, або при зміні цього режиму.

Слово стану принтера може бути отримане за допомогою функції 02h:

На вході: AH = 02h;

DX = номер принтера: 0, 1 або 2.

На виході: AH = слово стану принтера.

Цю функцію зручно використовувати перед початком друку для визначення готовності принтера до роботи.

Засоби нижнього рівня для роботи із принтером

Для друку символу на стандартному друкувальному пристрої LPT1 (він же PRN) ви можете використовувати функцію 05h переривання MS-DOS INT 21h:

На вході: AH = 05h;

DL = ASCII-код символу для друку.

На виході: AH = слово стану принтера (див. нижче).

Команда MS-DOS MODE може перепризначити стандартний пристрій друку LPT1 на асинхронний послідовний порт:

MODE LPT1:=COM1

Помітьте, що функція 05h переривання INT 21h не повертає стану принтера при помилці вводу/виводу. Замість цього викликається стандартний оброблювач критичних помилок MS-DOS, що виводить на екран добре знайоме вам повідомлення:

Write fault error writing device PRN

Abort, Retry, Ignore, Fail?

Ви можете відповісти Retry, нажавши клавішу "R", тоді MS-DOS виконає спробу повторити печатка символу. Якщо відповісти Abort (нажавши клавішу "A"), MS-DOS завершить роботу вашої програми.

Кафедра_КБПЗ_2023рік

Якщо вас не влаштовують дії, виконувані стандартним оброблювачем критичних помилок MS-DOS, ви можете скласти власний.

Більше цікаві можливості по керуванню процесом друку надає програма резидентного спулера друку PRINT.COM. Ви знаєте, що команда PRINT призначена для виконання друку у фоновому режимі.

Виявляється, що якщо запущено програму PRINT, інші програми можуть взаємодіяти з нею, управляючи процесом друку.

Для зв'язку зі спулером друку можна використовувати кілька функцій переривання INT 2Fh:

На вході: AH = 01h;
AL = номер виконуваної операції.

На виході: AH = 00 – спулер друку не встановлений, але його можна встановити, запустивши програму PRINT;
01 – спулер друку не встановлений і його установка неможлива (система не містить жодного принтера);
FFh – спулер установлений.

Приведемо формати регістрів для виконання різних операцій зі спулером друку.

На вході: AH = 01h;
AL = 0 – перевірити установку спулера друку.

На виході: AH = 00 – спулер друку не встановлений, але його можна встановити, запустивши програму PRINT;
01 – спулер друку не встановлений і його установка неможлива;
FFh – спулер установлений.

На вході: AH = 01h;

Кафедра_КБПЗ_2023рік

AL = 1 – передача файлу спулеру для друку;

DS:DX = адреса керуючого блоку:

Зсув Довжина

(+0) 1 рівень запиту, дорівнює 0;

(+1) 4 FAR-адреса рядка у форматі ASCIIZ, утримуючий шлях файлу.

На виході: AH = 00 – спулер друку не встановлений, але його можна встановити, запустивши програму PRINT;

01 – спулер друку не встановлений і його установка неможлива;

FFh – спулер установлений.

На вході: AH = 01h;

AL = 2 – скасувати печатка файлу;

DS:DX = адреса рядка у форматі ASCIIZ, утримуюче ім'я файлу, що видаляється із черги для друку.

На виході: AH = 00 – спулер друку не встановлений, але його можна встановити, запустивши програму PRINT;

01 – спулер друку не встановлений і його установка неможлива;

FFh – спулер установлений.

На вході: AH = 01h;

AL = 3 – скасувати печатка всіх файлів.

На виході: AH = 00 – спулер друку не встановлений, але його можна встановити, запустивши програму PRINT;

01 – спулер друку не встановлений і

Кафедра_КБПЗ_2023рік

його установка неможлива;

FFh – спулер установлений.

На вході: AH = 01h;

AL = 4 – визначити стан спулера

і заблокувати спулер.

На виході: DS:SI = адреса черги друку (масив рядків

у форматі ASCIIZ, кінець масиву

відзначений рядком, що складається з 0;

DX = кількість помилок при спробі надрукувати

останній символ;

AH = 00 – спулер друку не встановлений,

але його можна встановити, запустивши

програму PRINT;

01 – спулер друку не встановлений і

його установка неможлива;

FFh – спулер установлений.

На вході: AH = 01h;

AL = 5 – розблокувати спулер для продовження

друку.

На виході: AH = 00 – спулер друку не встановлений,

але його можна встановити, запустивши

програму PRINT;

01 – спулер друку не встановлений і

його установка неможлива;

FFh – спулер установлений.

Якщо після виклику перерахованих вище функцій прапор переносу CF установлений в 1, регістр AX містить код помилки.

Кафедра_КБПЗ_2023рік

Таблиця 3.2 – Формат слова стану

1	Неправильний код функції
2	Файл не знайдений
3	Шлях не знайдений
4	Занадто багато відкритих файлів
5	Доступ заборонений
6	Неправильний індекс (handle)
8	Переповнення черги
9	Зайнято
0Ch	Занадто довгий шлях і ім'я файлу (більше 64 байтів)
0Fh	Неправильне визначення диска

Програмування LPT порту під Windows

Розроблювачі Windows лінійки NT з метою забезпечення безпеки використання спільних апаратних ресурсів комп'ютера, заборонили до них прямий доступ із програм користувальницького режиму (по суті справи з будь-яких програм). Щоб все-таки звернутися до порту, необхідно всі операції проводити через драйвер. Але є вже кілька готових інструментів які дозволяють без проблем спілкуватися LPT портом в NT.

Почнемо спочатку з найпростішого, і в теж час дуже потужного засобу: бібліотеки inpoutx64.dll.

Бібліотека inpoutx64.dll

Цей комплекс складається з 3 файлів:

– динамічної бібліотеки inpoutx64.dll;

– статичної бібліотеки lib, необхідної на етапі складання програми;

– заголовного файлу із прототипами використовуваних функцій.

Бібліотека працює в такий спосіб: спочатку вона перевіряє що за ОС у Вас коштує. Якщо це 9x – то проблем немає: при звертанні до бібліотеки за допомогою функцій **Out64**(*адреса порту, байт даних*) – для запису в порт і

Кафедра_КБПЗ_2023рік

Inp64(*адреса порту*) – для читання з порту, вона конвертує ці звертання до стандартним і відомим нам функціям **_outp**(*адреса порту, байт даних*) і **_inp**(*адреса порту*). Якщо ж вона має справу з NT, то звертання до неї вона конвертує в запити до стандартного драйвера ОС, через який і йде обмін інформацією з портом.

Отже, розглянемо наступний варіант роботи з портами вводу/виводу в ОС Windows NT. Цим "варіантом" буде спеціальний драйвер **GiveIO.sys**, що дозволяє додатку, що запустив його, вільно працювати з будь-якими портами.

Головним достоїнством цього драйвера, є те, що звертання до порту можуть бути зроблені й за допомогою функцій, що працюють тільки в Windows 9x.

На початку роботи нашої програми для спілкування з портами необхідно звернутися до драйвера **GiveIO.sys**. При цьому він установить такі значення бітового масиву карти дозволу (I/O Permission Map) для нашого додатка, що для нього буде дозволений прямий доступ до будь-яких портів. I/O Permission Map це складова частина системи захисту вводу-виводу Windows NT. Він являє собою бітовий масив, кожний біт якого відповідає порту вводу-виводу. Якщо біт = 1, то доступ до порту закритий, якщо 0, те відкритий. Для будь-якого користувальницького додатка всі біти встановлені в 1, і відповідно ми не можемо прямо влізти в порт. GiveIO.sys влазить у цей масив і скрізь установлює 0 – тобто можемо звертатися до будь-якого порту з додатка.

Прийоми читання й запису даних в LPT порт

Коротка характеристика проекту:

– працює під будь-який ОС (NT);

– дозволяє читати й записувати дані в будь-який регістр LPT порту (якщо він цю можливість передбачає);

– додаток консольний, для доступу до портів використовуємо бібліотеку inpoutx64.dll.

Кафедра_КБПЗ_2023рік

Перш ніж рухатися далі й писати програму, необхідно розібратися з LPT портом, подивитися із чого він складається і як нам скористатися ним у своїх цілях.

Якщо говорити на побутовому рівні, то можна сказати, що LPT порт це набір контактів, на яких ми можемо встановити напругу 0 або +5 В (логічний 0 і 1) із програми або це може зробити зовнішній пристрій зовні. Давайте розберемося, якими контактами ми можемо оперувати, а якими ні.

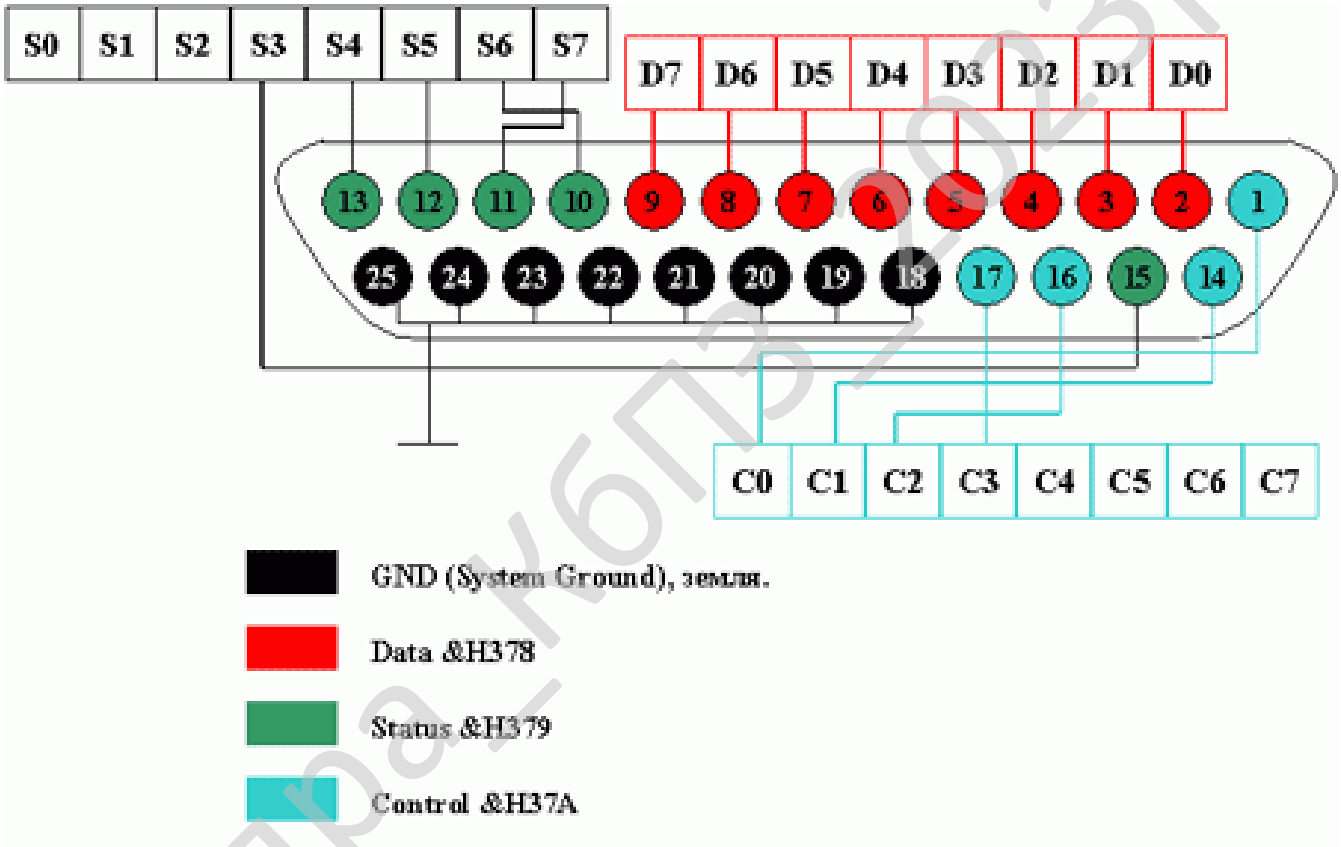

Рисунок 3.1 – Роз’єми LPT-порту

З рисунку 3.1 видно, що виводи порту можна розділити на чотири групи: це виводи **'заземлення'**. Вони позначені чорним кольором (контакти 18-25). Всі вони з'єднані між собою, тому для своїх розробок як землю можна використовувати кожний з них.

Червоним кольором позначені виводи так званого регістра **Data** (контакти 2-9). Під регістром будемо розуміти об'єднання групи контактів LPT порту. У

Кафедра_КБПЗ_2023рік

регістрі Data їхніх 8 штук. Цей регістр дозволяє нам як із програми, так і із зовнішнього пристрою встановити на його контактах логічний 0 або 1, тобто він двонаправлений. Щоб звертатися до цього регістра, треба знати його адресу: **0x378** – в 16-річній системі або **888** у десятковій.

Продовжимо розгляд порту. Залишилося ще два регістри. Наступним буде регістр **Status** (контакти 10-13, 15). Це односпрямований регістр. Управляти їм можна тільки із зовні, через зовнішній пристрій (мається на увазі змінювати дані на ньому, читати можна з будь-якого регістра в будь-яку сторону). Він має адресу **0x379** – в 16-річній системі або **889** у десятковій. І регістр **Control** (контакти 1, 14, 16-17). Він має всього 4 контакти й може управлятися тільки програмою. Його адреса: **890** у десятковій системі.

У підсумку одержали:

– 8 двонаправлених контактів (регістр Data) – дані туди може записати й програма й зовнішній пристрій;

– 5 односпрямованих контактів (регістр Status) – дані туди може записати тільки зовнішній пристрій;

– 4 односпрямованих контакти (регістр Control) – дані туди може записати тільки програма.

Вивід: у нас є 17 виводів якими ми можемо управляти за своїм розсудом.

Тепер розглянемо, а як відбувається запис і читання даних у регістри LPT порту, тобто як нам установити на потрібних виводах 0 або 1.

Запис/читання даних у регістр Data

Отже, розглянемо щоб на виводі регістра Data під номером 3 (3 – це номер виводу LPT порту) була встановлена логічна 1 (тобто щоб між ним і землею було +5 В) і на інших виводах цього регістра (2, 4-9 виводи порту) були нулі. Пишемо код:

```
int Address=888;
int data=2;
Out64(Address, data);
```

Кафедра_КбПЗ_2023рік

Використовується функцію Out64 бібліотеки inpoutx64.dll. Якщо цей код виконати, то вийде що на виводі порту 3 є +5 В, а на 2, 4-9 висить нуль. Першим параметром функції Out64 ми передаємо число 888. Це адреса регістра Data LPT порту. Тепер функція знає куди їй писати дані. Далі другим параметром ми передаємо число 2. Прошу звернути увагу, що двійка в десятковій системі числення. Що далі робить функція? Їй треба цю двійку запхнути в регістр Data, але от проблема: регістр зовсім не розуміє що таке 2. Він знає 0 або 1. Більше нічого. Тоді функція як би "переводить цю двійку у двійкову систему числення" і кожний розряд двійкового числа праворуч на уліво записує один по одному в регістр починаючи з молодшого розряду D0 (вивід 2 порту) і закінчуючи старшим D7 (вивід 9). Якщо Ви переведете число 2 з десяткової у двійкову систему числення то одержите **10**. Функція бере перший розряд двійкового числа – це **0** (саму праву) і пише її в D0, далі бере **1** і пише її в D1. Т.к. регістр 8-мі розрядний (у нього є 8 контактів), функція продовжує брати дані праворуч на ліво й писати в наступний біт регістра. Т.к. наше число закінчилося, то функція як би дописує нулями наше двійкове число ліворуч, щоб воно стало 8-мі розрядним.

В регістр Data запишемо число 245. Пишемо код:

```
int Address=888;
int data=245;
Out64(Address, data);
```

Знову переводимо 245 у двійкову систему числення й з праворуч наліво записуємо розряди числа в відповідні біти регістра. У підсумку одержимо, що на виводах LPT порту під номерами 2, 4, 6-9 є присутнім напруга +5 В, на виводах 3, 5 нуль.

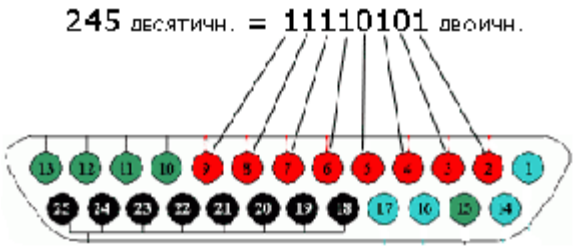

Рисунок 3.2 – Вивід даних на контакти порту

Кафедра_КБПЗ_2023рік

Ну що, тепер я думаю, із записом даних у регістр Data ми розібралися. Треба відзначити, що діапазон десяткових чисел, які можна записати в регістр Data лежить у межах від **0** до **255**. Регістр він у нас 8-мі розрядний, значить максимальне число комбінації 0 і 1 на його виводах становить 2^8-1=256-1=255.

Читання даних

Тепер давайте одержимо дані з порту, а саме з його регістра Data, до нас у програму. Ми хоти довідатися, на яких виводах регістра Data зараз високий рівень напруги, а на яких низький. Вище ми записали в порт число 245. Давайте його зараз одержимо з порту. Пишемо код:

```
int Address=888;
int data;
data = Inp64(Address);
```

Inp64 це функція для читання даних з порту бібліотеки inpoutx64.dll. Єдиним параметром для неї є адреса того регістра, звідки ми хочемо прочитати дані. На виході вона повертає десяткове число, що відповідає поточному вмісту регістра. Виконавши цей код, змінна data буде містити число 245. Щоб розібратися, переводимо число 245 з десяткової у двійкову й на виводах порту 2, 4, 6-9 зараз +5 В а на виводах 3, 5 0 В.

Як я вже згадав вище, у регістр Data дані записати може й зовнішній пристрій. Однак розгляд цього питання поки залишимо, тому що це зажадає зовнішніх джерел живлення. Спочатку, давайте повністю розберемося з базовими операціями.

Запис/читання даних у регістр Control

Регістр Control односпрямований, дані в нього може записати тільки наша програма. Зверніть увагу на кілька особливостей цього регістра. По-перше, про містить усього чотири робітники виводу. Значить у нього можна записати число в діапазоні від 0 до 2^4-1=16-1=15. По-друге, він має дуже неприємну особливість: деякі з його виводів інвертовані, тобто якщо Ви на цей вивід пишете 1, то на ній установлюється 0. І навпаки, читаєте 1, а насправді там 0. Тому, значення записуваних даних і даних, що читаються, не зовсім очевидні. Для розбору необхідно просто спробувати писати в регістр різні дані й дивитися, що виходить. Приведу приклад запису числа в регістр Control. Пишемо код:

Кафедра_КбПЗ_2023рік

```
int Address=890; //адреса регістра Control
int data=10;
Out64(Address, data);
```

І приклад читання:

```
int Address=890; //адреса регістра Control
int data;
data = Inp64(Address);
```

Запис/читання даних у регістр Status

Нарешті, добралися до регістра Status. Він односпрямований, дані в нього може записати тільки **зовнішній пристрій**, тобто ми в програмі можемо тільки читати вміст цього регістра. Причитав дані з Status, і перевівши їх у двійкове число, відразу досить важко зрозуміти що ж реально діється з напругами на виходах цього регістра. По-перше, він теж має інвертовані виводи, а по-друге робочими є біти під номерами 4-7, а 0-3 не використовуються, і отже число записується досить хитро. Щоб розібратися, краще просто кілька разів робити читання при різних даних у регістрі.

Для установки даних виконаєте такий код.

```
int Address=889; //адреса регістра Status
int data;
data = Inp64(Address);
```

Ви одержите деяке число. Тепер візьміть провідник і з'єднаєте їм кожний із земляних виводів порту (18-25) з яким-небудь виводом регістра Status (10-13, 15), наприклад з десятим. І знову виконаєте читання. Ви одержите інше число. Заберіть провідник. Прочитавши, одержите вихідне число. Як це працює? Вихідно, на всіх виводах цього регістра перебуває високий рівень напруги +5 В. Коли ми з'єднали один з його виводів із землею, то на ньому, відповідно, напруга стала рівною нулю, тобто логічний нуль. Тепер двійковим даним у регістрі відповідає інше десяткове число. Можна спробувати замикати й інші виводи регістра Status на землю, замикати відразу трохи. Щораз при читанні вийде різний результат.

Варто помітити, що при таких досвідах з регістром Status виникає не зовсім зрозуміла ситуація з іншими виводами порту LPT. Після першого замикання виводів Status, починають мигати виводи Data і Control. Це пов'язане з тим, що порт LPT призначений для підключення принтера, а виводи Status він

Кафедра_КБПЗ_2023рік

використовує, для того щоб повідомити комп'ютер деяку службову інформацію. Зміни на виводах Status реєструє системний драйвер операційної системи. Він же проводить і відповідні дії, для нас спостережувані у вигляді періодичної зміни стану інших виводів.

3.2 Розробка структурної схеми

На рисунку 3.3 зображена структурна схема системи. З неї бачимо, що система складається з наступних структурних блоків:

– Структурний блок, у якому реалізований роз’єм паралельного порту.

– Структурний блок, у якому реалізований буфер вводу даних.

– Структурний блок, у якому реалізований буфер керування сигналів.

– Структурний блок, у якому реалізований декодер адреси.

– Структурний блок, у якому реалізована шина адреси.

Крім того у структурній схемі відображені зовнішні підключення та переривання, за допомогою яких можливо керувати паралельним портом.

З шини даних, дані передаються у буфер вводу даних, та буфер сигналу керування.

Структурний блок, у якому реалізований роз’єм паралельного порту, складається з наступних регістрів:

– Регістр даних (програмний порт 0x378).

– Регістр стану (програмний порт 0x379).

– Регістр керування (програмний порт 0x37A).

У перший і третій можемо писати побітно. При побайтному записі значення байта можна вводити в десятковому, вісімковому або в шістнадцятковому форматі. Регістр статусу обновляється автоматично.

Опишемо ці регістри більш докладно.

Кафедра_КБПЗ_2023рік

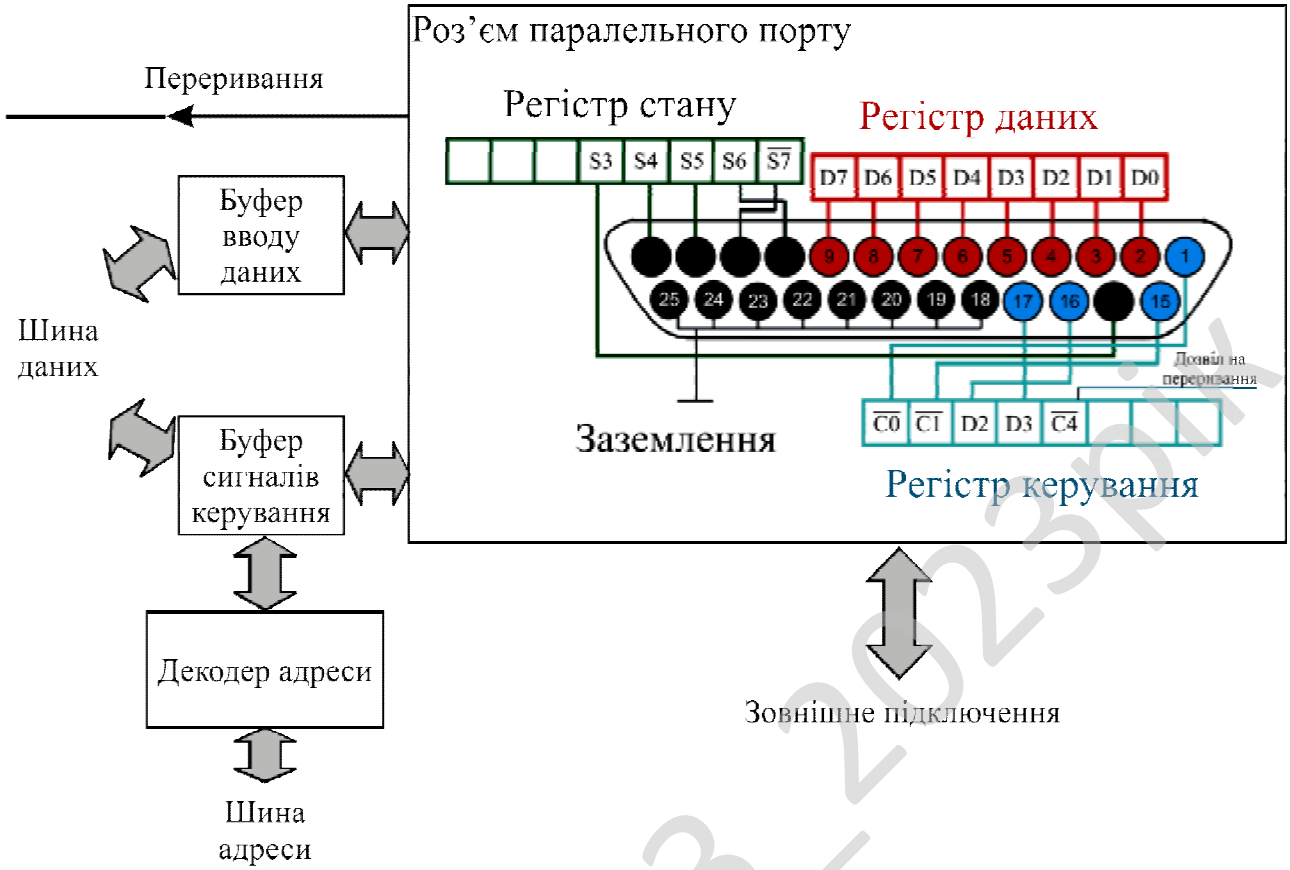

Рисунок 3.3 – Структурна схема системи

Порт 37Ah

Порт керування принтером, доступний для читання й запису.

Таблиця 3.3 – Формат регістра порту

Номер біта	Значення біта
0	строб даних, приймає значення 1 при виводі байта, підключений до 1 контакту рознімання, STROBE;
1	автоматичне переведення рядка після символу "повернення каретки" CR, контакт 14, AUTO LineFeed;
2	скидання принтера, активний рівень – 0, контакт 16, INIT
3	вибір принтера для роботи, контакт 17, SLCT IN;
4	дозвіл переривання від принтера, IRQ Enable;
5-7	дорівнює 0

Кафедра_КБПЗ_2023рік

Якщо переривання від принтера дозволені, вони виробляються, коли сигнал готовності принтера ACK (контакт роз'єму 10) приймає рівень логічного 0.

Порт 378h

Цей порт призначений для запису виведеного на принтер байта даних. Можливо також читання тільки що записаного байта.

Порт 379h

Порт стану принтера, доступний тільки для читання.

Звичайно рідко доводиться працювати із принтером на рівні портів вводу/виводу, тому що досить використовувати функції BIOS або MS-DOS, призначені для цього. Наведена вище інформація може придатися вам для розробки власного драйвера принтера або для підключення до принтерного порту якого-небудь іншого пристрою вводу/виводу, наприклад, аналого-цифрового перетворювача.

Таблиця 3.4 – Формат регістра порту

Номер біта	Значення біта
0-2	установлені в 0
3	сигнал помилки, активний рівень – 0, контакт роз'єму – 15, ERROR
4	принтер обраний, контакт 13, SLCT
5	кінець паперу, контакт 12, PE
6	готовність принтера, активний рівень – 0, контакт роз'єму – 10, ACK
7	0 – принтер зайнятий, перебуває в стані offline або відбулася помилка, контакт 11, BUSY

Для тих, хто буде використовувати принтерний порт для підключення апаратури, приведемо таблицю призначення контактів роз'ємів принтерного

Кафедра_КБПЗ_2023рік

порту (контакт PC) на комп'ютері й контактів рознімання безпосередньо на принтері (контакт принтера). Ці дані наведемо у таблиці 3.5.

Таблиця 3.5 – Призначення контактів роз’ємів принтерного порту (контакт PC) на комп'ютері й контактів рознімання безпосередньо на принтері (контакт принтера)

Контакт PC	Контакт принтера	Призначення	Вхід/вихід
1	1	Строб (STROBE)	Вихід, інверсія
2	2	Дані, біт 0	Вихід
3	3	Дані, біт 1	Вихід
4	4	Дані, біт 2	Вихід
5	5	Дані, біт 3	Вихід
6	6	Дані, біт 4	Вихід
7	7	Дані, біт 5	Вихід
8	8	Дані, біт 6	Вихід
9	9	Дані, біт 7	Вихід
10	10	Підтвердження, ACK	Вхід, інверсія
11	11	Зайнятість, BUSY	Вхід
12	12	Кінець паперу, PE	Вхід
13	13	Вибір, SLCT	Вхід
14	14	Автоматичне переведення рядка, Auto Line Feed	Вихід, інверсія
15	64	Помилка, ERROR	Вхід, інверсія
16	31	Скидання принтера, INIT	Вихід, інверсія
17	36	Принтер обраний, SLCT IN	Вихід, інверсія
18-25	16,17, 19-30, 33	Земля	-

Кафедра_КБПЗ_2023рік

Для сигналів, відзначених у таблиці словом "інверсія", використовується рівень логічного нуля в активному стані сигналу.

Використовуючи принтерний порт для керування зовнішніми пристроями, будьте обережні й виконуйте всі правила заземлення пристроїв. Якщо ваш пристрій не заземлений або заземлений неправильно, принтерний порт може вийти з ладу.

Варто також ураховувати, що навантаження на вихідну лінію принтерного порту не повинна перевищувати одного входу TTL.

Якщо за якимись причинами ви побажаєте працювати із принтером через порти вводу/виводу, вам необхідно вивчити часову діаграму принтерного порту. Вона представлена в таблиці 3.6.

Таблиця 3.6 – Рівень сигналу на лініях порту (часова діаграма)

Лінія	Рівень сигналу															
BUSY	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
ACK	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1
DATA	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
STROBE	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Для того, щоб вивести символ на принтер, програма спочатку повинна переконається, що рівень сигналу на лінії BUSY (біт 7 порту 379h) дорівнює 0, а рівень сигналу на лінії ACK (біт 6 порту 379h) – одиниці. Після цього варто встановити код виведеного символу на лініях DATA (порт 378h).

Потім не раніше, ніж через 0,5 мкс лінію STROBE (біт 0 порту 37Ah) необхідно перевести в стан логічного 0. При цьому виведений символ запишеться у внутрішній буфер принтера. Рівень логічного нуля необхідно втримувати протягом як мінімум 0,5 мкс. Це час потрібно для того, щоб символ записався в буфер принтера. Після витікання інтервалу часу лінію STROBE потрібно знову перевести в стан логічної одиниці.

Кафедра_КбПЗ_2023рік

Після того, як програма встановить лінію STROBE у стан логічного нуля, вихідна лінія принтера BUSY установлюється в одиницю, сигналізуючи про те, що принтер зайняти обробкою отриманого символу й тимчасово не може приймати інші символи.

Коли принтер повністю обробить виведений символ, лінія ACK перейде в стан 0. Приблизно через 5 мкс після цього лінія BUSY також перейде в стан 0.

Ще через 5 мкс лінія ACK прийме стан 1. Тепер принтер готовий прийнятий наступний символ даних, що роздруковуються.

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

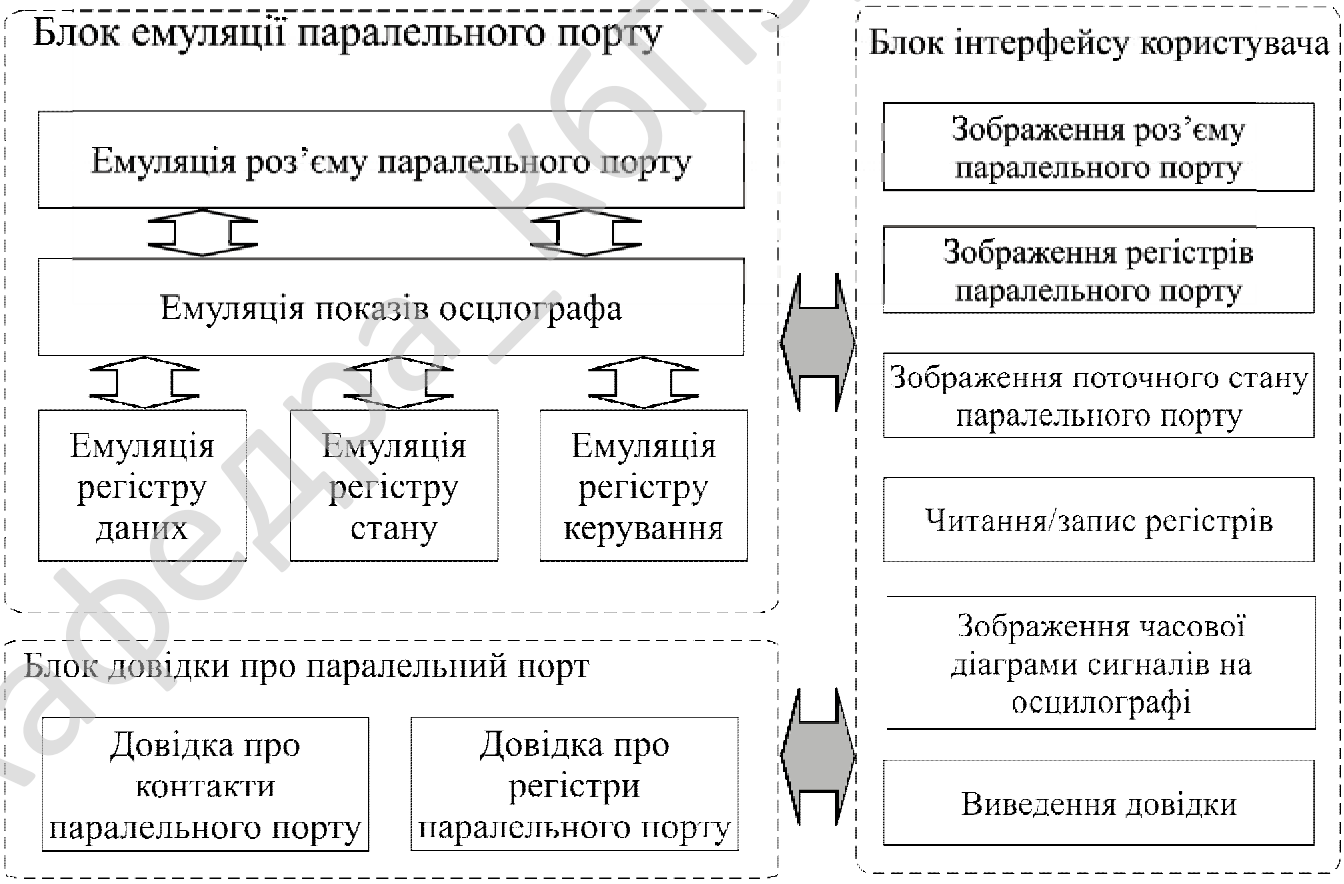

Рисунок 3.4 – Функціональна схема системи

Кафедра_КБПЗ_2023рік

Функціональна схема складається наступних функціональних блоків:

– Блок емуляції паралельного порту.

– Блок інтерфейсу користувача.

– Блок довідки про паралельний порт.

Розглянемо ці функціональні блоки більш детально.

Блок емуляції паралельного порту складається з наступних підблоків:

– Емуляція роз’єму паралельного порту.

– Емуляція показів осцилографа.

– Емуляція регістру даних.

– Емуляція регістру стану.

– Емуляція регістру керування.

Блок інтерфейсу користувача складається з наступних підблоків:

– Зображення роз’єму паралельного порту.

– Зображення регістрів паралельного порту.

– Зображення поточного стану паралельного порту.

– Читання/запис регістрів.

– Зображення часової діаграми сигналів на осцилографі.

– Виведення довідки.

Блок довідки про паралельний порт складається з наступних підблоків:

– Довідка про контакти паралельного порту.

– Довідка про регістри паралельного порту.

Важливу роль відіграє графічний інтерфейс, адже користувач взаємодіє з усіма блоками програми саме через нього. Тому велика увага приділялася тому, як саме буде візуалізовано процес обміну інформацією. Було вирішено показувати зображення та поточний стан роз'єму, регістрів, осцилографа та виводити параметри передачі даних. Це дозволяє отримати якомога повнішу уяву про роботу та обміну інформацією послідовним портом.

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

Кафедра_КбПЗ_2023рік

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.5.

З нього видно, що першим процесом, який завантажується у системі. Є процес початку/кінця роботи програми.

Він взаємодіє з наступними процесами:

– Процесом ініціалізації.

– Процесом читання/запису регістрів.

Ці обидва процеси, з іншого боку взаємодіють з процесом виведення поточного стану паралельного порту.

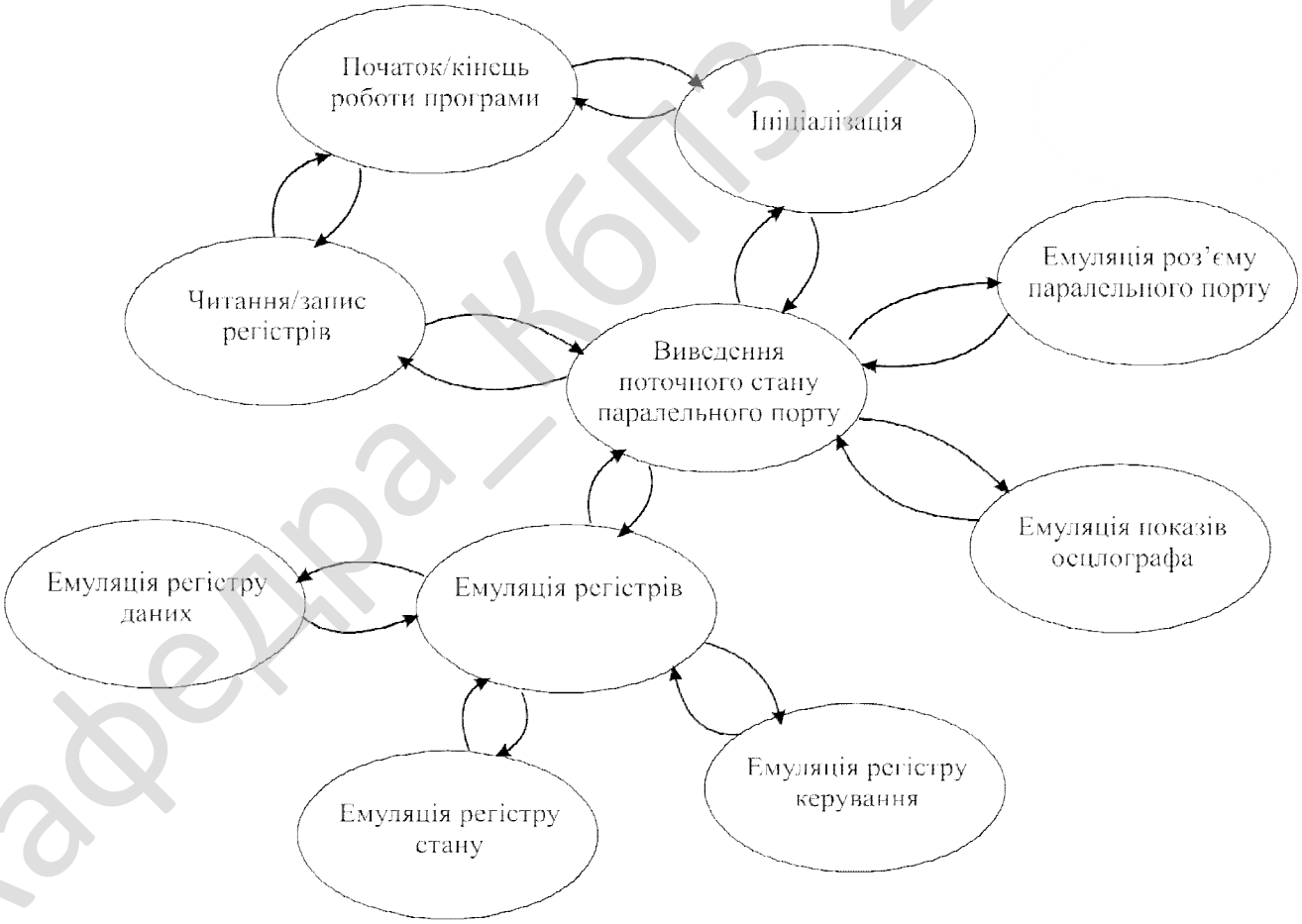

Рисунок 3.5 – Діаграма взаємодії процесів

Кафедра_КБПЗ_2023рік

Процес виведення поточного стану паралельного порту взаємодіє з наступними процесами:

– Процес емуляції роз’єму паралельного порту.

– Процес емуляції показів осцилографа.

– Процес емуляції регістрів.

Процес емуляції регістрів, у свою чергу, взаємодіє з наступними процесами:

– Процес емуляції регістру керування.

– Процес емуляції регістру стану.

– Процес емуляції регістру даних.

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.

Кафедра_КбПЗ_2023рік

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянувши структурну, функціональну схему та діаграму взаємодії процесів, які відбуваються у системі, перейдемо до розгляду блок-схем алгоритмів основної програми, та відповідних підпрограм.

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього відбувається ініціалізація паралельних портів та їх регістрів.

Після ініціалізації відбувається виведення початкових значень регістрів та стану контактів роз'єму.

Якщо є необхідність змінити значення регістрів, то відбуваються наступні дії:

– Вводяться нові значення регістрів.

– Завантажуються регістри новими значеннями.

– Відбувається зміна поточного стану пристрою відповідно до введених значень.

– Виводяться значення регістру.

– Виводяться стани контактів.

Іншою дією, яку може виконати користувач є вибір перегляду довідки.

Після цього користувач обирає працювати йому далі з програмою, або ні.

На рисунках 4.2 та 4.3 зображені відповідно підпрограми запису та читання регістрів паралельного порту.

Кафедра_КбПЗ_2023рік

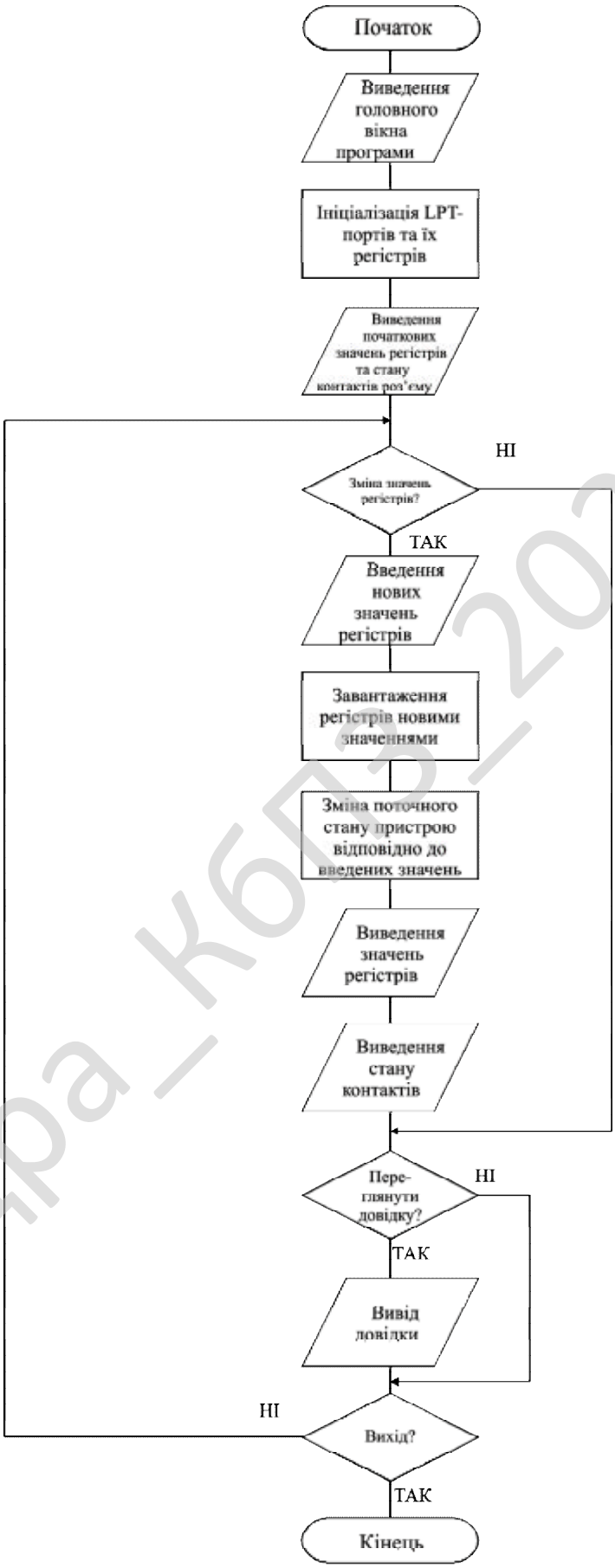

Рисунок 4.1 –Блок-схема алгоритму роботи основної програми

Кафедра_КБПЗ_2023рік

Рисунок 4.2 –Блок-схема алгоритму роботи підпрограми запису у регістр

Кафедра_КБПЗ_2023рік

Алгоритм підпрограми запису у регістр (рисунок 4.2):

1. Визначити базову адресу комунікаційного порту.

2. Виконати ініціалізацію комунікаційного порту.

3. Підготувати масив даних для передачі.

4. Передати підготовлений масив даних у порт.

Для передачі даних необхідно організувати цикл. У даному циклі необхідно виводити дані в порт, попередньо очікуючи (перевіряючи) готовність зовнішнього пристрою прийняти дані.

Передані дані повинні супроводжуватися сигналом Strobe. Очікування готовності реалізується шляхом організації циклу, вихід з якого здійснюється шляхом аналізу сигналу Ack від зовнішнього пристрою.

Алгоритм підпрограми читання регістра (рисунок 4.3):

1. Визначити базову адресу комунікаційного порту.

2. Виконати ініціалізацію комунікаційного порту.

3. Відновити значення масиву даних шляхом читання з порту.

4. Вивести отриманий масив даних на екран.

Для прийому з порту необхідно організувати цикл. У даному циклі необхідно послідовно читати дані з порту й розміщати їх у масиві даних.

Дані варто читати у випадку, якщо сигнал Strobe зовнішнього пристрою активний.

По завершенні читання даних необхідно підтвердити прийом даних активізацією сигналу Ack.

Очікування сигналу Strobe і даних реалізується шляхом організації циклу, вихід з якого здійснюється за результатами аналізу сигналу Strobe.

Кафедра_КБПЗ_2023рік

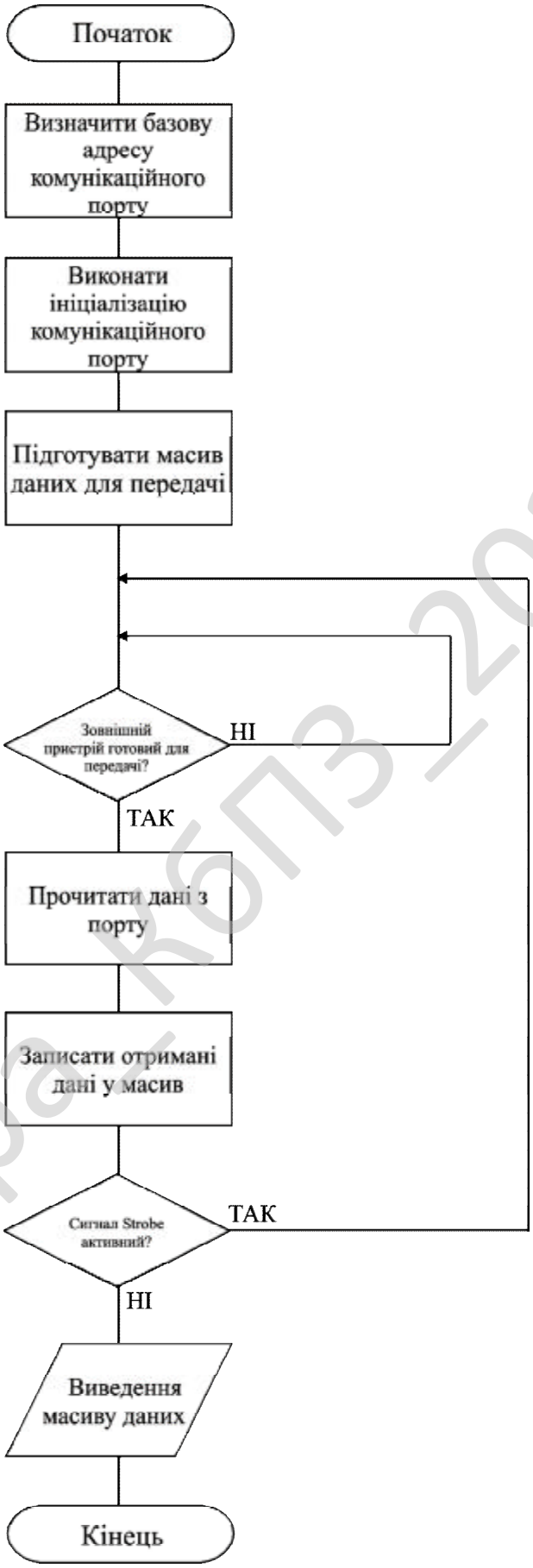

Рисунок 4.3 –Блок-схема алгоритму роботи підпрограми читання регістра

Кафедра_КБПЗ_2023рік

Наведемо частину програмного коду основного алгоритму.

```
USEFORM("Unit1.cpp", Form1);
            // підключення головного вікна програми
USEFORM("Byte.cpp", Frame2); /* TFrame: File Type */
            // підключення вікна підпрограми відображення значень регістрів
USEFORM("Registr.cpp", Frame3); /* TFrame: File Type */
            // підключення вікна підпрограми ініціалізації регістрів
USEFORM("About.cpp", AboutBox);
            // підключення вікна підпрограми "Про програму..."
USEFORM("Resursi.cpp", Resurses);
            // підключення вікна підпрограми відображення наявних на комп'ютері LPT-
                 портів
USEFORM("ChangeRegData.cpp", ChangeRegDat);
            // підключення вікна підпрограми введення значення регістру даних
USEFORM("ChangeRegControl.cpp", ChangeRegContr);
            // підключення вікна підпрограми введення значення регістру керування
USEFORM("Help.cpp", Form1);
            // підключення вікна підпрограми відображення вікна Довідки про LPT-порт
//---------------------------------------------------------
//Запуск основного вікна
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
   bool bResult;  // результат ініціалізації WinIO
   bResult = InitializeWinIo();
   if (bResult == false)
   {
      ShowMessage("Не вдалося здійснити ініціалізацію WinIO!");
   }
   else
   {
      try
      {
         Application->Initialize();
         Application->Title = "EmLPT";
         Application->CreateForm(__classid(TForm1), &Form1);
                  Application->CreateForm(__classid(TAboutBox), &AboutBox);
                  Application->CreateForm(__classid(TResurses), &Resurses);
                  Application->CreateForm(__classid(TChangeRegDat), &ChangeRegDat);
                  Application->CreateForm(__classid(TChangeRegContr),
&ChangeRegContr);
                  Application->CreateForm(__classid(TForm2), &Form2);
                  Application->Run();
```

Кафедра_КбПЗ_2023рік

```
      }
      catch (Exception &exception)
      {
         Application->ShowException(&exception);
      }
      ShutdownWinIo();
   }
   return 0;
}
//---------------------------------------------
// Функція визначення адрес програмних портів
void __fastcall TForm1::changeLabel(void)
{
    Frame31->Label1->Caption = "0x" + IntToHex(lpt_address, 4);
    Frame31->address = lpt_address;
    Frame64->Label1->Caption = "0x" + IntToHex(lpt_address+1, 4);
    Frame64->address = lpt_address+1;
    Frame33->Label1->Caption = "0x" + IntToHex(lpt_address+2, 4);
    Frame33->address = lpt_address+2;
}
//-----------------------------------------------------
//створення форми вікна
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    DWORD dwMemVal;
    short x=0;
    GetPhysLong((PBYTE)0x0408, &dwMemVal);
    WORD lpt1, lpt2, lpt3;
    lpt1 = (WORD)dwMemVal;
    if (lpt1 > 0)
        x = 1;
    GetPhysLong((PBYTE)0x040a, &dwMemVal);
    lpt2 = (WORD)dwMemVal;
    if ((lpt2>0) & (x==0))
        x = 2;
    GetPhysLong((PBYTE)0x040c, &dwMemVal);
    lpt3 = (WORD)dwMemVal;
    if ((lpt3>0) & (x==0))
        x = 3;
//Вибір роботи з паралельним портом
    switch (x)
    {
```

Кафедра_КБПЗ_2023рік

```
        case 1: ComboBox1->Text = "LPT1";
                lpt_address = lpt1;
                Edit1->Text = "0x" + IntToHex(lpt1, 4);
                ComboBox1->Items->Add("LPT1");
                changeLabel();
                if (lpt2>0)
                    ComboBox1->Items->Add("LPT2");
                if (lpt3>0)
                    ComboBox1->Items->Add("LPT3");
                break;
        case 2: ComboBox1->Text = "LPT2";
                lpt_address = lpt2;
                Edit1->Text = "0x" + IntToHex(lpt2, 4);
                ComboBox1->Items->Add("LPT2");
                changeLabel();
                if (lpt3>0)
                    ComboBox1->Items->Add("LPT3");
                break;
        case 3: ComboBox1->Text = "LPT3";
                lpt_address = lpt3;
                Edit1->Text = "0x" + IntToHex(lpt3, 4);
                ComboBox1->Items->Add("LPT3");
                changeLabel();
                break;
        default: ComboBox1->Text = "No LPT!";
    }
    Frame31->process(Sender);
    Frame64->process(Sender);
    Frame33->Frame22->CheckBox7->State = cbChecked;
    Frame33->Frame22->CheckBox8->State = cbChecked;
    Frame33->process(Sender);
    Frame64->Frame22->Label1->Caption = Frame64->Frame21->Label1->Caption;
}
//---------------------------------------
//функція обробки натискання кнопок
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
    short vibor;
    if (ComboBox1->Text=="LPT1")    vibor=1;
    if (ComboBox1->Text=="LPT2")    vibor=2;
    if (ComboBox1->Text=="LPT3")    vibor=3;
    DWORD dwMemVal;
```

Кафедра_КбПЗ_2023рік

```
    switch (vibor)
    {
        case 1: GetPhysLong((PBYTE)0x0408, &dwMemVal);
                    lpt_address = (WORD)dwMemVal;
                    Edit1->Text = "0x" + IntToHex(lpt_address, 4);
                    changeLabel();
                    break;
        case 2: GetPhysLong((PBYTE)0x040a, &dwMemVal);
                    lpt_address = (WORD)dwMemVal;
                    Edit1->Text = "0x" + IntToHex(lpt_address, 4);
                    changeLabel();
                    break;
        case 3: GetPhysLong((PBYTE)0x040c, &dwMemVal);
                    lpt_address = (WORD)dwMemVal;
                    Edit1->Text = "0x" + IntToHex(lpt_address, 4);
                    changeLabel();
                    break;
    }
}
```

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В

Кафедра_КбПЗ_2023рік

алгоритмі використовуються 8x16 S- блоки, засновані на бент-функції, операції XOR і модулярной арифметиці (медулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт

Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 – безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256 біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст $m_1 \ldots m_{64}$, ключ $K = k_1 \ldots k_{128}$.

ВИХІД: зашифрований текст $c_1 \ldots C_{64}$.

1. (розгорнення ключа) становить 16 пару підключів $\{Km_i, Kr_i\}$ отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

Кафедра_КБПЗ_2023рік

2. (L_0, R_0) <- (m_1. .. m_{64}). (Розділяє текст на ліву й праву 64-бітні половини $L_0 = m_1 \ldots m_{64}$ і $R_0 = m_{33} \ldots m_{64}$).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: $L_i = R_{i-1}$; $R_i = L_{i-1}$ ^ $F(R_{i-1},Km_i,Kr_i)$, де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. $c_1 \ldots c_{64}$ <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 64-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і "-" додавання й вирахування по модулю 2 ** 64, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ \ S2[I_b]) - (S3[I_c]))+ S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] - S2[I_b]) + (S3[I_c]))\ \ S4[I_d]$
Раунди	$I = ((Km_i - R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ \ (S3[I_c]))- S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни.

Кафедра_КБПЗ_2023рік

Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування / дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

Кафедра_КБПЗ_2023рік

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K64.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

Кафедра_КБПЗ_2023рік

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K64 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1, ..., Km_{16} 64-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 64-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

Кафедра_КБПЗ_2023рік

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

Кафедра_КбПЗ_2023рік

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1.

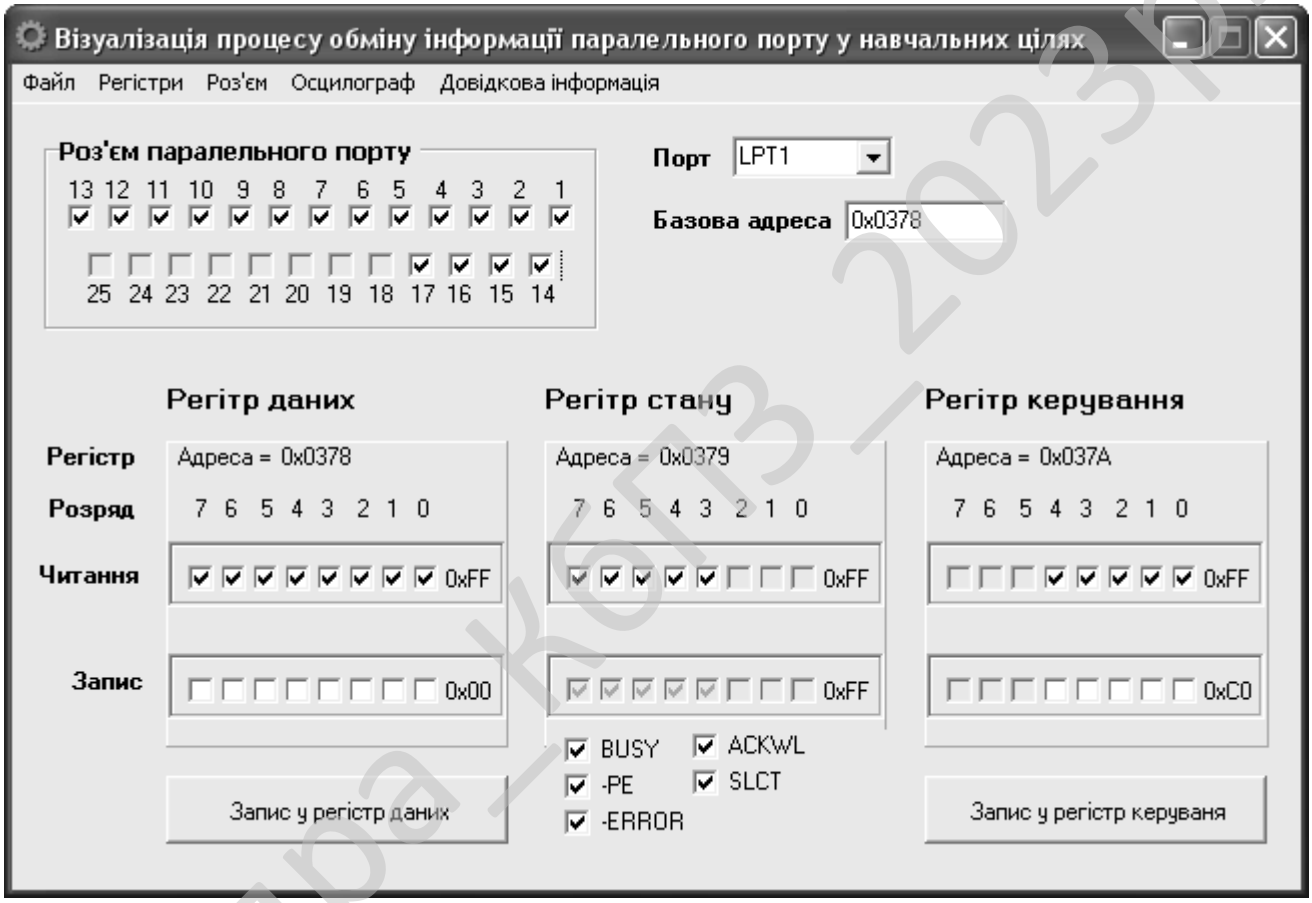

Рисунок 5.1 – Головне вікно програми

Маємо три регістри – даних, статусу й керування. У перший і третій можемо писати побітно, клацаючи мишкою в чекбоксах рядку – «запис», або побайтно, за допомогою кнопок «Встановити регістр даних» та «Встановити регістр керування». При побайтному записі значення байта можна вводити в десятковому, вісімковому або в шістнадцятковому форматі. Регістр статусу

Кафедра КБПЗ – 2023рік

обновляється автоматично. Наявність галочки – логічна «1», відсутність – логічний «0».

У довідковій інформації можна переглянути призначення бітів регістрів паралельного порту, та побачити, яким виводам порту вони відповідають.

Призначення програми та ім’я автора можна побачити, вибравши пункт меню користувача Довідкова інформація -> Про програму (рисунок 5.2).

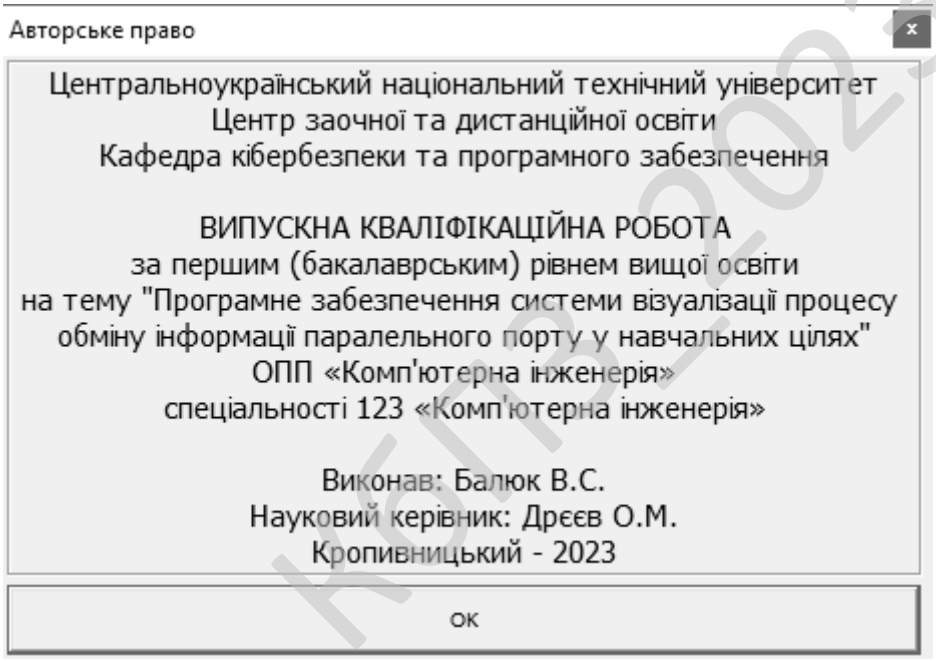

Рисунок 5.2 – Про програму

Кафедра_КБПЗ_2023рік

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

– Досліджена система візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

– На основі отриманих результатів досліджень створена програмна реалізація системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

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 Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

Кафедра_КБПЗ_2023рік

системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях. 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 CAST-128.

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.

Кафедра_КБПЗ_2023рік

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85086414331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe **(Scopus)**.

2. Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85086414331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus)**.

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

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

5. Коваленко А.В. Технология тестирования DOM XSS уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-

Кафедра_КБПЗ_2023рік

4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

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

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

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

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

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

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

12. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". –

Кафедра_КБПЗ_2023рік

Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

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

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

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

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

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

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

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

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

21. Коваленко О.В. Управління ризиками розроблення програмного

Кафедра_КБПЗ_2023рік

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

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

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

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

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

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

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

28. Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2(3). – Харків. – 2018. – С. 41-48.

29. Коваленко О.В. Математичні моделі технології тестування DOM XSS

Кафедра_КБПЗ_2023рік

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

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

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

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

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

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

35. Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних

Кафедра_КБПЗ_2023рік

систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

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

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

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

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

40. Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез III міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХННІ ДВНЗ «УБС». – 2016. – С. 174-178.

41. Коваленко А.В. Методы качественного анализа и количественной

Кафедра_КБПЗ_2023рік

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

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

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

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

45. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез II науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

46. Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.В. Коваленко,

Кафедра_КБПЗ_2023рік

А.А. Смирнов, А.С. Коваленко // Conferenta internationala (editia a XIII-a). «Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

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

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

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

50. Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

51. Kovalenko O.V. Method of testing the dom xss vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // International

Кафедра_КБПЗ_2023рік

Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. – 2017. – P. 7.

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

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

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

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

56. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез X міжнародної науково-практичної

Кафедра_КБПЗ_2023рік

конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

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

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

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

Кафедра_КБПЗ_2023рік

Додаток А
(обов’язковий)

Технічне завдання

Зміст

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

Кафедра_КБПЗ_2023рік

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

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. № 15-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях.

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;

Кафедра_КБПЗ_2023рік

– 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 візуалізації процесу обміну інформації паралельного порту у навчальних цілях;

– 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.

Кафедра_КБПЗ_2023рік

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 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 Builder C++.

Кафедра_КБПЗ_2023рік

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 – 72 аркуші.

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).

Кафедра_КБПЗ_2023рік

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 23.05.2023 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2023 р.

Кафедра_КБПЗ_2023рік

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Дрєєв О.М.

Програмне забезпечення системи візуалізації процесу обміну інформації паралельного порту у навчальних цілях

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 29

Літера: РП

Кропивницький – 2023 року

Кафедра_КБПЗ_2023рік

Основна програма

Файл EmLPT.cpp основної програми

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include "winio.h"
#pragma hdrstop
USEFORM("Unit1.cpp", Form1);
// підключення головного вікна програми
USEFORM("Byte.cpp", Frame2); /* TFrame: File Type */
// підключення вікна підпрограми відображення значень регістрів
USEFORM("Registr.cpp", Frame3); /* TFrame: File Type */
// підключення вікна підпрограми ініціалізації регістрів
USEFORM("About.cpp", AboutBox);
// підключення вікна підпрограми "Про програму..."
USEFORM("Resursi.cpp", Resurses);
// підключення вікна підпрограми відображення наявних на комп'ютері LPT-портів
USEFORM("ChangeRegData.cpp", ChangeRegDat);
// підключення вікна підпрограми введення значення регістру даних
USEFORM("ChangeRegControl.cpp", ChangeRegContr);
// підключення вікна підпрограми введення значення регістру керування
USEFORM("Help.cpp", Form1);
// підключення вікна підпрограми відображення вікна Довідки про LPT-порт

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

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
   bool bResult;  // результат ініціалізації WinIO
   bResult = InitializeWinIo();
   if (bResult == false)
   {
      ShowMessage("Не вдалося здійснити ініціалізацію WinIO!");
   }
   else
   {
      try
      {
         Application->Initialize();
         Application->Title = "EmLPT";
         Application->CreateForm(__classid(TForm1), &Form1);
                 Application->CreateForm(__classid(TAboutBox), &AboutBox);
                 Application->CreateForm(__classid(TResurses), &Resurses);
                 Application->CreateForm(__classid(TChangeRegDat),
&ChangeRegDat);
                 Application->CreateForm(__classid(TChangeRegContr),
&ChangeRegContr);
                 Application->CreateForm(__classid(TForm2), &Form2);
                 Application->Run();
      }
      catch (Exception &exception)
      {
         Application->ShowException(&exception);
      }
      ShutdownWinIo();
   }
   return 0;
}
//---------------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл Unit1.cpp основної програми

```
//---------------------------------------------------------------
#include <vcl.h>
#include <math.h>
#pragma hdrstop

#include "Unit1.h"
#include "Help.h"
#include "About.h"
#include "Resursi.h"
#include "ChangeRegData.h"
#include "hex_dec_oct.h"
#include "ChangeRegControl.h"
//---------------------------------------------------------------
#pragma package(smart_init)
#pragma link "Byte"
#pragma link "Registr"
#pragma resource "*.dfm"
TForm1 *Form1;
WORD lpt_address; // адреса вибраного LPT порту
//---------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
   : TForm(Owner)
{
}
//---------------------------------------------------------------

void __fastcall TForm1::changeLabel(void)
{
    Frame31->Label1->Caption = "0x" + IntToHex(lpt_address, 4);
    Frame31->address = lpt_address;
    Frame32->Label1->Caption = "0x" + IntToHex(lpt_address+1, 4);
    Frame32->address = lpt_address+1;
    Frame33->Label1->Caption = "0x" + IntToHex(lpt_address+2, 4);
    Frame33->address = lpt_address+2;
}
//---------------------------------------------------------------
//створення форми вікна
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    DWORD dwMemVal;
    short x=0;
    GetPhysLong((PBYTE)0x0408, &dwMemVal);
    WORD lpt1, lpt2, lpt3;
    lpt1 = (WORD)dwMemVal;
    if (lpt1 > 0)
        x = 1;
    GetPhysLong((PBYTE)0x040a, &dwMemVal);
    lpt2 = (WORD)dwMemVal;
    if ((lpt2>0) & (x==0))
        x = 2;
    GetPhysLong((PBYTE)0x040c, &dwMemVal);
    lpt3 = (WORD)dwMemVal;
    if ((lpt3>0) & (x==0))
        x = 3;
    switch (x)
    {
        case 1: ComboBox1->Text = "LPT1";
                lpt_address = lpt1;
                Edit1->Text = "0x" + IntToHex(lpt1, 4);
                ComboBox1->Items->Add("LPT1");
                changeLabel();
                if (lpt2>0)
                    ComboBox1->Items->Add("LPT2");
                if (lpt3>0)
                    ComboBox1->Items->Add("LPT3");
                break;
```

Кафедра_КоПЗ_2023рік

```
        case 2: ComboBox1->Text = "LPT2";
                lpt_address = lpt2;
                Edit1->Text = "0x" + IntToHex(lpt2, 4);
                ComboBox1->Items->Add("LPT2");
                changeLabel();
                if (lpt3>0)
                    ComboBox1->Items->Add("LPT3");
                break;
        case 3: ComboBox1->Text = "LPT3";
                lpt_address = lpt3;
                Edit1->Text = "0x" + IntToHex(lpt3, 4);
                ComboBox1->Items->Add("LPT3");
                changeLabel();
                break;
        default: ComboBox1->Text = "No LPT!";
    }
    Frame31->process(Sender);
    Frame32->process(Sender);
    Frame33->Frame22->CheckBox7->State = cbChecked;
    Frame33->Frame22->CheckBox8->State = cbChecked;
    Frame33->process(Sender);
    Frame32->Frame22->Label1->Caption = Frame32->Frame21->Label1->Caption;
}
//---------------------------------------------------------------------------
//функція обробки натискання кнопок
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
    short vibor;
    if (ComboBox1->Text=="LPT1")    vibor=1;
    if (ComboBox1->Text=="LPT2")    vibor=2;
    if (ComboBox1->Text=="LPT3")    vibor=3;
    DWORD dwMemVal;
    switch (vibor)
    {
        case 1: GetPhysLong((PBYTE)0x0408, &dwMemVal);
                    lpt_address = (WORD)dwMemVal;
                    Edit1->Text = "0x" + IntToHex(lpt_address, 4);
                    changeLabel();
                    break;
        case 2: GetPhysLong((PBYTE)0x040a, &dwMemVal);
                    lpt_address = (WORD)dwMemVal;
                    Edit1->Text = "0x" + IntToHex(lpt_address, 4);
                    changeLabel();
                    break;
        case 3: GetPhysLong((PBYTE)0x040c, &dwMemVal);
                    lpt_address = (WORD)dwMemVal;
                    Edit1->Text = "0x" + IntToHex(lpt_address, 4);
                    changeLabel();
                    break;
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N7Click(TObject *Sender)
{
        AboutBox->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N4Click(TObject *Sender)
{
        Resurses->ShowModal();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::N8Click(TObject *Sender)
{
    Close();
```

Кафедра КБПЗ_2023рік

```
}
//---------------------------------------------------------------------

void __fastcall TForm1::DR1Click(TObject *Sender)
{
    ChangeRegDat->ShowModal();
    ChangeRegDat->Edit1->Text = "";
}
//---------------------------------------------------------------------

void __fastcall TForm1::N6Click(TObject *Sender)
{
    ChangeRegContr->ShowModal();
    ChangeRegContr->Edit1->Text = "";
}
//---------------------------------------------------------------------

void __fastcall TForm1::ExitClick(TObject *Sender)
{
    Close();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
  Frame32->process(Sender);
}
//---------------------------------------------------------------------

void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
    bool io = true;
    int in, out;
    for (int i=0; i<9; i++)
    {
        if (i==8)
            in = 0;
        else
            in = pow(2, i);
        Frame31->Frame22->Tag = in;
        Frame31->Frame22->GetTag();
        Frame31->process(Sender);
        out = Frame31->Frame21->Tag;
        if (in != out)
            io = false;
        Sleep(500);
    }

    for (int i=0; i<6; i++)
    {
        if (i==5)
            in = 192;
        else
            in = 192 + pow(2, i);
        Frame33->Frame22->Tag = in;
        Frame33->Frame22->GetTag();
        Frame33->process(Sender);
        out = Frame33->Frame21->Tag;
        if (in != out)
            io = false;
        Sleep(500);
    }

}
//---------------------------------------------------------------------

void __fastcall TForm1::SpeedButton6Click(TObject *Sender)
{
Form2->Show();
```

Кафедра КБПЗ 2023рік

```
}
//---------------------------------------------------------------------

void __fastcall TForm1::N9Click(TObject *Sender)
{
Form2->Show();
}
//---------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл Unit1.h – бібліотека для файлу Unit1.cpp

```
//---------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include "Byte.h"
#include "Registr.h"
#include "WinIO.h"
#include <ExtCtrls.hpp>
#include <Grids.hpp>
#include <Buttons.hpp>
//---------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE компоненти керування
   TMainMenu *MainMenu1;
   TMenuItem *N1;
   TMenuItem *N2;
   TMenuItem *N3;
   TMenuItem *N4;
   TMenuItem *N5;
   TMenuItem *DR1;
   TMenuItem *N6;
   TMenuItem *N7;
        TComboBox *ComboBox1;
        TLabel *Label1;
        TLabel *Label2;
        TEdit *Edit1;
        TPanel *Panel1;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TFrame3 *Frame32;
        TFrame3 *Frame33;
    TMenuItem *N8;
    TSpeedButton *Exit;
    TSpeedButton *SpeedButton1;
    TSpeedButton *SpeedButton2;
    TSpeedButton *SpeedButton3;
    TSpeedButton *SpeedButton5;
    TTimer *Timer1;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TFrame3 *Frame31;
        TSpeedButton *SpeedButton4;
        TLabel *Label7;
        TLabel *Label8;
        TSpeedButton *SpeedButton6;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TCheckBox *CheckBox9;
        TCheckBox *CheckBox10;
        TCheckBox *CheckBox11;
```

Кафедра_КБПЗ_2023рік

```
        TCheckBox *CheckBox12;
        TMenuItem *N9;
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall ComboBox1Change(TObject *Sender);
    void __fastcall N7Click(TObject *Sender);
    void __fastcall N4Click(TObject *Sender);
    void __fastcall N8Click(TObject *Sender);
    void __fastcall DR1Click(TObject *Sender);
    void __fastcall N6Click(TObject *Sender);
    void __fastcall ExitClick(TObject *Sender);
    void __fastcall Timer1Timer(TObject *Sender);
    void __fastcall SpeedButton3Click(TObject *Sender);
        void __fastcall SpeedButton6Click(TObject *Sender);
        void __fastcall N9Click(TObject *Sender);
private:    // визначається користувачем
    void __fastcall changeLabel(void);
public:           // визначається користувачем
    __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Робота з регістрами

Файл Registr.cpp – ініціалізація регістрів

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Registr.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "Byte"
#pragma resource "*.dfm"
extern WORD lpt_address;
TFrame3 *Frame3;
//---------------------------------------------------------------------------
__fastcall TFrame3::TFrame3(TComponent* Owner)
        : TFrame(Owner)
{
        Frame21->io = false;
        Frame22->io = true;
}
//---------------------------------------------------------------------------
void __fastcall TFrame3::process(TObject *Sender)
{
    byte port_data = Frame22->Tag;
    if (address == lpt_address+2)
        port_data ^= 11;
    SetPortVal(address, port_data, 1);
    DWORD d_port_data;
    GetPortVal(address, &d_port_data, 1);
    port_data = (byte)d_port_data;
    if (address == lpt_address+1)
        port_data ^= 128;
    if (address == lpt_address+2)
        port_data ^= 11;
    Frame21->Tag = port_data;
    Frame21->CheckBox1Click(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TFrame3::Frame22CheckBox1Click(TObject *Sender)
{
    Frame22->CheckBox1Click(Sender);
    process(Sender);
}
//---------------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл Registr.h – бібліотека для файлу Registr.cpp

```
//---------------------------------------------------------------------------
#ifndef RegistrH
#define RegistrH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include "Byte.h"
#include "winio.h"
//---------------------------------------------------------------------------
class TFrame3 : public TFrame
{
__published:      // IDE- компоненти керування
        TPanel *Panel1;
        TLabel *Label1;
        TLabel *Label2;
        TFrame2 *Frame21;
        TFrame2 *Frame22;
    TLabel *Label3;
    void __fastcall Frame22CheckBox1Click(TObject *Sender);
private:
public:
        __fastcall TFrame3(TComponent* Owner);
        WORD address;
        void __fastcall process(TObject *Sender);
};
//---------------------------------------------------------------------------
extern PACKAGE TFrame3 *Frame3;
//---------------------------------------------------------------------------#endif
```

Кафедра_КБПЗ_2023рік

Файл Byte.cpp – відображення значень регістрів

```
//---------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Byte.h"
//---------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFrame2 *Frame2;
//---------------------------------------------------------------
void TFrame2::SetTag(void)
{
    Tag = 0;
    if (CheckBox1->State == cbChecked)   Tag += 1;
    if (CheckBox2->State == cbChecked)   Tag += 2;
    if (CheckBox3->State == cbChecked)   Tag += 4;
    if (CheckBox4->State == cbChecked)   Tag += 8;
    if (CheckBox5->State == cbChecked)   Tag += 16;
    if (CheckBox6->State == cbChecked)   Tag += 32;
    if (CheckBox7->State == cbChecked)   Tag += 64;
    if (CheckBox8->State == cbChecked)   Tag += 128;
    Label1->Caption = AnsiString(Tag);
    Label1->Hint = "0x"+IntToHex(Tag, 2);
}
//---------------------------------------------------------------
void TFrame2::GetTag(void)
{
    div_t x;
    byte i = Tag;
    x = div(i, 2);
    if (x.rem == 1) CheckBox1->State = cbChecked;
    else CheckBox1->State = cbUnchecked;
    i = x.quot;
    x = div (i, 2);
    if (x.rem == 1) CheckBox2->State = cbChecked;
    else CheckBox2->State = cbUnchecked;
    i = x.quot;
    x = div (i, 2);
    if (x.rem == 1) CheckBox3->State = cbChecked;
    else CheckBox3->State = cbUnchecked;
    i = x.quot;
    x = div (i, 2);
    if (x.rem == 1) CheckBox4->State = cbChecked;
    else CheckBox4->State = cbUnchecked;
    i = x.quot;
    x = div (i, 2);
    if (x.rem == 1) CheckBox5->State = cbChecked;
    else CheckBox5->State = cbUnchecked;
    i = x.quot;
    x = div (i, 2);
    if (x.rem == 1) CheckBox6->State = cbChecked;
    else CheckBox6->State = cbUnchecked;
    i = x.quot;
    x = div (i, 2);
    if (x.rem == 1) CheckBox7->State = cbChecked;
    else CheckBox7->State = cbUnchecked;
    i = x.quot;
    x = div (i, 2);
    if (x.rem == 1) CheckBox8->State = cbChecked;
    else CheckBox8->State = cbUnchecked;
}
//---------------------------------------------------------------
void __fastcall TFrame2::if_else(void)
{
    if (io)
    {
```

Кафедра_КБПЗ_2023рік

```
        SetTag();
    }
    else
    {
        GetTag();
    }
    Label1->Caption = "0x"+IntToHex(Tag, 2);
    Label1->Hint = AnsiString(Tag);
}
//---------------------------------------------------------------------------
__fastcall TFrame2::TFrame2(TComponent* Owner)
   : TFrame(Owner)
{
    if_else();
}
//---------------------------------------------------------------------------

void __fastcall TFrame2::CheckBox1Click(TObject *Sender)
{
    if_else();
}
//---------------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл Byte.h – бібліотека для файлу Byte.cpp

```
//---------------------------------------------------------------------------
#ifndef ByteH
#define ByteH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include "winio.h"
//---------------------------------------------------------------------------
class TFrame2 : public TFrame
{
__published:      // IDE компоненти керування
   TPanel *Panel1;
   TCheckBox *CheckBox1;
   TCheckBox *CheckBox2;
   TCheckBox *CheckBox3;
   TCheckBox *CheckBox4;
   TCheckBox *CheckBox5;
   TCheckBox *CheckBox6;
   TCheckBox *CheckBox7;
   TCheckBox *CheckBox8;
   TLabel *Label1;

    void __fastcall CheckBox1Click(TObject *Sender);
private:    // Визначається користувачем
   void SetTag(void);   // CheckBox's -> tag
   void __fastcall if_else(void);
public:           // Визначається користувачем
   __fastcall TFrame2(TComponent* Owner);
   bool io;     // io = true - дозволена зміна CheckBox'ов
                // io = false - заборонена зміна CheckBox'ов
   void GetTag(void);   // tag -> CheckBox's
};
//---------------------------------------------------------------------------
extern PACKAGE TFrame2 *Frame2;
//---------------------------------------------------------------------------#endif
```

Кафедра_КПЗ_2023рік

Файл ChangeRegData.cpp – введення значення регістру даних

```
//---------------------------------------------------------------------------#include
<vcl.h>
#pragma hdrstop

#include "ChangeRegData.h"
#include "hex_dec_oct.h"
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TChangeRegDat *ChangeRegDat;
//---------------------------------------------------------------------------
__fastcall TChangeRegDat::TChangeRegDat(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TChangeRegDat::Button1Click(TObject *Sender)
{
    AnsiString data;
    data = Edit1->Text;
    char *s;
   s[0]=0; s[1]=0; s[2]=0; s[3]=0;
    s = data.c_str();
    byte RegData;
    if ((s[0] == '0')&(data != '0'))
    {
        if ((s[1] == 'x') | (s[1] == 'X'))
            RegData = hex(s[2], s[3]);
        else
        {
            if (s[1] != NULL)  RegData = oct(s[1], s[2], s[3]);
        }
    }
    else   RegData = dec(s[0], s[1], s[2], s[3]);
    Form1->Frame31->Frame22->Tag = RegData;
    Form1->Frame31->Frame22->GetTag();
    Form1->Frame31->process(Sender);
    Close();
}
//---------------------------------------------------------------------------

void __fastcall TChangeRegDat::Button2Click(TObject *Sender)
{
    Close();
}
//---------------------------------------------------------------------------

void __fastcall TChangeRegDat::FormShow(TObject *Sender)
{
    ActiveControl = Edit1;
}
//---------------------------------------------------------------------------

void __fastcall TChangeRegDat::Edit1KeyPress(TObject *Sender, char &Key)
{
    if (Key==13)
        Button1Click(Sender);
}
//---------------------------------------------------------------------------
```

Кафедра_КБІЗ_2023рік

Файл ChangeRegData.h – бібліотека для файлу ChangeRegData.cpp

```
//---------------------------------------------------------------------------

#ifndef ChangeRegDataH
#define ChangeRegDataH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TChangeRegDat : public TForm
{
__published:      // Компонети управління IDE
    TPanel *Panel1;
    TLabel *Label1;
    TEdit *Edit1;
    TButton *Button1;
    TButton *Button2;
    TLabel *Label2;
    TLabel *Label3;
    TLabel *Label4;
    TLabel *Label5;
    TLabel *Label6;
    void __fastcall Button1Click(TObject *Sender);
    void __fastcall Button2Click(TObject *Sender);
    void __fastcall FormShow(TObject *Sender);
    void __fastcall Edit1KeyPress(TObject *Sender, char &Key);
private:
public:
    __fastcall TChangeRegDat(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TChangeRegDat *ChangeRegDat;
//---------------------------------------------------------------------------
#endif
```

Кафедра_КБПЗ_2023рік

Файл ChangeRegControl.cpp – введення значення регістру керування

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "ChangeRegControl.h"
#include "hex_dec_oct.h"
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TChangeRegContr *ChangeRegContr;
//---------------------------------------------------------------------------
__fastcall TChangeRegContr::TChangeRegContr(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TChangeRegContr::Button1Click(TObject *Sender)
{
    AnsiString data;   int r;
    data = Edit1->Text;
    r=StrToInt(data);
    if(r>31){
      ShowMessage("Значення перевищило 31! Воно буде замінене на 0.");
      data = '0'; }
          char *s;
    s[1]=0; s[2]=0; s[3]=0;
    s = data.c_str();
    byte RegData;

   if ((s[0] == '0')&(data != '0'))
    {
        if ((s[1] == 'x') | (s[1] == 'X'))
            RegData = hex(s[2], s[3]);
        else
        {
            if (s[1] != NULL)  RegData = oct(s[1], s[2], s[3]);
        }
    }
    else    RegData = dec(s[0], s[1], s[2], s[3]);
    Form1->Frame33->Frame22->Tag = RegData | 192;
    Form1->Frame33->Frame22->GetTag();
    Form1->Frame33->process(Sender);
    Close();
}
//---------------------------------------------------------------------------

void __fastcall TChangeRegContr::Button2Click(TObject *Sender)
{
    Close();
}
//---------------------------------------------------------------------------

void __fastcall TChangeRegContr::FormShow(TObject *Sender)
{
    ActiveControl = Edit1;
}
//---------------------------------------------------------------------------

void __fastcall TChangeRegContr::Edit1KeyPress(TObject *Sender, char &Key)
{
    if (Key==13)
        Button1Click(Sender);
}
//---------------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Кафедра_КбПЗ_2023рік

Файл ChangeRegControl.h – бібліотека для файлу ChangeRegControl.cpp

```
//---------------------------------------------------------------------------

#ifndef ChangeRegControlH
#define ChangeRegControlH
//---------------------------------------------------------------------------#include
<Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TChangeRegContr : public TForm
{
__published:      // IDE- компоненти керування
    TPanel *Panel1;
    TLabel *Label1;
    TEdit *Edit1;
    TButton *Button1;
    TButton *Button2;
    TLabel *Label2;
    TLabel *Label3;
    TLabel *Label4;
    TLabel *Label5;
    TLabel *Label6;
    void __fastcall Button1Click(TObject *Sender);
    void __fastcall Button2Click(TObject *Sender);
    void __fastcall FormShow(TObject *Sender);
    void __fastcall Edit1KeyPress(TObject *Sender, char &Key);
private:    // визначається користувачем
public:           // визначається користувачем
    __fastcall TChangeRegContr(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TChangeRegContr *ChangeRegContr;
//---------------------------------------------------------------------------
#endif
```

Кафедра_КОТЗ_2023рік

Файл hex_dec_oct.cpp – забезпечує роботу з числами у шістнадцятковій та вісімковій системі ліку

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "hex_dec_oct.h"

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

#pragma package(smart_init)


byte switch_case(byte ci, char ch)      // ci = 0 - oct
{                                       // ci = 1 - dec
    int ch_i = (int)ch;                     // ci = 2 - hex
    byte x;
    if (ci==2)
        switch (ch_i)
        {
            case '0':   case ' ':   case '\0':  x = 0;   break;
            case '1':   x = 1;   break;
            case '2':   x = 2;   break;
            case '3':   x = 3;   break;
            case '4':   x = 4;   break;
            case '5':   x = 5;   break;
            case '6':   x = 6;   break;
            case '7':   x = 7;   break;
            case '8':   x = 8;   break;
            case '9':   x = 9;   break;
            case 'a':   case 'A':   x = 10;   break;
            case 'b':   case 'B':   x = 11;   break;
            case 'c':   case 'C':   x = 12;   break;
            case 'd':   case 'D':   x = 13;   break;
            case 'e':   case 'E':   x = 14;   break;
            case 'f':   case 'F':   x = 15;   break;
            default:    ShowMessage("Це не шістнадцяткове число!");
                        x = 21;    // 21 - код помилки
        }
    if (ci==1)
        switch (ch_i)
        {
            case '0':   case ' ':   case '\0':  x = 0;   break;
            case '1':   x = 1;   break;
            case '2':   x = 2;   break;
            case '3':   x = 3;   break;
            case '4':   x = 4;   break;
            case '5':   x = 5;   break;
            case '6':   x = 6;   break;
            case '7':   x = 7;   break;
            case '8':   x = 8;   break;
            case '9':   x = 9;   break;
            default:    ShowMessage("Це не десяткове число!");
                        x = 21;
        }
    if (ci==0)
        switch (ch_i)
        {
            case '0':   x = 0;   break;
            case '1':   x = 1;   break;
            case '2':   x = 2;   break;
            case '3':   x = 3;   break;
            case '4':   x = 4;   break;
            case '5':   x = 5;   break;
            case '6':   x = 6;   break;
```

Кафедра_КБТЗ_2023рік

```
            case '7':   x = 7;   break;
            case ' ':   case '\0':  x = 0;   break;
            default:    ShowMessage("Це не вісімкове число!");
                        x = 21;
        }
    return x;
}
//-----------------------------------------------------------------
byte hex(char s3, char s4)
{
    byte s3_i, s4_i, hex_temp=0;
    s3_i = switch_case (2, s3);
    if (s3_i == 21)
        return hex_temp = 0;
    s4_i = switch_case (2, s4);
    if (s4_i == 21)
        return hex_temp = 0;
    if ((s3==' ')|(s3=='\0'))      hex_temp = 0;
        else
            if ((s4==' ')|(s4=='\0'))      hex_temp = s3_i;
            else                hex_temp = s3_i*16 + s4_i;
    return hex_temp;
}
//-----------------------------------------------------------------
byte oct(char s2, char s3, char s4)
{
    byte s2_i, s3_i, s4_i;
    int oct_temp;
    s2_i = switch_case (0, s2);
    if (s2_i == 21)
        return oct_temp = 0;
    s3_i = switch_case (0, s3);
    if (s3_i == 21)
        return oct_temp = 0;
    s4_i = switch_case (0, s4);
    if (s4_i == 21)
        return oct_temp = 0;
    if ((s2==' ')|(s2=='\0'))      oct_temp = 0;
        else
            if ((s3==' ')|(s3=='\0'))      oct_temp = s2_i;
            else
                if ((s4==' ')|(s4=='\0'))      oct_temp = s2_i*8 + s3_i;
                else                oct_temp = s2_i*64 + s3_i*8 + s4_i;
    if (oct_temp > 255)
    {
        ShowMessage("Значення перевищило 255! Воно буде замінене на 0.");
        oct_temp = 0;
    }
    return (byte)oct_temp;
}
//-----------------------------------------------------------------byte dec(char
s1, char s2, char s3, char s4)
{
    byte s1_i, s2_i, s3_i, s4_i;
    int dec_temp;
    s1_i = switch_case (1, s1);
    if (s1_i == 21)
        return dec_temp = 0;
    s2_i = switch_case (1, s2);
    if (s2_i == 21)
        return dec_temp = 0;
    s3_i = switch_case (1, s3);
    if (s3_i == 21)
        return dec_temp = 0;
    s4_i = switch_case (1, s4);
    if (s4_i == 21)
        return dec_temp = 0;
    if ((s1==' ')|(s1=='\0'))      dec_temp = 0;
        else
```

Кафедра_КбПЗ_2023рік

```
            if ((s2==' ')|(s2=='\0'))      dec_temp = s1_i;
            else
                if ((s3==' ')|(s3=='\0'))      dec_temp = s1_i*10+s2_i;
                else
                    if ((s4==' ')|(s4=='\0'))      dec_temp =
s1_i*100+s2_i*10+s3_i;
                    else      dec_temp = s1_i*1000+s2_i*100+s3_i*10+s4_i;
    if (dec_temp > 255)
    {
        ShowMessage("Значення перевищило 255! Воно буде замінене на 0.");
        dec_temp = 0;
    }
    return (byte)dec_temp;
}
//-------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл hex_dec_oct.h – бібліотека для файлу hex_dec_oct.cpp

```
//---------------------------------------------------------------
#ifndef hex_dec_octH
#define hex_dec_octH
//---------------------------------------------------------------
#endif
byte switch_case (byte, char); //визначення системи ліку числа
byte hex (char, char); //переведення рядка в 16-ве число
byte oct (char, char, char); //переведення рядка в 8-ве число
byte dec (char, char, char, char); //переведення рядка в 10-ве //число
```

Кафедра_КбПЗ_2023рік

Додаткові функції програми

Файл Resursi.cpp – відображення наявних на комп'ютері LPT-портів

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Resursi.h"
#include "WinIO.h"
//---------------------------------------------------------------------------

#pragma package(smart_init)
#pragma resource "*.dfm"
TResurses *Resurses;

//---------------------------------------------------------------------------
__fastcall TResurses::TResurses(TComponent* Owner)
        : TForm(Owner)
{
}

//---------------------------------------------------------------------------
void __fastcall TResurses::Button1Click(TObject *Sender)
{
        Close();
}

//---------------------------------------------------------------------------
//створення форми вікна
void __fastcall TResurses::FormCreate(TObject *Sender)
{
    DWORD dwMemVal;
    byte count_port=0;
    GetPhysLong((PBYTE)0x0408, &dwMemVal);
    WORD lpt1, lpt2, lpt3;
    lpt1 = (WORD)dwMemVal;
    if (lpt1 > 0)
    {
        Resurses->Label_lpt1->Caption = "0x" + IntToHex(lpt1, 4);
        count_port++;
    }
    else
        Resurses->Label_lpt1->Caption = "Немає доступу";
    GetPhysLong((PBYTE)0x040a, &dwMemVal);
    lpt2 = (WORD)dwMemVal;
    if (lpt2 > 0)
    {
        Resurses->Label_lpt2->Caption = "0x" + IntToHex(lpt2, 4);
        count_port++;
    }
    else
        Resurses->Label_lpt2->Caption = "Немає доступу";
    GetPhysLong((PBYTE)0x040c, &dwMemVal);
    lpt3 = (WORD)dwMemVal;
    if (lpt3 > 0)
    {
        Resurses->Label_lpt3->Caption = "0x" + IntToHex(lpt3, 4);
        count_port++;
    }
    else
        Resurses->Label_lpt3->Caption = "Немає доступу";
    Label7->Caption = AnsiString(count_port);
}
//---------------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Кафедра_КбПЗ_2023рік

Файл Resursi.h – бібліотека для файлу Resursi.cpp

```
//---------------------------------------------------------------------------

#ifndef ResursiH
#define ResursiH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TResurses : public TForm
{
__published:      // IDE компоненти керування
        TPanel *Panel1;
        TLabel *Label1;
        TButton *Button1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label_lpt1;
        TLabel *Label_lpt2;
        TLabel *Label_lpt3;
    TLabel *Label7;
    TLabel *Label8;
        void __fastcall Button1Click(TObject *Sender);
    void __fastcall FormCreate(TObject *Sender);
private:    // визначається користувачем
public:           // визначається користувачем
        __fastcall TResurses(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TResurses *Resurses;
//---------------------------------------------------------------------------
                                    #endif
```

Кафедра_КОПЗ_2023рік

Файл About.cpp – відображення вікна "Про програму..."

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "About.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TAboutBox *AboutBox;
//---------------------------------------------------------------------------
__fastcall TAboutBox::TAboutBox(TComponent* AOwner)
      : TForm(AOwner)
{
}
//---------------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл About.h – бібліотека для файлу About.cpp

```
//---------------------------------------------------------------------------
#ifndef AboutH
#define AboutH
//---------------------------------------------------------------------------
#include <vcl\System.hpp>
#include <vcl\Windows.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Classes.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TAboutBox : public TForm
{
__published:
      TPanel *Panel1;
      TLabel *ProductName;
      TLabel *Copyright;
      TButton *OKButton;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
private:
public:
      virtual __fastcall TAboutBox(TComponent* AOwner);
};
//---------------------------------------------------------------------------
extern PACKAGE TAboutBox *AboutBox;
//---------------------------------------------------------------------------
#endif
```

Кафедра_КБПЗ_2023рік

Файл Help.cpp – відображення вікна Довідки про LPT-порт

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Help.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button1Click(TObject *Sender)
{
Form2->Close();
}
//---------------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл Help.h – бібліотека для файлу Help.cpp

```
#ifndef HelpH
#define HelpH
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:      // Компонети управління IDE
        TImage *Image1;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TLabel *Label13;
        TLabel *Label14;
        TLabel *Label15;
        TLabel *Label16;
        TLabel *Label19;
        TLabel *Label20;
        TLabel *Label21;
        TLabel *Label22;
        TLabel *Label23;
        TLabel *Label24;
        TLabel *Label25;
        TLabel *Label26;
        TLabel *Label27;
        TLabel *Label28;
        TLabel *Label29;
        TLabel *Label30;
        TLabel *Label31;
        TLabel *Label17;
        TLabel *Label18;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:         // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік