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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи керування комунальних теплових мереж”

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

Керівник проекту
кандидат технічних наук, доцент
_____ Сергій СМІРНОВ
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

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

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

Теслі Сергію Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи керування комунальних теплових мереж*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи керування комунальних теплових мереж*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Тесля С.В. Дослідження та програмна реалізація системи керування комунальних теплових мереж. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи керування комунальних теплових мереж.

Метою розробки є дослідження та програмна реалізація системи керування комунальних теплових мереж.

Об'єктом дослідження є процес керування комунальних теплових мереж.

Предметом дослідження є методи керування комунальних теплових мереж.

Методи дослідження базуються на методах теорії автоматизованого управління , методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи керування комунальних теплових мереж.

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

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

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

Програму розроблено в середовищі Delphi 10.4 Sydney.

Ключові слова: комп'ютерна інженерія, автоматизоване керування

ABSTRACT

Teslia S.V. Research and software implementation of the control system of communal heating networks. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of higher education the software which is intended for system of management of municipal thermal networks is developed.

The purpose of development is research and software implementation of the control system of municipal heating networks.

The object of research is the process of management of communal heating networks.

The subject of research is the methods of management of municipal heating networks.

Research methods are based on the methods of the theory of automated control, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the control system of communal heating networks.

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

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

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

The program is developed in the environment of Delphi 10.4 Sydney.

Keywords: computer engineering, automated control

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АТМ – автоматична телефонна мережа

БДрП – багатошарові друковані плати

ДП – диспечерський пункт

ДДрП – двосторонні друковані плати

ДрП – друковані плати

ЕОМ – електронно–обчислювальна машина

ЕРЕ – електрорадіоелементи

ЗВІС – зверхвеликі інтегральні схеми

ЗП – запам'ятовувальний пристрій

ІМС – інтегральні мікросхеми

КП – контрольований пункт

ЛЕП – високовольтні лінії електропередач

МТМ – міська телефонна мережа

ОДрП – однобічні друковані плати

РЕМ – розподільні електричні мережі

ТММ – телеметрична мережа

УКХ – ультракороткі хвилі

ЦП – центральний процесор

RISC – Reduced Instruction Set Computers

ВСТУП

Актуальність теми. Розвиток сучасної обчислювальної техніки, електроніки дозволяє її використовувати для завдань керування комунальних теплових мереж [1-5]. Використання таких систем покликано поліпшити якість, ефективність тих або інших виробничих цілей.

Існує кілька наукових напрямків, в основі яких лежить об'єднання обчислювальної техніки й електроніки з технологічними процесами, радіоапаратурою [3-8]. Одним з напрямків використання мікропроцесорної техніки, є її застосування для реалізації контролю параметрів різних комунальних мереж та управління ними. В даному проекті за основу взята тепломережа.

Системи теплопостачання будуються переважно по 3-східчастій схемі, частинами якої є:

1. Джерела тепла різних типів, з'єднані між собою в єдину закільцьовану систему

2. Центральні теплові пункти (центральні теплові пункти), приєднані до магістральних теплових мереж з високою температурою теплоносія (130...150°С). У центральні теплові пункти температура плавно знижується до максимальної температури 110 °С, виходячи з потреб індивідуальні теплові пункти. У малих систем рівень центральних теплових пунктів може бути відсутнім.

3. Індивідуальні теплові пункти, що одержують теплову енергію від центральні теплові пункти і які забезпечують теплопостачання об'єкта.

Принциповою особливістю розв'язків є те, що вся система заснована на принципі 2-х трубного розведення, яке є кращим техніко-економічним компромісом. Такий розв'язок дозволяє знизити втрати тепла й споживання електроенергії в порівнянні із широко розповсюдженими в Україні 4-х трубної або 1-але трубної з відкритим водоразбором системами, інвестиції в модернізацію яких без зміни їх структури не ефективні. Витрати на обслуговування таких систем постійно збільшуються. Тим часом, саме економічний ефект є основним

критерієм доцільності розвитку й технічного вдосконалювання системи. Очевидно, що при спорудженні нових систем слід ухвалювати апробовані на практиці оптимальні розв'язки. Якщо ж мова йде про капітальний ремонт системи теплопостачання неоптимальної структури, економічно вигідно переходити до 2-х трубній системі з індивідуальними тепловими пунктами в кожному будинку.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи керування комунальних теплових мереж.

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

– Огляд існуючих систем керування комунальних теплових мереж.

– Дослідження системи керування комунальних теплових мереж.

– Програмна реалізація системи керування комунальних теплових мереж.

Об'єктом дослідження є процес керування комунальних теплових мереж.

Предметом дослідження є методи керування комунальних теплових мереж.

Методи дослідження базуються на методах теорії автоматизованого управління , методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод керування комунальних теплових мереж.

– Розроблено вітчизняний продукт керування комунальних теплових мереж, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі керування комунальних теплових мереж.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи керування комунальних теплових мереж, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

При забезпеченні споживачів теплом і гарячою водою компанія, що управляє, має постійні витрати, структура яких виглядає в такий спосіб:

– витрати на виробіток тепла для споживання;

– втрати в джерелах тепла внаслідок недосконалості способів вироблення тепла;

– втрати тепла в теплових магістралях;

– витрати на електроенергію.

Кожна із цих складових може бути знижена при оптимальному керуванні й застосуванні сучасних засобів автоматизації на кожному рівні.

Джерела тепла

Відомо, що для систем теплопостачання кращими є більші джерела комбінованого виробітку тепла й електроенергії або такі джерела, у яких тепло є вторинним продуктом, наприклад, продуктом промислових процесів. Саме на основі таких принципів виникла ідея центрального теплопостачання. У якості резервних джерел тепла використовуються котельні, що працюють на різних видах палива, газові турбіни та інше. Якщо газові котельні служать основним джерелом тепла, вони повинні працювати з автоматичною оптимізацією процесу горіння. Тільки так можна одержати економію й знизити викиди в порівнянні з розподіленим виробітком тепла в кожному будинку.

Насосні станції

Тепло із джерел тепла передається в магістральні теплові мережі. Теплоносій перекачується мережними насосами, які працюють безупинно. Тому добору й способу експлуатації насосів повинне приділятися особлива увага. Режим роботи насоса залежить від режимів теплових пунктів. Зниження витрати

на центральні теплові пункти спричиняє небажане збільшення напору насоса (насосів). Збільшення напору негативно впливає на все компоненти системи. У найкращому разі збільшується тільки гідравлічний шум. У кожному разі губиться електрична енергія. У цих умовах безумовний економічний ефект забезпечується при частотному керуванні насосами. Використовуються різні алгоритми керування. У базовій схемі контролер підтримує постійний перепад тиску на насосі шляхом зміни частоти обертання. У зв'язку з тим, що зі зменшенням витрати теплоносія знижуються втрати тиску в трасах (квадратична залежність), можна знизити також задане значення (уставку) перепаду тиску. Таке керування насосами називається пропорційним і дозволяє додатково знизити витрати на роботу насоса. Більш ефективне управління насосами з корекцією завдання по “вилученій точці”. У цьому випадку виміряється перепад тиску в кінцевих точках магістральних мереж. Поточні значення перепаду тиску компенсують тиску на насосній станції.

1.2 Область застосування

Центральні теплові пункти

У сучасних системах теплопостачання центральні теплові пункти відіграють дуже важливу роль. Енергозберігаюча система теплопостачання повинна працювати із застосуванням індивідуальних теплових пунктів. Однак це не виходить, що центральні теплові пункти будуть закриватися: вони виконують функцію гідравлічного стабілізатора й одночасно розділяють систему теплопостачання на окремі підсистеми. із центральних теплових пунктів у випадку застосування індивідуальні теплові пункти виключаються системи центрального гарячого водопостачання. При цьому через центральні теплові пункти проходять тільки 2 труби, розділені теплообмінником, який відокремлює систему магістральних трас від системи індивідуальні теплові пункти. Таким чином, система індивідуальних теплових пунктів може працювати з іншими

температурами теплоносія, а також з меншими динамічними тисками. Це гарантує стабільну роботу індивідуальні теплові пункти й одночасно спричиняє скорочення інвестицій на індивідуальні теплові пункти. Температура подачі із центральні теплові пункти коректується відповідно до температурного графіка по температурі зовнішнього повітря з урахуванням літнього обмеження, яке залежить від потреби системи ГВС в індивідуальні теплові пункти. Мова йде про попереднє коректування параметрів теплоносія, що дозволяє знизити втрати тепла у вторинних трасах, а також побільшати термін служби компонентів теплової автоматики в індивідуальні теплові пункти.

Індивідуальні теплові пункти

Робота індивідуальні теплові пункти впливає на економічність усієї системи теплопостачання. індивідуальні теплові пункти – стратегічно важлива частина системи теплопостачання. Перехід від 4-х трубної системи до сучасної 2-х трубної сполучений з певними труднощами. По-перше, це спричиняє необхідність інвестицій, по-друге, без наявності певного “ноу-хау” впровадження індивідуальні теплові пункти може навпаки побільшати поточні витрати керуючої компанії. Принцип роботи індивідуальні теплові пункти полягає в тому, що тепловий пункт перебуває безпосередньо в будинку, який опалюється й для якого готується гаряча вода. При цьому до будинку підключене тільки 3 труби: 2 для теплоносія й 1 для холодного водопостачання. Таким чином, спрощується структура трубопроводів системи, і при плановому ремонті трас відразу має місце економія на прокладці труб.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи керування комунальних теплових мереж, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Керування контуром опалення

Контролер індивідуальні теплові пункти управляє тепловою потужністю системи опалення, змінюючи температуру теплоносія. Уставка температури опалення визначається по температурі зовнішнього повітря й кривій опалення (погодозалежне керування). Крива опалення визначається з урахуванням інерційності будинку.

Інерційність будинку

Інерційність будинків значно впливає на результат погодозалежного керування опаленням. Сучасний контролер індивідуальні теплові пункти повинен ураховувати цей фактор, що впливає. Інерційність будинку визначається значенням постійної часу будинку, який перебуває в діапазоні від 10 годин у панельних будинків до 35 годин у цегельних будинків. Контролер індивідуальні теплові пункти визначає на підставі постійної часу будинку так звану "комбіновану" температуру зовнішнього повітря, яка й використовується в якості коригувального сигналу в автоматичній системі регулювання температури води на опалення.

Сила вітру

Вітер суттєво впливає на температуру приміщення особливо у висотних будинках, розташованих на відкритих територіях. Алгоритм корекції температури води на опалення, що враховує вплив вітру, забезпечує до 10% економії теплової енергії.

Обмеження температури зворотної води

Усі описані вище види керування побічно впливають на зниження температури зворотної води. Ця температура є головним показником економічної роботи системи теплопостачання. При різних режимах роботи індивідуальні теплові пункти температура зворотної води може бути знижена за допомогою функцій обмеження. Однак усі функції обмеження спричиняють відхилення від комфортних умов, і їх застосування повинне мати техніко-економічне обґрунтування. У незалежних схемах підключення контуру опалення при економічній роботі теплообмінника різниця температур зворотної води первинного контуру й контуру опалення не повинна перевищувати 5°С. Економічність забезпечується функцією динамічного обмеження температури зворотної води (DRT – differential of return temperature): при перевищенні заданого значення різниці температур зворотної води первинного контуру й контуру опалення контролер знижує витрата теплоносія в первинному контурі. При цьому знижується й пікове навантаження.

1. Розподіл теплового навантаження споживачів теплової енергії в системі теплопостачання між джерелами теплової енергії, що поставляють теплову енергію в даній системі теплопостачання, здійснюється органом, уповноваженим відповідно до справжнього закону на твердження схеми теплопостачання, шляхом внесення щорічно змін у схему теплопостачання.

2. Для розподілу теплового навантаження споживачів теплової енергії всі теплопостачальні організації, що володіють джерелами теплової енергії в даній системі теплопостачання, зобов'язано представити в орган, уповноважений відповідно до справжнього закону на твердження схеми теплопостачання, заявку.

1) про кількість теплової енергії, яку теплопостачальна організація зобов'язується поставляти споживачам і теплопостачальним організаціям у даній системі теплопостачання;

2) про обсяг потужності джерел теплової енергії, яку теплопостачальна організація зобов'язується підтримувати;

3) про чинні тарифи в сфері теплопостачання й прогнозних питомих змінних витратах на виробництво теплової енергії, теплоносія й підтримка потужності.

3. У схемі теплопостачання повинні бути певні умови, при наявності яких існує можливість поставок теплової енергії споживачам від різних джерел теплової енергії при збереженні надійності теплопостачання. При наявності таких умов розподіл теплового навантаження між джерелами теплової енергії здійснюється на конкурсній основі відповідно до критерію мінімальних питомих змінних витрат на виробництво теплової енергії джерелами теплової енергії, обумовленими в порядку, установленому основами ціноутворення в сфері теплопостачання, затвердженими Урядом України, на підставі заявок організацій, що володіють джерелами теплової енергії, і нормативів, що враховуються при регулюванні тарифів в області теплопостачання на відповідний період регулювання.

4. Якщо теплопостачальна організація не згодна з розподілом теплового навантаження, здійсненим у схемі теплопостачання, вона має право оскаржити розв'язок про такий розподіл, прийняте органом, уповноваженим відповідно до справжнього закону на твердження схеми теплопостачання, в уповноважений Урядом України орган виконавчої влади.

5. Теплопостачальні організації й тепломережеві організації, що здійснюють свою діяльність в одній системі теплопостачання, щорічно до початку опалювального періоду зобов'язано містити між собою угода про керування системою теплопостачання відповідно до правил організації теплопостачання, затвердженими Урядом України.

6. Предметом зазначеної в частині 5 справжньої статті угоди є порядок взаємних дій по забезпеченню функціонування системи теплопостачання відповідно до вимог справжнього закону. Обов'язковими умовами зазначеної угоди є:

1) визначення співпідпорядкованості диспетчерських служб теплопостачальних організацій і тепломережевих організацій, порядок їх взаємодії;

2) порядок організації налагодження теплових мереж і регулювання роботи системи теплопостачання;

3) порядок забезпечення доступу сторін угоди або, за взаємною домовленістю сторін угоди, іншої організації до теплових мереж для здійснення налагодження теплових мереж і регулювання роботи системи теплопостачання;

4) порядок взаємодії теплопостачальних організацій і тепломережевих організацій у надзвичайних ситуаціях і аварійних ситуаціях.

7. У випадку, якщо теплопостачальні організації й тепломережеві організації не уклали зазначене в справжній роботі угода, порядок керування системою теплопостачання визначається угодою, укладеною на попередній опалювальний період, а якщо така угода не полягала раніше, зазначений порядок установлюється органом, уповноваженим відповідно до справжнього закону на твердження схеми теплопостачання.

Датчики ОВЕН ПД 100

Датчики ОВЕН ПД 100 моделі 311, 371, 381 являють собою перетворювачі надлишкового тиску з керамічною вимірювальною мембраною, сенсором на основі технології ТНК і кабельним уведенням стандарту EN175301-803 (DIN43650 A).

Дана модель характеризується найбільш бюджетною ціною й стійкістю до агресивних середовищ.

Перетворювачі даної моделі призначені для систем регулювання й керування на об'єктах житлово-комунального господарства: прямих і зворотних трубопроводах мережної води систем ГВС/ХВС, теплолічильниках, станціях підкачування води й т.п., де не потрібна висока точність вимірів.

Основні характеристики перетворювача тиску для ЖКГ

– вимір надлишкового тиску нейтральних до кераміки AL2 ПРО3 і нержавіючу сталь AISI 304S середовищ (гази, пара, вода, слабоагресивні рідини).

– перетворення тиску в уніфікований сигнал постійного струму 4...20 мА.

– верхня межа вимірюваного тиску (ВМВТ) – від 0,1 до 10,0 Мпа.

– перевантажувальна здатність – не менш 200% ВМВТ.

– основна наведена погрішність – 1,0; 0,5 % ВМВТ.

– ступінь захисту корпуса й электрорознімання перетворювача – IP65.

– завадостійкість задовольняють вимогам до встаткування класу А за ДСТ 30804.6.2-2013.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion,

навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи керування комунальних теплових мереж.

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

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

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Постановка завдання й необхідні функції системи

У роботі визначені функції створюваної системи керування комунальних теплових мереж. Були обрані основні й допоміжні засоби автоматизації й передачі даних. Зроблена розробка SCADA-системи для забезпечення працездатності системи в цілому.

Необхідні й достатні функції системи:

1. Інформаційні функції:

– Вимір і контроль технологічних параметрів.

– Сигналізація й реєстрація відхилень параметрів від установлених границь.

– Формування й видача оперативних даних персоналу.

– Архівування й перегляд історії параметрів.

2. Керуючі функції:

– Автоматичне регулювання важливих параметрів процесу.

– Дистанційне керування периферійними пристроями (насосами).

– Технологічні захисти й блокування.

3. Сервісні функції:

– Самодіагностика програмно-технічного комплексу в реальному часі.

– Передача даних на диспетчерський пункт за розкладом, по запиту й по виникненню позаштатної ситуації.

– Тестування працездатності й правильності функціонування обчислювальних пристроїв і каналів уведення/виводу.

Вибір основних засобів автоматизації відбувався в основному по трьом факторам – це ціна, надійність і універсальність настроювання й програмування.

Так, для самостійної роботи в центральні теплові пункти й для передачі даних були обрані вільно-програмувальні контролери серії PCD2-PCD3 фірми Saia-Burgess. Для створення диспетчерського місця була обрана розроблена SCADA-Система керування комунальних теплових мереж. Для передачі даних ухвалене рішення використовувати звичайний стільниковий зв'язок: використовувати звичайний голосовий канал для передачі даних і SMS-повідомлення для оперативного повідомлення персоналу про виникнення позаштатних ситуацій.

Як і в багатьох подібних системах, управлінські функції для безпосереднього впливу на регулюючі механізми віддаються на нижній рівень, а вже керування всією системою в цілому – на верхній. Опис роботи нижнього рівня (контролерів) і процесу передачі даних я свідомо опускаю й перейду відразу на опис верхнього.

Для зручності використання диспетчерське місце оснащене персональним комп'ютером (ПК) із двома моніторами. Дані із усіх пунктів стікаються на диспетчерський контролер і через інтерфейс RS-232 передаються в OPC-сервер, що працює на ПК. Проект реалізований у керування комунальних теплових мереж версії 6 і розрахований на 2048 каналів. Це перший етап впровадження описуваної системи.

Особливістю реалізації поставленого завдання в керування комунальних теплових мереж є спроба створення багатовіконого інтерфейсу з можливістю спостереження за процесом теплопостачання в режимі on-line, як на схемі міста Кропивницький, так і на мнемосхемах теплових пунктів. Використання багатовіконого інтерфейсу дозволяє розв'язати проблеми виводу великої кількості інформації на дисплей диспетчера, яка повинна бути достатня й у той же час ненадлишкова. Принцип багатовіконого інтерфейсу дозволяє мати доступ до будь-яких параметрів процесу відповідно до ієрархічної структури вікон. А також спрощується впровадження системи на об'єкті, тому що такий інтерфейс по зовнішньому вигляду досить схожий на широко розповсюджені продукти

сімейства Microsoft і має схоже встаткування меню й панелей інструментів, знайомих будь-якому користувачеві персонального комп'ютера.

На головному екрані системи схематично відображена магістральна тепломережа із вказівкою джерела тепла (ТЕЦ) і центральних теплових пунктів (з першого по сьомий). На екран виведена інформація про виникнення позаштатних ситуацій на об'єктах зовнішня температура, що тече, повітря, дата й час останньої передачі даних з кожного пункту. Об'єкти теплопостачання постачені спливаючими підказками. При виникненні позаштатної ситуації – об'єкт на схемі починає «мигати», і з'являються запис про подію й червоний миготливий індикатор у звіті тривог поруч із датою й часом передачі даних. Є можливість перегляду укрупнених теплових параметрів по центральні теплові пункти й по всій тепломережі в цілому. Для цього необхідно відключити показ списку звіту тривог і попереджень (кнопка «ОТіП»).

Перехід на мнемосхему теплового пункту можливий двома способами – необхідно клацнути мишкою по значкові на схемі міста Кропивницький або по кнопці з написом теплового пункту.

Мнемосхема теплового пункту відкривається на другому екрані. Це зроблене як для зручності спостереження за конкретною ситуацією на центральні теплові пункти, так і для спостереження за загальним станом системи. На цих екранах у режимі реального часу візуалізуються всі контрольовані й регульовані параметри, у тому числі й параметри, які зчитуються з теплолічильников. Усе технологічне встаткування й засобу виміри постачені спливаючими підказками відповідно до технічної документації.

Зображення встаткування й засобів автоматизації на мнемосхемі максимально наближене до реального виду.

На наступному рівні багатовіконого інтерфейсу здійснюється безпосереднє керування процесом теплопередачі, зміна настроювань, перегляд характеристик працюючого встаткування, спостереження за параметрами в реальному часі з історією змін.

На екрані керування контролером є можливість змінити телефонні номери для передачі SMS-повідомлень, заборонити або дозволити передачу аварійних і інформаційних повідомлень, управляти періодичністю й величиною передачі даних, задавати параметри для самодіагностики засобів виміру. На екрані теплорозраховника можна переглядати всі налаштуванні параметри, змінювати доступні настроювання й управляти режимом обміну даними з контролером.

Спливаючі панелі для керуючого встаткування (регулювальний клапан і насосні групи) відображають поточний стан цього встаткування, відомості про помилки й деякі параметри, необхідні для самодіагностики й перевірки. Так, для насосів дуже важливими параметрами є тиск сухого ходу, час наробітку на відмову й затримка для включення.

Екрани для спостереження за параметрами й регулюючими контурами в графічному виді з можливістю перегляду історії зміни, куди виведені всі контрольовані параметри теплового пункту. Вони згруповані по фізичному змісту (температура, тиск, витрата, кількість тепла, теплова потужність, висвітлення). На екран регулюючих контурів виведені всі контури керування параметрами й відображається поточне значення параметра, задане з урахуванням зони нечутливості, положення клапана й обраний закон регулювання. Усі ці дані на екранах розбиті на сторінки, подібно загальноприйнятому оформленню в Windows-Додатках.

Усі екрани можна переміщати по просторі двох моніторів, одночасно виконуючи кілька завдань. У режимі реального часу доступні всі необхідні параметри для безаварійної роботи системи розподілу тепла.

Переваги й недоліки

У даній роботі автор не ставить завдання оцінити економічний ефект від впровадження системи керування в цифрах. Однак економія очевидна через скорочення персоналу, зайнятого в обслуговуванні системи, значного зменшення кількості аварій. Крім того, очевидний екологічний ефект. А також слід зазначити, що впровадження такої системи дозволяє оперативно реагувати й

усувати ситуації, які можуть привести до непередбачених наслідків. Строк окупності всього комплексу робіт (будівництво теплотраси й теплових пунктів, монтаж і налагодження, автоматизація й диспетчеризація) для замовника складе 5-6 років.

Можна привести переваги працюючої системи керування:

– Наочність вистави інформації на графічному зображенні об'єкта;

– Що стосується анімаційних елементів, то вони спеціальним образом додавалися в проект для поліпшення візуального ефекту від перегляду програми.

Перспективи розвитку системи

Особливостями теплопостачання є твердий взаємовплив режимів теплопостачання й теплоспоживання, а також множинність точок поставки декількох товарів (теплова енергія, потужність, теплоносій, гаряча вода). Ціль теплопостачання, не забезпечення генерації й транспорту, а підтримка якості названих товарів для кожного споживача.

Ця мета досягалася відносно ефективно при стабільних витратах теплоносія у всіх елементах системи. Застосовуване в нас “якісне” регулювання по самій своїй суті має на увазі зміна тільки температури теплоносія. Поява будинків з регульованим споживанням забезпечило непередбачуваність гідравлічних режимів у мережах при збереженні сталості витрат у самих будинках. Скарги в сусідніх будинках довелося ліквідувати завищеною циркуляцією й відповідними масовими перетопами.

Застосовувані сьогодні гідравлічні розрахункові моделі, не дивлячись на їхнє періодичне калібрування, не можуть забезпечити облік відхилень витрат на введеннях будинків через зміну внутрішніх тепловиділень і споживання гарячої води, а також впливу сонця, вітру й дощу. При фактичному якісно-кількісному регулюванні, необхідно “бачити” систему в реальному часі й забезпечити:

– контроль максимальної кількості точок поставки;

– відомість поточних балансів відпустки, втрат і споживання;

– керуючий вплив при неприпустимому порушенні режимів.

Керування повинне бути максимально автоматизованим, інакше його просто неможливо реалізувати. Завдання полягало в тому, щоб добитися цього без надмірних витрат на встаткування контрольних точок.

Сьогодні, коли у великій кількості будинків є вимірювальні системи з витратомірами, датчиками температури й тиску, використовувати їх тільки для фінансових розрахунків нерозумно. Система керування комунальних теплових мереж побудована, в основному, на узагальненні й аналізі інформації « від споживача».

При створенні системи керування комунальних теплових мереж були переборені типові проблеми застарілих систем:

– залежність від коректність обчислення приладу облік й вірогідності даних у архівах;

– неможливість відомості оперативних балансів через нестиковки часу вимірів;

– неможливість контролю процесів що швидко змінюються;

– невідповідність новим вимогам інформаційної безпеки.

Ефекти від впровадження системи

Служби по роботі зі споживачами:

– визначення реальних балансів по всіх видах товарів і комерційних втрат:

– визначення можливих забалансових доходів;

– контроль фактичного споживання потужності й відповідності її ТУ на підключення;

– уведення обмежень відповідних до рівня платежів;

– перехід на двоставочний тариф;

– контроль КПЕ для всіх служб, що працюють зі споживачами, і оцінка якості їх роботи.

Експлуатація:

– визначення технологічних втрат і балансів у теплових мережах;

– диспетчерське й аварійне керування по фактичних режимах;

– підтримка оптимальних температурних графіків;

– контроль стану мереж;

– налагодження режимів теплопостачання;

– контроль відключень і порушень режимів.

Розвиток і інвестиції:

– достовірна оцінка результатів впровадження проектів поліпшень;

– оцінка ефектів інвестиційних витрат;

– розробка схем теплопостачання в реальних електронних моделях;

– оптимізація діаметрів і конфігурації мережі;

– зниження витрат на підключення при обліку реальних резервів пропускної здатності й енергозбереження в споживачів;

– планування ремонтів

– організація спільної роботи ТЕЦ і котелень.

Впровадження системи керування комунальних теплових мереж опалення, вентиляції, гарячого водопостачання є основним підходом до економії теплової енергії. Установка систем автоматичного регулювання в індивідуальних теплових пунктах знижує споживання тепла в житловому секторі на 5-10%, а в адміністративних приміщеннях на 40%. Найбільший ефект виходить за рахунок оптимального регулювання у весняно-осінній період опалювального сезону, коли автоматика центральних теплових пунктів практично не виконує повною мірою свої функціональні можливості.

Регулювання теплового режиму будинку

Керування тепловим режимом зводиться до підтримки його на заданому рівні або зміні відповідно до заданого закону.

На теплових пунктах проводиться регулювання в основному дві видів теплового навантаження: гарячого водопостачання й опалення.

Для обох видів теплового навантаження системи керування комунальних теплових мереж повинна підтримувати незмінними задані значення температури води гарячого водопостачання й повітря в опалювальних приміщеннях.

Відмінною рисою регулювання опалення є його більша теплова інерційність, тоді як інерційність системи гарячого водопостачання значно менше. Тому завдання стабілізації температури повітря в опалювальному приміщенні значно складніше, чим завдання стабілізації температури гарячої води в системі гарячого водопостачання.

Основними впливами, що обурюють, є зовнішні метеоумови: температура зовнішнього повітря, вітер, сонячна радіація.

Існують наступні принципово можливі схеми регулювання:

– регулювання за відхиленнями внутрішньої температури приміщень від заданої шляхом впливу на витрату води, що надходить у систему опалення;

– регулювання залежно від збурювання зовнішніх параметрів, що приводять до відхилення внутрішньої температури від заданої;

– регулювання залежно від змін зовнішньої температури й усередині приміщення (за збуренням й за відхиленнями).

Внутрішні, що обурюють впливи на тепловий режим будинку незначні.

Для методу регулювання за збуренням в якості сигналів, що дозволяють відслідковувати зовнішню температуру, можуть бути обрані:

– температура води, що надходить у систему опалення;

– кількість теплоти, що надходить у систему опалення:

– витрата теплоносія.

Система керування комунальних теплових мереж повинна враховувати наступні режими роботи системи централізованого теплопостачання, при яких:

– регулювання температури води на теплоджерелі не ведеться по поточній зовнішній температурі, яка є основним, що обурює фактором для внутрішньої температури. Температура мережної води на теплоджерелі визначається по температурі повітря за тривалий період з урахуванням прогнозу

й розташовуваної теплової потужності встаткування. Транспортне запізнювання, вимірюване годинником, також приводить до невідповідності в абонента температури мережної води поточній зовнішній температурі;

– гідравлічні режими теплових мереж вимагають обмеження максимального, а іноді й мінімального витрат мережної води на теплову підстанцію;

– навантаження гарячого водопостачання впливає на режими роботи опалювальних систем, приводячи до змінних протягом доби температур води в системі опалення або витратам мережної води на систему опалення залежно від виду системи теплопостачання, схеми приєднання підігрівників гарячого водопостачання й схеми опалення.

Система регулювання за збуренням

Для системи регулювання за збуренням характерно те, що:

– існує пристрій, що вимірює величину збурювання;

– за результатами вимірів регулятор здійснює керуючий вплив на витрату теплоносія;

– на регулятор надходить інформація про температуру усередині приміщення;

– основне збурювання – температура зовнішнього повітря, яка контролюється системи керування комунальних теплових мереж, тому збурювання буде називатися контрольованим.

Варіанти схем регулювання за збуренням при зазначених сигналах, що вище відслідковують:

– регулювання температури води, що надходить у систему опалення по поточній температурі зовнішнього повітря;

– регулювання витрати теплоти, що подаємося в систему опалення по поточній температурі зовнішнього повітря;

– регулювання витрати мережної води по температурі зовнішнього повітря.

Незалежно від способу регулювання автоматична система регулювання теплопостачання у своєму составі повинна містити наступні основні елементи:

– первинні вимірювальні пристрої – датчики температури, витрати, тиску, перепаду тиску;

– вторинні вимірювальні пристрої;

– виконавчі механізми, що містять регулювальні органі й приводи;

– мікропроцесорні регулятори;

– нагрівальні прилади (бойлери, калорифери, радіатори).

Датчики системи керування комунальних теплових мереж теплопостачання

Основні параметри теплопостачання, які за допомогою системи керування комунальних теплових мереж підтримуються відповідно до завдання, широко відомі.

У системах опалення, вентиляції й гарячого водопостачання звичайно виміряється температура, витрата, тиск, перепад тиску. У деяких системах виміряється теплове навантаження. Методи й способи виміру параметрів теплоносіїв традиційні.

Автоматичні регулятори

Автоматичний регулятор – це засіб автоматизації, що одержує, підсилює й перетворююче сигнал відключення регульованої величини, що й цілеспрямовано впливає на об'єкт регулювання.

У цей час в о сновному застосовують цифрові регулятори на базі мікропроцесорів. При цьому звичайно в одному мікропроцесорному контролері реалізуються кілька регуляторів для систем опалення, вентиляції й гарячого водопостачання.

Більшість вітчизняних і закордонних контролерів для систем теплопостачання мають однакові функціональні можливості:

– залежно від температури зовнішнього повітря регулятор забезпечує необхідну температуру теплоносія на опалення будинку за графіком, управляючи

регулювальним клапаном з електроприводом, установленим на трубопроводі тепломережі;

– автоматичне коректування опалювального графіка проводиться відповідно до потреб конкретного будинку. Для найбільшої ефективності заощадження тепла графік подачі постійно коректується з урахуванням реальних умов теплопункту, клімату, тепловтрат приміщення;

– економія теплоносія в нічний час досягається за рахунок тимчасового методу регулювання. Зміна завдання на часткове зниження теплоносія залежить від зовнішньої температури так, щоб, з однієї сторони зменшити споживання тепла, з інший, не проморозити й ранком вчасно прогріти приміщення. При цьому автоматично розраховується момент включення денного режиму опалення, або інтенсивного прогріву для досягнення потрібної температури приміщення в потрібний час;

– контролери дозволяють здійснювати забезпечення можливе низької температури, що вертається води. При цьому передбачається захист системи від заморожування;

– проводиться автоматичне коректування, задана в системі гарячого водопостачання. Коли споживання в системі гарячого водопостачання невелике, припустимі більші відхилення в температурі (збільшення зони нечутливості). При цьому шток клапана не буде мінятися занадто часто, і строк його служби протриває. При збільшенні навантаження зона нечутливості автоматично зменшується, і точність регулювання зростає;

– спрацьовує сигналізація перевищення уставок. Звичайно виробляються наступні сигнали тривоги:

– сигнал тривоги по температурі, у випадку відмінності реальної від заданої температури;

– сигнал тривоги від насоса надходить у випадку збою в роботі;

– сигнал тривоги від датчика тиску в розширювальному баку;

– сигнал тривоги по строкові експлуатації надходить, якщо встаткування відробило встановлений строк;

– сигнал загальної тривоги – якщо контролер зареєстрував один або більш сигналів тривоги;

– ведеться реєстрація параметрів регульованого об'єкта й передача його на ЕОМ.

Регулювальні органі

Виконавчий пристрій – це одне з ланок системи керування комунальних теплових мереж, призначених для безпосереднього впливу на об'єкт регулювання. У загальному випадку виконавчий пристрій складається з виконавчого механізму й регулювального органа.

В автоматичних системах регулювання теплопостачання застосовуються, в основному, електричні (електромагнітні й електродвигательні).

Регулювальний орган призначений для зміни витрати речовини або енергії в об'єкті регулювання. Розрізняють дозуючі й дросельні регулювальні органі. До дозуючих ставляться такі пристрої, які змінюють витрату речовини за рахунок зміни продуктивності агрегатів (дозатори, живильники, насоси).

Дросельні регулювальні органі являють собою змінний гідравлічний опір, що змінює витрата речовини за рахунок зміни свого прохідного перетину. До них ставляться регулювальні клапани, елеватори, повторні заслінки, крани і т.д.

Регулювальні органі характеризуються багатьма параметрами, основними з яких є: пропускна здатність K v, умовний тиск P y, перепад тиску на регулювальному органі D y, і умовний прохід Д y .

Крім наведених параметрів регулювального органа, що визначають в основному їхню конструкцію й розміри, є й інші характеристики, які враховуються при виборі регулювального органа залежно від конкретних умов їх застосування.

Найбільш важливої є пропускна характеристика, яка встановлює залежність пропускної здатності щодо переміщення затвора при постійному перепаді тиску.

Дросельні регулювальні клапана профилюються звичайно з лінійної або рівнопроцентною пропускною характеристикою.

При лінійній пропускній характеристиці збільшення пропускної здатності відбувається пропорційно збільшенню переміщення затвора.

При рівнопроцентній пропускній характеристиці збільшення пропускної здатності (при зміні переміщення затвора) іде пропорційно поточному значенню пропускної здатності.

У робочих умовах вид пропускної характеристики змінюється залежно від перепаду тиску на клапані. При пом регулювальний клапан характеризується видатковою характеристикою, яка являє собою залежність відносної витрати середовища від ступеня відкриття регулюючого органа.

Найменше значення пропускної здатності, при якому зберігається пропускна характеристика в межах установленого допуску, оцінюється як мінімальна пропускна здатність.

У багатьох випадках автоматизації виробничих процесів регулювальний орган повинен мати широкий діапазон зміни пропускної здатності, який являє собою відношення умовної пропускної здатності до мінімальної пропускної здатності.

Необхідною умовою надійної роботи автоматичної системи регулювання є правильний вибір форми пропускної характеристики регулювального клапана.

Для конкретної системи видаткова характеристика визначається значеннями параметрів середовища, що протікають через клапан, і його пропускною характеристикою. У загальному випадку видаткова характеристика відрізняється від пропускний, тому що параметри середовища (в основному тиск і перепад тисків), як правило, залежать від значення витрати. Тому завдання

вибору кращої пропускної характеристики регулювального клапана розбивається на два етапи:

– вибір форми видаткової характеристики, що забезпечує сталість коефіцієнта передачі регулювального клапана у всьому діапазоні навантажень;

– вибір форми пропускної характеристики, що забезпечує при даних параметрах середовища бажану форму видаткової характеристики.

При модернізації систем опалення, вентиляції й гарячого водопостачання задані розміри типової мережі, розташовуваний напір і первісний тиск середовища, регулювальний орган вибирають так, щоб при мінімальній витраті через клапан втрата в ньому відповідала надлишковому тиску середовища, що налаштовується джерелом, а форма видаткової характеристики була близька до заданої. Метод гідравлічного розрахунків при виборі регулювального клапана досить трудомісткий.

3.2 Розробка структурної схеми

В процесі практичної реалізації теоретичних принципів розробки системи, розглянутих вище, була розроблена структурна схема (рисунок 3.1) в якій були розглянуті аспекти побудови системи керування тепломережею.

Розглядаючи її можна побачити що система розбита на дві основних частини апаратна частина (винесена на плакат «Функціональна схема апаратної частини») та програмна частина (винесена на плакат «Функціональна схема програмної частини»), взаємозв’язок котрих відбувається за допомогою інтерфейсу RS-232 та інтерфейсу Windows.

Вхідні дані подають поточне значення спостерігаємих об`єктів (тепломереж). Спостерігання відбувається за допомогою датчиків тиску температури, вологості та інших датчиків.

Головним модулем апаратної частини являється мікроконтролер AT90S1200, якій керує вхідними потоками і подає дані на інтерфейс RS-232.

Головним модулем у програмної частини є розроблене у магістерській роботі програмне забезпечення.

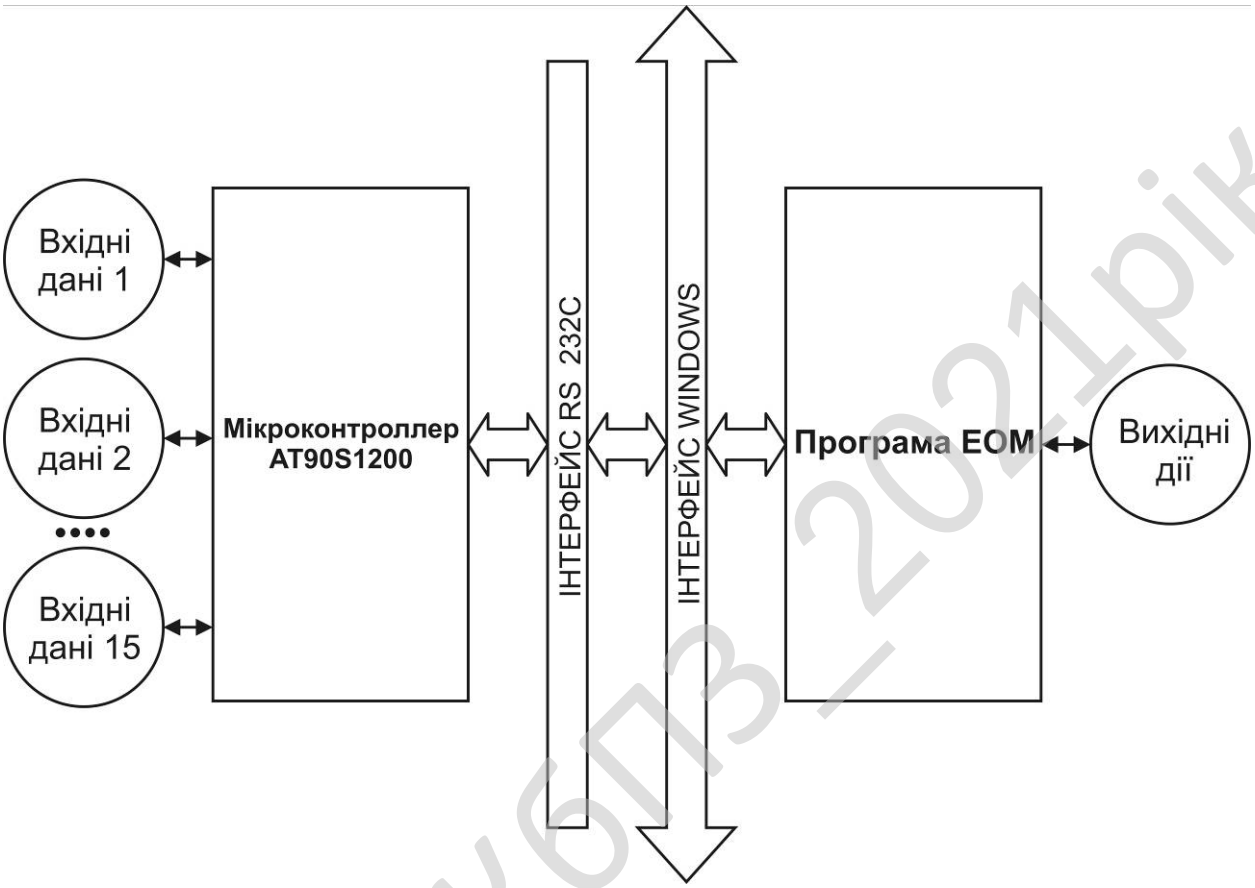

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема апаратної частини де детально розглядається частина системи отримання даних. Виміри які виконуються на тепломережі за допомогою датчиків з інтерфейсами (в залежності від типа датчика інтерфейси різні) подаються на мікроконтролер AT90S1200 який обробляє отримані дані утворюючи чергу подання даних на інтерфейс RS-232.

Розглянемо детально використовуємі датчики.

Датчики виміру температури. Температура – найбільш важливий показник теплової насосної станції. Відповідно до технічного завдання система повинна забезпечувати вимір температури теплоносія в трубопроводі, що подає, а також у зворотному трубопроводі. Крім того, контроль перегріву підшипників насосів і електродвигунів також доцільно здійснювати шляхом виміру температури. Найпоширеніші термопари, термоперетворювачи опору, напівпровідникові терморезистори, кремнієві (у тому числі й інтегральні) термодатчики. Для виміру температури теплоносія доцільно застосувати термоперетворювачі TD4A, TD5A які забезпечують не тільки контроль температури на поверхні, а також забезпечити контроль температури в рідкому середовищі . В зв'язку з тим, що передбач ається, що в насосній станції не буде обслуговуючого персоналу, то з метою підвищення надійності апаратно-програмного комплексу доцільний постійний контроль температури повітря в насосній. Для цього теж може бути обрано TD4A або TD5A.

При застосуванні будь-якого термодатчика необхідно в комплекті з ним застосовувати проміжний перетворювач (інтерфейс), призначений для перетворення сигналу термодатчика в уніфікований сигнал постійного струму 0-5 ма або напругу 10В. Принцип дії перетворювача заснований на статичній автокомпенсації. Сигнал від термометра надходить на вимірювальний міст і далі на вхідний підсилювач, виконаний за схемою модулятор-демодулятор.

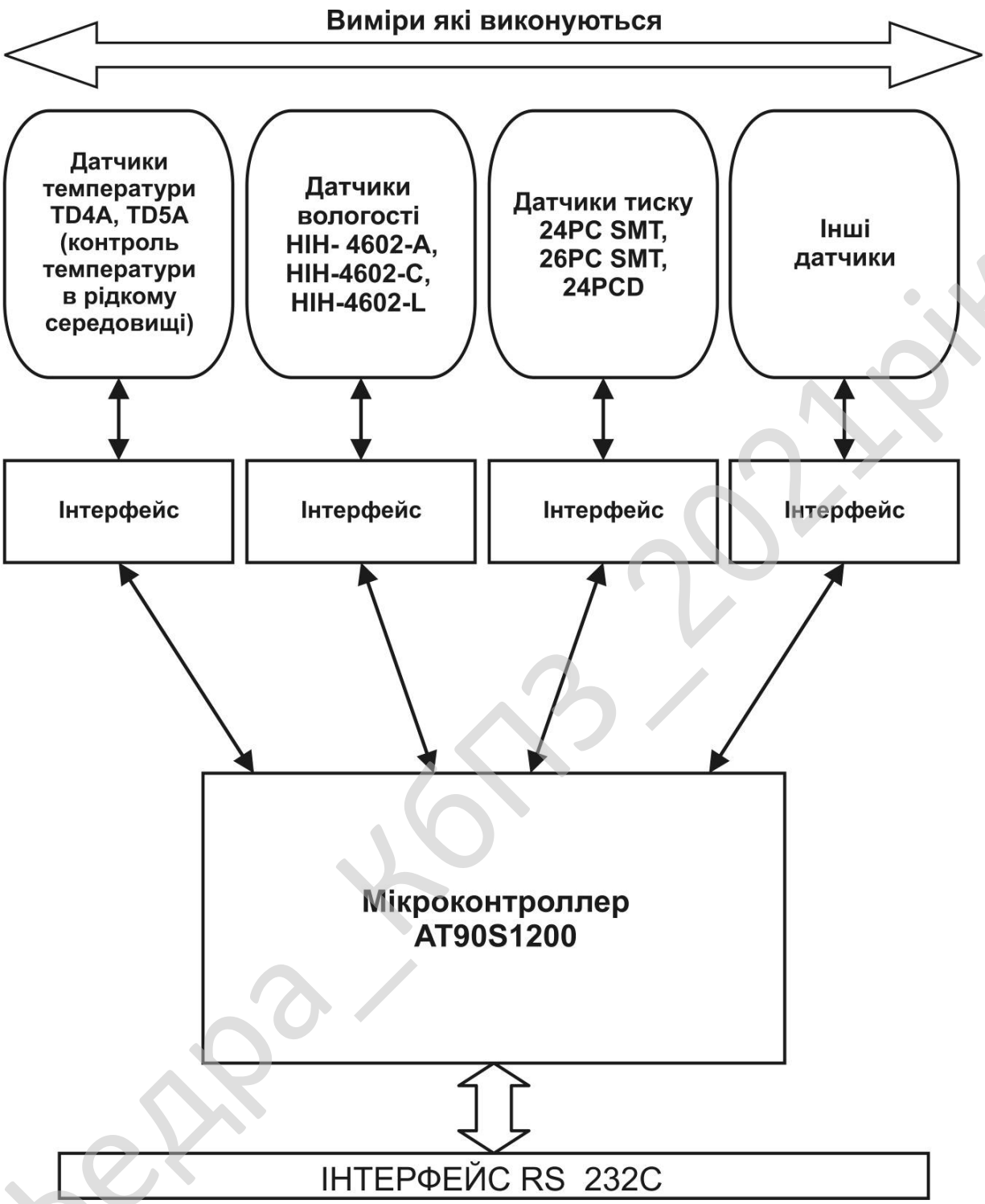

Рисунок 3.2 – Функціональна схема системи

Демодульованний сигнал підсилюється вихідним підсилювачем постійного струму, вихідний сигнал якого надходить на навантаження й пристрій зворотного зв'язка. Вхідні й вихідні ланцюги не мають гальванічного зв'язку з ланцюгами живлення й між собою. Всі типи перетворювачів є одноканальними, тобто для кожного термометра повинен використовуватися свій перетворювач.

Датчики для виміру тиску. Тиск – параметр, що характеризує протікання процесів на ТНС. При виборі датчиків тиски керуються вимогою перетворення величини тиску в уніфікований вихідний сигнал. Існує кілька різних типів датчиків:

– датчики тиску з мембранами (прогини мембрани перетворяться в зміни опору резистора або в зміну індуктивності обмоток вихідного перетворювача);

– датчики тиску з мембранами й п’єзоелементами (виникнення електричних зарядів на робочих гранях п’єзоелемента при додатку до нього тиску);

– датчики тиску з мембранами й тензометричними перетворювачами (тиск, прикладений до мембрани, перетвориться в зміну опору тензоелемента);

– ємнісні датчики тиску (тиск, прикладений до мембрани, перетвориться в зміну опору тензоелемента);

– датчики тиску з манометричними трубчастими пружинами.

Порівняльний аналіз датчиків тиску з різними принципами дії показав, що найбільше доцільно в телемеханічній системі застосувати датчики 24PC SMT, 26PC SMT, 24PCD, принцип дії яких заснован на прогині металевої мембрани (чутливий елемент), що спочатку перетвориться в зміну опору потенціометра, а потім останнє – у струм на виході датчика.

Датчик вологості. Забезпечує контроль приміщень тепломереж. Використовуються датчики HIH- 4602-A, HIH-4602-C, HIH-4602-L в залежності від типу приміщення. Також могуть бути застосовані інші менш важливі датчики що забезпечують охорону пре мішень.

Датчик пожежної сигналізації. У цей час використовуються системи автоматичного виявлення пожежі по трьох факторах: теплу, диму, полум'ю. Найпоширеніші теплові пожежні оповіщувачі наступних типів:

– максимальної дії, що спрацьовують при перевищенні температурою розрахункової величини;

– максимально-диференціальні, об'єднуючі властивості оповіщувачів максимального й диференціального типів;

– диференціальні, реагуючі на швидке підвищення температури.

Всі існуючі теплові оповіщувачі виявляють пожежу, коли вона досягає значних розмірів. Час виявлення пожежі дозволяє знизити використання пожежних оповіщувачів, що формують сигнал пожежної тривоги з появою пульсації температури конвективного потоку над вогнищем пожежі. Такий оповіщувач відповідає наступним вимогам: реагує на змінну складову коливань температури в визначеному частотному діапазоні, не видає сигналів тривоги при впливі факторів, що заважають, створюваною роботою встаткування. Для підвищення надійності системи пожежної сигналізації в телемеханічній системі встановлені додаткові датчики диму.

Датчики охоронної сигналізації. Повинні забезпечувати недоторканність пункту обліку теплової енергії. Можливе застосування наступних систем охорони: шлейфового типу, на базі інфрачервоних світлових передавачів і приймачів, на базі радіохвиль, на базі ультразвуку. Найбільш проста й дешева система шлейфового типу. У ній використовуються замикаючі або електричні контакти, що розмикаються, тобто електричне коло замикається або розмикається механічним способом. Шлейф з'єднується з перетворювачем охоронної сигналізація. При обриві шлейфа на виході перетворювача охоронної сигналізація з'являється сигнал тривоги, що надходить у передавальну апаратуру ТММ для передачі сигналу на диспетчерський пункт. До складу системи охоронної сигналізації введений вимикач входу-виходу, що приводить до затримки на кілька секунд у дії системи й що дозволяє входити й виходити з охоронюваного

об'єкта, не викликаючи сигналу тривоги. Сигнал тривоги, що надійшов на диспетчерський пункт із охоронюваного об'єкта, може скасувати тільки прибулий на теплову насосну станцію обслуговуючий персонал.

Датчики для сигналізації затоплення напрямка мережних труб. Можливе застосування двох варіантів датчиків: поплавкового й реле рівня. У цей час існують поплавкові датчики заводського виготовлення, наприклад, датчик рівня РО-1. Можливе настроювання на чотири значення рівня води. Реле рівня засновано на замиканні контакту при зіткненні з рідиною. Існують наступних видів таких реле: РІС -101-011, РІС-101011И, РІС-101021, РМ-51. Для контролю за даним параметром ефективним буде використання датчика з контактними електродами (реле рівня), тому що він простий, дешевий і надійний.

3.4 Розробка діаграми процесів

Важливим критерієм при розробці програмного забезпечення це є грамотна розробка структури роботи системи. На діаграмі процесів зображеній на рисунку 3.3 можна точно зрозуміти як працює і взаємодіє розроблене програмне забезпечення, в цілому.

Починається і закінчується програма в першому блоці це є основною точкою відрахунку діаграми. При переміщенні по стрілках можна побачити загальну схему взаємодії блоків і їх входження один в одного.

Як можна побачити з основного блоку програми управляння переходить спочатку до модулю захисту та потім до головного вікна програми з якого можно потрапити у вікно налагодження інтерфейсу ПЗ та параметрів виведення інформації. Далі через головне вікно та обробки отриманої інформації через інтерфейс RS-232C відбувається зв'язок з вхідними даними (датчиками).

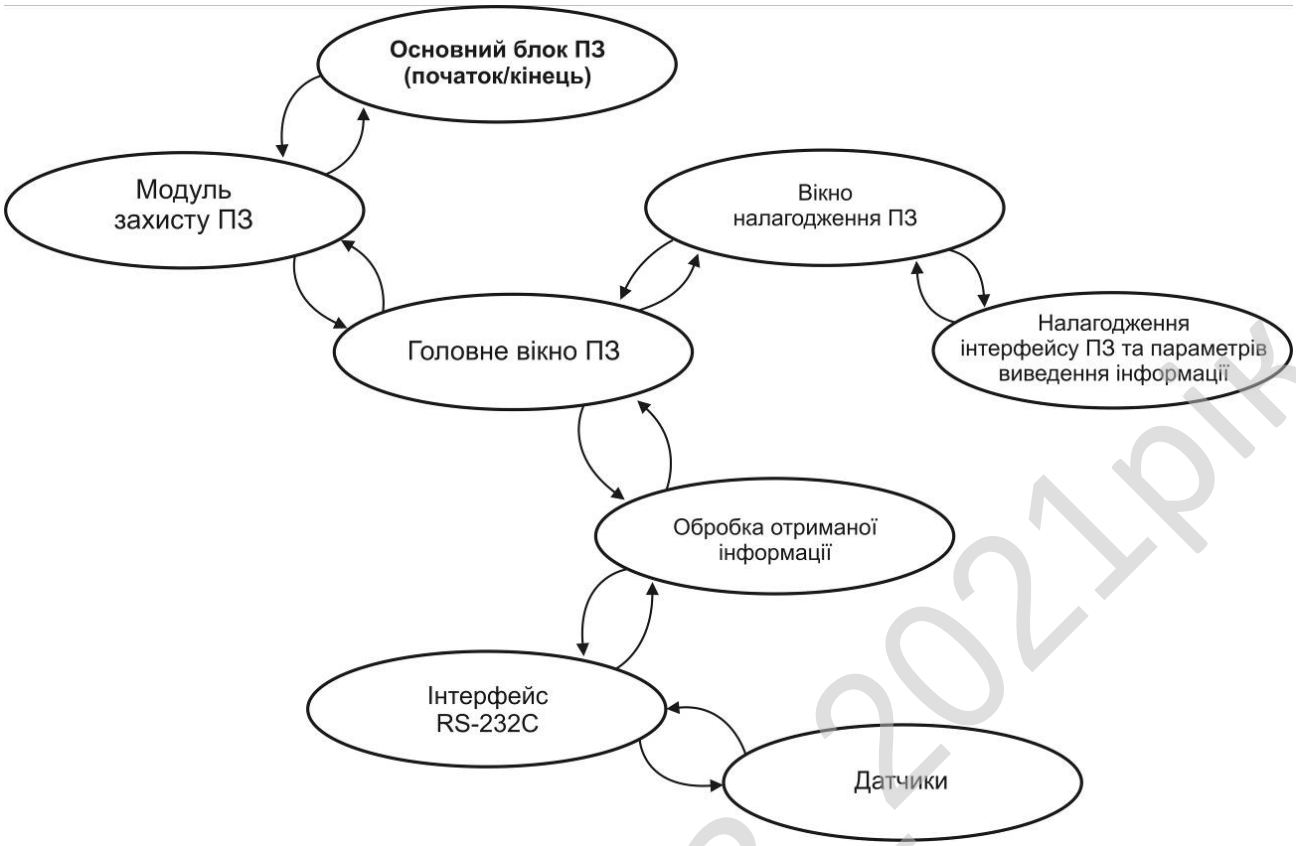

Рисунок 3.3 – Діаграма процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунках 4.1, 4.2, 4.3 зображена розроблена блок-схема програми. Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно є розробка блок-схем.

З основного алгоритму програми винесені дві основних підпрограми це підпрограма налагодження параметрів (рисунок 4.1) та блок-схема підпрограми отримання та обробки даних (рисунок 4.2).

Виділимо у розробленій блок-схемі основні блоки.

Початок роботи та перевірки:

– Початкова ініціалізація змінних та виділення пам`яті.

– Підключення бібліотек взаємодії з RS-232.

– Підключення додаткових модулів.

– Ініціалізація та спроба доступу до RS-232.

– Встановлення початкових значень програмного забезпечення.

– Виведення головного вікна ПЗ на екран.

Робота програми:

– Очікування дій користувача.

– Підпрограма налагодження параметрів виведення даних.

– Підпрограма отримання та обробки даних.

Завершення роботи:

– Виведення вікна повідомлення про завершення роботи ПЗ.

– Приховання головного вікна ПЗ на екран.

– Перехід в режим завершення роботи ПЗ.

– Відключення додаткових модулів.

– Звільнення ресурсів програми.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів взаємодії я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові процедури та функції що забезпечили вирішення проблем . Розглянемо ряд розроблених процедур та функцій:

– procedure RunInConsoleMode – Перевірка введених параметрів у ПЗ;

– function GetFileStream: TFileStream – Отримання потоку данних;

– function GetModifyTime: Longint – Отримання модифікованого часу;

– function GetMethodValue – Отримання значень;

– procedure Initialize – Початкова ініціалізація;

– procedure AddrEdtChange(Sender: TObject) – Процедура виникає при зміні мережної адреси пристрою;

– procedure SetInputState(const Value: integer) – Процедура виникає при підборі візуального стану програми;

– procedure CommPortDriver1ReceiveData(Sender: TObject; DataPtr: Pointer; DataSize: Integer) – Процедура виникає при відповіді пристрою;

– procedure SetOutput(const Value: String) – Дана процедура виникає, коли ми намагаємося послати команду пристрою;

– function GetInput: String – Повертає тіло пакета без мережної адреси й контрольної суми;

– function CompareCheckSum(AStr: String; CS: Char): boolean – Функція порівняння контрольної суми з отриманими даними;

– function CheckSum(AStr: String): Char – Функція відповідальна за підрахунок контрольної суми.

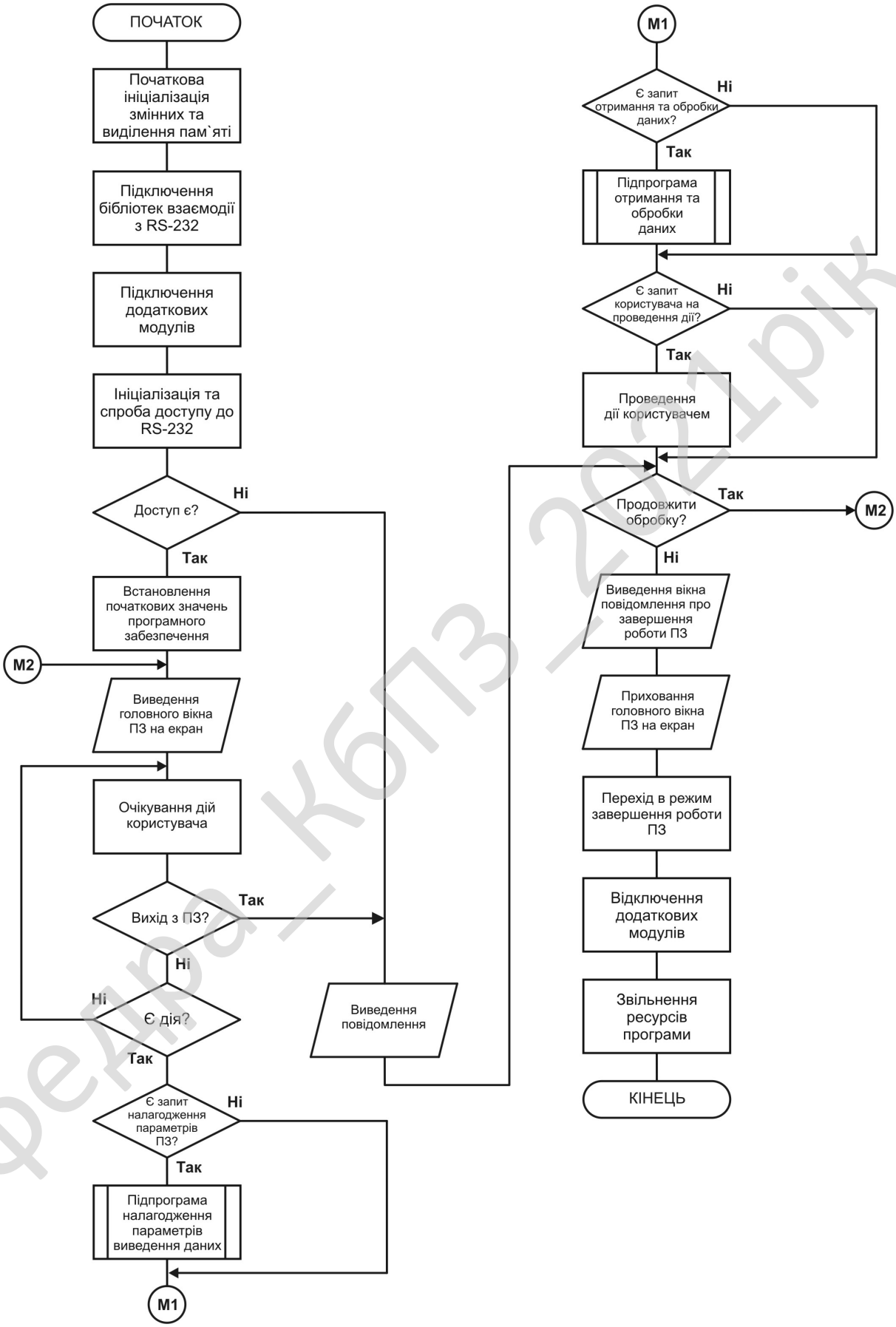

Рисунок 4.1 – Блок схема основної частини ПЗ

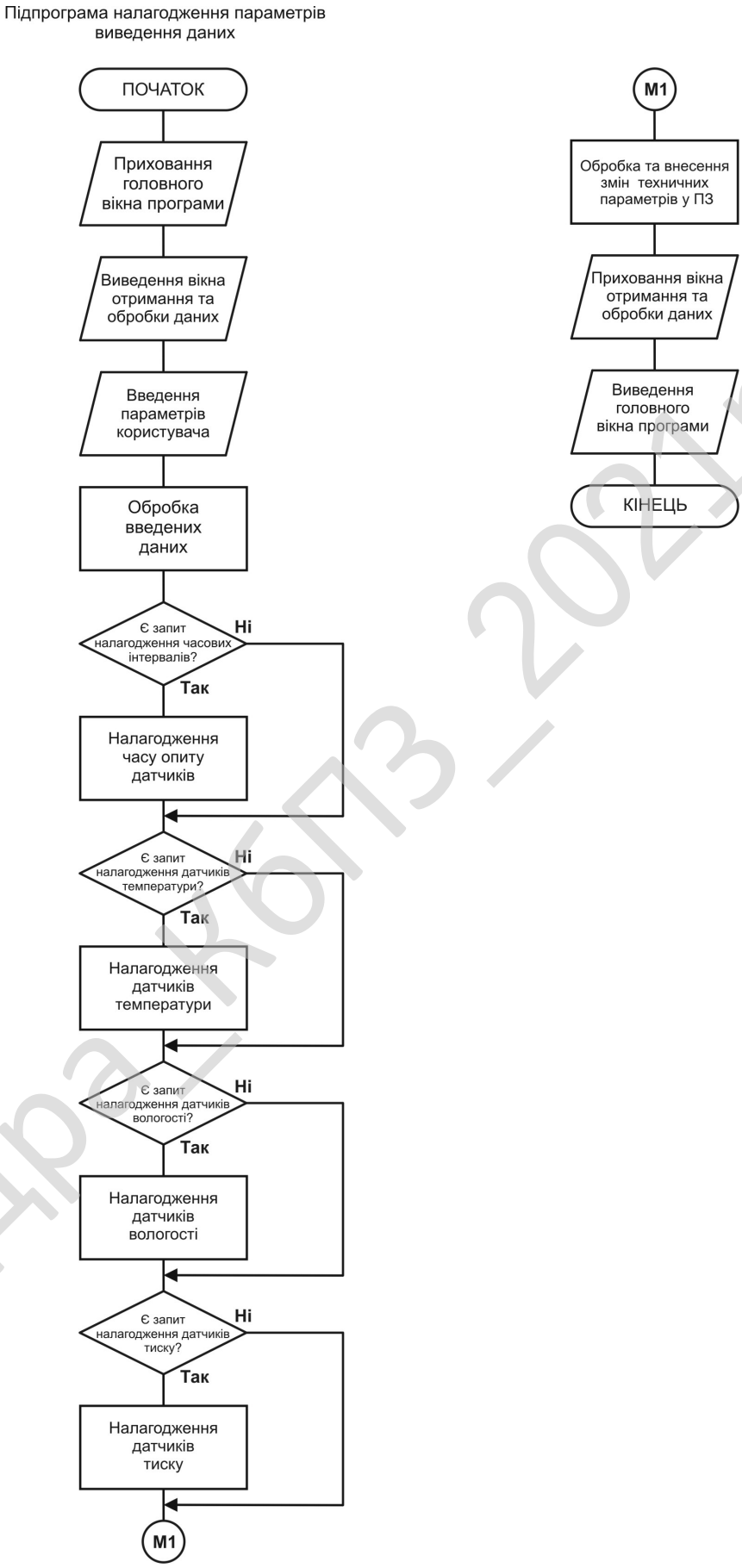

Рисунок 4.2 – Блок-схема підпрограми налагодження параметрів

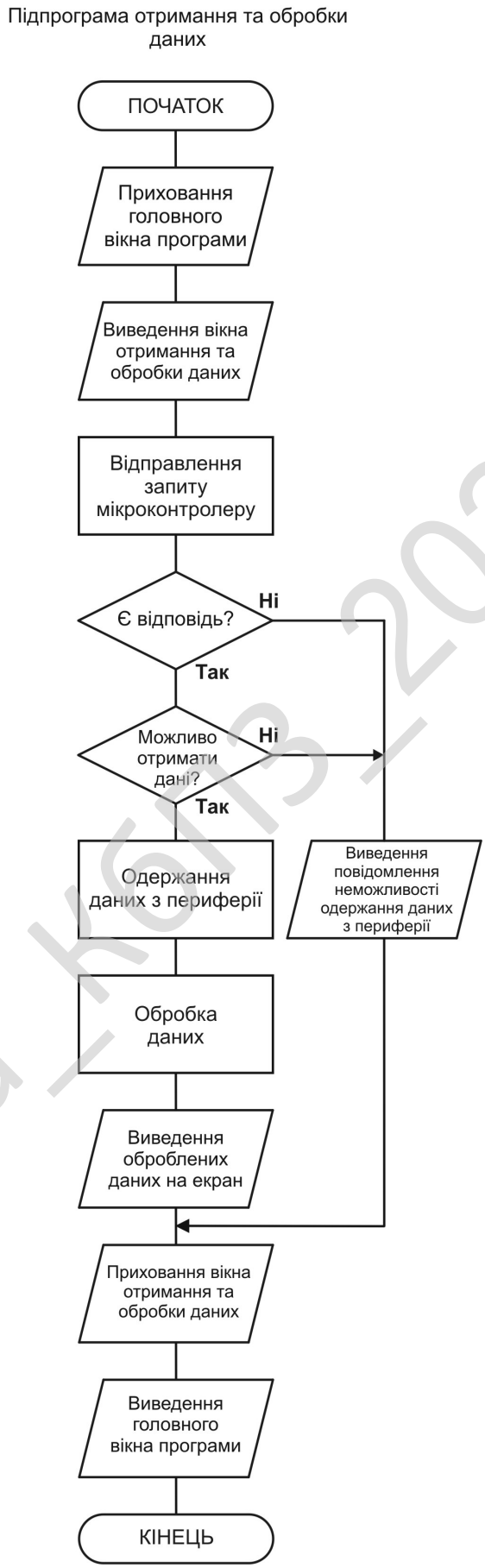

Рисунок 4.3 – Блок-схема підпрограми отримання та обробки даних

Розробка алгоритмів обробки даних та взаємодії з інтерфейсом RS - 232C

Розроблене ПЗ здійснює прийом інформаційних байтів з пункту обліку теплової енергії (вхідні дані) і їхній аналіз. По результаті аналізу дані в зручному для користувача виді виводяться на екран монітора. Дані також можуть бути збережені на ЕОМ, у якій зберігаються абсолютно всі параметри вимірів. Програма в залежності від настройок проводить ручний (статичне) чі автоматичний контроль системи. Програма дозволяє робити логічні й математичні операції над даними, що надійшли. Блок моніторингу дозволяє побачити в реальному часі дані, що надходять, і миттєві їхні значення.

Розглянемо розроблений код взаємодії з RS-232C та проведення спостереження за об'єктом.

У перше поле (Edit) вводимо мережеву адресу теплолічильника. За умовчанням він дорівнює 0. За допомогою другого ми надсилаємо команди теплолічильнику. Третє поле (Edit) служить висновку інформації, яку теплолічильник посилає.

```
//Визначаємо символьний масив  PXXXX=^ TTESLIA;
type    TTESLIA=array[1..255] of Char;
//Функція відповідальна за підрахунок контрольної суми
function TForm1.CheckSum(AStr: String): Char;
//Уводимо свої цілочислені змінні
var crc,i: Integer;
begin
crc:=0;
for i := 1 to Length(AStr) do
crc:=crc+Ord(AStr[i]);
crc:=(crc and $3F) + $30;
Result:=Chr(crc);
end;

//Функція порівняння контрольної суми з отриманими
//даними
function TForm1.CompareCheckSum(AStr: String; CS: Char): boolean;
begin
Result:=CheckSum(AStr)=CS;
```

```
end;
//Повертає тіло пакета без мережної адреси й
//контрольної суми
function TForm1.GetInput: String;
var l:integer;
begin
Result:='';
l:=Length(FInput);
if InputState = 1 then
begin
 if StartTime+3000 < GetTickCount then InputState := 2;
 Exit;
end;
if l<3 then Exit;
if CompareCheckSum(Copy(FInput,1,l-2),Copy(FInput,l-1,1)[1])=true  then
begin
InputState := 0;
NetNumber:=FInput[1];
AddrEdt.Text:=NetNumber;
Result:=copy(FInput,2,l-3);
end
else
InputState := 3;
end;
//Дана процедура виникає, коли ми намагаємося
//послати команду пристрою
procedure TForm1.SetOutput(const Value: String);
var XXXX:TXXXX;
    S:String;
    L,i:Integer;
begin
S:=NetNumber+Value;
S:=S+CheckSum(S)+#13;
L:=Length(S);
if L>255 then Exit;
for i:=1 to L do XXXX[i] := S[i];
InputState := 1;
FInput:='';
CommPortDriver1.SendData(@XXXX,L);
StartTime:=GetTickCount;
end;
//Процедура виникає при запуску програми
```

```
        procedure TForm1.FormCreate(Sender: TObject);
        begin
        NetNumber:='0';
        CommPortDriver1.Connect;
        end;
        //Процедура виникає при виході із програми
        procedure TForm1.FormDestroy(Sender: TObject);
        begin
        CommPortDriver1.Disconnect;
        end;
        //Процедура виникає при відповіді пристрою
        procedure TForm1.CommPortDriver1ReceiveData(Sender: TObject;  DataPtr:
Pointer; DataSize: Integer);
        var PX:PXXXX; i:integer;
        begin
        InputState := 4;
        Application.ProcessMessages;
        FInput:='';
        PX:=DataPtr;
        for i := 1 to DataSize do
        begin
         FInput:=FInput+PX^[i];
        end;
        InputState := 5;
        Application.ProcessMessages;
        Edit2.Text:=Input;
        end;
        //Процедура виникає при підборі візуального
        //стану програми
        procedure TForm1.SetInputState(const Value: integer);
        begin
        FInputState := Value;
        case Value of
            0: Caption:='Дані успішно прийняті';
           1: Caption:='Чекаємо відповіді';
           2: Caption:='Таймаут';
           3: Caption:='Пакет прийнятий з помилкою';
           4: Caption:='Приймаємо відповідь';
           5: Caption:='Відповідь отримана';
        end;
        end;
        //Процедура виникає при натисканні клавіші "Відправити"
```

```
procedure TForm1.SendBtnClick(Sender: TObject);
begin
Output:=OutputEdt.Text;
SendBtn.Enabled:=False;
repeat
 Edit2.Text:=Input;
until InputState<>1;
SendBtn.Enabled:=True;
end;
//Процедура виникає при зміні мережної адреси пристрою
procedure TForm1.AddrEdtChange(Sender: TObject);
begin
NetNumber:=AddrEdt.Text[1];
end;
```

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму теорії автоматизованого управління DSA.

Відправник і одержувач електронного документа використовують при обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$. Число Y є відкритим ключем для перевірки підпису відправника. Число Y передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-). У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M , відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана. Потім одержувач обчислює значення $w = 1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$. Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$. Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Підготовка програми для AVR-мікроконтролера виконується на персональному комп'ютері й складається з наступних етапів:

– створення тексту програми;

– трансляція тексту в машинні коди й виправлення синтаксичних помилок;

– налагодження програми, тобто усунення логічних помилок;

– остаточне програмування AVR-мікроконтролера.

Кожний з етапів вимагає використання спеціальних програмних і апаратних засобів. Нижче перераховані найбільш доступні з них на сьогоднішній день. Базові програмні засоби фірми Атмел поширюються безкоштовно, у той час як апаратні засоби мають свою вартість. Слід зазначити, що крім безкоштовних програмних засобів фірми Атмел, що дозволяють програмувати тільки мовою асемблера, фірмами IAR SYSTEMS, CMX CORPORATION, KANDA SYSTEMS Ltd. і іншими розроблені засоби підтримки програмування на мовах високого рівня С и BASIC, а також операційні системи реального часу.

Розглянемо більш докладно етапи підготовки програми для AVR.

Якщо Ви працюєте в середовищі, то для виконання першого й другого етапу Вам доведеться скористатися різними засобами. Для створення тексту програми підійде будь-який текстовий редактор, що формує на виході ASCII-Файли.

Для роботи в середовищі Windows фірмою Атмел пропонується програма WAVRASM, що дозволяє виконати створення тексту програми і його трансляцію усередині однієї оболонки й забезпечує додатковий сервіс для швидкого пошуку синтаксичних помилок у тексті програми.

Результатом Вашої роботи на першому етапі є файл <ім'я_файлу>.asm, що містить текст програми (розширення ім'я файлу звичайно вказує на мову

програмування) і є вхідним для програм-трансляторів, які, у свою чергу, створюють чотири нових файли: файл лістингу (<ім'я_файлу>.lst), об'єктний файл (<ім'я_файлу>.obj), файл-прошивання FLASH-пам'яті (<ім'я_файлу>.hex), файл-прошивання EEPROM-пам'яті (<ім'я_файлу>.eep). Файл лістингу – це звіт транслятора про свою роботу. У ньому приводиться трансльована програма у вигляді вихідного тексту, кожному рядку якого зіставлені відповідні двійкові коди. Крім того, лістинг містить повідомлення про виявлені помилки.Об'єктний файл використовується надалі як вхідний для програми-відладчика AVRSTUDIO і має спеціальний формат. Файли прошивання FLASH і EEPROM блоків пам'яті призначені для роботи з будь-якими послідовними й паралельними програматорами AVR і мають стандартні формати.

Наступним етапом підготовки програми є її налагодження, що може виконуватися двома основними способами: на персональному комп'ютері за допомогою програми-симулятора або в реальній мікропроцесорній системі. Два ці способи взаємно доповнюють один одного. Програма-симулятор AVRSTUDIO відображає на екрані комп'ютера Вашу програму й стан внутрішніх регістрів AVR. Таким чином, стає можливим спостерігати зміни змінних, які відбуваються усередині мікроконтролера при виконанні тих або інших команд програми. Відзначимо, що в реальній системі за допомогою осцилографа неможливо переглянути стан внутрішніх регістрів. Використання симуляторів ефективно при налагодженні підпрограм, які виконують чисельну обробку внутрішніх данных. В той же час, налагодження підпрограм, пов'язаних з якими-небудь зовнішніми елементами, зручно виконувати безпосередньо в робочій системі. Для налагодження програми в робочій системі, крім програмних засобів, потрібні також і апаратні. Нижче наведені представлені різні варіанти побудови відладочної системи, що відрізняються своєю вартістю й можливостями.

Найбільш швидкий, не потребуючі пайки спосіб побудови мікропроцесорної системи на основі AVR – це придбання комплекту AVR STARTER KIT фірми Атмел, що містить плату DEVELOPMENT BOARD, книгу

"Development tool user's guide", дискети із програмним забезпеченням, з повною документацією на всі типи AVR і численними прикладами прикладних програм для AVR. Плата DEVELOPMENT BOARD містить панельки для підключення базових типів AVR-мікроконтролерів в DIP-корпусах; джерело живлення; послідовний програматор, вузол інтерфейсу RS-232 для зв'язку з комп'ютером по асинхронному послідовному каналі; набори з 8 светлодиодів і з 8 кнопкових перемикачів, які можна підключати до виводів портів мікроконтролера; роз'єм, через які за допомогою гнучких кабелів можна нарощувати мікропроцесорну систему. Універсальність DEVELOPMENT BOARD зручна для навчання й для макетування нових розробок. Разом з тим, для багатьох конкретних проектів може не підійти конструктивна реалізація DEVELOPMENT BOARD або надлишковим буде використання на цій платі джерела живлення, послідовного програматора й панелей під різні типи корпусів. У такому випадку виконують спеціалізовану розробку, що задовольняє вимогам конкретного завдання. Одним з найважливіших достоїнств AVR-мікроконтролерів є те, що всі його апаратні ресурси "заховані" усередині, і тому схема включення AVR дуже проста. Така простота й мініатюрність дозволяє в багатьох конкретних додатках не виносити цифрову частину на окрему плату, а поміщати AVR безпосередньо усередині аналогового блоку. Для навчання технології програмування AVR Ви можете зібрати макет схеми за 15 мінут.

Отже, якщо Ви вирішили працювати не із платою DEVELOPMENT BOARD, а зі своєю власною мікро процесорною системою, то Вам необхідно придбати ще який-небудь програмуючий пристрій. Як згадувалося вище, існує два способи програмування AVR-мікроконтролерів: послідовне програмування через SPI-інтерфейс і паралельне програмування.

Паралельне програмування, що вимагає добування AVR-мікроконтролера із системи й установки його в програматор, досить незручно на етапі налагодження програми. Для програмування Вашої мікропроцесорної системи по SPI-інтерфейсі безпосередньо в системі фірмою Атмел пропонується програма

AVRPROG і програмуючий SPI-кабель, у який убудований послідовний програматор. Крім того, функцією програмування по послідовному SPI-інтерфейсі володіє паралельний програматор FLASHER. Спосіб налагодження мікропроцесорної системи за допомогою SPI-інтерфейсу відрізняється своєю дешевиною, але, однак, має й недоліки. По-перше, щораз при внесенні змін у програму Ви перепрограмуєте FLASH-пам'ять мікроконтролера, кількість циклів перезапису якої обмежено хоч і досить більшим, але все-таки кінцевим числом. По-друге, описаний спосіб не дає можливості покрокового налагодження програми.

У зв'язку із цим, фірмою Атмел розроблені могутніші, але й більш дорогі внутрісхемні емулятори (in-circuit emulator) ICEPRO і megaICEPRO. Вони являють собою мікропроцесорні пристрої, які з однієї сторони зв'язуються з Вашою мікропроцесорною системою через панель, призначену для установки AVR-мікроконтролера, а з іншого боку – з персональним комп'ютером і працюють під керуванням уже згадуваної програми фірми Атмел AVRSTUDIO. Внутрісхемні емулятори дозволяють виконувати програму у Вашій системі в покроковому режимі й необмежене число раз вносити зміни в програму. При роботі із внутрісхемним емулятором Ви одночасно можете на екрані комп'ютера спостерігати стан внутрішніх ресурсів процесора, а на мікропроцесорній платі – реакцію системи на ті або інші команди програми.

Завершальним етапом програмування AVR-мікроконтролера є занесення до пам’яті вже налагодженої програми. Воно може бути виконане так само, як і при налагодженні програми, через SPI-інтерфейс. Однак необхідно пам'ятати, що послідовне програмування молодших моделей AVR не дозволяє змінювати FUSE-біти мікроконтролера.

Якщо в мікропроцесорній системі не передбачені SPI-інтерфейс а також при серійному виробництві для підвищення швидкості програмування великої кількості мікроконтролерів зручно використовувати програматори, які виконують паралельне програмування.

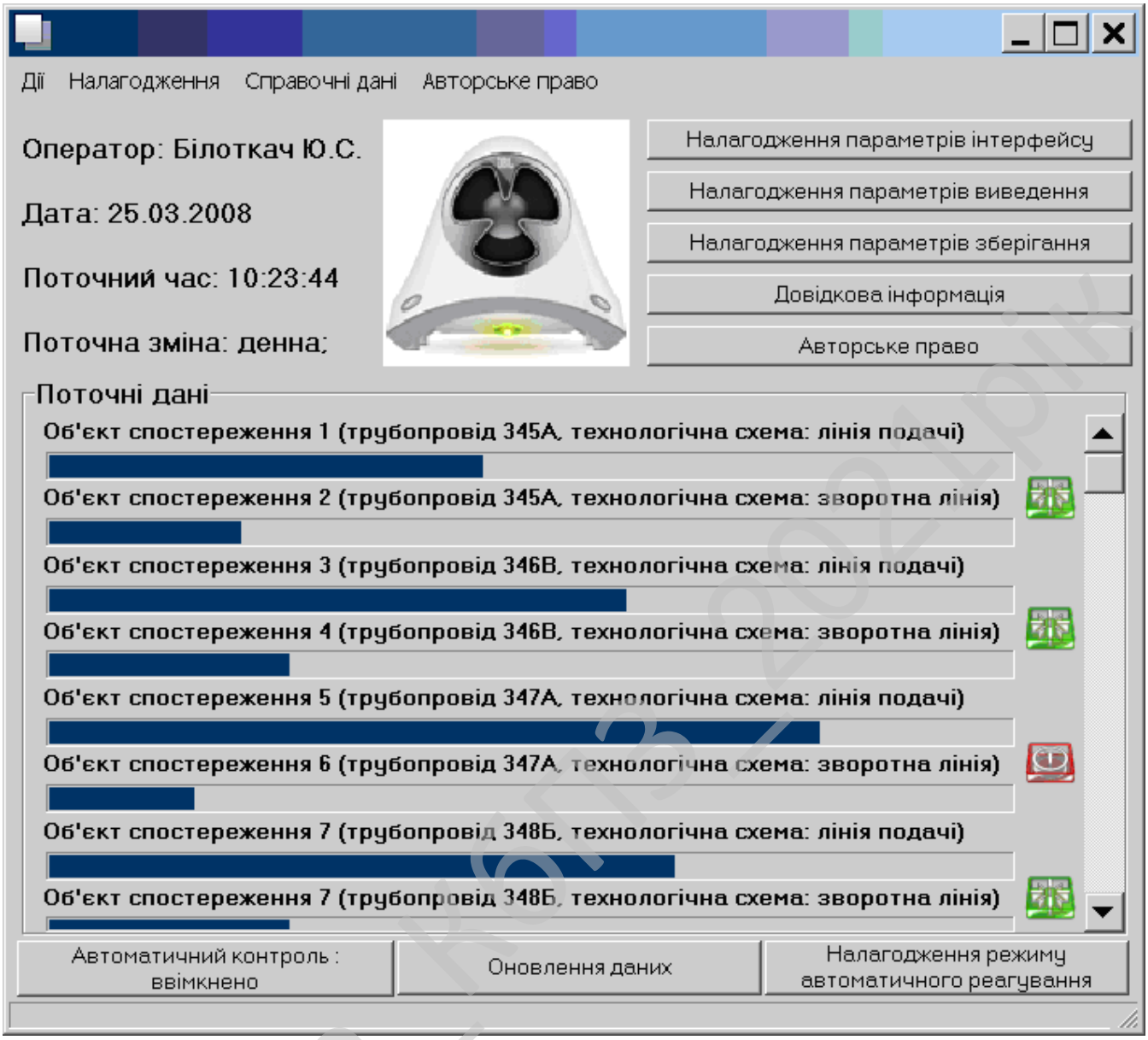

Рисунок 5.1 – Головне вікно програми

На рисунку 5.1 зображене основне вікно програми. На ньому ми бачимо, що на головне вікно виведені усі основні параметри , які потрібні для контролю та управління системою тепломережі.

При цьому існує можливість виконувати наступні дії:

– Налагодження параметрів інтерфейсу.
– Налагодження параметрів виводу.
– Налагодження параметрів зберігання.

– Видача довідкової інформації. Окремо це вікно зображене на рисунку 5.2.

– Видача даних авторського права. Окремо це вікно зображене на рисунку 5.3.

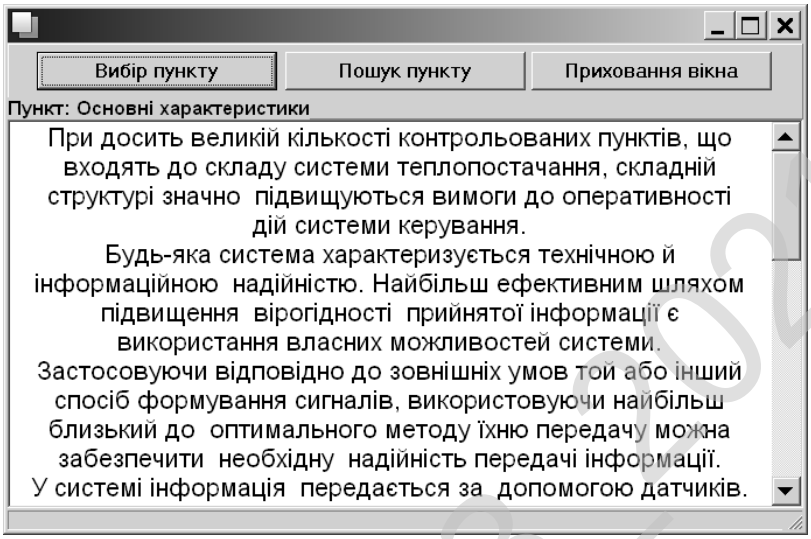

Рисунок 5.2 – Вікно додаткової інформації

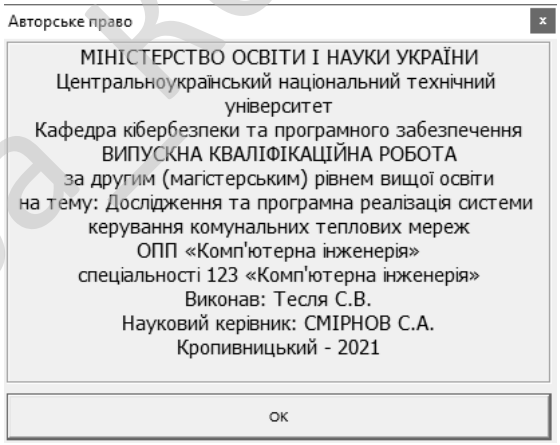

Рисунок 5.3 – Вікно авторського права

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи керування комунальних теплових мереж.

Метою розробки є дослідження та програмна реалізація системи керування комунальних теплових мереж.

Об'єктом дослідження є процес керування комунальних теплових мереж.

Предметом дослідження є методи керування комунальних теплових мереж.

Методи дослідження базуються на методах теорії автоматизованого управління , методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод керування комунальних теплових мереж.

– Розроблено вітчизняний продукт керування комунальних теплових мереж, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи керування комунальних теплових мереж.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	21 (2 ост. цифри № зал)
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	21000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

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

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

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

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

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

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

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

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

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

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

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 100 = 168 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

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

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

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

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

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

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

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

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

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

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$ = 567,5/(60·8) = 1,2 ставки.

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	13000	39000
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	1,2	11500	41400
Інженер-системотехнік	0,25	11500	8625
Адміністратор мережі	0,5	11500	17250
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 8,25	-	$Ф_{роб}$ = 297225

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{297225}{8,25 \cdot 60} = 600 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На к ожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

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

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

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

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми HARD.kiev.ua за 5.11.21 – джерело https://hard.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11186
Системний блок		6490
Процесор	Intel Xeon E5-2680 SR0KH 2.70GHz/20M LGA2011	1638
Системна плата	Intel X79 LGA2011 DDR3 USB 3.0 WiFi SL ATX	1050
Відеокарта	Видеокарта HP Quadro FX 580 450Mhz PCI E 2.0 1024Mb 1600Mhz 128bit DisplayPort	860
Жорсткий диск	SSD диск Micron RealSSD P400m 200G SAS 6G MLC 2.5 (MTFDEAK200MAS 2S1AA)	980
Оперативна пам’ять	Samsung DDR3-1333 8Gb PC3-10600R EC Registered (M393B1K70CH0-CH9Q5)	468
DVD-привод	Super Multi LG SATA DVD±RW R+22x 22x, RW+8x/-6x, DL+16x/-12x, RAM 12x SecurDisc, black (GH22NS40RBB)	432
Корпус	ASUS TA-861 500W FSP ATX-500V (Black/Silver panel) ATX (90-PL861AF5C4 53CZ)	822
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: Al Type Cards, black	240
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3200
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V (BE525-RS)	1496

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни. Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	11186	8948,8	98436,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	8948,8	98436,8
Копіюв. апарат	1	5965	540	5940
Всього	–	–	–	214803,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	214804	-	-
Всього по групі	214804	50	107402
Група 5, 6			
4. Вимірювальні пристрої	5190	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	28000	25	-
Всього по групі	33190	-	8297,5
Нематеріальні активи			
7. Нематеріальні активи	21000	10	2100
Разом	$К_р$ = 1676994		$А_р$ = 188199,5

Примітка: вартість автомобіля приймаємо рівною нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 600 \cdot 209/21 = 5971 \text{ грн.}$$

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

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

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

$$З_д = 5971 \cdot 10 \cdot 0{,}01 = 597 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

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

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

$$С_{оц} = 0{,}01 \cdot 22(5971+597) = 1445 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 5971 \cdot 15 \cdot 0{,}01 = 896 \text{ грн.}$$

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

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

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

Згідно виданих викладачем норм приймаємо 0,5 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 100 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

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

$$З_{М1} = 100 \cdot 0,5 \cdot 3 = 150 \text{ грн.}$$

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

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 12 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 12 грн./шт.

$$З_{М2} = 12 \cdot 21 + 12 = 264 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (150 + 264 + 1702)/21 = 101 \text{ грн.}$$

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

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

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

$$О_п = 5971 \cdot 15 \cdot 0,01 = 896 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 21 прим.):

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

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

$$A_м = 188200 \cdot 3/(21 \cdot 12) = 2240 \text{ грн.}$$

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

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

$$C_п = 5971+597+1445+896+101+896+2240 = 12146 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 12146 = 6073 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	5971
2. Додаткова зарплата виконавців	$З_д$	597
3. Відрахування на соціальні потреби	$C_{оц}$	1445
4. Загальногосподарські витрати	$Г_{осп}$	896
5. Витрати на матеріали	$З_М$	101
6. Освоєння нових операційних систем, мов програмування	$О_п$	896

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	2240
8. Повна собівартість програмного забезпечення	$С_п$	12146
9. Плановий прибуток	$П_р$	6073
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	18219
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3643,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	21862,8

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	21863
Всього капітальних витрат	–	21863

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	80520	40260
2. Витрати на електроенергію	$З_{ел}$	-	0
3. Витрати на амортизацію	$З_{ам}$	-	5466
Всього витрат за рік	$І$	80520	45726

Витрати на обслуговування:

$$З_р = T_р \cdot З_г \cdot (1 + 0{,}01 \cdot H_q) \cdot (1 + 0{,}01 \cdot H_c), \quad (7.23)$$

де: $T_р$ – кількість годин, що витрачається на обслуговування на рік, *год.* (приймаємо 1000 год.);

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

$$З_{р\,баз} = 1000 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 = 80520 \ грн,$$

до:

$$З_{р\,нов} = 500 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 = 40260 \ грн.$$

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

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

$$З_{ел\,баз} = З_{ел\,нов}$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації, %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	21863	–	5465,75
Всього відрахувань	-	–	21863	–	5465,75

7.8 Визначення економічної ефективності програмної продукції

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

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

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (18218 - 12146) \cdot 21 - (0,05 \cdot 1408000 + 0,5 \cdot 214804 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 21000) \cdot 3/12 = 80462 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{268994}{(18218 - 12146) \cdot 21 \cdot 12 / 3} = 0,5 \text{ років}.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (80520\text{-}45726)\text{-}0,25 \cdot 21863 = 29328 \text{ грн.}$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	21
2. Повна собівартість розробленої програми	Грн.	12146
3. Ціна розробленої програми	Грн.	18218
4. Плановий прибуток від реалізації одної програми	Грн.	6072
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1676994
7. Загальний прибуток від реалізації програмної продукції	Грн.	127512
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	80462
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	21863
11.Величина економічного ефекту у користувача програмної продукції	Грн.	29328
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Року	0,6

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{21863}{80520 - 45726} = 0,6\,року\,.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя. Охорона праці є складовою частиною безпеки життя [1].

Законом України “Про охорону праці” [2] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

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

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та

«Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

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

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

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

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

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

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	6,4
Довжина	8,4
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,7
Обсяг, V	$м^3$	не менше 20.0	20,1

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

У зазначеному приміщенні працює 8 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-58	0,11
Тепла	23-25	50-70	0,1	24-25	56-69	0,12

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер *Epson L3151*, електродвигуни вентиляторів ЕОМ.

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

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р. можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V -го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при

фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

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

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

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Початкові данні для розрахунку захисного заземлення:

Тип заземлення: робоче заземлення нульової точки трансформатора. Заземленню підлягає виробниче обладнання організації. Напруга – 220/380 В. Розташування заземлюючих електродів – по контуру.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача методом коефіцієнта використання заземлювачів.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – садова земля, нижнього шару грунта – супісок. Умовна товщина верхнього шару грунта: *H=1,1 м*. Для захисного заземлення: застосовуються вертикальні електроди – пруток довжиною *L=2 м.* Відстань між вертикальними заземлювачами (електродами) *A=2 м*. Діаметр ве ртикального електрода (прутка) *D=40 мм*, Тип горизонтального заземлювача: металева полоса. Розміри перетину з'єднуючої полоси: 80 х 6 мм. (*b=80 мм*.). Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Глибина закладення контура заземлення *t= 0,7 м*.

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр.13-16 [4], або будь якої відповідної іншої, наприклад, програми «Електрик» (автор Сумеркін Є.О.). Додаткову перевірку отриманих результатів можна робити за допомогою он-лайн сервісу URL: https://kalk.pro/electricity/earthing/.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+2/2=1,7 \text{ м.}$$

Еквівалентний питомий опір грунта [3]:

$$\rho_{екв} = \psi \rho_1 \rho_2 L / [\rho_1 \psi(L-H+t) + \rho_2 \psi(H-t)] = 153{,}54 \text{ Ом}$$

де

ψ = 1,1 табличне значення коефіцієнта сезонності у багатошаровому грунті;

ρ_1 = 150 Ом*м. – табличне значення питомого опору верхнього шару грунта;

ρ_2= 100 Ом*м. – табличне значення питомого опору нижнього шару грунта.

Опір розтіканню електричного струму одного електрода вертикального заземлювача:

$$R_O = [\rho_1/(2\pi*L)]*[(ln)2L/D)+0{,}5ln[(4T+L)/(4T-L)] = 59{,}9 \text{ Ом.}$$

Опір розтіканню електричного струму заземлювача, який нормується, якщо $\rho_{екв} > 100$ (у нас $\rho_{екв} = 153{,}54 > 100$):

$$R = R_{3H} * \rho_{екв}/100 = 4*194{,}5/100 = 6{,}1 \text{ Ом.}$$

Опір розтіканню електричного струму горизонтального заземлювача (полоси):

$$R_П = 0{,}366(\rho_{екв}\ \psi / L_П \cdot \eta_П)\lg(2/ L_П * L_П/bt) = 8{,}35 \text{ Ом.}$$

де $\eta_П$ = 0,27 – табличне значення коефіцієнта використання горизонтального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення A/L; довжина горизонтального заземлювача (полоси) при розташуванні заземлювачів по контуру:

$$L_П = A*n = 2*15 = 30 \text{ м.;}$$

де n – ітераційна кількість вертикальних заземлювачів.

Загальний опір штучного заземлюючого пристрою розтіканню електричного струму на землю [4]:

$$R_B = R_П R / (R_П - R) = 4{,}0 \text{ Ом.}$$

Кількість вертикальних заземлювачів:

$$n = R_O / R_B * \eta_B = 15 \text{ шт.}$$

де $\eta_B = 0{,}47$ – табличне значення коефіцієнта використання вертикального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення A/L.

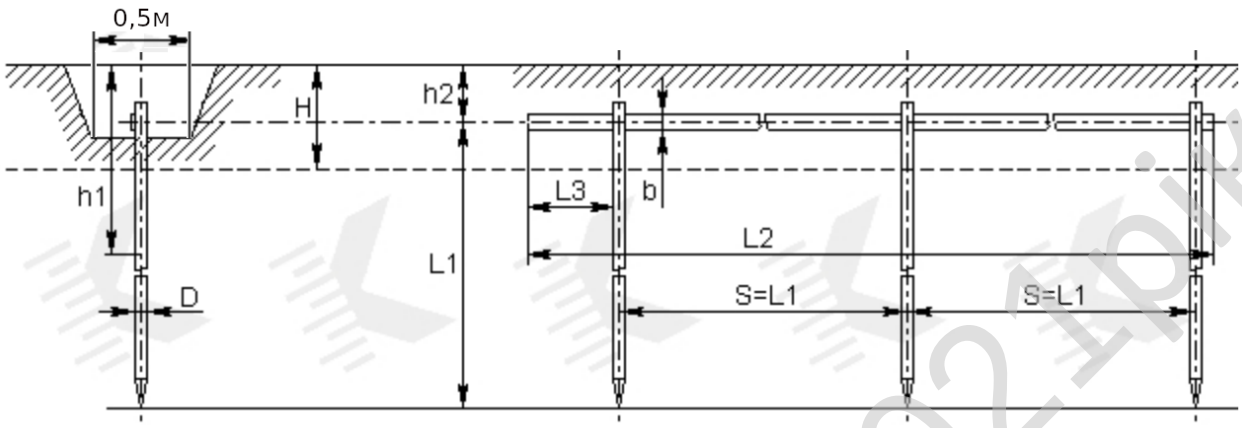

Рисунок 8.1 – Штучний заземлювач

8.6 Висновки до розділу

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

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи керування комунальних теплових мереж.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів керування комунальних теплових мереж.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем керування комунальних теплових мереж.

– Досліджена система керування комунальних теплових мереж.

– На основі отриманих результатів досліджень створена програмна реалізація системи керування комунальних теплових мереж.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання керування комунальних теплових мереж.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DSA.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 29328 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,6 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тесля С.В. Дослідження та програмна реалізація системи керування комунальних теплових мереж // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Щелкунов Н.Н., Дианов А.П. Микропроцессорные средства и системы. М., Радио и связь, 1989.-288с.

3. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М., Энергоатомиздат, 1990.-224с.

4. Майоров В.Г., Гаврилов А.И. Практический курс программирования микропроцессорных систем. М., Машиностроение, 1989.-272с.

5. Каган Б.М., Сташин В.В. Основы проектирования микропроцессорных устройств автоматики. М., Энергоатомиздат, 1987.-304с.

6. Рафикузаман М. Микропроцессоры и машинное проектирование микропроцессорных систем. В 2 кн. Кн.1. М.,Мир.,1988.-312с.

7. Уильямс Г.Б. Отладка микропроцессорных систем. М., Энергоатомиздат. 1988.-253с.

8. Фергуссон Дж., Макари Л., Уилльямз П. Обслуживание микропроцессорных систем. М., Мир, 1989.-336с.

9. Корнев В.В., Киселев А.В. Современные микропроцессоры.-М.:НОЛИДЖ,1998.-240с.: ил.

10. 15.Евстифеев А.В. Микроконтроллеры AVR семейства Classic фирмы “Atmel”.-М.:Издательский дом «Додэк-XXI» ,2002.-288с.: ил.

11. Бродин В.Б., Шагурин И.И. Микроконтроллеры. Архитектура, программирование, интерфейс.- М.: Издательство ЭКОМ, 1999.-400 с.:илл.

12. Кожанова А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / О.А. Смірнов,

А.С. Кожанова, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2013. – Вип. 6(113). – С. 255-257.

13. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.

14. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.

15. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.

16. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

17. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

18. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

19. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.

20. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.

21. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.

22. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.

23. Коваленко А.С. Удосконалення методу технічного обслуговування об'єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.

24. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 69-72.

25. Kovalenko A.S. Information model and its element for displaying information on technical condition of objects of integrated information system / A.S. Kovalenko, A.A. Smirnov, A.V. Kovalenko, A.P. Dorensky // International Journal of Computational Engineering Research (IJCER). – India: Delhi, 2016. – Volume 6, Issue 1. – P. 21-27.

26. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.

27. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем / А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.

28. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.

29. Коваленко А.С. Задачи распознавания ситуаций в системах организационной стратегии интеграции производства и операций / А.С. Коваленко, А.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVI міжнар. наук.-практ. сем., 11-12 квіт. 2014 р., м. Кіровоград: зб. тез. – Кіровоград: КНТУ, 2014. – С. 53-55.

30. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.

31. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко //

Комп’ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28 -31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.

32. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп’ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

33. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.

34. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.

35. Коваленко А.С. Метод автоматизованої перевірки результатів вимірювання параметрів об’єкті в інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Стратегія якості у промисловості і освіті: XI міжнар. конф., 1 – 5 черв. 2015 р., м. Варна, Болгарія.: зб. матер. – Варна: ТУВ, 2015. – С. 423-426.

36. Коваленко А.С. Обґрунтування необхідності створення розподіленої бази даних для забезпечення захисту рухомих повітряних об`єктів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Перспективні напрями захисту інформації: І всеукр. наук.-практ. конф., 07 вер. 2015 р., м. Одеса: зб. тез доп. – Одеса: ОНАЗ, 2015. – С. 35-39.

37. Коваленко А.С. Розробка інформаційної моделі автоматизованої оцінки технічного стану інтегральної інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформаційні технології та взаємодії (ІТ & І): ІІ

між нар. наук.-практ. конф., 3-5 лист. 2015 р., м. Київ: тези доп. – Київ: КНУ ім. Т. Шевченка, 2015. – С. 41-42.

38. Коваленко А.С. Разработка метода усовершенствования технического обслуживания интегрированной информационной системы / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Информационные и телекоммуникационные технологии: образование, наука, практика: II междунар. научн.-практ. конф., 3-4 дек. 2015 г., г. Алматы, Казахстан: сб. труд. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – Т.2. – С. 423-427.

39. Королюк Н.А. Оценка временных интервалов работы лица, принимающего решение, на автоматизированном командном пункте / Н.А. Королюк, А.И. Тимочко // Системи обробки інформації. – Х.: ХУПС, 2005. – Вип. 8 (48). – С. 51-54.

40. Костерев В.В. Надёжность технических систем и управление риском: учебн. пособ. / В.В. Костерев. – М.: МИФИ, 2008. – 280 с.

41. Костюков А.В. Підвищення операційної ефективності підприємств на основі моніторингу в реальному часі. / А.В. Костюков, В.М. Костюков. – М.: Машинобудування, 2009. – 192 с.

42. Лазарев А.А. Выбор показателя затрат для анализа сравнительной экономической эффективности техники конечного потребления / А.А. Лазарев, М.В. Бейлин // Сборник научных трудов ХГПУ.– Х.: ХГПУ, 1999. – Вып. 74. – С. 27-29.

43. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 1. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 308 с.

44. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 2. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 208 с.

45. Лапсарь А.П. Метод оценки состояния сложных технических объектов для синтеза быстродействующих прогнозирующих систем / А.П. Лапсарь // Измерительная техника. – 2004. – № 2. – С. 7-10.

46. Линейные задачи оптимизации: Учеб. пособие / С.В. Лутманов. – Пермь: ЛИТЕР-А, 2004. – Ч.1. – Линейное программирование. – 128 с.

47. Литвак Б.Г. Экспертные технологии в управлении. Учебное пособие / Б.Г. Литвак. – М.: Дело, 2014. – 318 с.

48. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

49. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

50. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

51. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін -т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. - Кіровоград: КІСМ, 1997. - 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.21.0021.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Тесля С.В.			Дослідження та програмна реалізація системи керування комунальних теплових мереж	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнов С.А.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи керування комунальних теплових мереж.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи керування комунальних теплових мереж.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи керування комунальних теплових мереж;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4 Sydney.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних умов праці на робочому місці програміста.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 97 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

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

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнов С.А.

Дослідження та програмна реалізація

системи керування комунальних теплових мереж

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 76

Літера: РП

Кропивницький – 2021 року

Файл проекту ПЗ

```
program TEPLOMERAGA;
// Copyright (C)  //
// Teslia S.V.,  2021, CNTU //
uses
  ShareMem, Windows, SysUtils,
  Forms,  Dialogs,
  Buttons in 'Buttons.pas' {TeplMerButtons},
  MyTLstreams in 'MyTLstreams.pas' {TeplMerMyTLstreams},
  TextTestRunner in 'TextTestRunner.pas'{TeplMerTestRunner},
  TeplMerEditors in 'TeplMerEditors.pas' {TeplMerEditors},
  TeplMerAbout in 'TeplMerAbout.pas' {TeplMerAboutBox},
  RS in 'RS_232C.pas';
{$R *.RES}
{$R versioninfo.res }
const
rcs_id :string = '#(@)$Id: TEPLOMERAGA.dpr,v 1.0 2008 judc Exp $';
SwitchChars = ['-','/'];
procedure RunInConsoleMode;
var
  i :Integer;
begin
  try
    if not IsConsole then
      Windows.AllocConsole;
    for i := 1 to ParamCount do
    begin
      if not (ParamStr(i)[1] in SwitchChars) then
         RegisterModuleTests(ParamStr(i));
    end;
    TextTestRunner.RunRegisteredTests(rxbHaltOnFailures);
  except
    on e:Exception do
      Writeln(Format('%s: %s', [e.ClassName, e.Message]));
  end;
end;
begin
    RegisterExpectedMemoryLeak(36, 1); // TWinHelpViewer x 1
    RegisterExpectedMemoryLeak(20, 3); // TObjectList x 3
    RegisterExpectedMemoryLeak(20, 3);
    RegisterExpectedMemoryLeak(52, 1); // THelpManager x 1
  if FindCmdLineSwitch('c', SwitchChars, true) then
    RunInConsoleMode
  else
  begin
    Application.Initialize;
    Application.Title := ' Project TEPLOMERAGA ';
if not SysUtils.FindCmdLineSwitch('nologo', ['/','-'], true) then
    TEPLOMERAGAAbout.Splash;
Application.CreateForm(TTeplMerButtons, TeplMerButtons);
Application.CreateForm(TTeplMerMyTLstreams, TeplMerMyTLstreams);
Application.CreateForm(TTeplMerTestRunner, TeplMerTestRunner);
Application.CreateForm(TTeplMerEditors, TeplMerEditors);
Application.CreateForm(TTeplMerAboutBox, TeplMerAboutBox);
    try
      Application.Run;
    except
       on e:Exception do
         ShowMessage(e.Message);
    end;
  end;
end.
```

Файл організації потоку

```
unit MyTLstreams;
// Copyright (C)  //
// Teslia S.V.  //
// 2021, CNTU //
interface

uses Windows, ActiveX, SysUtils, Classes, VirtIntf, AxCtrls;

type

  TMyTLstreams = class(TStreamAdapter, IStreamModifyTime)
  protected
    FModifyTime: Longint;
  public
constructor Create(Stream: TStream; Ownership: TStreamOwnership = soReference);
function Write(pv: Pointer; cb: Longint; pcbWritten: PLongint): HResult;
override;
function Stat(out statstg: TStatStg; grfStatFlag: Longint): HResult; override;
function GetModifyTime: Longint; virtual; stdcall;
procedure SetModifyTime(Value: Longint); virtual; stdcall;
  end;
 TBILOTKACH=array[1..255] of Char; PXXXX=^ TBILOTKACH;

  TIMyTLMemoryStream = class(TMyTLstreams)
  private
    function GetMemoryStream: TMemoryStream;
  public
 constructor
Create(Stream:TMemoryStream;Ownership:TStreamOwnership=soReference);
    property MemoryStream: TMemoryStream read GetMemoryStream;
  end;

  TIFileStream = class(TStreamAdapter, IStreamModifyTime)
  private
    function GetFileStream: TFileStream;
  public
    constructor Create(const FileName: string; Mode: Word);
    function Commit(grfCommitFlags: Longint): HResult; override;
    function Stat(out statstg: TStatStg; grfStatFlag: Longint): HResult;
override;
    function GetModifyTime: Longint; stdcall;
    procedure SetModifyTime(Time: Longint); stdcall;
    property FileStream: TFileStream read GetFileStream;
  end;

  TForm1 = class(TOleStream)
  private
    FStreamModifyTime: IStreamModifyTime;
  public
    constructor Create(AStream: IStream);
    function GetModifyTime: Longint;
    procedure SetModifyTime(Time: Longint);
  end;

  TExceptionHandler = procedure;

const
  ExceptionHandler: TExceptionHandler = nil;

implementation

uses Libc;

constructor TMyTLstreams.Create(Stream: TStream;
  Ownership: TStreamOwnership);
begin
  inherited Create(Stream, Ownership);
```

```
    FModifyTime := DateTimeToFileDate(Now);
  end;

  function TMyTLstreams.Write(pv: Pointer; cb: Longint;
    pcbWritten: PLongint): HResult;
  begin
    Result := inherited Write(pv, cb, pcbWritten);
    FModifyTime := DateTimeToFileDate(Now);
  end;

  function TMyTLstreams.Stat(out statstg: TStatStg; grfStatFlag: Longint):
HResult;
  var
    DosFileTime: Longint;
    LocalFileTime: TFileTime;
  begin
    Result := inherited Stat(statstg, grfStatFlag);
    if Result <> 0 then Exit;
    DosFileTime := GetModifyTime;
    DosDateTimeToFileTime(LongRec(DosFileTime).Hi, LongRec(DosFileTime).Lo,
LocalFileTime);
    LocalFileTimeToFileTime(LocalFileTime, statstg.mtime);
  end;

  function TMyTLstreams.GetModifyTime: Longint;
  begin
    Result := FModifyTime;
  end;

  procedure TMyTLstreams.SetModifyTime(Value: Longint);
  begin
    FModifyTime := Value;
  end;

  constructor TIMyTLMemoryStream.Create(Stream: TMemoryStream;
    Ownership: TStreamOwnership);
  begin
    if Stream = nil then
    begin
      Ownership := soOwned;
      Stream := TMemoryStream.Create;
    end;
    inherited Create(Stream, Ownership);
  end;

  function TIMyTLMemoryStream.GetMemoryStream: TMemoryStream;
  begin
    Result := TMemoryStream(Stream);
  end;

  constructor TIFileStream.Create(const FileName: string; Mode: Word);
  begin
    if Mode = fmCreate then
      unlink(PChar(FileName));
    inherited Create(TFileStream.Create(FileName, Mode), soOwned);
  end;

  function TIFileStream.GetFileStream: TFileStream;
  begin
    Result := TFileStream(Stream);
  end;

function TIFileStream.Stat(out statstg: TStatStg; grfStatFlag: Longint):
HResult;
  begin
    Result := inherited Stat(statstg, grfStatFlag);
    if Result <> 0 then Exit;
    GetFileTime(TFileStream(Stream).Handle, @statstg.ctime, @statstg.atime,
@statstg.mtime);
```

```
end;

function TIFileStream.GetModifyTime: Longint;
begin
  Result := FileGetDate(FileStream.Handle);
end;

function TForm1.CheckSum(AStr: String): Char;
var crc,i: Integer;
begin
crc:=0;
for i := 1 to Length(AStr) do
crc:=crc+Ord(AStr[i]);
crc:=(crc and $3F) + $30;
Result:=Chr(crc);
end;
function TForm1.CompareCheckSum(AStr: String; CS: Char): boolean;
begin
Result:=CheckSum(AStr)=CS;
end;

function TForm1.GetInput: String;
var l:integer;
begin
Result:='';
l:=Length(FInput);
if InputState = 1 then
begin
 if StartTime+3000 < GetTickCount then InputState := 2;
 Exit;
end;
if l<3 then Exit;
if CompareCheckSum(Copy(FInput,1,l-2),Copy(FInput,l-1,1)[1])=true  then
begin
InputState := 0;
NetNumber:=FInput[1];
AddrEdt.Text:=NetNumber;
Result:=copy(FInput,2,l-3);
end
else
InputState := 3;
end;

procedure TForm1.SetOutput(const Value: String);
var XXXX:TXXXX;
    S:String;
    L,i:Integer;
begin
S:=NetNumber+Value;
S:=S+CheckSum(S)+#13;
L:=Length(S);
if L>255 then Exit;
for i:=1 to L do XXXX[i] := S[i];
InputState := 1;
FInput:='';
CommPortDriver1.SendData(@XXXX,L);
StartTime:=GetTickCount;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
NetNumber:='0';
CommPortDriver1.Connect;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
CommPortDriver1.Disconnect;
end;
```

```
 procedure TForm1.CommPortDriver1ReceiveData(Sender: TObject;  DataPtr:
Pointer; DataSize: Integer);
 var PX:PXXXX; i:integer;
 begin
 InputState := 4;
 Application.ProcessMessages;
 FInput:='';
 PX:=DataPtr;
 for i := 1 to DataSize do
 begin
 FInput:=FInput+PX^[i];
 end;
 InputState := 5;
 Application.ProcessMessages;
 Edit2.Text:=Input;
 end;

 procedure TForm1.SetInputState(const Value: integer);
 begin
 FInputState := Value;
 case Value of
     0: Caption:='Данные успешно приняты';
      1: Caption:='Ждем ответа';
      2: Caption:='Таймаут';
      3: Caption:='Пакет принят с ошибкой';
      4: Caption:='Принимаем ответ';
      5: Caption:='Ответ получен';
 end;
 end;

 procedure TForm1.SendBtnClick(Sender: TObject);
 begin
 Output:=OutputEdt.Text;
 SendBtn.Enabled:=False;
 repeat
  Edit2.Text:=Input;
 until InputState<>1;
 SendBtn.Enabled:=True;
 end;

 procedure TForm1.AddrEdtChange(Sender: TObject);
 begin
 NetNumber:=AddrEdt.Text[1];
 end;

 procedure TIFileStream.SetModifyTime(Time: Longint);
 begin
   FileSetDate(FileStream.Handle, Time);
 end;

 function TIFileStream.Commit(grfCommitFlags: Longint): HResult;
 begin
   FlushFileBuffers(FileStream.Handle);
   Result := inherited Commit(grfCommitFlags);
 end;

 constructor TVirtualStream.Create(AStream: IStream);
 begin
   inherited Create(AStream);
   if AStream.QueryInterface(IStreamModifyTime, FStreamModifyTime) <> 0 then
     FStreamModifyTime := nil;
 end;

 function TVirtualStream.GetModifyTime: Longint;
 begin
   if FStreamModifyTime <> nil then
     Result := FStreamModifyTime.GetModifyTime
   else
```

```
    Result := 0;
end;

procedure TVirtualStream.SetModifyTime(Time: Longint);
begin
  if FStreamModifyTime <> nil then
    FStreamModifyTime.SetModifyTime(Time);
end;

end.
```

Файл обробки та виведення даних

```
unit TeplMerEditors;
// Copyright (C)  //
// Teslia S.V.  //
// 2021, CNTU //
interface

uses
  Types, SysUtils, Classes, TypInfo, Variants, DesignIntf, DesignMenus;

type
  TInstProp = record
    Instance: TPersistent;
    PropInfo: PPropInfo;
  end;

  PInstPropList = ^TInstPropList;
  TInstPropList = array[0..1023] of TInstProp;

  TPropertyEditor = class(TBasePropertyEditor, IProperty, IProperty70)
  private
    FDesigner: IDesigner;
    FPropList: PInstPropList;
    FPropCount: Integer;
    FAncestorList: TList;
    FRoot: TComponent;
    FAncestor: TPersistent;
    FRootAncestor: TComponent;
    FLookingFor: TComponent;
    FDoneLooking: Boolean;
    procedure AddAncestor(Component: TComponent);
    procedure GetLookupInfo(var Ancestor: TPersistent;
      var Root, LookupRoot, RootAncestor: TComponent);
    function GetPrivateDirectory: string;
    procedure WriteComponentSimulation(Component: TComponent);
  protected
    procedure SetPropEntry(Index: Integer; AInstance: TPersistent;
      APropInfo: PPropInfo); override;
  protected
    function GetFloatValue: Extended;
    function GetFloatValueAt(Index: Integer): Extended;
    function GetInt64Value: Int64;
    function GetInt64ValueAt(Index: Integer): Int64;
    function GetMethodValue: TMethod;
    function GetMethodValueAt(Index: Integer): TMethod;
    function GetOrdValue: Longint;
    function GetOrdValueAt(Index: Integer): Longint;
    function GetStrValue: string;
    function GetStrValueAt(Index: Integer): string;
    function GetVarValue: Variant;
    function GetVarValueAt(Index: Integer): Variant;
    function GetIntfValue: IInterface;
    function GetIntfValueAt(Index: Integer): IInterface;
    procedure Modified;
    procedure SetFloatValue(Value: Extended);
    procedure SetMethodValue(const Value: TMethod);
    procedure SetInt64Value(Value: Int64);
    procedure SetOrdValue(Value: Longint);
    procedure SetStrValue(const Value: string);
    procedure SetVarValue(const Value: Variant);
    procedure SetIntfValue(const Value: IInterface);
  protected
    function GetEditValue(out Value: string): Boolean;
    function HasInstance(Instance: TPersistent): Boolean;
    function GetIsDefault: Boolean; virtual;
  public
    constructor Create(const ADesigner: IDesigner; APropCount: Integer);
override;
```

```
  destructor Destroy; override;
  procedure Activate; virtual;
  function AllEqual: Boolean; virtual;
  function AutoFill: Boolean; virtual;
  procedure Edit; virtual;
  function GetAttributes: TPropertyAttributes; virtual;
  function GetComponent(Index: Integer): TPersistent;
  function GetEditLimit: Integer; virtual;
  function GetName: string; virtual;
  procedure GetProperties(Proc: TGetPropProc); virtual;
  function GetPropInfo: PPropInfo; virtual;
  function GetPropType: PTypeInfo;
  function GetValue: string; virtual;
  function GetVisualValue: string;
  procedure GetValues(Proc: TGetStrProc); virtual;
  procedure Initialize; override;
  procedure Revert;
  procedure SetValue(const Value: string); virtual;
  function ValueAvailable: Boolean;
  property Designer: IDesigner read FDesigner;
  property PrivateDirectory: string read GetPrivateDirectory;
  property PropCount: Integer read FPropCount;
  property Value: string read GetValue write SetValue;
end;

TOrdinalProperty = class(TeplMerEditors)
  function AllEqual: Boolean; override;
  function GetEditLimit: Integer; override;
end;

TIntegerProperty = class(TOrdinalProperty)
public
  function GetValue: string; override;
  procedure SetValue(const Value: string); override;
end;

TCharProperty = class(TOrdinalProperty)
public
  function GetValue: string; override;
  procedure SetValue(const Value: string); override;
end;

TEnumProperty = class(TOrdinalProperty)
public
  function GetAttributes: TPropertyAttributes; override;
  function GetValue: string; override;
  procedure GetValues(Proc: TGetStrProc); override;
  procedure SetValue(const Value: string); override;
end;

TInt64Property = class(TeplMerEditors)
public
  function AllEqual: Boolean; override;
  function GetEditLimit: Integer; override;
  function GetValue: string; override;
  procedure SetValue(const Value: string); override;
end;

TFloatProperty = class(TeplMerEditors)
public
  function AllEqual: Boolean; override;
  function GetValue: string; override;
  procedure SetValue(const Value: string); override;
end;

TStringProperty = class(TeplMerEditors)
public
  function AllEqual: Boolean; override;
  function GetEditLimit: Integer; override;
```

```
      function GetValue: string; override;
      procedure SetValue(const Value: string); override;
    end;

    TNestedProperty = class(TeplMerEditors)
    public
      constructor Create(Parent: TeplMerEditors); reintroduce;
      destructor Destroy; override;
    end;

    TSetTeplMerElementProperty = class(TNestedProperty)
    private
      FElement: Integer;
    protected
      constructor Create(Parent: TeplMerEditors; AElement: Integer);
reintroduce;
      property Element: Integer read FElement;
      function GetIsDefault: Boolean; override;
    public
      function AllEqual: Boolean; override;
      function GetAttributes: TPropertyAttributes; override;
      function GetName: string; override;
      function GetValue: string; override;
      procedure GetValues(Proc: TGetStrProc); override;
      procedure SetValue(const Value: string); override;
     end;

    TSetProperty = class(TOrdinalProperty)
    public
      function GetAttributes: TPropertyAttributes; override;
      procedure GetProperties(Proc: TGetPropProc); override;
      function GetValue: string; override;
    end;

    TClassProperty = class(TeplMerEditors)
    public
      function GetAttributes: TPropertyAttributes; override;
      procedure GetProperties(Proc: TGetPropProc); override;
      function GetValue: string; override;
    end;

    TMethodProperty = class(TeplMerEditors, IMethodProperty)
    public
      function AllNamed: Boolean; virtual;
      function AllEqual: Boolean; override;
      procedure Edit; override;
      function GetAttributes: TPropertyAttributes; override;
      function GetEditLimit: Integer; override;
      function GetValue: string; override;
      procedure GetValues(Proc: TGetStrProc); override;
      procedure SetValue(const AValue: string); override;
      function GetFormMethodName: string; virtual;
      function GetTrimmedEventName: string;
    end;

    TComponentProperty = class(TeplMerEditors, IReferenceProperty)
    protected
      function FilterFunc(const ATestEditor: IProperty): Boolean;
      function GetComponentReference: TComponent; virtual;
      function GetSelections: IDesignerSelections; virtual;
    public
      function AllEqual: Boolean; override;
      procedure Edit; override;
      function GetAttributes: TPropertyAttributes; override;
      procedure GetProperties(Proc: TGetPropProc); override;
      function GetEditLimit: Integer; override;
      function GetValue: string; override;
      procedure GetValues(Proc: TGetStrProc); override;
      procedure SetValue(const Value: string); override;
```

```
    end;

    TInterfaceProperty = class(TComponentProperty)
    private
      FGetValuesStrProc: TGetStrProc;
    protected
      procedure ReceiveComponentNames(const S: string);
      function GetComponent(const AInterface: IInterface): TComponent;
      function GetComponentReference: TComponent; override;
      function GetSelections: IDesignerSelections; override;
    public
      function AllEqual: Boolean; override;
      procedure GetValues(Proc: TGetStrProc); override;
      procedure SetValue(const Value: string); override;
    end;

    TComponentNameProperty = class(TStringProperty)
    public
      function GetAttributes: TPropertyAttributes; override;
      function GetEditLimit: Integer; override;
    end;

    TDateProperty = class(TeplMerEditors)
      function GetAttributes: TPropertyAttributes; override;
      function GetValue: string; override;
      procedure SetValue(const Value: string); override;
    end;

    TTimeProperty = class(TeplMerEditors)
      function GetAttributes: TPropertyAttributes; override;
      function GetValue: string; override;
      procedure SetValue(const Value: string); override;
    end;

    TDateTimeProperty = class(TeplMerEditors)
      function GetAttributes: TPropertyAttributes; override;
      function GetValue: string; override;
      procedure SetValue(const Value: string); override;
    end;

    TVariantProperty = class(TeplMerEditors)
      function GetAttributes: TPropertyAttributes; override;
      function GetValue: string; override;
      procedure SetValue(const Value: string); override;
      procedure GetProperties(Proc: TGetPropProc); override;
    end;

  procedure GetComponentProperties(const Components: IDesignerSelections;
    Filter: TTypeKinds; const Designer: IDesigner; Proc: TGetPropProc;
    EditorFilterFunc: TeplMerEditorsFilterFunc = nil);

  type
    TTeplMerComponentEditor = class(TBaseComponentEditor, IComponentEditor)
    private
      FComponent: TComponent;
      FDesigner: IDesigner;
    public
      constructor Create(AComponent: TComponent; ADesigner: IDesigner);
override;
      procedure Edit; virtual;
      procedure ExecuteVerb(Index: Integer); virtual;
      function GetComponent: TComponent;
      function GetDesigner: IDesigner;
      function GetVerb(Index: Integer): string; virtual;
      function GetVerbCount: Integer; virtual;
      function IsInInlined: Boolean;
      procedure Copy; virtual;
      procedure PrepareItem(Index: Integer; const AItem: IMenuItem); virtual;
      property Component: TComponent read FComponent;
```

```
      property Designer: IDesigner read GetDesigner;
    end;

    TDefaultEditor = class(TTeplMerComponentEditor, IDefaultEditor)
    private
      FFirst: IProperty;
      FBest: IProperty;
      FContinue: Boolean;
      procedure CheckEdit(const Prop: IProperty);
    protected
      procedure EditProperty(const Prop: IProperty; var Continue: Boolean);
virtual;
    public
      procedure Edit; override;
    end;

  function GeTTeplMerComponentEditor(Component: TComponent;
    const Designer: IDesigner): IComponentEditor;

  type

    TeplMerSelectionEditor = class(TBaseSelectionEditor, ISelectionEditor)
    private
      FDesigner: IDesigner;
    public
      constructor Create(const ADesigner: IDesigner); override;
      procedure ExecuteVerb(Index: Integer; const List: IDesignerSelections);
virtual;
      function GetVerb(Index: Integer): string; virtual;
      function GetVerbCount: Integer; virtual;
      procedure RequiresUnits(Proc: TGetStrProc); virtual;
      procedure PrepareItem(Index: Integer; const AItem: IMenuItem); virtual;
      property Designer: IDesigner read FDesigner;
    end;

  function GeTeplMerSelectionEditors(const Designer: IDesigner):
ISelectionEditorList; overload;
  function GeTeplMerSelectionEditors(const Designer: IDesigner;
    const Selections: IDesignerSelections): ISelectionEditorList; overload;
  function GeTeplMerSelectionEditors(const Designer: IDesigner;
    Component: TComponent): ISelectionEditorList; overload;

  type

    TEditActionSelectionEditor = class(TeplMerSelectionEditor)
    private
      procedure HandleToBack(Sender: TObject);
      procedure HandleToFront(Sender: TObject);
    protected
      function GetEditState: TEditState;
      procedure EditAction(Action: TEditAction);
      procedure HandleCopy(Sender: TObject);
      procedure HandleCut(Sender: TObject);
      procedure HandleDelete(Sender: TObject);
      procedure HandlePaste(Sender: TObject);
      procedure HandleSelectAll(Sender: TObject);
      procedure HandleUndo(Sender: TObject);
    public
      function GetVerb(Index: Integer): string; override;
      function GetVerbCount: Integer; override;
      procedure PrepareItem(Index: Integer; const AItem: IMenuItem); override;
    end;

  type
    TCustomModule = class(TBaseCustomModule, ICustomModule)
    private
      FRoot: TComponent;
      FDesigner: IDesigner;
      FFinder: TClassFinder;
```

```
    public
      constructor Create(ARoot: TComponent; const ADesigner: IDesigner);
override;
      destructor Destroy; override;
      procedure ExecuteVerb(Index: Integer); virtual;
      function GetAttributes: TCustomModuleAttributes; virtual;
      function GetVerb(Index: Integer): string; virtual;
      function GetVerbCount: Integer; virtual;
      procedure Saving; virtual;
      procedure PrepareItem(Index: Integer; const AItem: IMenuItem); virtual;
      procedure ValidateComponent(Component: TComponent); virtual;
      function ValidateComponentClass(ComponentClass: TComponentClass): Boolean;
virtual;
      function Nestable: Boolean; virtual;
      property Root: TComponent read FRoot;
      property Designer: IDesigner read FDesigner;
    end;

  function ClassInheritsFrom(ClassType: TClass; const ClassName: string):
Boolean;
  function AncestorNameMatches(ClassType: TClass; AncestorClass: TClass):
Boolean;
  type
    TGetTopLevelComponentFunc = function(Ignoring: TComponent = nil):
TComponent;
  var
    GetTopLevelComponentFunc: TGetTopLevelComponentFunc;

  function PossibleStream(const S: string): Boolean;
  type
    TGroupChangeProc = procedure(AGroup: Integer);
  function NewEditorGroup: Integer;
  procedure FreeEditorGroup(Group: Integer);
  procedure NotifyGroupChange(AProc: TGroupChangeProc);
  procedure UnnotifyGroupChange(AProc: TGroupChangeProc);

  var
    GReferenceExpandable: Boolean = True;
    GShowReadOnlyProps: Boolean = True;

  implementation

  uses DesignConst, Consts, RTLConsts, Contnrs, Proxies;

  function PossibleStream(const S: string): Boolean;
  var
    I: Integer;
  begin
    Result := True;
    for I := 1 to Length(S) - 6 do
    begin
  if ((S[I] in ['O','o']) and (CompareText(Copy(S, I, 6), 'OBJECT') = 0)) or
   ((S[I] in ['I','i']) and (CompareText(Copy(S, I, 6), 'INLINE') = 0)) then
       Exit;
      if not (S[I] in [' ',#9, #13, #10]) then Break;
    end;
    Result := False;
  end;

  constructor TeplMerEditors.Create(const ADesigner: IDesigner;
    APropCount: Integer);
  begin
    inherited Create(ADesigner, APropCount);
    FDesigner := ADesigner;
    GetMem(FPropList, APropCount * SizeOf(TInstProp));
    FPropCount := APropCount;
  end;

  destructor TeplMerEditors.Destroy;
```

```
begin
  if FPropList <> nil then
    FreeMem(FPropList, FPropCount * SizeOf(TInstProp));
end;

procedure TeplMerEditors.Activate;
begin
end;

function TeplMerEditors.AllEqual: Boolean;
begin
  Result := FPropCount = 1;
end;

procedure TeplMerEditors.Edit;
type
  TGetStrFunc = function(const Value: string): Integer of object;
var
  I: Integer;
  Values: TStringList;
  AddValue: TGetStrFunc;
begin
  if not AutoFill then Exit;
  Values := TStringList.Create;
  Values.Sorted := paSortList in GetAttributes;
  try
    AddValue := Values.Add;
    GetValues(TGetStrProc(AddValue));
    if Values.Count > 0 then
    begin
      I := Values.IndexOf(Value) + 1;
      if I = Values.Count then I := 0;
      Value := Values[I];
    end;
  finally
    Values.Free;
  end;
end;

function TeplMerEditors.AutoFill: Boolean;
begin
  Result := Assigned(GetPropInfo^.SetProc);
end;

function TeplMerEditors.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paRevertable];
end;

function TeplMerEditors.GetComponent(Index: Integer): TPersistent;
begin
  Result := FPropList^[Index].Instance;
end;


function TeplMerEditors.GetFloatValue: Extended;
begin
  Result := GetFloatValueAt(0);
end;

function TeplMerEditors.GetFloatValueAt(Index: Integer): Extended;
begin
  with FPropList^[Index] do Result := GetFloatProp(Instance, PropInfo);
end;

function TeplMerEditors.GetMethodValue: TMethod;
begin
  Result := GetMethodValueAt(0);
end;
```

```
function TeplMerEditors.GetMethodValueAt(Index: Integer): TMethod;
begin
  with FPropList^[Index] do Result := GetMethodProp(Instance, PropInfo);
end;

function TeplMerEditors.GetEditLimit: Integer;
begin
  Result := 2047;
end;

function TeplMerEditors.GetName: string;
begin
  Result := FPropList^[0].PropInfo^.Name;
end;

function TeplMerEditors.GetOrdValue: Longint;
begin
  Result := GetOrdValueAt(0);
end;

function TeplMerEditors.GetOrdValueAt(Index: Integer): Longint;
begin
  with FPropList^[Index] do Result := GetOrdProp(Instance, PropInfo);
end;

function TeplMerEditors.GetPrivateDirectory: string;
begin
  Result := '';
  if Designer <> nil then
    Result := Designer.GetPrivateDirectory;
end;

procedure TeplMerEditors.GetProperties(Proc: TGetPropProc);
begin
end;

function TeplMerEditors.GetPropInfo: PPropInfo;
begin
  Result := FPropList^[0].PropInfo;
end;

function TeplMerEditors.GetPropType: PTypeInfo;
begin
  Result := FPropList^[0].PropInfo^.PropType^;
end;

function TeplMerEditors.GetStrValue: string;
begin
  Result := GetStrValueAt(0);
end;

function TeplMerEditors.GetStrValueAt(Index: Integer): string;
begin
  with FPropList^[Index] do Result := GetStrProp(Instance, PropInfo);
end;

function TeplMerEditors.GetVarValue: Variant;
begin
  Result := GetVarValueAt(0);
end;

function TeplMerEditors.GetVarValueAt(Index: Integer): Variant;
begin
  with FPropList^[Index] do Result := GetVariantProp(Instance, PropInfo);
end;

function TeplMerEditors.GetValue: string;
begin
```

```
  Result := srUnknown;
end;

function TeplMerEditors.GetVisualValue: string;
begin
  if AllEqual then
    Result := GetValue
  else
    Result := '';
end;

procedure TeplMerEditors.GetValues(Proc: TGetStrProc);
begin
end;

procedure TeplMerEditors.Initialize;
begin
end;

procedure TeplMerEditors.Modified;
begin
  if Designer <> nil then
    Designer.Modified;
end;

procedure TeplMerEditors.SetFloatValue(Value: Extended);
var
  I: Integer;
begin
  for I := 0 to FPropCount - 1 do
    with FPropList^[I] do SetFloatProp(Instance, PropInfo, Value);
  Modified;
end;

procedure TeplMerEditors.SetMethodValue(const Value: TMethod);
var
  I: Integer;
begin
  for I := 0 to FPropCount - 1 do
    with FPropList^[I] do SetMethodProp(Instance, PropInfo, Value);
  Modified;
end;

procedure TeplMerEditors.SetOrdValue(Value: Longint);
var
  I: Integer;
begin
  for I := 0 to FPropCount - 1 do
    with FPropList^[I] do SetOrdProp(Instance, PropInfo, Value);
  Modified;
end;

procedure TeplMerEditors.SetPropEntry(Index: Integer;
  AInstance: TPersistent; APropInfo: PPropInfo);
begin
  with FPropList^[Index] do
  begin
    Instance := AInstance;
    PropInfo := APropInfo;
  end;
end;

procedure TeplMerEditors.SetStrValue(const Value: string);
var
  I: Integer;
begin
  for I := 0 to FPropCount - 1 do
    with FPropList^[I] do SetStrProp(Instance, PropInfo, Value);
  Modified;
```

```
end;

procedure TeplMerEditors.SetVarValue(const Value: Variant);
var
  I: Integer;
begin
  for I := 0 to FPropCount - 1 do
    with FPropList^[I] do SetVariantProp(Instance, PropInfo, Value);
  Modified;
end;

procedure TeplMerEditors.Revert;
var
  I: Integer;
begin
  if Designer <> nil then
    for I := 0 to FPropCount - 1 do
      with FPropList^[I] do Designer.Revert(Instance, PropInfo);
end;

procedure TeplMerEditors.SetValue(const Value: string);
begin
end;

function TeplMerEditors.ValueAvailable: Boolean;
var
  I: Integer;
  S: string;
begin
  Result := True;
  for I := 0 to FPropCount - 1 do
  begin
    if (FPropList^[I].Instance is TComponent) and
      (csCheckPropAvail in TComponent(FPropList^[I].Instance).ComponentStyle)
then
    begin
      try
        S := GetValue;
        AllEqual;
      except
        Result := False;
      end;
      Exit;
    end;
  end;
end;

function TeplMerEditors.GetInt64Value: Int64;
begin
  Result := GetInt64ValueAt(0);
end;

function TeplMerEditors.GetInt64ValueAt(Index: Integer): Int64;
begin
  with FPropList^[Index] do Result := GetInt64Prop(Instance, PropInfo);
end;

procedure TeplMerEditors.SetInt64Value(Value: Int64);
var
  I: Integer;
begin
  for I := 0 to FPropCount - 1 do
    with FPropList^[I] do SetInt64Prop(Instance, PropInfo, Value);
  Modified;
end;

function TeplMerEditors.GetIntfValue: IInterface;
begin
  Result := GetIntfValueAt(0);
```

```
end;

function TeplMerEditors.GetIntfValueAt(Index: Integer): IInterface;
begin
  with FPropList^[Index] do Result := GetInterfaceProp(Instance, PropInfo);
end;

procedure TeplMerEditors.SetIntfValue(const Value: IInterface);
var
  I: Integer;
begin
  for I := 0 to FPropCount - 1 do
    with FPropList^[I] do SetInterfaceProp(Instance, PropInfo, Value);
  Modified;
end;

function TeplMerEditors.GetEditValue(out Value: string): Boolean;
begin
  Result := False;
  try
    Value := GetValue;
    Result := Assigned(GetPropInfo^.SetProc);
  except
    on E: EPropWriteOnly do Value := sNotAvailable;
    on E: Exception do Value := Format('(%s)', [E.Message]);
  end;
end;

function TeplMerEditors.HasInstance(Instance: TPersistent): Boolean;
var
  I: Integer;
begin
  Result := True;
  for I := 0 to FPropCount - 1 do
    if FPropList^[I].Instance = Instance then Exit;
  Result := False;
end;

type
  TComponentHack = class(TComponent);

procedure TeplMerEditors.WriteComponentSimulation(Component: TComponent);
  function FindAncestor(const Name: string): TComponent;
  var
    I: Integer;
  begin
    for I := 0 to FAncestorList.Count - 1 do
    begin
      Result := FAncestorList[I];
      if SameText(Result.Name, Name) then Exit;
    end;
    Result := nil;
  end;
var
  OldAncestor: TPersistent;
  OldRoot, OldRootAncestor: TComponent;
  OldAncestorList: TList;
  TempAncestor: TPersistent;
begin
  if FDoneLooking then
    Exit;

  OldAncestor := FAncestor;
  OldRootAncestor := FRootAncestor;
  try
    if Assigned(FAncestorList) then
      FAncestor := FindAncestor(Component.Name);
    if FLookingFor = Component then
    begin
```

```
        FDoneLooking := True
      end
      else if SameText(FLookingFor.Name, Component.Name) then
      begin
        FDoneLooking := True;
      end
      else
      begin

        if (FAncestor = nil) and (Component <> Designer.Root)
          and IsProxyClass(Component.ClassType) then
        begin
          TempAncestor := Designer.FindRootAncestor(Component.ClassName);
          if TempAncestor <> nil then
          begin
            FAncestor := TempAncestor;
            FRootAncestor := TComponent(FAncestor);
          end;

          InlineRoot := ActiveDesigner.OpenRootClass(Component.ClassName);
          if InlineRoot <> nil then
          begin
            FAncestor := InlineRoot.GetRoot;
            FRootAncestor := TComponent(FAncestor);
          end;
        end;

        OldAncestorList := FAncestorList;
        OldRoot := FRoot;
        OldRootAncestor := FRootAncestor;
        try
          FAncestorList := nil;
            try
              if (FAncestor <> nil) and (FAncestor is TComponent) then
              begin
                if csInline in TComponent(FAncestor).ComponentState then
                  FRootAncestor := TComponent(FAncestor);
                FAncestorList := TList.Create;
                TComponentHack(FAncestor).GetChildren(AddAncestor,
FRootAncestor);
              end;
              if csInline in Component.ComponentState then
                FRoot := Component;
              TComponentHack(Component).GetChildren(WriteComponentSimulation,
FRoot);
            finally
              FAncestorList.Free;
            end;
        finally
          FAncestorList := OldAncestorList;
          if not FDoneLooking then
          begin
            FRoot := OldRoot;
            FRootAncestor := OldRootAncestor;
          end;
        end;
      end;
    finally
      if not FDoneLooking then
      begin
        FAncestor := OldAncestor;
        FRootAncestor := OldRootAncestor;
      end
    end;
  end;

  function TeplMerEditors.GetIsDefault: Boolean;
    function CheckProperties(AnObject: TObject): Boolean;
    var
```

```
      PropList: PPropList;
      PropInfo: PPropInfo;
      I, Count: Integer;
    begin
      Result := True;
      Count := GetTypeData(AnObject.ClassInfo)^.PropCount;
      if Count > 0 then
      begin
        GetMem(PropList, Count * SizeOf(Pointer));
        try
          GetPropInfos(AnObject.ClassInfo, PropList);
          for I := 0 to Count - 1 do
          begin
            PropInfo := PropList^[I];
            if PropInfo = nil then
              Break;
            if not IsDefaultPropertyValue(AnObject, PropInfo, GetLookupInfo)
then
            begin
              Result := False;
              Break;
            end;
          end;
        finally
          FreeMem(PropList, Count * SizeOf(Pointer));
        end;
      end;
    end;

  var
    FirstInstance: TObject;
    FirstPropInfo: PPropInfo;

    SubObject: TObject;
    OldAncestor: TPersistent;

  begin
    Result := True;
    if PropCount > 0 then
    begin
      if not AllEqual then
      begin
        Result := False;
        Exit;
      end;

      FirstInstance := FPropList^[0].Instance;
      FirstPropInfo := FPropList^[0].PropInfo;
      if IsStoredProp(FirstInstance, FirstPropInfo) then
      begin
        if (Designer.AncestorDesigner <> nil) then
        begin
          FRootAncestor := Designer.AncestorDesigner.Root;
          FAncestor := FRootAncestor;
        end
        else
        begin
          FRootAncestor := nil;
          FAncestor := nil;
        end;
        FRoot := Designer.Root;

        if FirstInstance is TComponent then
        begin
          FLookingFor := TComponent(FirstInstance);
          if csAncestor in FLookingFor.ComponentState then
          begin
            FDoneLooking := False;
            WriteComponentSimulation(FRoot);
```

```
          end
          else
          begin
            FRootAncestor := nil;
            FAncestor := nil;
          end;
        end
        else
        begin
          FRootAncestor := nil;
          FAncestor := nil;
        end;

        Result := IsDefaultPropertyValue(FirstInstance, FirstPropInfo,
GetLookupInfo);
        if not Result then
        begin
          if FirstPropInfo^.PropType^.Kind = tkClass then
          begin
            SubObject := GetObjectProp(FirstInstance, FirstPropInfo);

            OldAncestor := FAncestor;
            try
              if AncestorIsValid(FAncestor, FRoot, FRootAncestor) then
                FAncestor := TPersistent(GetOrdProp(FAncestor, FirstPropInfo));
              Result := CheckProperties(SubObject);
            finally
              FAncestor := OldAncestor;
            end;

            if SubObject is TCollection then
            begin
              if not AncestorIsValid(FAncestor, FRoot, FRootAncestor) or
                not CollectionsEqual(TCollection(SubObject),
                  TCollection(GetOrdProp(FAncestor, FirstPropInfo)),
                    FRoot, FRootAncestor) then
                    Result := False;
            end;
          end;
        end;
      end;
    end;
  end;

  procedure TeplMerEditors.AddAncestor(Component: TComponent);
  begin
    FAncestorList.Add(Component);
  end;

  procedure TeplMerEditors.GetLookupInfo(var Ancestor: TPersistent;
    var Root, LookupRoot, RootAncestor: TComponent);
  begin
    Ancestor := FAncestor;
    Root := FRoot;
    LookupRoot := FRoot;
    RootAncestor := FRootAncestor;
  end;

  function TOrdinalProperty.AllEqual: Boolean;
  var
    I: Integer;
    V: Longint;
  begin
    Result := False;
    if PropCount > 1 then
    begin
      V := GetOrdValue;
      for I := 1 to PropCount - 1 do
        if GetOrdValueAt(I) <> V then Exit;
```

```
  end;
  Result := True;
end;

function TOrdinalProperty.GetEditLimit: Integer;
begin
  Result := 63;
end;

function TIntegerProperty.GetValue: string;
begin
  with GetTypeData(GetPropType)^ do
    if OrdType = otULong then
      Result := IntToStr(Cardinal(GetOrdValue))
    else
      Result := IntToStr(GetOrdValue);
end;

procedure TIntegerProperty.SetValue(const Value: String);

  procedure Error(const Args: array of const);
  begin
    raise EPropertyError.CreateResFmt(@SOutOfRange, Args);
  end;

var
  L: Int64;
begin
  L := StrToInt64(Value);
  with GetTypeData(GetPropType)^ do
    if OrdType = otULong then
    begin
      if (L < Cardinal(MinValue)) or (L > Cardinal(MaxValue)) then
        Error([Int64(Cardinal(MinValue)), Int64(Cardinal(MaxValue))]);
    end
    else if (L < MinValue) or (L > MaxValue) then
      Error([MinValue, MaxValue]);
  SetOrdValue(L);
end;

function TeplMerCharProperty.GetValue: string;
var
  Ch: Char;
begin
  Ch := Chr(GetOrdValue);
  if Ch in [#33..#127] then
    Result := Ch else
    FmtStr(Result, '#%d', [Ord(Ch)]);
end;

procedure TeplMerCharProperty.SetValue(const Value: string);
var
  L: Longint;
begin
  if Length(Value) = 0 then L := 0 else
    if Length(Value) = 1 then L := Ord(Value[1]) else
      if Value[1] = '#' then L := StrToInt(Copy(Value, 2, Maxint)) else
        raise EPropertyError.CreateRes(@SInvalidPropertyValue);
  with GetTypeData(GetPropType)^ do
    if (L < MinValue) or (L > MaxValue) then
      raise EPropertyError.CreateResFmt(@SOutOfRange, [MinValue, MaxValue]);
  SetOrdValue(L);
end;

function TeplMerEnumProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paValueList, paSortList, paRevertable];
end;
```

```
function TeplMerEnumProperty.GetValue: string;
var
  L: Longint;
begin
  L := GetOrdValue;
  with GetTypeData(GetPropType)^ do
    if (L < MinValue) or (L > MaxValue) then L := MaxValue;
  Result := GetEnumName(GetPropType, L);
end;

procedure TeplMerEnumProperty.GetValues(Proc: TGetStrProc);
var
  I: Integer;
  EnumType: PTypeInfo;
begin
  EnumType := GetPropType;
  with GetTypeData(EnumType)^ do
  begin
    if MinValue < 0 then
    begin
      Proc(GetEnumName(EnumType, 0));
      Proc(GetEnumName(EnumType, 1));
    end
    else
      for I := MinValue to MaxValue do Proc(GetEnumName(EnumType, I));
  end;
end;

procedure TeplMerEnumProperty.SetValue(const Value: string);
var
  I: Integer;
begin
  I := GetEnumValue(GetPropType, Value);
  with GetTypeData(GetPropType)^ do
    if (I < MinValue) or (I > MaxValue) then
      raise EPropertyError.CreateRes(@SInvalidPropertyValue);
  SetOrdValue(I);
end;

function TBoolProperty.GetValue: string;
begin
  Result := BooleanIdents[GetOrdValue <> 0];
end;

procedure TBoolProperty.GetValues(Proc: TGetStrProc);
begin
  Proc(BooleanIdents[False]);
  Proc(BooleanIdents[True]);
end;

procedure TBoolProperty.SetValue(const Value: string);
var
  I: Integer;
begin
  if SameText(Value, BooleanIdents[False]) then
    I := 0
  else if SameText(Value, BooleanIdents[True]) then
    I := -1
  else
    I := StrToInt(Value);
  SetOrdValue(I);
end;


function TInt64Property.AllEqual: Boolean;
var
  I: Integer;
  V: Int64;
begin
```

```
  Result := False;
  if PropCount > 1 then
  begin
    V := GetInt64Value;
    for I := 1 to PropCount - 1 do
      if GetInt64ValueAt(I) <> V then Exit;
  end;
  Result := True;
end;

function TInt64Property.GetEditLimit: Integer;
begin
  Result := 63;
end;

function TInt64Property.GetValue: string;
begin
  Result := IntToStr(GetInt64Value);
end;

procedure TInt64Property.SetValue(const Value: string);
begin
  SetInt64Value(StrToInt64(Value));
end;

function TFloatProperty.AllEqual: Boolean;
var
  I: Integer;
  V: Extended;
begin
  Result := False;
  if PropCount > 1 then
  begin
    V := GetFloatValue;
    for I := 1 to PropCount - 1 do
      if GetFloatValueAt(I) <> V then Exit;
  end;
  Result := True;
end;

function TFloatProperty.GetValue: string;
const
  Precisions: array[TFloatType] of Integer = (7, 15, 18, 18, 18);
begin
  Result := FloatToStrF(GetFloatValue, ffGeneral,
    Precisions[GetTypeData(GetPropType)^.FloatType], 0);
end;

procedure TFloatProperty.SetValue(const Value: string);
begin
  SetFloatValue(StrToFloat(Value));
end;

function TStringProperty.AllEqual: Boolean;
var
  I: Integer;
  V: string;
begin
  Result := False;
  if PropCount > 1 then
  begin
    V := GetStrValue;
    for I := 1 to PropCount - 1 do
      if GetStrValueAt(I) <> V then Exit;
  end;
  Result := True;
end;

function TStringProperty.GetEditLimit: Integer;
```

```
  begin
    if GetPropType^.Kind = tkString then
      Result := GetTypeData(GetPropType)^.MaxLength
    else
      Result := inherited GetEditLimit;
  end;

  function TStringProperty.GetValue: string;
  begin
    Result := GetStrValue;
  end;

  procedure TStringProperty.SetValue(const Value: string);
  begin
    SetStrValue(Value);
  end;

  function TComponentNameProperty.GetAttributes: TPropertyAttributes;
  begin
    Result := [paNotNestable];
  end;

  function TComponentNameProperty.GetEditLimit: Integer;
  begin
    Result := MaxIdentLength;
  end;

  constructor TNestedProperty.Create(Parent: TPropertyEditor);
  begin
    FDesigner := Parent.Designer;
    FPropList := Parent.FPropList;
    FPropCount := Parent.PropCount;
  end;

  destructor TNestedProperty.Destroy;
  begin
  end;

  constructor TSetTeplMerElementProperty.Create(Parent: TPropertyEditor;
AElement: Integer);
  begin
    inherited Create(Parent);
    FElement := AElement;
  end;

  function TSetTeplMerElementProperty.AllEqual: Boolean;
  var
    I: Integer;
    S: TIntegerSet;
    V: Boolean;
  begin
    Result := False;
    if PropCount > 1 then
    begin
      Integer(S) := GetOrdValue;
      V := FElement in S;
      for I := 1 to PropCount - 1 do
      begin
        Integer(S) := GetOrdValueAt(I);
        if (FElement in S) <> V then Exit;
      end;
    end;
    Result := True;
  end;

  function TSetTeplMerElementProperty.GetAttributes: TPropertyAttributes;
  begin
    Result := [paMultiSelect, paValueList, paSortList];
  end;
```

```
function TSetTeplMerElementProperty.GetName: string;
begin
  Result := GetEnumName(GetTypeData(GetPropType)^.CompType^, FElement);
end;

function TSetTeplMerElementProperty.GetValue: string;
var
  S: TIntegerSet;
begin
  Integer(S) := GetOrdValue;
  Result := BooleanIdents[FElement in S];
end;

procedure TSetTeplMerElementProperty.GetValues(Proc: TGetStrProc);
begin
  Proc(BooleanIdents[False]);
  Proc(BooleanIdents[True]);
end;

procedure TSetTeplMerElementProperty.SetValue(const Value: string);
var
  S: TIntegerSet;
begin
  Integer(S) := GetOrdValue;
  if CompareText(Value, BooleanIdents[True]) = 0 then
    Include(S, FElement)
  else
    Exclude(S, FElement);
  SetOrdValue(Integer(S));
end;

function TSetTeplMerElementProperty.GetIsDefault: Boolean;
var
  S: TIntegerSet;
  ShouldBeInSet: Boolean;
  HasStoredProc: Integer;
  ProcAsInt: Integer;
begin
  Result := inherited GetIsDefault;
  if not Result then
  begin
    ProcAsInt := Integer(PPropInfo(GetPropInfo)^.StoredProc);
    HasStoredProc := ProcAsInt and $FFFFFF00;
    if HasStoredProc = 0 then
    begin
      Integer(S) := PPropInfo(GetPropInfo)^.Default;
      ShouldBeInSet := FElement in S;
      Integer(S) := GetOrdValue;
      if ShouldBeInSet then
        Result := FElement in S
      else
        Result := not (FElement in S);
    end;
  end;
end;

function TSetProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paSubProperties, paReadOnly, paRevertable];
end;

procedure TSetProperty.GetProperties(Proc: TGetPropProc);
var
  I: Integer;
begin
  with GetTypeData(GetTypeData(GetPropType)^.CompType^)^ do
    for I := MinValue to MaxValue do
      Proc(TSetTeplMerElementProperty.Create(Self, I));
```

```
end;

function TSetProperty.GetValue: string;
var
  S: TIntegerSet;
  TypeInfo: PTypeInfo;
  I: Integer;
begin
  Integer(S) := GetOrdValue;
  TypeInfo := GetTypeData(GetPropType)^.CompType^;
  Result := '[';
  for I := 0 to SizeOf(Integer) * 8 - 1 do
    if I in S then
    begin
      if Length(Result) <> 1 then Result := Result + ',';
      Result := Result + GetEnumName(TypeInfo, I);
    end;
  Result := Result + ']';
end;

function TClassProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paSubProperties, paReadOnly];
end;

procedure TClassProperty.GetProperties(Proc: TGetPropProc);
var
  I: Integer;
  J: Integer;
  Components: IDesignerSelections;
begin
  Components := TDesignerSelections.Create;
  for I := 0 to PropCount - 1 do
  begin
    J := GetOrdValueAt(I);
    if J <> 0 then
      Components.Add(TComponent(GetOrdValueAt(I)));
  end;
  if Components.Count > 0 then
    GetComponentProperties(Components, tkProperties, Designer, Proc);
end;

function TClassProperty.GetValue: string;
begin
  FmtStr(Result, '(%s)', [GetPropType^.Name]);
end;

procedure TComponentProperty.Edit;
var
  Temp: TComponent;
begin
  if (Designer.GetShiftState * [ssCtrl, ssLeft] = [ssCtrl, ssLeft]) then
  begin
    Temp := GetComponentReference;
    if Temp <> nil then
      Designer.SelectComponent(Temp)
    else
      inherited Edit;
  end
  else
    inherited Edit;
end;

function TComponentProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect];
  if Assigned(GetPropInfo^.SetProc) then
    Result := Result + [paValueList, paSortList, paRevertable]
  else
```

```
    Result := Result + [paReadOnly];
  if GReferenceExpandable and (GetComponentReference <> nil) and AllEqual then
    Result := Result + [paSubProperties, paVolatileSubProperties];
end;

function TComponentProperty.GetSelections: IDesignerSelections;
var
  I: Integer;
begin
  Result := nil;
  if (GetComponentReference <> nil) and AllEqual then
  begin
    Result := TDesignerSelections.Create;
    for I := 0 to PropCount - 1 do
      Result.Add(TComponent(GetOrdValueAt(I)));
  end;
end;

procedure TComponentProperty.GetProperties(Proc: TGetPropProc);
var
  LComponents: IDesignerSelections;
  LDesigner: IDesigner;
begin
  LComponents := GetSelections;
  if LComponents <> nil then
  begin
    if not Supports(FindRootDesigner(LComponents[0]), IDesigner, LDesigner)
then
      LDesigner := Designer;
    GetComponentProperties(LComponents, tkAny, LDesigner, Proc, FilterFunc);
  end;
end;

function TComponentProperty.GetEditLimit: Integer;
begin
  Result := 127;
end;

function TComponentProperty.GetValue: string;
begin
  Result := Designer.GetComponentName(GetComponentReference);
end;

procedure TComponentProperty.GetValues(Proc: TGetStrProc);
begin
  Designer.GetComponentNames(GetTypeData(GetPropType), Proc);
end;

procedure TComponentProperty.SetValue(const Value: string);
var
  Component: TComponent;
begin
  if Value = '' then
    Component := nil
  else
  begin
    Component := Designer.GetComponent(Value);
    if not (Component is GetTypeData(GetPropType)^.ClassType) then
      raise EPropertyError.CreateRes(@SInvalidPropertyValue);
  end;
  SetOrdValue(LongInt(Component));
end;

function TComponentProperty.AllEqual: Boolean;
var
  I: Integer;
  LInstance: TComponent;
begin
  Result := False;
```

```
  LInstance := TComponent(GetOrdValue);
  if PropCount > 1 then
    for I := 1 to PropCount - 1 do
      if TComponent(GetOrdValueAt(I)) <> LInstance then
        Exit;
  Result := Supports(FindRootDesigner(LInstance), IDesigner);
end;

function TComponentProperty.GetComponentReference: TComponent;
begin
  Result := TComponent(GetOrdValue);
end;

function TComponentProperty.FilterFunc(const ATestEditor: IProperty): Boolean;
begin
  Result := not (paNotNestable in ATestEditor.GetAttributes);
end;

function TInterfaceProperty.AllEqual: Boolean;
var
  I: Integer;
  LInterface: IInterface;
begin
  Result := False;
  LInterface := GetIntfValue;
  if PropCount > 1 then
    for I := 1 to PropCount - 1 do
      if GetIntfValueAt(I) <> LInterface then
        Exit;
  Result := Supports(FindRootDesigner(GetComponent(LInterface)), IDesigner);
end;

function TInterfaceProperty.GetComponent(const AInterface: IInterface):
TComponent;
var
  ICR: IInterfaceComponentReference;
begin
  if (AInterface <> nil) and
     Supports(AInterface, IInterfaceComponentReference, ICR) then
    Result := ICR.GetComponent
  else
    Result := nil;
end;

function TInterfaceProperty.GetComponentReference: TComponent;
begin
  Result := GetComponent(GetIntfValue);
end;

function TInterfaceProperty.GetSelections: IDesignerSelections;
var
  I: Integer;
begin
  Result := nil;
  if (GetIntfValue <> nil) and AllEqual then
  begin
    Result := TDesignerSelections.Create;
    for I := 0 to PropCount - 1 do
      Result.Add(GetComponent(GetIntfValueAt(I)));
  end;
end;

procedure TInterfaceProperty.ReceiveComponentNames(const S: string);
var
  Temp: TComponent;
  Intf: IInterface;
begin
  Temp := Designer.GetComponent(S);
  if Assigned(FGetValuesStrProc) and
```

```
      Assigned(Temp) and
      Supports(TObject(Temp), GetTypeData(GetPropType)^.Guid, Intf) then
     FGetValuesStrProc(S);
 end;

 procedure TInterfaceProperty.GetValues(Proc: TGetStrProc);
 begin
   FGetValuesStrProc := Proc;
   try
     Designer.GetComponentNames(GetTypeData(TypeInfo(TComponent)),
ReceiveComponentNames);
   finally
     FGetValuesStrProc := nil;
   end;
 end;

 procedure TInterfaceProperty.SetValue(const Value: string);
 var
   Intf: IInterface;
   Component: TComponent;
 begin
   if Value = '' then
     Intf := nil
   else
   begin
     Component := Designer.GetComponent(Value);
     if (Component = nil) or
       not Supports(TObject(Component), GetTypeData(GetPropType)^.Guid, Intf)
then
       raise EPropertyError.CreateRes(@SInvalidPropertyValue);
   end;
   SetIntfValue(Intf);
 end;

 function TMethodProperty.AllEqual: Boolean;
 var
   I: Integer;
   V, T: TMethod;
 begin
   Result := False;
   if PropCount > 1 then
   begin
     V := GetMethodValue;
     for I := 1 to PropCount - 1 do
     begin
       T := GetMethodValueAt(I);
       if (T.Code <> V.Code) or (T.Data <> V.Data) then Exit;
     end;
   end;
   Result := True;
 end;

 function TMethodProperty.AllNamed: Boolean;
 var
   I: Integer;
 begin
   Result := True;
   for I := 0 to PropCount - 1 do
     if GetComponent(I).GetNamePath = '' then
     begin
       Result := False;
       Break;
     end;
 end;

 procedure TMethodProperty.Edit;
 var
   FormMethodName: string;
 begin
```

```
  if not AllNamed then
    raise EPropertyError.CreateRes(@SCannotCreateName);
  FormMethodName := GetValue;
  if (FormMethodName = '') or
    Designer.MethodFromAncestor(GetMethodValue) then
  begin
    if FormMethodName = '' then
      FormMethodName := GetFormMethodName;
    if FormMethodName = '' then
      raise EPropertyError.CreateRes(@SCannotCreateName);
    SetValue(FormMethodName);
  end;
  Designer.ShowMethod(FormMethodName);
end;

function TMethodProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paValueList, paSortList, paRevertable];
end;

function TMethodProperty.GetEditLimit: Integer;
begin
  Result := MaxIdentLength;
end;

function TMethodProperty.GetFormMethodName: string;
var
  I: Integer;
begin
  if GetComponent(0) = Designer.GetRoot then
  begin
    Result := Designer.GetRootClassName;
    if (Result <> '') and (Result[1] = 'T') then
      Delete(Result, 1, 1);
  end
  else
  begin
    Result := Designer.GetObjectName(GetComponent(0));
    for I := Length(Result) downto 1 do
      if Result[I] in ['.', '[', ']', '-', '>'] then
        Delete(Result, I, 1);
  end;
  if Result = '' then
    raise EPropertyError.CreateRes(@SCannotCreateName);
  Result := Result + GetTrimmedEventName;
end;

function TMethodProperty.GetTrimmedEventName: string;
begin
  Result := GetName;
  if (Length(Result) >= 2) and
    (Result[1] in ['O', 'o']) and (Result[2] in ['N', 'n']) then
    Delete(Result,1,2);
end;

function TMethodProperty.GetValue: string;
begin
  Result := Designer.GetMethodName(GetMethodValue);
end;

procedure TMethodProperty.GetValues(Proc: TGetStrProc);
begin
  Designer.GetMethods(GetTypeData(GetPropType), Proc);
end;

procedure TMethodProperty.SetValue(const AValue: string);

  procedure CheckChainCall(const MethodName: string; Method: TMethod);
  var
```

```
      Persistent: TPersistent;
      Component: TComponent;
      InstanceMethod: string;
      Instance: TComponent;
    begin
      Persistent := GetComponent(0);
      if Persistent is TComponent then
      begin
        Component := TComponent(Persistent);
        if (Component.Name <> '') and (Method.Data <> Designer.GetRoot) and
          (TObject(Method.Data) is TComponent) then
        begin
          Instance := TComponent(Method.Data);
          InstanceMethod := Instance.MethodName(Method.Code);
          if InstanceMethod <> '' then
            Designer.ChainCall(MethodName, Instance.Name, InstanceMethod,
              GetTypeData(GetPropType));
        end;
      end;
    end;

  var
    NewMethod: Boolean;
    CurValue: string;
    OldMethod: TMethod;
  begin
    if not AllNamed then
      raise EPropertyError.CreateRes(@SCannotCreateName);
    CurValue:= GetValue;
    if (CurValue <> '') and (AValue <> '') and (SameText(CurValue, AValue) or
      not Designer.MethodExists(AValue)) and not
Designer.MethodFromAncestor(GetMethodValue) then
      Designer.RenameMethod(CurValue, AValue)
    else
    begin
      NewMethod := (AValue <> '') and not Designer.MethodExists(AValue);
      OldMethod := GetMethodValue;
      SetMethodValue(Designer.CreateMethod(AValue, GetTypeData(GetPropType)));
      if NewMethod then
      begin
        if (PropCount = 1) and (OldMethod.Data <> nil) and (OldMethod.Code <>
nil) then
          CheckChainCall(AValue, OldMethod);
        Designer.ShowMethod(AValue);
      end;
    end;
  end;

  function TDateProperty.GetAttributes: TPropertyAttributes;
  begin
    Result := [paMultiSelect, paRevertable];
  end;

  function TDateProperty.GetValue: string;
  var
    DT: TDateTime;
  begin
    DT := GetFloatValue;
    if DT = 0.0 then Result := '' else
    Result := DateToStr(DT);
  end;

  procedure TDateProperty.SetValue(const Value: string);
  var
    DT: TDateTime;
  begin
    if Value = '' then DT := 0.0
    else DT := StrToDate(Value);
    SetFloatValue(DT);
```

```
end;

function TTimeProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paRevertable];
end;

function TTimeProperty.GetValue: string;
var
  DT: TDateTime;
begin
  DT := GetFloatValue;
  if DT = 0.0 then Result := '' else
  Result := TimeToStr(DT);
end;

procedure TTimeProperty.SetValue(const Value: string);
var
  DT: TDateTime;
begin
  if Value = '' then DT := 0.0
  else DT := StrToTime(Value);
  SetFloatValue(DT);
end;

function TDateTimeProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paRevertable];
end;

function TDateTimeProperty.GetValue: string;
var
  DT: TDateTime;
begin
  DT := GetFloatValue;
  if DT = 0.0 then Result := '' else
  Result := DateTimeToStr(DT);
end;

procedure TDateTimeProperty.SetValue(const Value: string);
var
  DT: TDateTime;
begin
  if Value = '' then DT := 0.0
  else DT := StrToDateTime(Value);
  SetFloatValue(DT);
end;

type
  TPropInfoList = class
  private
    FList: PPropList;
    FCount: Integer;
    FSize: Integer;
    function Get(Index: Integer): PPropInfo;
  public
    constructor Create(Instance: TPersistent; Filter: TTypeKinds);
    destructor Destroy; override;
    function Contains(P: PPropInfo): Boolean;
    procedure Delete(Index: Integer);
    procedure Intersect(List: TPropInfoList);
    property Count: Integer read FCount;
    property Items[Index: Integer]: PPropInfo read Get; default;
  end;

constructor TPropInfoList.Create(Instance: TPersistent; Filter: TTypeKinds);
begin
  FCount := GetPropList(Instance.ClassInfo, Filter, nil);
  FSize := FCount * SizeOf(Pointer);
```

```
  GetMem(FList, FSize);
  GetPropList(Instance.ClassInfo, Filter, FList);
end;

destructor TPropInfoList.Destroy;
begin
  if FList <> nil then FreeMem(FList, FSize);
end;

function TPropInfoList.Contains(P: PPropInfo): Boolean;
var
  I: Integer;
begin
  for I := 0 to FCount - 1 do
    with FList^[I]^ do
      if (PropType^ = P^.PropType^) and (CompareText(Name, P^.Name) = 0) then
      begin
        Result := True;
        Exit;
      end;
  Result := False;
end;

procedure TPropInfoList.Delete(Index: Integer);
begin
  Dec(FCount);
  if Index < FCount then
    Move(FList^[Index + 1], FList^[Index],
      (FCount - Index) * SizeOf(Pointer));
end;

function TPropInfoList.Get(Index: Integer): PPropInfo;
begin
  Result := FList^[Index];
end;

procedure TPropInfoList.Intersect(List: TPropInfoList);
var
  I: Integer;
begin
  for I := FCount - 1 downto 0 do
    if not List.Contains(FList^[I]) then Delete(I);
end;

function InterfaceInheritsFrom(Child, Parent: PTypeData): Boolean;
begin
  while (Child <> nil) and (Child <> Parent) and (Child^.IntfParent <> nil) do
    Child := GetTypeData(Child^.IntfParent^);
  Result := (Child <> nil) and (Child = Parent);
end;

type
  PPropertyClassRec = ^TPropertyClassRec;
  TPropertyClassRec = record
    Group: Integer;
    PropertyType: PTypeInfo;
    PropertyName: string;
    ComponentClass: TClass;
    ClassGroup: TPersistentClass;
    EditorClass: TPropertyEditorClass;
  end;

  PPropertyMapperRec = ^TPropertyMapperRec;
  TPropertyMapperRec = record
    Group: Integer;
    Mapper: TPropertyMapperFunc;
  end;

var
```

```
    PropertyClassList: TList;
    PropertyMapperList: TList = nil;

procedure RegisterPropertyEditor(PropertyType: PTypeInfo; ComponentClass:
TClass;
    const PropertyName: string; EditorClass: TPropertyEditorClass);
 var
   P: PPropertyClassRec;
 begin
   if PropertyClassList = nil then
     PropertyClassList := TList.Create;
   New(P);
   P.Group := CurrentGroup;
   P.PropertyType := PropertyType;
   P.ComponentClass := ComponentClass;
   P.PropertyName := '';
   P.ClassGroup := nil;
   if Assigned(ComponentClass) then P^.PropertyName := PropertyName;
   P.EditorClass := EditorClass;
   PropertyClassList.Insert(0, P);
 end;

 procedure SetPropertyEditorGroup(EditorClass: TPropertyEditorClass;
   GroupClass: TPersistentClass);
 var
   P: PPropertyClassRec;
   I: Integer;
 begin
   for I := 0 to PropertyClassList.Count - 1 do
   begin
     P := PropertyClassList[I];
     if P^.EditorClass = EditorClass then
     begin
       P^.ClassGroup := ClassGroupOf(GroupClass);
       Exit;
     end;
   end;
 end;

 function GetEditorClass(PropInfo: PPropInfo;
   Obj: TPersistent): TPropertyEditorClass;
 var
   PropType: PTypeInfo;
   P, C: PPropertyClassRec;
   I: Integer;
 begin
   if PropertyMapperList <> nil then
   begin
     for I := 0 to PropertyMapperList.Count -1 do
      with PPropertyMapperRec(PropertyMapperList[I])^ do
      begin
       Result := Mapper(Obj, PropInfo);
       if Result <> nil then Exit;
      end;
   end;
   PropType := PropInfo^.PropType^;
   I := 0;
   C := nil;
   while I < PropertyClassList.Count do
   begin
     P := PropertyClassList[I];

     if ( (P^.PropertyType = PropType) or
          ((P^.PropertyType^.Kind = PropType.Kind) and
           (P^.PropertyType^.Name = PropType.Name)
          )
        ) or
        ( (PropType^.Kind = tkClass) and
          (P^.PropertyType^.Kind = tkClass) and
```

```
GetTypeData(PropType)^.ClassType.InheritsFrom(GetTypeData(P^.PropertyType)^.Clas
sType)
        ) or
        ( (PropType^.Kind = tkInterface) and
          (P^.PropertyType^.Kind = tkInterface) and
          InterfaceInheritsFrom(GetTypeData(PropType),
GetTypeData(P^.PropertyType))
        ) then
       if ((P^.ComponentClass = nil) or (Obj.InheritsFrom(P^.ComponentClass)))
and
          ((P^.ClassGroup = nil) or (P^.ClassGroup = ClassGroupOf(Obj))) and
          ((P^.PropertyName = '') or (CompareText(PropInfo^.Name,
P^.PropertyName) = 0)) then
         if (C = nil) or
            ((C^.ComponentClass = nil) and (P^.ComponentClass <> nil)) or
            ((C^.PropertyName = '') and (P^.PropertyName <> ''))
            or
            ((C^.PropertyType <> PropType) and (P^.PropertyType = PropType))
            or
            ( (P^.PropertyType <> C^.PropertyType) and
              ( (
                  (P^.PropertyType^.Kind = tkClass) and
                  (C^.PropertyType^.Kind = tkClass) and
                  GetTypeData(P^.PropertyType)^.ClassType.InheritsFrom(
                    GetTypeData(C^.PropertyType)^.ClassType)
                ) or
                ( (P^.PropertyType^.Kind = tkInterface) and
                  (C^.PropertyType^.Kind = tkInterface) and
                  InterfaceInheritsFrom(GetTypeData(P^.PropertyType),
GetTypeData(C^.PropertyType))
                )
              )
            ) or
            ( (P^.ComponentClass <> nil) and (C^.ComponentClass <> nil) and
              (P^.ComponentClass <> C^.ComponentClass) and
              (P^.ComponentClass.InheritsFrom(C^.ComponentClass))
            ) then
           C := P;
     Inc(I);
   end;
   if C <> nil then
     Result := C^.EditorClass else
     Result := PropClassMap[PropType^.Kind];
 end;

 procedure GetComponentProperties(const Components: IDesignerSelections;
   Filter: TTypeKinds; const Designer: IDesigner; Proc: TGetPropProc;
   EditorFilterFunc: TPropertyEditorFilterFunc);
 var
   I, J, CompCount: Integer;
   CompType: TClass;
   Candidates: TPropInfoList;
   PropLists: TList;
   EditorInstance: TBasePropertyEditor;
   Editor: IProperty;
   EdClass: TPropertyEditorClass;
   PropInfo: PPropInfo;
   AddEditor: Boolean;
   Obj: TPersistent;
 begin
   if (Components = nil) or (Components.Count = 0) then Exit;
   CompCount := Components.Count;
   Obj := Components[0];
   CompType := Components[0].ClassType;
   Candidates := TPropInfoList.Create(Components[0], Filter);
   try
     for I := Candidates.Count - 1 downto 0 do
     begin
```

```
        PropInfo := Candidates[I];
        EdClass := GetEditorClass(PropInfo, Obj);
        if EdClass = nil then
          Candidates.Delete(I)
        else
        begin
          EditorInstance := EdClass.Create(Designer, 1);
          Editor := EditorInstance as IProperty;
          TPropertyEditor(EditorInstance).SetPropEntry(0, Components[0],
PropInfo);
          TPropertyEditor(EditorInstance).Initialize;
          with PropInfo^ do
            if (GetProc = nil) or
               (not GShowReadOnlyProps and
                ((PropType^.Kind <> tkClass) and
                 (SetProc = nil))) or
               ((CompCount > 1) and
                not (paMultiSelect in Editor.GetAttributes)) or
               not Editor.ValueAvailable or
               (Assigned(EditorFilterFunc) and not EditorFilterFunc(Editor))
then
              Candidates.Delete(I);
        end;
      end;
      PropLists := TList.Create;
      try
        PropLists.Capacity := CompCount;
        for I := 0 to CompCount - 1 do
          PropLists.Add(TPropInfoList.Create(Components[I], Filter));
        for I := 0 to CompCount - 1 do
          Candidates.Intersect(TPropInfoList(PropLists[I]));
        for I := 0 to CompCount - 1 do
          TPropInfoList(PropLists[I]).Intersect(Candidates);
        for I := 0 to Candidates.Count - 1 do
        begin
          EdClass := GetEditorClass(Candidates[I], Obj);
          if EdClass = nil then Continue;
          EditorInstance := EdClass.Create(Designer, CompCount);
          Editor := EditorInstance as IProperty;
          AddEditor := True;
          for J := 0 to CompCount - 1 do
          begin
            if (Components[J].ClassType <> CompType) and
              (GetEditorClass(TPropInfoList(PropLists[J])[I],
                Components[J]) <> EdClass) then
            begin
              AddEditor := False;
              Break;
            end;
            TPropertyEditor(EditorInstance).SetPropEntry(J, Components[J],
              TPropInfoList(PropLists[J])[I]);
          end;
          if AddEditor then
          begin
            TPropertyEditor(EditorInstance).Initialize;
            if Editor.ValueAvailable then Proc(Editor);
          end;
        end;
      finally
        for I := 0 to PropLists.Count - 1 do TPropInfoList(PropLists[I]).Free;
        PropLists.Free;
      end;
    finally
      Candidates.Free;
    end;
  end;

  procedure RegisterPropertyMapper(Mapper: TPropertyMapperFunc);
  var
```

```
    P: PPropertyMapperRec;
  begin
    if PropertyMapperList = nil then
      PropertyMapperList := TList.Create;
    New(P);
    P^.Group := CurrentGroup;
    P^.Mapper := Mapper;
    PropertyMapperList.Insert(0, P);
  end;

  constructor TTeplMerComponentEditor.Create(AComponent: TComponent; ADesigner:
IDesigner);
  begin
    inherited Create(AComponent, ADesigner);
    FComponent := AComponent;
    FDesigner := ADesigner;
  end;

  procedure TTeplMerComponentEditor.Edit;
  begin
    if GetVerbCount > 0 then ExecuteVerb(0);
  end;

  function TTeplMerComponentEditor.GetComponent: TComponent;
  begin
    Result := FComponent;
  end;

  function TTeplMerComponentEditor.GetDesigner: IDesigner;
  begin
    Result := FDesigner;
  end;

  function TTeplMerComponentEditor.GetVerbCount: Integer;
  begin
    Result := 0;
  end;

  function TTeplMerComponentEditor.IsInInlined: Boolean;
  begin
    Result := csInline in Component.Owner.ComponentState;
  end;

  procedure TDefaultEditor.CheckEdit(const Prop: IProperty);
  begin
    if FContinue then
      EditProperty(Prop, FContinue);
  end;

  procedure TDefaultEditor.EditProperty(const Prop: IProperty;
    var Continue: Boolean);
  var
    PropName: string;
    BestName: string;
    MethodProperty: IMethodProperty;

    procedure ReplaceBest;
    begin
      FBest := Prop;
      if FFirst = FBest then FFirst := nil;
    end;

  begin
    if not Assigned(FFirst) and
      Supports(Prop, IMethodProperty, MethodProperty) then
      FFirst := Prop;
    PropName := Prop.GetName;
    BestName := '';
    if Assigned(FBest) then BestName := FBest.GetName;
```

```
  if CompareText(PropName, 'ONCREATE') = 0 then
    ReplaceBest
  else if CompareText(BestName, 'ONCREATE') <> 0 then
    if CompareText(PropName, 'ONCHANGE') = 0 then
      ReplaceBest
    else if CompareText(BestName, 'ONCHANGE') <> 0 then
      if CompareText(PropName, 'ONCHANGED') = 0 then
        ReplaceBest
      else if CompareText(BestName, 'ONCHANGED') <> 0 then
        if CompareText(PropName, 'ONCLICK') = 0 then
          ReplaceBest;
end;

procedure TDefaultEditor.Edit;
var
  Components: IDesignerSelections;
begin
  Components := TDesignerSelections.Create;
  FContinue := True;
  Components.Add(Component);
  FFirst := nil;
  FBest := nil;
  try
    GetComponentProperties(Components, tkAny, Designer, CheckEdit);
    if FContinue then
      if Assigned(FBest) then
        FBest.Edit
      else if Assigned(FFirst) then
        FFirst.Edit;
  finally
    FFirst := nil;
    FBest := nil;
  end;
end;

type
  PComponentClassRec = ^TComponentClassRec;
  TComponentClassRec = record
    Group: Integer;
    ComponentClass: TComponentClass;
    EditorClass: TTeplMerComponentEditorClass;
  end;

var
  ComponentClassList: TList = nil;

procedure RegisterComponentEditor(ComponentClass: TComponentClass;
  ComponentEditor: TTeplMerComponentEditorClass);
var
  P: PComponentClassRec;
begin
  if ComponentClassList = nil then
    ComponentClassList := TList.Create;
  New(P);
  P.Group := CurrentGroup;
  P.ComponentClass := ComponentClass;
  P.EditorClass := ComponentEditor;
  ComponentClassList.Insert(0, P);
end;

function GeTTeplMerComponentEditor(Component: TComponent;
  const Designer: IDesigner): IComponentEditor;
var
  P: PComponentClassRec;
  I: Integer;
  ComponentClass: TComponentClass;
  EditorClass: TTeplMerComponentEditorClass;
begin
  ComponentClass := TComponentClass(TPersistent);
```

```
  EditorClass := TDefaultEditor;
  if ComponentClassList <> nil then
    for I := 0 to ComponentClassList.Count-1 do
    begin
      P := ComponentClassList[I];
      if (Component is P^.ComponentClass) and
        (P^.ComponentClass <> ComponentClass) and
        (P^.ComponentClass.InheritsFrom(ComponentClass)) then
      begin
        EditorClass := P^.EditorClass;
        ComponentClass := P^.ComponentClass;
      end;
    end;
  Result := EditorClass.Create(Component, Designer) as IComponentEditor;
end;


constructor TeplMerSelectionEditor.Create(const ADesigner: IDesigner);
begin
  inherited Create(ADesigner);
  FDesigner := ADesigner;
end;

function TeplMerSelectionEditor.GetVerbCount: Integer;
begin
  Result := 0;
end;

type
  TeplMerSelectionEditorList = class(TInterfacedObject, ISelectionEditorList)
  private
    FList: IInterfaceList;
  protected
    procedure Add(AEditor: ISelectionEditor);
  public
    constructor Create;
    function Get(Index: Integer): ISelectionEditor;
    function GetCount: Integer;
    property Count: Integer read GetCount;
    property Items[Index: Integer]: ISelectionEditor read Get; default;
  end;

procedure TeplMerSelectionEditorList.Add(AEditor: ISelectionEditor);
begin
  FList.Add(AEditor);
end;

constructor TeplMerSelectionEditorList.Create;
begin
  inherited;
  FList := TInterfaceList.Create;
end;

function TeplMerSelectionEditorList.Get(Index: Integer): ISelectionEditor;
begin
  Result := FList[Index] as ISelectionEditor;
end;

function TeplMerSelectionEditorList.GetCount: Integer;
begin
  Result := FList.Count;
end;

type
  TeplMerSelectionEditorDefinition = class(TObject)
  private
    FGroup: Integer;
    FClass: TClass;
    FEditor: TeplMerSelectionEditorClass;
```

```
    public
      constructor Create(AClass: TClass; AEditor: TeplMerSelectionEditorClass);
      function Matches(AClass: TClass): Boolean;
      property Editor: TeplMerSelectionEditorClass read FEditor;
    end;

    TeplMerSelectionEditorDefinitionList = class(TObjectList)
    protected
      function GetItem(Index: Integer): TeplMerSelectionEditorDefinition;
      procedure SetItem(Index: Integer; AObject:
TeplMerSelectionEditorDefinition);
      procedure FreeEditorGroup(AGroup: Integer);
    public
      property Items[Index: Integer]: TeplMerSelectionEditorDefinition read
GetItem write SetItem; default;
    end;

  constructor TeplMerSelectionEditorDefinition.Create(AClass: TClass;
    AEditor: TeplMerSelectionEditorClass);
  begin
    inherited Create;
    FGroup := CurrentGroup;
    FClass := AClass;
    FEditor := AEditor;
  end;

  function TeplMerSelectionEditorDefinition.Matches(AClass: TClass): Boolean;
  begin
    Result := AClass.InheritsFrom(FClass);
  end;

  procedure TeplMerSelectionEditorDefinitionList.FreeEditorGroup(AGroup:
Integer);
  var
    I: Integer;
  begin
    for I := Count - 1 downto 0 do
      if Items[I].FGroup = AGroup then
        Delete(I);
  end;

  function TeplMerSelectionEditorDefinitionList.GetItem(Index: Integer):
TeplMerSelectionEditorDefinition;
  begin
    Result := TeplMerSelectionEditorDefinition(inherited Items[Index]);
  end;

  procedure TeplMerSelectionEditorDefinitionList.SetItem(Index: Integer;
    AObject: TeplMerSelectionEditorDefinition);
  begin
    inherited Items[Index] := AObject;
  end;

  var
    SelectionEditorDefinitionList: TeplMerSelectionEditorDefinitionList;

  procedure RegisterSelectionEditor(AClass: TClass; AEditor:
TeplMerSelectionEditorClass);
  begin
    if not Assigned(SelectionEditorDefinitionList) then
      SelectionEditorDefinitionList :=
TeplMerSelectionEditorDefinitionList.Create;

SelectionEditorDefinitionList.Add(TeplMerSelectionEditorDefinition.Create(AClass
, AEditor));
  end;

  function GeTeplMerSelectionEditors(const Designer: IDesigner;
    const Selections: IDesignerSelections): ISelectionEditorList; overload;
```

```
  var
    LList: TeplMerSelectionEditorList;
    I: Integer;
    LCommonClass, LClass: TClass;
  begin
    LList := TeplMerSelectionEditorList.Create;
    Result := LList as ISelectionEditorList;

    if Selections.Count > 0 then
    begin
      LCommonClass := Selections[0].ClassType;
      for I := 1 to Selections.Count - 1 do
      begin
        LClass := Selections[I].ClassType;
        while LCommonClass <> TObject do
          if not LClass.InheritsFrom(LCommonClass) then
            LCommonClass := LCommonClass.ClassParent
          else
            Break;
      end;

for I := 0 to SelectionEditorDefinitionList.Count - 1 do
if SelectionEditorDefinitionList[I].Matches(LCommonClass) then
LList.Add(SelectionEditorDefinitionList[I].Editor.Create(Designer) as
ISelectionEditor);
    end;
  end;

  function GeTeplMerSelectionEditors(const Designer: IDesigner):
ISelectionEditorList;
  var
    LSelections: IDesignerSelections;
  begin
    LSelections := TDesignerSelections.Create;
    Designer.GetSelections(LSelections);
    Result := GeTeplMerSelectionEditors(Designer, LSelections);
  end;

  function GeTeplMerSelectionEditors(const Designer: IDesigner;
    Component: TComponent): ISelectionEditorList;
  var
    LSelections: IDesignerSelections;
  begin
    LSelections := TDesignerSelections.Create;
    LSelections.Add(Component);
    Result := GeTeplMerSelectionEditors(Designer, LSelections);
  end;

  constructor TCustomModule.Create(ARoot: TComponent; const ADesigner:
IDesigner);
  begin
    inherited Create(ARoot, ADesigner);
    FRoot := ARoot;
    FDesigner := ADesigner;
  end;

  destructor TCustomModule.Destroy;
  begin
    FFinder.Free;
    inherited;
  end;

  function TCustomModule.GetAttributes: TCustomModuleAttributes;
  begin
    Result := [];
  end;

  function TCustomModule.GetVerb(Index: Integer): string;
  begin
```

```
    Result := '';
  end;

  function TCustomModule.GetVerbCount: Integer;
  begin
    Result := 0;
  end;

  function TCustomModule.Nestable: Boolean;
  begin
    Result := False;
  end;

  procedure TCustomModule.ValidateComponent(Component: TComponent);
  begin
    if not ValidateComponentClass(TComponentClass(Component.ClassType)) then
raise Exception.CreateResFmt(@sClassNotApplicable, [Component.ClassName]);
  end;

  function TCustomModule.ValidateComponentClass(ComponentClass:
TComponentClass): Boolean;
  var
    Base: TComponent;
  begin
    if FFinder = nil then
    begin
      Base := Root;
      if Base.Owner <> nil then
        Base := Base.Owner;
      FFinder := TClassFinder.Create(TPersistentClass(Base.ClassType));
    end;
    while IsProxyClass(ComponentClass) do
      ComponentClass := TComponentClass(ComponentClass.ClassParent);
    Result := FFinder.GetClass(ComponentClass.ClassName) = ComponentClass;
  end;

  function ClassInheritsFrom(ClassType: TClass; const ClassName: string):
Boolean;
  begin
    Result := True;
    while ClassType <> nil do
      if ClassType.ClassNameIs(ClassName) then
        Exit
      else
        ClassType := ClassType.ClassParent;
    Result := False;
  end;

  function AncestorNameMatches(ClassType: TClass; AncestorClass: TClass):
Boolean;
  begin
    Result := ClassType.InheritsFrom(AncestorClass) or
      not ClassInheritsFrom(ClassType, AncestorClass.ClassName);
  end;

  var
    GroupNotifyList: TList;
    EditorGroupList: TBits;

  function NewEditorGroup: Integer;
  begin
    if EditorGroupList = nil then
      EditorGroupList := TBits.Create;
    CurrentGroup := EditorGroupList.OpenBit;
    EditorGroupList[CurrentGroup] := True;
    Result := CurrentGroup;
  end;

  procedure NotifyGroupChange(AProc: TGroupChangeProc);
```

```
begin
  UnnotifyGroupChange(AProc);
  if not Assigned(GroupNotifyList) then
    GroupNotifyList := TList.Create;
  GroupNotifyList.Add(@AProc);
end;

procedure UnnotifyGroupChange(AProc: TGroupChangeProc);
begin
  if Assigned(GroupNotifyList) then
    GroupNotifyList.Remove(@AProc);
end;

procedure FreeEditorGroup(Group: Integer);
var
  I: Integer;
  P: PPropertyClassRec;
  C: PComponentClassRec;
  M: PPropertyMapperRec;
  SelectionDef: TeplMerSelectionEditorDefinition;
begin
  I := PropertyClassList.Count - 1;
  while I > -1 do
  begin
    P := PropertyClassList[I];
    if P.Group = Group then
    begin
      PropertyClassList.Delete(I);
      Dispose(P);
    end;
    Dec(I);
  end;
  I := ComponentClassList.Count - 1;
  while I > -1 do
  begin
    C := ComponentClassList[I];
    if C.Group = Group then
    begin
      ComponentClassList.Delete(I);
      Dispose(C);
    end;
    Dec(I);
  end;
  if PropertyMapperList <> nil then
    for I := PropertyMapperList.Count-1 downto 0 do
    begin
      M := PropertyMapperList[I];
      if M.Group = Group then
      begin
        PropertyMapperList.Delete(I);
        Dispose(M);
      end;
    end;
  if SelectionEditorDefinitionList <> nil then
    for I := SelectionEditorDefinitionList.Count-1 downto 0 do
    begin
      SelectionDef := SelectionEditorDefinitionList[I];
      if SelectionDef.FGroup = Group then
        SelectionEditorDefinitionList.Delete(I);
    end;
  if Assigned(GroupNotifyList) then
    for I := GroupNotifyList.Count - 1 downto 0 do
      TGroupChangeProc(GroupNotifyList[I])(Group);
  if (Group >= 0) and (Group < EditorGroupList.Size) then
    EditorGroupList[Group] := False;
end;

type
  TVariantTypeProperty = class(TNestedProperty)
```

```
  public
    function AllEqual: Boolean; override;
    function GetAttributes: TPropertyAttributes; override;
    function GetName: string; override;
    function GetValue: string; override;
    procedure GetValues(Proc: TGetStrProc); override;
    procedure SetValue(const Value: string); override;
   end;

function TVariantTypeProperty.AllEqual: Boolean;
var
  i: Integer;
  V1, V2: Variant;
begin
  Result := False;
  if PropCount > 1 then
  begin
    V1 := GetVarValue;
    for i := 1 to PropCount - 1 do
    begin
      V2 := GetVarValueAt(i);
      if VarType(V1) <> VarType(V2) then Exit;
    end;
  end;
  Result := True;
end;

function TVariantTypeProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paValueList, paSortList];
end;

function TVariantTypeProperty.GetName: string;
begin
  Result := 'Type';
end;

function TVariantTypeProperty.GetValue: string;
begin
  case VarType(GetVarValue) and varTypeMask of
    Low(VarTypeNames)..High(VarTypeNames):
      Result := VarTypeNames[VarType(GetVarValue)];
    varString:
      Result := SString;
  else
    Result := SUnknown;
  end;
end;

procedure TVariantTypeProperty.GetValues(Proc: TGetStrProc);
var
  i: Integer;
begin
  for i := 0 to High(VarTypeNames) do
    if VarTypeNames[i] <> '' then
      Proc(VarTypeNames[i]);
  Proc(SString);
end;

procedure TVariantTypeProperty.SetValue(const Value: string);

  function GetSelectedType: Integer;
  var
    i: Integer;
  begin
    Result := -1;
    for i := 0 to High(VarTypeNames) do
      if VarTypeNames[i] = Value then
      begin
```

```
        Result := i;
        break;
      end;
    if (Result = -1) and (Value = SString) then
      Result := varString;
  end;

var
  NewType: Integer;
  V: Variant;
begin
  V := GetVarValue;
  NewType := GetSelectedType;
  case NewType of
    varEmpty: VarClear(V);
    varNull: V := NULL;
    -1: raise Exception.CreateRes(@SUnknownType);
  else
    try
      VarCast(V, V, NewType);
    except
      VarClear(V);
      VarCast(V, V, NewType);
    end;
  end;
  SetVarValue(V);
end;

function TVariantProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paMultiSelect, paSubProperties];
end;

procedure TVariantProperty.GetProperties(Proc: TGetPropProc);
begin
  Proc(TVariantTypeProperty.Create(Self));
end;

function TVariantProperty.GetValue: string;

  function GetVariantStr(const Value: Variant): string;
  begin
    case VarType(Value) of
      varBoolean:
        Result := BooleanIdents[Value = True];
      varCurrency:
        Result := CurrToStr(Value);
    else
      Result := VarToStrDef(Value, SNull);
    end;
  end;

var
  Value: Variant;
begin
  Value := GetVarValue;
  if VarType(Value) <> varDispatch then
    Result := GetVariantStr(Value)
  else
    Result := 'ERROR';
end;

procedure TVariantProperty.SetValue(const Value: string);

  function Cast(var Value: Variant; NewType: Integer): Boolean;
  var
    V2: Variant;
  begin
    Result := True;
```

```
      if NewType = varCurrency then
        Result := AnsiPos(CurrencyString, Value) > 0;
      if Result then
      try
        VarCast(V2, Value, NewType);
        Result := (NewType = varDate) or (VarToStr(V2) = VarToStr(Value));
        if Result then Value := V2;
      except
        Result := False;
      end;
    end;

  var
    V: Variant;
    OldType: Integer;
  begin
    OldType := VarType(GetVarValue);
    V := Value;
    if Value = '' then
      VarClear(V) else
    if (CompareText(Value, SNull) = 0) then
      V := NULL else
    if not Cast(V, OldType) then
      V := Value;
    SetVarValue(V);
  end;

  function TEditActionSelectionEditor.GetVerb(Index: Integer): string;
  begin
    case Index of
      0: Result := sEditSubmenu;
      1: Result := sControlSubmenu;
    end;
  end;

  function TEditActionSelectionEditor.GetVerbCount: Integer;
  begin
    Result := 2;
  end;

  procedure TEditActionSelectionEditor.PrepareItem(Index: Integer; const AItem:
IMenuItem);
  var
    LEditState: TEditState;
  begin
    case Index of
      0:
        with AItem do
        begin
          AddLine;
        end;
      1:
        with AItem do
        begin
          LEditState := GetEditState;
          Visible := esCanZOrder in LEditState;
          AddItem(sToFrontControl, 0, False, True, HandleToFront);
          AddItem(sToBackControl, 0, False, True, HandleToBack);
        end;
    end;
  end;

  procedure TEditActionSelectionEditor.HandleToFront(Sender: TObject);
  begin
    EditAction(eaBringToFront);
  end;

  procedure TEditActionSelectionEditor.HandleToBack(Sender: TObject);
  begin
```

```
  EditAction(eaSendToBack);
end;

procedure TEditActionSelectionEditor.HandleUndo(Sender: TObject);
begin
  EditAction(eaUndo);
end;

end.
```

Файл інтерфейсу

```
unit MyButtons;
// Copyright (C)  //
// Teslia S.V.  //
// 2021, CNTU //
interface
TeplMer
uses Windows, Messages, Classes, Controls, Forms, Graphics, StdCtrls,
  ExtCtrls, CommCtrl;

type
  TButtonLayout = (blGlyphLeft, blGlyphRight, blGlyphTop, blGlyphBottom);
  TButtonState = (bsUp, bsDisabled, bsDown, bsExclusive);
  TButtonStyle = (bsAutoDetect, bsWin31, bsNew);
  TNumGlyphs = 1..4;

  TTeplMerSpeedButtonLink = class;

  TTeplMerSpeedButton = class(TControlActionLink)
  protected
    FClient: TTeplMerSpeedButtonLink;
    procedure AssignClient(AClient: TObject); override;
    function IsCheckedLinked: Boolean; override;
    function IsGroupIndexLinked: Boolean; override;
    procedure SetGroupIndex(Value: Integer); override;
    procedure SetChecked(Value: Boolean); override;
  end;

  TTeplMerSpeedButtonLink = class(TGraphicControl)
  private
    FGroupIndex: Integer;
    FGlyph: Pointer;
    FDown: Boolean;
    FDragging: Boolean;
    FAllowAllUp: Boolean;
    FLayout: TButtonLayout;
    FSpacing: Integer;
    FTransparent: Boolean;
    FMargin: Integer;
    FFlat: Boolean;
    FMouseInControl: Boolean;
    procedure GlyphChanged(Sender: TObject);
    procedure UpdateExclusive;
    function GetGlyph: TBitmap;
    procedure SetGlyph(Value: TBitmap);
    function GetNumGlyphs: TNumGlyphs;
    procedure SetNumGlyphs(Value: TNumGlyphs);
    procedure SetDown(Value: Boolean);
    procedure SetFlat(Value: Boolean);
    procedure SetAllowAllUp(Value: Boolean);
    procedure SetGroupIndex(Value: Integer);
    procedure SetLayout(Value: TButtonLayout);
    procedure SetSpacing(Value: Integer);
    procedure SetTransparent(Value: Boolean);
    procedure SetMargin(Value: Integer);
    procedure UpdateTracking;
    procedure WMLButtonDblClk(var Message: TWMLButtonDown); message
WM_LBUTTONDBLCLK;
    procedure CMEnabledChanged(var Message: TMessage); message
CM_ENABLEDCHANGED;
    procedure CMButtonPressed(var Message: TMessage); message
CM_BUTTONPRESSED;
    procedure CMDialogChar(var Message: TCMDialogChar); message CM_DIALOGCHAR;
    procedure CMFontChanged(var Message: TMessage); message CM_FONTCHANGED;
    procedure CMTextChanged(var Message: TMessage); message CM_TEXTCHANGED;
    procedure CMSysColorChange(var Message: TMessage); message
CM_SYSCOLORCHANGE;
    procedure CMMouseEnter(var Message: TMessage); message CM_MOUSEENTER;
```

```
      procedure CMMouseLeave(var Message: TMessage); message CM_MOUSELEAVE;
    protected
      FState: TButtonState;
      procedure ActionChange(Sender: TObject; CheckDefaults: Boolean); override;
      function GetActionLinkClass: TControlActionLinkClass; override;
      function GetPalette: HPALETTE; override;
      procedure Loaded; override;
      procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
        X, Y: Integer); override;
      procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
      procedure MouseUp(Button: TMouseButton; Shift: TShiftState;
        X, Y: Integer); override;
      procedure Paint; override;
      property MouseInControl: Boolean read FMouseInControl;
    public
      constructor Create(AOwner: TComponent); override;
      destructor Destroy; override;
      procedure Click; override;
    published
      property Action;
      property AllowAllUp: Boolean read FAllowAllUp write SetAllowAllUp default
False;
      property Anchors;
      property BiDiMode;
      property Constraints;
      property GroupIndex: Integer read FGroupIndex write SetGroupIndex default
0;
      property Down: Boolean read FDown write SetDown default False;
      property Caption;
      property Enabled;
      property Flat: Boolean read FFlat write SetFlat default False;
      property Font;
      property Glyph: TBitmap read GetGlyph write SetGlyph;
      property Layout: TButtonLayout read FLayout write SetLayout default
blGlyphLeft;
      property Margin: Integer read FMargin write SetMargin default -1;
      property NumGlyphs: TNumGlyphs read GetNumGlyphs write SetNumGlyphs
default 1;
      property ParentFont;
      property ParentShowHint;
      property ParentBiDiMode;
      property PopupMenu;
      property ShowHint;
      property Spacing: Integer read FSpacing write SetSpacing default 4;
      property Transparent: Boolean read FTransparent write SetTransparent
default True;
      property Visible;
      property OnClick;
      property OnDblClick;
      property OnMouseDown;
      property OnMouseMove;
      property OnMouseUp;
    end;

    TBitBtnKind = (bkCustom, bkOK, bkCancel, bkHelp, bkYes, bkNo, bkClose,
      bkAbort, bkRetry, bkIgnore, bkAll);

    TBitBtn = class(TButton)
    private
      FCanvas: TCanvas;
      FGlyph: Pointer;
      FStyle: TButtonStyle;
      FKind: TBitBtnKind;
      FLayout: TButtonLayout;
      FSpacing: Integer;
      FMargin: Integer;
      IsFocused: Boolean;
      FModifiedGlyph: Boolean;
      FMouseInControl: Boolean;
```

```
      procedure DrawItem(const DrawItemStruct: TDrawItemStruct);
      procedure SetGlyph(Value: TBitmap);
      function GetGlyph: TBitmap;
      function GetNumGlyphs: TNumGlyphs;
      procedure SetNumGlyphs(Value: TNumGlyphs);
      procedure GlyphChanged(Sender: TObject);
      function IsCustom: Boolean;
      function IsCustomCaption: Boolean;
      procedure SetStyle(Value: TButtonStyle);
      procedure SetKind(Value: TBitBtnKind);
      function GetKind: TBitBtnKind;
      procedure SetLayout(Value: TButtonLayout);
      procedure SetSpacing(Value: Integer);
      procedure SetMargin(Value: Integer);
      procedure CNMeasureItem(var Message: TWMMeasureItem); message
CN_MEASUREITEM;
      procedure CNDrawItem(var Message: TWMDrawItem); message CN_DRAWITEM;
      procedure CMFontChanged(var Message: TMessage); message CM_FONTCHANGED;
      procedure CMEnabledChanged(var Message: TMessage); message
CM_ENABLEDCHANGED;
      procedure CMMouseEnter(var Message: TMessage); message CM_MOUSEENTER;
      procedure CMMouseLeave(var Message: TMessage); message CM_MOUSELEAVE;
      procedure WMLButtonDblClk(var Message: TWMLButtonDblClk);
        message WM_LBUTTONDBLCLK;
    protected
      procedure ActionChange(Sender: TObject; CheckDefaults: Boolean); override;
      procedure CreateHandle; override;
      procedure CreateParams(var Params: TCreateParams); override;
      function GetPalette: HPALETTE; override;
      procedure SetButtonStyle(ADefault: Boolean); override;
    public
      constructor Create(AOwner: TComponent); override;
      destructor Destroy; override;
      procedure Click; override;
    published
      property Action;
      property Anchors;
      property BiDiMode;
      property Cancel stored IsCustom;
      property Caption stored IsCustomCaption;
      property Constraints;
      property Default stored IsCustom;
      property Enabled;
      property Glyph: TBitmap read GetGlyph write SetGlyph stored IsCustom;
      property Kind: TBitBtnKind read GetKind write SetKind default bkCustom;
      property Layout: TButtonLayout read FLayout write SetLayout default
blGlyphLeft;
      property Margin: Integer read FMargin write SetMargin default -1;
      property ModalResult stored IsCustom;
      property NumGlyphs: TNumGlyphs read GetNumGlyphs write SetNumGlyphs stored
IsCustom default 1;
      property ParentShowHint;
      property ParentBiDiMode;
      property ShowHint;
      property Style: TButtonStyle read FStyle write SetStyle default
bsAutoDetect;
      property Spacing: Integer read FSpacing write SetSpacing default 4;
      property TabOrder;
      property TabStop;
      property Visible;
      property OnEnter;
      property OnExit;
    end;

  function DrawButtonFace(Canvas: TCanvas; const Client: TRect;
    BevelWidth: Integer; Style: TButtonStyle; IsRounded, IsDown,
    IsFocused: Boolean): TRect;

  implementation
```

```
uses Consts, SysUtils, ActnList, ImgList, Themes;

{$R Buttons.res}

var
  BitBtnResNames: array[TBitBtnKind] of PChar = (
    nil, 'BBOK', 'BBCANCEL', 'BBHELP', 'BBYES', 'BBNO', 'BBCLOSE',
    'BBABORT', 'BBRETRY', 'BBIGNORE', 'BBALL');
  BitBtnCaptions: array[TBitBtnKind] of Pointer = (
    nil, @SOKButton, @SCancelButton, @SHelpButton, @SYesButton, @SNoButton,
    @SCloseButton, @SAbortButton, @SRetryButton, @SIgnoreButton,
    @SAllButton);
  BitBtnModalResults: array[TBitBtnKind] of TModalResult = (
    0, mrOk, mrCancel, 0, mrYes, mrNo, 0, mrAbort, mrRetry, mrIgnore,
    mrAll);

var
  BitBtnGlyphs: array[TBitBtnKind] of TBitmap;

function DrawButtonFace(Canvas: TCanvas; const Client: TRect;
  BevelWidth: Integer; Style: TButtonStyle; IsRounded, IsDown,
  IsFocused: Boolean): TRect;
var
  NewStyle: Boolean;
  R: TRect;
  DC: THandle;
begin
  NewStyle := ((Style = bsAutoDetect) and NewStyleControls) or (Style =
bsNew);

  R := Client;
  with Canvas do
  begin
    if NewStyle then
    begin
      Brush.Color := clBtnFace;
      Brush.Style := bsSolid;
      DC := Canvas.Handle;

      if IsDown then
      begin
        DrawEdge(DC, R, BDR_SUNKENINNER, BF_TOPLEFT);
        DrawEdge(DC, R, BDR_SUNKENOUTER, BF_BOTTOMRIGHT);
        Dec(R.Bottom);
        Dec(R.Right);
        Inc(R.Top);
        Inc(R.Left);
        DrawEdge(DC, R, BDR_SUNKENOUTER, BF_TOPLEFT or BF_MIDDLE);
      end
      else
      begin
        DrawEdge(DC, R, BDR_RAISEDOUTER, BF_BOTTOMRIGHT);
        Dec(R.Bottom);
        Dec(R.Right);
        DrawEdge(DC, R, BDR_RAISEDINNER, BF_TOPLEFT);
        Inc(R.Top);
        Inc(R.Left);
        DrawEdge(DC, R, BDR_RAISEDINNER, BF_BOTTOMRIGHT or BF_MIDDLE);
      end;
    end
    else
    begin
      Pen.Color := clWindowFrame;
      Brush.Color := clBtnFace;
      Brush.Style := bsSolid;
      Rectangle(R.Left, R.Top, R.Right, R.Bottom);

      if IsRounded then
```

```
      begin
        Pixels[R.Left, R.Top] := clBtnFace;
        Pixels[R.Left, R.Bottom - 1] := clBtnFace;
        Pixels[R.Right - 1, R.Top] := clBtnFace;
        Pixels[R.Right - 1, R.Bottom - 1] := clBtnFace;
      end;

      if IsFocused then
      begin
        InflateRect(R, -1, -1);
        Brush.Style := bsClear;
        Rectangle(R.Left, R.Top, R.Right, R.Bottom);
      end;

      InflateRect(R, -1, -1);
      if not IsDown then
        Frame3D(Canvas, R, clBtnHighlight, clBtnShadow, BevelWidth)
      else
      begin
        Pen.Color := clBtnShadow;
        PolyLine([Point(R.Left, R.Bottom - 1), Point(R.Left, R.Top),
          Point(R.Right, R.Top)]);
      end;
    end;
  end;

  Result := Rect(Client.Left + 1, Client.Top + 1,
    Client.Right - 2, Client.Bottom - 2);
  if IsDown then OffsetRect(Result, 1, 1);
end;

function GetBitBtnGlyph(Kind: TBitBtnKind): TBitmap;
begin
  if BitBtnGlyphs[Kind] = nil then
  begin
    BitBtnGlyphs[Kind] := TBitmap.Create;
    BitBtnGlyphs[Kind].LoadFromResourceName(HInstance, BitBtnResNames[Kind]);
  end;
  Result := BitBtnGlyphs[Kind];
end;

type
  TGlyphList = class(TImageList)
  private
    Used: TBits;
    FCount: Integer;
    function AllocateIndex: Integer;
  public
    constructor CreateSize(AWidth, AHeight: Integer);
    destructor Destroy; override;
    function AddMasked(Image: TBitmap; MaskColor: TColor): Integer;
    procedure Delete(Index: Integer);
    property Count: Integer read FCount;
  end;

  TGlyphCache = class
  private
    GlyphLists: TList;
  public
    constructor Create;
    destructor Destroy; override;
    function GetList(AWidth, AHeight: Integer): TGlyphList;
    procedure ReturnList(List: TGlyphList);
    function Empty: Boolean;
  end;

  TButtonGlyph = class
  private
    FOriginal: TBitmap;
```

```
    FGlyphList: TGlyphList;
    FIndexs: array[TButtonState] of Integer;
    FTransparentColor: TColor;
    FNumGlyphs: TNumGlyphs;
    FOnChange: TNotifyEvent;
    procedure GlyphChanged(Sender: TObject);
    procedure SetGlyph(Value: TBitmap);
    procedure SetNumGlyphs(Value: TNumGlyphs);
    procedure Invalidate;
    function CreateButtonGlyph(State: TButtonState): Integer;
    procedure DrawButtonGlyph(Canvas: TCanvas; const GlyphPos: TPoint;
      State: TButtonState; Transparent: Boolean);
    procedure DrawButtonText(Canvas: TCanvas; const Caption: string;
      TextBounds: TRect; State: TButtonState; BiDiFlags: Longint);
    procedure CalcButtonLayout(Canvas: TCanvas; const Client: TRect;
      const Offset: TPoint; const Caption: string; Layout: TButtonLayout;
      Margin, Spacing: Integer; var GlyphPos: TPoint; var TextBounds: TRect;
      BiDiFlags: Longint);
  public
    constructor Create;
    destructor Destroy; override;
    function Draw(Canvas: TCanvas; const Client: TRect; const Offset: TPoint;
      const Caption: string; Layout: TButtonLayout; Margin, Spacing: Integer;
      State: TButtonState; Transparent: Boolean; BiDiFlags: Longint): TRect;
    property Glyph: TBitmap read FOriginal write SetGlyph;
    property NumGlyphs: TNumGlyphs read FNumGlyphs write SetNumGlyphs;
    property OnChange: TNotifyEvent read FOnChange write FOnChange;
  end;

constructor TGlyphList.CreateSize(AWidth, AHeight: Integer);
begin
  inherited CreateSize(AWidth, AHeight);
  Used := TBits.Create;
end;

destructor TGlyphList.Destroy;
begin
  Used.Free;
  inherited Destroy;
end;

function TGlyphList.AllocateIndex: Integer;
begin
  Result := Used.OpenBit;
  if Result >= Used.Size then
  begin
    Result := inherited Add(nil, nil);
    Used.Size := Result + 1;
  end;
  Used[Result] := True;
end;

function TGlyphList.AddMasked(Image: TBitmap; MaskColor: TColor): Integer;
begin
  Result := AllocateIndex;
  ReplaceMasked(Result, Image, MaskColor);
  Inc(FCount);
end;

procedure TGlyphList.Delete(Index: Integer);
begin
  if Used[Index] then
  begin
    Dec(FCount);
    Used[Index] := False;
  end;
end;

constructor TGlyphCache.Create;
```

```
begin
  inherited Create;
  GlyphLists := TList.Create;
end;

destructor TGlyphCache.Destroy;
begin
  GlyphLists.Free;
  inherited Destroy;
end;

function TGlyphCache.GetList(AWidth, AHeight: Integer): TGlyphList;
var
  I: Integer;
begin
  for I := GlyphLists.Count - 1 downto 0 do
  begin
    Result := GlyphLists[I];
    with Result do
      if (AWidth = Width) and (AHeight = Height) then Exit;
  end;
  Result := TGlyphList.CreateSize(AWidth, AHeight);
  GlyphLists.Add(Result);
end;

procedure TGlyphCache.ReturnList(List: TGlyphList);
begin
  if List = nil then Exit;
  if List.Count = 0 then
  begin
    GlyphLists.Remove(List);
    List.Free;
  end;
end;

function TGlyphCache.Empty: Boolean;
begin
  Result := GlyphLists.Count = 0;
end;

var
  GlyphCache: TGlyphCache = nil;
  ButtonCount: Integer = 0;

constructor TButtonGlyph.Create;
var
  I: TButtonState;
begin
  inherited Create;
  FOriginal := TBitmap.Create;
  FOriginal.OnChange := GlyphChanged;
  FTransparentColor := clOlive;
  FNumGlyphs := 1;
  for I := Low(I) to High(I) do
    FIndexs[I] := -1;
  if GlyphCache = nil then GlyphCache := TGlyphCache.Create;
end;

destructor TButtonGlyph.Destroy;
begin
  FOriginal.Free;
  Invalidate;
  if Assigned(GlyphCache) and GlyphCache.Empty then
  begin
    GlyphCache.Free;
    GlyphCache := nil;
  end;
  inherited Destroy;
end;
```

```
procedure TButtonGlyph.Invalidate;
var
  I: TButtonState;
begin
  for I := Low(I) to High(I) do
  begin
    if FIndexs[I] <> -1 then FGlyphList.Delete(FIndexs[I]);
    FIndexs[I] := -1;
  end;
  GlyphCache.ReturnList(FGlyphList);
  FGlyphList := nil;
end;

procedure TButtonGlyph.GlyphChanged(Sender: TObject);
begin
  if Sender = FOriginal then
  begin
    FTransparentColor := FOriginal.TransparentColor;
    Invalidate;
    if Assigned(FOnChange) then FOnChange(Self);
  end;
end;

procedure TButtonGlyph.SetGlyph(Value: TBitmap);
var
  Glyphs: Integer;
begin
  Invalidate;
  FOriginal.Assign(Value);
  if (Value <> nil) and (Value.Height > 0) then
  begin
    FTransparentColor := Value.TransparentColor;
    if Value.Width mod Value.Height = 0 then
    begin
      Glyphs := Value.Width div Value.Height;
      if Glyphs > 4 then Glyphs := 1;
      SetNumGlyphs(Glyphs);
    end;
  end;
end;

procedure TButtonGlyph.SetNumGlyphs(Value: TNumGlyphs);
begin
  if (Value <> FNumGlyphs) and (Value > 0) then
  begin
    Invalidate;
    FNumGlyphs := Value;
    GlyphChanged(Glyph);
  end;
end;

function TButtonGlyph.CreateButtonGlyph(State: TButtonState): Integer;
const
  ROP_DSPDxax = $00E20746;
var
  TmpImage, DDB, MonoBmp: TBitmap;
  IWidth, IHeight: Integer;
  IRect, ORect: TRect;
  I: TButtonState;
  DestDC: HDC;
begin
  if (State = bsDown) and (NumGlyphs < 3) then State := bsUp;
  Result := FIndexs[State];
  if Result <> -1 then Exit;
  if (FOriginal.Width or FOriginal.Height) = 0 then Exit;
  IWidth := FOriginal.Width div FNumGlyphs;
  IHeight := FOriginal.Height;
  if FGlyphList = nil then
```

```
    begin
      if GlyphCache = nil then GlyphCache := TGlyphCache.Create;
      FGlyphList := GlyphCache.GetList(IWidth, IHeight);
    end;
    TmpImage := TBitmap.Create;
    try
      TmpImage.Width := IWidth;
      TmpImage.Height := IHeight;
      IRect := Rect(0, 0, IWidth, IHeight);
      TmpImage.Canvas.Brush.Color := clBtnFace;
      TmpImage.Palette := CopyPalette(FOriginal.Palette);
      I := State;
      if Ord(I) >= NumGlyphs then I := bsUp;
      ORect := Rect(Ord(I) * IWidth, 0, (Ord(I) + 1) * IWidth, IHeight);
      case State of
        bsUp, bsDown,
        bsExclusive:
          begin
            TmpImage.Canvas.CopyRect(IRect, FOriginal.Canvas, ORect);
            if FOriginal.TransparentMode = tmFixed then
              FIndexs[State] := FGlyphList.AddMasked(TmpImage,
FTransparentColor)
            else
              FIndexs[State] := FGlyphList.AddMasked(TmpImage, clDefault);
          end;
        bsDisabled:
          begin
            MonoBmp := nil;
            DDB := nil;
            try
              MonoBmp := TBitmap.Create;
              DDB := TBitmap.Create;
              DDB.Assign(FOriginal);
              DDB.HandleType := bmDDB;
              if NumGlyphs > 1 then
              with TmpImage.Canvas do
              begin
                CopyRect(IRect, DDB.Canvas, ORect);
                MonoBmp.Monochrome := True;
                MonoBmp.Width := IWidth;
                MonoBmp.Height := IHeight;
                DDB.Canvas.Brush.Color := clWhite;
                MonoBmp.Canvas.CopyRect(IRect, DDB.Canvas, ORect);
                Brush.Color := clBtnHighlight;
                DestDC := Handle;
                SetTextColor(DestDC, clBlack);
                SetBkColor(DestDC, clWhite);
                BitBlt(DestDC, 0, 0, IWidth, IHeight,
                       MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
                DDB.Canvas.Brush.Color := clGray;
                MonoBmp.Canvas.CopyRect(IRect, DDB.Canvas, ORect);
                Brush.Color := clBtnShadow;
                DestDC := Handle;
                SetTextColor(DestDC, clBlack);
                SetBkColor(DestDC, clWhite);
                BitBlt(DestDC, 0, 0, IWidth, IHeight,
                       MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
                DDB.Canvas.Brush.Color := ColorToRGB(FTransparentColor);
                MonoBmp.Canvas.CopyRect(IRect, DDB.Canvas, ORect);
                Brush.Color := clBtnFace;
                DestDC := Handle;
                SetTextColor(DestDC, clBlack);
                SetBkColor(DestDC, clWhite);
                BitBlt(DestDC, 0, 0, IWidth, IHeight,
                       MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
              end
              else
              begin
                with MonoBmp do
```

```
                begin
                  Assign(FOriginal);
                  HandleType := bmDDB;
                  Canvas.Brush.Color := clBlack;
                  Width := IWidth;
                  if Monochrome then
                  begin
                    Canvas.Font.Color := clWhite;
                    Monochrome := False;
                    Canvas.Brush.Color := clWhite;
                  end;
                  Monochrome := True;
                end;
                with TmpImage.Canvas do
                begin
                  Brush.Color := clBtnFace;
                  FillRect(IRect);
                  Brush.Color := clBtnHighlight;
                  SetTextColor(Handle, clBlack);
                  SetBkColor(Handle, clWhite);
                  BitBlt(Handle, 1, 1, IWidth, IHeight,
                    MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
                  Brush.Color := clBtnShadow;
                  SetTextColor(Handle, clBlack);
                  SetBkColor(Handle, clWhite);
                  BitBlt(Handle, 0, 0, IWidth, IHeight,
                    MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
                end;
              end;
            finally
              DDB.Free;
              MonoBmp.Free;
            end;
            FIndexs[State] := FGlyphList.AddMasked(TmpImage, clDefault);
          end;
      end;
    finally
      TmpImage.Free;
    end;
    Result := FIndexs[State];
    FOriginal.Dormant;
  end;

  procedure TButtonGlyph.DrawButtonGlyph(Canvas: TCanvas; const GlyphPos:
TPoint;
    State: TButtonState; Transparent: Boolean);
  var
    Index: Integer;
    Details: TThemedElementDetails;
    R: TRect;
    Button: TThemedButton;
  begin
    if FOriginal = nil then Exit;
    if (FOriginal.Width = 0) or (FOriginal.Height = 0) then Exit;
    Index := CreateButtonGlyph(State);
    with GlyphPos do
    begin
      if ThemeServices.ThemesEnabled then
      begin
        R.TopLeft := GlyphPos;
        R.Right := R.Left + FOriginal.Width div FNumGlyphs;
        R.Bottom := R.Top + FOriginal.Height;
        case State of
          bsDisabled:
            Button := tbPushButtonDisabled;
          bsDown,
          bsExclusive:
            Button := tbPushButtonPressed;
        else
```

```
          Button := tbPushButtonNormal;
        end;
        Details := ThemeServices.GetElementDetails(Button);
        ThemeServices.DrawIcon(Canvas.Handle, Details, R, FGlyphList.Handle,
Index);
      end
      else
        if Transparent or (State = bsExclusive) then
        begin
          ImageList_DrawEx(FGlyphList.Handle, Index, Canvas.Handle, X, Y, 0, 0,
            clNone, clNone, ILD_Transparent)
        end
        else
          ImageList_DrawEx(FGlyphList.Handle, Index, Canvas.Handle, X, Y, 0, 0,
            ColorToRGB(clBtnFace), clNone, ILD_Normal);
    end;
  end;

  procedure TButtonGlyph.DrawButtonText(Canvas: TCanvas; const Caption: string;
    TextBounds: TRect; State: TButtonState; BiDiFlags: LongInt);
  begin
    with Canvas do
    begin
      Brush.Style := bsClear;
      if State = bsDisabled then
      begin
        OffsetRect(TextBounds, 1, 1);
        Font.Color := clBtnHighlight;
        DrawText(Handle, PChar(Caption), Length(Caption), TextBounds,
          DT_CENTER or DT_VCENTER or BiDiFlags);
        OffsetRect(TextBounds, -1, -1);
        Font.Color := clBtnShadow;
        DrawText(Handle, PChar(Caption), Length(Caption), TextBounds,
          DT_CENTER or DT_VCENTER or BiDiFlags);
      end else
        DrawText(Handle, PChar(Caption), Length(Caption), TextBounds,
          DT_CENTER or DT_VCENTER or BiDiFlags);
    end;
  end;

  procedure TButtonGlyph.CalcButtonLayout(Canvas: TCanvas; const Client: TRect;
    const Offset: TPoint; const Caption: string; Layout: TButtonLayout; Margin,
    Spacing: Integer; var GlyphPos: TPoint; var TextBounds: TRect;
    BiDiFlags: LongInt);
  var
    TextPos: TPoint;
    ClientSize, GlyphSize, TextSize: TPoint;
    TotalSize: TPoint;
  begin
    if (BiDiFlags and DT_RIGHT) = DT_RIGHT then
      if Layout = blGlyphLeft then Layout := blGlyphRight
      else
        if Layout = blGlyphRight then Layout := blGlyphLeft;
    ClientSize := Point(Client.Right - Client.Left, Client.Bottom -
      Client.Top);

    if FOriginal <> nil then
      GlyphSize := Point(FOriginal.Width div FNumGlyphs, FOriginal.Height) else
      GlyphSize := Point(0, 0);

    if Length(Caption) > 0 then
    begin
      TextBounds := Rect(0, 0, Client.Right - Client.Left, 0);
      DrawText(Canvas.Handle, PChar(Caption), Length(Caption), TextBounds,
        DT_CALCRECT or BiDiFlags);
      TextSize := Point(TextBounds.Right - TextBounds.Left, TextBounds.Bottom -
        TextBounds.Top);
    end
    else
```

```
begin
  TextBounds := Rect(0, 0, 0, 0);
  TextSize := Point(0,0);
end;

if Layout in [blGlyphLeft, blGlyphRight] then
begin
  GlyphPos.Y := (ClientSize.Y - GlyphSize.Y + 1) div 2;
  TextPos.Y := (ClientSize.Y - TextSize.Y + 1) div 2;
end
else
begin
  GlyphPos.X := (ClientSize.X - GlyphSize.X + 1) div 2;
  TextPos.X := (ClientSize.X - TextSize.X + 1) div 2;
end;

if (TextSize.X = 0) or (GlyphSize.X = 0) then
  Spacing := 0;

if Margin = -1 then
begin
  if Spacing = -1 then
  begin
    TotalSize := Point(GlyphSize.X + TextSize.X, GlyphSize.Y + TextSize.Y);
    if Layout in [blGlyphLeft, blGlyphRight] then
      Margin := (ClientSize.X - TotalSize.X) div 3
    else
      Margin := (ClientSize.Y - TotalSize.Y) div 3;
    Spacing := Margin;
  end
  else
  begin
    TotalSize := Point(GlyphSize.X + Spacing + TextSize.X, GlyphSize.Y +
      Spacing + TextSize.Y);
    if Layout in [blGlyphLeft, blGlyphRight] then
      Margin := (ClientSize.X - TotalSize.X + 1) div 2
    else
      Margin := (ClientSize.Y - TotalSize.Y + 1) div 2;
  end;
end
else
begin
  if Spacing = -1 then
  begin
    TotalSize := Point(ClientSize.X - (Margin + GlyphSize.X), ClientSize.Y -
      (Margin + GlyphSize.Y));
    if Layout in [blGlyphLeft, blGlyphRight] then
      Spacing := (TotalSize.X - TextSize.X) div 2
    else
      Spacing := (TotalSize.Y - TextSize.Y) div 2;
  end;
end;

case Layout of
  blGlyphLeft:
    begin
      GlyphPos.X := Margin;
      TextPos.X := GlyphPos.X + GlyphSize.X + Spacing;
    end;
  blGlyphRight:
    begin
      GlyphPos.X := ClientSize.X - Margin - GlyphSize.X;
      TextPos.X := GlyphPos.X - Spacing - TextSize.X;
    end;
  blGlyphTop:
    begin
      GlyphPos.Y := Margin;
      TextPos.Y := GlyphPos.Y + GlyphSize.Y + Spacing;
    end;
```

```
      blGlyphBottom:
        begin
          GlyphPos.Y := ClientSize.Y - Margin - GlyphSize.Y;
          TextPos.Y := GlyphPos.Y - Spacing - TextSize.Y;
        end;
    end;

    with GlyphPos do
    begin
      Inc(X, Client.Left + Offset.X);
      Inc(Y, Client.Top + Offset.Y);
    end;

    if ThemeServices.ThemesEnabled then
      OffsetRect(TextBounds, TextPos.X + Client.Left, TextPos.Y + Client.Top)
    else
      OffsetRect(TextBounds, TextPos.X + Client.Left + Offset.X, TextPos.Y +
Client.Top + Offset.Y);

  end;

  function TButtonGlyph.Draw(Canvas: TCanvas; const Client: TRect;
    const Offset: TPoint; const Caption: string; Layout: TButtonLayout;
    Margin, Spacing: Integer; State: TButtonState; Transparent: Boolean;
    BiDiFlags: LongInt): TRect;
  var
    GlyphPos: TPoint;
  begin
    CalcButtonLayout(Canvas, Client, Offset, Caption, Layout, Margin, Spacing,
      GlyphPos, Result, BiDiFlags);
    DrawButtonGlyph(Canvas, GlyphPos, State, Transparent);
    DrawButtonText(Canvas, Caption, Result, State, BiDiFlags);
  end;

  procedure TTeplMerSpeedButton.AssignClient(AClient: TObject);
  begin
    inherited AssignClient(AClient);
    FClient := AClient as TTeplMerSpeedButtonLink;
  end;

  function TTeplMerSpeedButton.IsCheckedLinked: Boolean;
  begin
    Result := inherited IsCheckedLinked and (FClient.GroupIndex <> 0) and
      FClient.AllowAllUp and (FClient.Down = (Action as TCustomAction).Checked);
  end;

  function TTeplMerSpeedButton.IsGroupIndexLinked: Boolean;
  begin
    Result := (FClient is TTeplMerSpeedButtonLink) and
      (TTeplMerSpeedButtonLink(FClient).GroupIndex = (Action as
TCustomAction).GroupIndex);
  end;

  procedure TTeplMerSpeedButton.SetChecked(Value: Boolean);
  begin
    if IsCheckedLinked then TTeplMerSpeedButtonLink(FClient).Down := Value;
  end;

  procedure TTeplMerSpeedButton.SetGroupIndex(Value: Integer);
  begin
    if IsGroupIndexLinked then TTeplMerSpeedButtonLink(FClient).GroupIndex :=
Value;
  end;

  constructor TTeplMerSpeedButtonLink.Create(AOwner: TComponent);
  begin
    FGlyph := TButtonGlyph.Create;
    TButtonGlyph(FGlyph).OnChange := GlyphChanged;
    inherited Create(AOwner);
```

```
  SetBounds(0, 0, 23, 22);
  ControlStyle := [csCaptureMouse, csDoubleClicks];
  ParentFont := True;
  Color := clBtnFace;
  FSpacing := 4;
  FMargin := -1;
  FLayout := blGlyphLeft;
  FTransparent := True;
  Inc(ButtonCount);
end;

destructor TTeplMerSpeedButtonLink.Destroy;
begin
  Dec(ButtonCount);
  inherited Destroy;
  TButtonGlyph(FGlyph).Free;
end;

procedure TTeplMerSpeedButtonLink.Paint;
const
  DownStyles: array[Boolean] of Integer = (BDR_RAISEDINNER, BDR_SUNKENOUTER);
  FillStyles: array[Boolean] of Integer = (BF_MIDDLE, 0);
var
  PaintRect: TRect;
  DrawFlags: Integer;
  Offset: TPoint;
  Button: TThemedButton;
  ToolButton: TThemedToolBar;
  Details: TThemedElementDetails;
begin
  if not Enabled then
  begin
    FState := bsDisabled;
    FDragging := False;
  end
  else if FState = bsDisabled then
    if FDown and (GroupIndex <> 0) then
      FState := bsExclusive
    else
      FState := bsUp;
  Canvas.Font := Self.Font;

  if ThemeServices.ThemesEnabled then
  begin
    PerformEraseBackground(Self, Canvas.Handle);

    if not Enabled then
      Button := tbPushButtonDisabled
    else
      if FState in [bsDown, bsExclusive] then
        Button := tbPushButtonPressed
      else
        if MouseInControl then
          Button := tbPushButtonHot
        else
          Button := tbPushButtonNormal;

    ToolButton := ttbToolbarDontCare;
    if FFlat then
    begin
      case Button of
        tbPushButtonDisabled:
          Toolbutton := ttbButtonDisabled;
        tbPushButtonPressed:
          Toolbutton := ttbButtonPressed;
        tbPushButtonHot:
          Toolbutton := ttbButtonHot;
        tbPushButtonNormal:
          Toolbutton := ttbButtonNormal;
```

```
        end;
      end;

      PaintRect := ClientRect;
      if ToolButton = ttbToolbarDontCare then
      begin
        Details := ThemeServices.GetElementDetails(Button);
        ThemeServices.DrawElement(Canvas.Handle, Details, PaintRect);
        PaintRect := ThemeServices.ContentRect(Canvas.Handle, Details,
PaintRect);
      end
      else
      begin
        Details := ThemeServices.GetElementDetails(ToolButton);
        ThemeServices.DrawElement(Canvas.Handle, Details, PaintRect);
        PaintRect := ThemeServices.ContentRect(Canvas.Handle, Details,
PaintRect);
      end;

      if Button = tbPushButtonPressed then
      begin
        if ToolButton <> ttbToolbarDontCare then
          Canvas.Font.Color := clHighlightText;
        Offset := Point(1, 0);
      end
      else
        Offset := Point(0, 0);
      TButtonGlyph(FGlyph).Draw(Canvas, PaintRect, Offset, Caption, FLayout,
FMargin, FSpacing, FState, Transparent,
        DrawTextBiDiModeFlags(0));
    end
    else
    begin
      PaintRect := Rect(0, 0, Width, Height);
      if not FFlat then
      begin
        DrawFlags := DFCS_BUTTONPUSH or DFCS_ADJUSTRECT;
        if FState in [bsDown, bsExclusive] then
          DrawFlags := DrawFlags or DFCS_PUSHED;
        DrawFrameControl(Canvas.Handle, PaintRect, DFC_BUTTON, DrawFlags);
      end
      else
      begin
        if (FState in [bsDown, bsExclusive]) or
          (FMouseInControl and (FState <> bsDisabled)) or
          (csDesigning in ComponentState) then
          DrawEdge(Canvas.Handle, PaintRect, DownStyles[FState in [bsDown,
bsExclusive]],
            FillStyles[Transparent] or BF_RECT)
        else if not Transparent then
        begin
          Canvas.Brush.Color := Color;
          Canvas.FillRect(PaintRect);
        end;
        InflateRect(PaintRect, -1, -1);
      end;
      if FState in [bsDown, bsExclusive] then
      begin
        if (FState = bsExclusive) and (not FFlat or not FMouseInControl) then
        begin
          Canvas.Brush.Bitmap := AllocPatternBitmap(clBtnFace, clBtnHighlight);
          Canvas.FillRect(PaintRect);
        end;
        Offset.X := 1;
        Offset.Y := 1;
      end
      else
      begin
        Offset.X := 0;
```

```
        Offset.Y := 0;
      end;
      TButtonGlyph(FGlyph).Draw(Canvas, PaintRect, Offset, Caption, FLayout,
FMargin,
        FSpacing, FState, Transparent, DrawTextBiDiModeFlags(0));
    end;
  end;

  procedure TTeplMerSpeedButtonLink.UpdateTracking;
  var
    P: TPoint;
  begin
    if FFlat then
    begin
      if Enabled then
      begin
        GetCursorPos(P);
        FMouseInControl := not (FindDragTarget(P, True) = Self);
        if FMouseInControl then
          Perform(CM_MOUSELEAVE, 0, 0)
        else
          Perform(CM_MOUSEENTER, 0, 0);
      end;
    end;
  end;

  procedure TTeplMerSpeedButtonLink.Loaded;
  var
    State: TButtonState;
  begin
    inherited Loaded;
    if Enabled then
      State := bsUp
    else
      State := bsDisabled;
    TButtonGlyph(FGlyph).CreateButtonGlyph(State);
  end;

  procedure TTeplMerSpeedButtonLink.MouseDown(Button: TMouseButton; Shift:
TShiftState;
    X, Y: Integer);
  begin
    inherited MouseDown(Button, Shift, X, Y);
    if (Button = mbLeft) and Enabled then
    begin
      if not FDown then
      begin
        FState := bsDown;
        Invalidate;
      end;
      FDragging := True;
    end;
  end;

  procedure TTeplMerSpeedButtonLink.MouseMove(Shift: TShiftState; X, Y:
Integer);
  var
    NewState: TButtonState;
  begin
    inherited MouseMove(Shift, X, Y);
    if FDragging then
    begin
      if not FDown then NewState := bsUp
      else NewState := bsExclusive;
      if (X >= 0) and (X < ClientWidth) and (Y >= 0) and (Y <= ClientHeight)
then
        if FDown then NewState := bsExclusive else NewState := bsDown;
      if NewState <> FState then
      begin
```

```
        FState := NewState;
        Invalidate;
      end;
    end
    else if not FMouseInControl then
      UpdateTracking;
  end;

  procedure TTeplMerSpeedButtonLink.MouseUp(Button: TMouseButton; Shift:
TShiftState;
    X, Y: Integer);
  var
    DoClick: Boolean;
  begin
    inherited MouseUp(Button, Shift, X, Y);
    if FDragging then
    begin
      FDragging := False;
      DoClick := (X >= 0) and (X < ClientWidth) and (Y >= 0) and (Y <=
ClientHeight);
      if FGroupIndex = 0 then
      begin
        FState := bsUp;
        FMouseInControl := False;
        if DoClick and not (FState in [bsExclusive, bsDown]) then
          Invalidate;
      end
      else
        if DoClick then
        begin
          SetDown(not FDown);
          if FDown then Repaint;
        end
        else
        begin
          if FDown then FState := bsExclusive;
          Repaint;
        end;
      if DoClick then Click;
      UpdateTracking;
    end;
  end;

  procedure TTeplMerSpeedButtonLink.Click;
  begin
    inherited Click;
  end;

  function TTeplMerSpeedButtonLink.GetPalette: HPALETTE;
  begin
    Result := Glyph.Palette;
  end;

  function TTeplMerSpeedButtonLink.GetActionLinkClass: TControlActionLinkClass;
  begin
    Result := TTeplMerSpeedButton;
  end;

  function TTeplMerSpeedButtonLink.GetGlyph: TBitmap;
  begin
    Result := TButtonGlyph(FGlyph).Glyph;
  end;

  procedure TTeplMerSpeedButtonLink.SetGlyph(Value: TBitmap);
  begin
    TButtonGlyph(FGlyph).Glyph := Value;
    Invalidate;
  end;
```

```
function TTeplMerSpeedButtonLink.GetNumGlyphs: TNumGlyphs;
begin
  Result := TButtonGlyph(FGlyph).NumGlyphs;
end;

procedure TTeplMerSpeedButtonLink.SetNumGlyphs(Value: TNumGlyphs);
begin
  if Value < 0 then Value := 1
  else if Value > 4 then Value := 4;
  if Value <> TButtonGlyph(FGlyph).NumGlyphs then
  begin
    TButtonGlyph(FGlyph).NumGlyphs := Value;
    Invalidate;
  end;
end;

procedure TTeplMerSpeedButtonLink.GlyphChanged(Sender: TObject);
begin
  Invalidate;
end;

procedure TTeplMerSpeedButtonLink.UpdateExclusive;
var
  Msg: TMessage;
begin
  if (FGroupIndex <> 0) and (Parent <> nil) then
  begin
    Msg.Msg := CM_BUTTONPRESSED;
    Msg.WParam := FGroupIndex;
    Msg.LParam := Longint(Self);
    Msg.Result := 0;
    Parent.Broadcast(Msg);
  end;
end;

procedure TTeplMerSpeedButtonLink.SetDown(Value: Boolean);
begin
  if FGroupIndex = 0 then Value := False;
  if Value <> FDown then
  begin
    if FDown and (not FAllowAllUp) then Exit;
    FDown := Value;
    if Value then
    begin
      if FState = bsUp then Invalidate;
      FState := bsExclusive
    end
    else
    begin
      FState := bsUp;
      Repaint;
    end;
    if Value then UpdateExclusive;
  end;
end;

procedure TTeplMerSpeedButtonLink.SetFlat(Value: Boolean);
begin
  if Value <> FFlat then
  begin
    FFlat := Value;
    Invalidate;
  end;
end;

procedure TTeplMerSpeedButtonLink.SetGroupIndex(Value: Integer);
begin
  if FGroupIndex <> Value then
  begin
```

```
    FGroupIndex := Value;
    UpdateExclusive;
  end;
end;

procedure TTeplMerSpeedButtonLink.SetLayout(Value: TButtonLayout);
begin
  if FLayout <> Value then
  begin
    FLayout := Value;
    Invalidate;
  end;
end;

procedure TTeplMerSpeedButtonLink.SetMargin(Value: Integer);
begin
  if (Value <> FMargin) and (Value >= -1) then
  begin
    FMargin := Value;
    Invalidate;
  end;
end;

procedure TTeplMerSpeedButtonLink.SetSpacing(Value: Integer);
begin
  if Value <> FSpacing then
  begin
    FSpacing := Value;
    Invalidate;
  end;
end;

procedure TTeplMerSpeedButtonLink.SetTransparent(Value: Boolean);
begin
  if Value <> FTransparent then
  begin
    FTransparent := Value;
    if Value then
      ControlStyle := ControlStyle - [csOpaque] else
      ControlStyle := ControlStyle + [csOpaque];
    Invalidate;
  end;
end;

procedure TTeplMerSpeedButtonLink.SetAllowAllUp(Value: Boolean);
begin
  if FAllowAllUp <> Value then
  begin
    FAllowAllUp := Value;
    UpdateExclusive;
  end;
end;

procedure TTeplMerSpeedButtonLink.WMLButtonDblClk(var Message:
TWMLButtonDown);
begin
  inherited;
  if FDown then DblClick;
end;

procedure TTeplMerSpeedButtonLink.CMEnabledChanged(var Message: TMessage);
const
  NewState: array[Boolean] of TButtonState = (bsDisabled, bsUp);
begin
  TButtonGlyph(FGlyph).CreateButtonGlyph(NewState[Enabled]);
  UpdateTracking;
  Repaint;
end;
```

```
procedure TTeplMerSpeedButtonLink.CMButtonPressed(var Message: TMessage);
var
  Sender: TTeplMerSpeedButtonLink;
begin
  if Message.WParam = FGroupIndex then
  begin
    Sender := TTeplMerSpeedButtonLink(Message.LParam);
    if Sender <> Self then
    begin
      if Sender.Down and FDown then
      begin
        FDown := False;
        FState := bsUp;
        if (Action is TCustomAction) then
          TCustomAction(Action).Checked := False;
        Invalidate;
      end;
      FAllowAllUp := Sender.AllowAllUp;
    end;
  end;
end;

procedure TTeplMerSpeedButtonLink.CMDialogChar(var Message: TCMDialogChar);
begin
  with Message do
    if IsAccel(CharCode, Caption) and Enabled and Visible and
      (Parent <> nil) and Parent.Showing then
    begin
      Click;
      Result := 1;
    end else
      inherited;
end;

procedure TTeplMerSpeedButtonLink.CMFontChanged(var Message: TMessage);
begin
  Invalidate;
end;

procedure TTeplMerSpeedButtonLink.CMTextChanged(var Message: TMessage);
begin
  Invalidate;
end;

procedure TTeplMerSpeedButtonLink.CMSysColorChange(var Message: TMessage);
begin
  with TButtonGlyph(FGlyph) do
  begin
    Invalidate;
    CreateButtonGlyph(FState);
  end;
end;

procedure TTeplMerSpeedButtonLink.CMMouseEnter(var Message: TMessage);
var
  NeedRepaint: Boolean;
begin
  inherited;
  NeedRepaint := FFlat and not FMouseInControl and Enabled and (DragMode <>
dmAutomatic) and (GetCapture = 0);
  if (NeedRepaint or ThemeServices.ThemesEnabled) and not (csDesigning in
ComponentState) then
  begin
    FMouseInControl := True;
    if Enabled then
      Repaint;
  end;
end;
```

```
 procedure TTeplMerSpeedButtonLink.CMMouseLeave(var Message: TMessage);
 var
   NeedRepaint: Boolean;
 begin
   inherited;
   NeedRepaint := FFlat and FMouseInControl and Enabled and not FDragging;
   if NeedRepaint or ThemeServices.ThemesEnabled then
   begin
     FMouseInControl := False;
     if Enabled then
       Repaint;
   end;
 end;

 procedure TTeplMerSpeedButtonLink.ActionChange(Sender: TObject; CheckDefaults:
Boolean);

   procedure CopyImage(ImageList: TCustomImageList; Index: Integer);
   begin
     with Glyph do
     begin
       Width := ImageList.Width;
       Height := ImageList.Height;
       Canvas.Brush.Color := clFuchsia;
       Canvas.FillRect(Rect(0,0, Width, Height));
       ImageList.Draw(Canvas, 0, 0, Index);
     end;
   end;

 begin
   inherited ActionChange(Sender, CheckDefaults);
   if Sender is TCustomAction then
     with TCustomAction(Sender) do
     begin
       if CheckDefaults or (Self.GroupIndex = 0) then
         Self.GroupIndex := GroupIndex;
       if (Glyph.Empty) and (ActionList <> nil) and (ActionList.Images <> nil)
and
         (ImageIndex >= 0) and (ImageIndex < ActionList.Images.Count) then
         CopyImage(ActionList.Images, ImageIndex);
     end;
 end;

 constructor TBitBtn.Create(AOwner: TComponent);
 begin
   FGlyph := TButtonGlyph.Create;
   TButtonGlyph(FGlyph).OnChange := GlyphChanged;
   inherited Create(AOwner);
   FCanvas := TCanvas.Create;
   FStyle := bsAutoDetect;
   FKind := bkCustom;
   FLayout := blGlyphLeft;
   FSpacing := 4;
   FMargin := -1;
   ControlStyle := ControlStyle + [csReflector];
   DoubleBuffered := True;
 end;

 destructor TBitBtn.Destroy;
 begin
   inherited Destroy;
   TButtonGlyph(FGlyph).Free;
   FCanvas.Free;
 end;

 procedure TBitBtn.CreateHandle;
 var
   State: TButtonState;
 begin
```

```
  if Enabled then
    State := bsUp
  else
    State := bsDisabled;
  inherited CreateHandle;
  TButtonGlyph(FGlyph).CreateButtonGlyph(State);
end;

procedure TBitBtn.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  with Params do Style := Style or BS_OWNERDRAW;
end;

procedure TBitBtn.SetButtonStyle(ADefault: Boolean);
begin
  if ADefault <> IsFocused then
  begin
    IsFocused := ADefault;
    Refresh;
  end;
end;

procedure TBitBtn.Click;
var
  Form: TCustomForm;
  Control: TWinControl;
begin
  case FKind of
    bkClose:
      begin
        Form := GetParentForm(Self);
        if Form <> nil then Form.Close
        else inherited Click;
      end;
    bkHelp:
      begin
        Control := Self;
        while (Control <> nil) and (Control.HelpContext = 0) do
          Control := Control.Parent;
        if Control <> nil then Application.HelpContext(Control.HelpContext)
        else inherited Click;
      end;
    else
      inherited Click;
  end;
end;

procedure TBitBtn.CNMeasureItem(var Message: TWMMeasureItem);
begin
  with Message.MeasureItemStruct^ do
  begin
    itemWidth := Width;
    itemHeight := Height;
  end;
end;

procedure TBitBtn.CNDrawItem(var Message: TWMDrawItem);
begin
  DrawItem(Message.DrawItemStruct^);
end;

procedure TBitBtn.DrawItem(const DrawItemStruct: TDrawItemStruct);
var
  IsDown, IsDefault: Boolean;
  State: TButtonState;
  R: TRect;
  Flags: Longint;
  Details: TThemedElementDetails;
```

```
    Button: TThemedButton;
    Offset: TPoint;
  begin
    FCanvas.Handle := DrawItemStruct.hDC;
    R := ClientRect;

    with DrawItemStruct do
    begin
      FCanvas.Handle := hDC;
      FCanvas.Font := Self.Font;
      IsDown := itemState and ODS_SELECTED <> 0;
      IsDefault := itemState and ODS_FOCUS <> 0;

      if not Enabled then State := bsDisabled
      else if IsDown then State := bsDown
      else State := bsUp;
    end;

    if ThemeServices.ThemesEnabled then
    begin
      if not Enabled then
        Button := tbPushButtonDisabled
      else
        if IsDown then
          Button := tbPushButtonPressed
        else
          if FMouseInControl then
            Button := tbPushButtonHot
          else
            if IsFocused or IsDefault then
              Button := tbPushButtonDefaulted
            else
              Button := tbPushButtonNormal;

      Details := ThemeServices.GetElementDetails(Button);
      ThemeServices.DrawParentBackground(Handle, DrawItemStruct.hDC, @Details,
True);
      ThemeServices.DrawElement(DrawItemStruct.hDC, Details,
DrawItemStruct.rcItem);
      R := ThemeServices.ContentRect(FCanvas.Handle, Details,
DrawItemStruct.rcItem);

      if Button = tbPushButtonPressed then
        Offset := Point(1, 0)
      else
        Offset := Point(0, 0);
      TButtonGlyph(FGlyph).Draw(FCanvas, R, Offset, Caption, FLayout, FMargin,
FSpacing, State, False,
        DrawTextBiDiModeFlags(0));

      if IsFocused and IsDefault then
      begin
        FCanvas.Pen.Color := clWindowFrame;
        FCanvas.Brush.Color := clBtnFace;
        DrawFocusRect(FCanvas.Handle, R);
      end;
    end
    else
    begin
      R := ClientRect;

      Flags := DFCS_BUTTONPUSH or DFCS_ADJUSTRECT;
      if IsDown then Flags := Flags or DFCS_PUSHED;
      if DrawItemStruct.itemState and ODS_DISABLED <> 0 then
        Flags := Flags or DFCS_INACTIVE;

      if IsFocused or IsDefault then
      begin
        FCanvas.Pen.Color := clWindowFrame;
```

```
        FCanvas.Pen.Width := 1;
        FCanvas.Brush.Style := bsClear;
        FCanvas.Rectangle(R.Left, R.Top, R.Right, R.Bottom);

        InflateRect(R, -1, -1);
      end;

      if IsDown then
      begin
        FCanvas.Pen.Color := clBtnShadow;
        FCanvas.Pen.Width := 1;
        FCanvas.Brush.Color := clBtnFace;
        FCanvas.Rectangle(R.Left, R.Top, R.Right, R.Bottom);
        InflateRect(R, -1, -1);
      end
      else
        DrawFrameControl(DrawItemStruct.hDC, R, DFC_BUTTON, Flags);

      if IsFocused then
      begin
        R := ClientRect;
        InflateRect(R, -1, -1);
      end;

      FCanvas.Font := Self.Font;
      if IsDown then
        OffsetRect(R, 1, 1);
      TButtonGlyph(FGlyph).Draw(FCanvas, R, Point(0,0), Caption, FLayout,
FMargin,
        FSpacing, State, False, DrawTextBiDiModeFlags(0));

      if IsFocused and IsDefault then
      begin
        R := ClientRect;
        InflateRect(R, -4, -4);
        FCanvas.Pen.Color := clWindowFrame;
        FCanvas.Brush.Color := clBtnFace;
        DrawFocusRect(FCanvas.Handle, R);
      end;
    end;

    FCanvas.Handle := 0;
  end;

  procedure TBitBtn.CMFontChanged(var Message: TMessage);
  begin
    inherited;
    Invalidate;
  end;

  procedure TBitBtn.CMEnabledChanged(var Message: TMessage);
  begin
    inherited;
    Invalidate;
  end;

  procedure TBitBtn.WMLButtonDblClk(var Message: TWMLButtonDblClk);
  begin
    Perform(WM_LBUTTONDOWN, Message.Keys, Longint(Message.Pos));
  end;

  function TBitBtn.GetPalette: HPALETTE;
  begin
    Result := Glyph.Palette;
  end;

  procedure TBitBtn.SetGlyph(Value: TBitmap);
  begin
    TButtonGlyph(FGlyph).Glyph := Value as TBitmap;
```

```
    FModifiedGlyph := True;
    Invalidate;
  end;

  function TBitBtn.GetGlyph: TBitmap;
  begin
    Result := TButtonGlyph(FGlyph).Glyph;
  end;

  procedure TBitBtn.GlyphChanged(Sender: TObject);
  begin
    Invalidate;
  end;

  function TBitBtn.IsCustom: Boolean;
  begin
    Result := Kind = bkCustom;
  end;

  procedure TBitBtn.SetStyle(Value: TButtonStyle);
  begin
    if Value <> FStyle then
    begin
      FStyle := Value;
      Invalidate;
    end;
  end;

  procedure TBitBtn.SetKind(Value: TBitBtnKind);
  begin
    if Value <> FKind then
    begin
      if Value <> bkCustom then
      begin
        Default := Value in [bkOK, bkYes];
        Cancel := Value in [bkCancel, bkNo];

        if ((csLoading in ComponentState) and (Caption = '')) or
          (not (csLoading in ComponentState)) then
        begin
          if BitBtnCaptions[Value] <> nil then
            Caption := LoadResString(BitBtnCaptions[Value]);
        end;

        ModalResult := BitBtnModalResults[Value];
        TButtonGlyph(FGlyph).Glyph := GetBitBtnGlyph(Value);
        NumGlyphs := 2;
        FModifiedGlyph := False;
      end;
      FKind := Value;
      Invalidate;
    end;
  end;

  function TBitBtn.IsCustomCaption: Boolean;
  begin
    Result := AnsiCompareStr(Caption, LoadResString(BitBtnCaptions[FKind])) <>
0;
  end;

  function TBitBtn.GetKind: TBitBtnKind;
  begin
    if FKind <> bkCustom then
      if ((FKind in [bkOK, bkYes]) xor Default) or
        ((FKind in [bkCancel, bkNo]) xor Cancel) or
        (ModalResult <> BitBtnModalResults[FKind]) or
        FModifiedGlyph then
        FKind := bkCustom;
    Result := FKind;
```

```
end;

procedure TBitBtn.SetLayout(Value: TButtonLayout);
begin
  if FLayout <> Value then
  begin
    FLayout := Value;
    Invalidate;
  end;
end;

function TBitBtn.GetNumGlyphs: TNumGlyphs;
begin
  Result := TButtonGlyph(FGlyph).NumGlyphs;
end;

procedure TBitBtn.SetNumGlyphs(Value: TNumGlyphs);
begin
  if Value < 0 then Value := 1
  else if Value > 4 then Value := 4;
  if Value <> TButtonGlyph(FGlyph).NumGlyphs then
  begin
    TButtonGlyph(FGlyph).NumGlyphs := Value;
    Invalidate;
  end;
end;

procedure TBitBtn.SetSpacing(Value: Integer);
begin
  if FSpacing <> Value then
  begin
    FSpacing := Value;
    Invalidate;
  end;
end;

procedure TBitBtn.SetMargin(Value: Integer);
begin
  if (Value <> FMargin) and (Value >= - 1) then
  begin
    FMargin := Value;
    Invalidate;
  end;
end;

procedure TBitBtn.ActionChange(Sender: TObject; CheckDefaults: Boolean);

  procedure CopyImage(ImageList: TCustomImageList; Index: Integer);
  begin
    with Glyph do
    begin
      Width := ImageList.Width;
      Height := ImageList.Height;
      Canvas.Brush.Color := clFuchsia;
      Canvas.FillRect(Rect(0,0, Width, Height));
      ImageList.Draw(Canvas, 0, 0, Index);
    end;
  end;

begin
  inherited ActionChange(Sender, CheckDefaults);
  if Sender is TCustomAction then
    with TCustomAction(Sender) do
    begin
      if (Glyph.Empty) and (ActionList <> nil) and (ActionList.Images <> nil)
and
        (ImageIndex >= 0) and (ImageIndex < ActionList.Images.Count) then
        CopyImage(ActionList.Images, ImageIndex);
    end;
```

```
  end;

  procedure DestroyLocals; far;
  var
    I: TBitBtnKind;
  begin
    for I := Low(TBitBtnKind) to High(TBitBtnKind) do
      BitBtnGlyphs[I].Free;
  end;

  procedure TBitBtn.CMMouseEnter(var Message: TMessage);
  begin
    inherited;
    if ThemeServices.ThemesEnabled and not FMouseInControl and not (csDesigning
in ComponentState) then
    begin
      FMouseInControl := True;
      Repaint;
    end;
  end;

  procedure TBitBtn.CMMouseLeave(var Message: TMessage);
  begin
    inherited;
    if ThemeServices.ThemesEnabled and FMouseInControl then
    begin
      FMouseInControl := False;
      Repaint;
    end;
  end;

  end.
```

Файл форми авторського права

```
unit TeplMerAboutBox;
// Copyright (C)  //
// Teslia S.V.  //
// 2021, CNTU //
interface

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

type
  TTeplMerAboutBox1 = class(TTeplMerAboutBox)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

implementation

uses Unit1;

{$R *.dfm}

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

end.
```