

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ДОВГАНЬ ДЕНИС

ВАРІАНТ 9

(Курсовий проект)

Чернівці, 2015

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ЗАТВЕРДЖУЮ
завідуючий кафедри
_____ проф. С.В. Мельничук
„_____” _____ 2014 р.

ВАРІАНТ 9

ЛИСТ ЗАТВЕРДЖЕННЯ

482.362. 6050102-02 09-1 ЛЗ

УЗГОДЖЕНО
Керівник курсового проекту
асистент кафедри КСМ
_____ М.І. Скрипський
«_____» _____ 2015 р.

Виконавець
студент 1-го курсу
_____ Д.В. Довгань
«_____» _____ 2015 р.

Спеціальність 7.0915.01 “Комп’ютерні системи та мережі”

“ ” 2015 p.

(прізвище, ім'я, по батькові)

3. Вихідні дані до проекту(роботи) _____

- Вивести дані на екран, використовуючи метод базового класу.*

- 2) *Екранні форми роботи програми.*

[illegible]

6. Консультанти з проекту (роботи), з вказівкою відділів, які до них належать

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____ Скрипський М.І.
(підпис)

Завдання прийняв до виконання _____ Довгань Д.В.
(підпис)

Календарний план

№ п/п	Назва етапів курсового проекту (роботи)	Термін виконання етапів роботи	Примітки
1.	<i>Проведення огляду літератури по темі курсової роботи</i>		
2.	<i>Аналіз ООП та його основних принципів</i>		
3.	<i>Обґрунтування завдання на курсовий проект</i>		
4.	<i>Формування вимог до програми та оформлення технічного завдання</i>		
5.	<i>Розробка інтерфейсу користувача</i>		
6.	<i>Розроблення алгоритмів функціонування програми</i>		
7.	<i>Програмування розроблених алгоритмів</i>		
8.	<i>Тестування програми</i>		
9.	<i>Оформлення програмної документації</i>		
10.	<i>Захист курсового проекту</i>	<i>згідно розкладу</i>	

Студент _____ Довгань Д.В.
(підпис)

Керівник проекту _____ Скрипський М.І.
(підпис)

АНОТАЦІЯ

Метою розробки програми є вивчення реального прикладу програми, побудованої на принципах ООП, напрацювання вмінь та навичок, зокрема: збереження інформації у файл із можливістю відкрити цей файл на іншому комп'ютері, реалізація друку, робота з графічними компонентами.

В програмі реалізовані два класи, один з яких наслідує інший. В дочірньому класі запрограмовані нові властивості та методи, в реалізаціях яких присутнє звернення до методів батьківського класу.

Окрім цього, в програмі реалізована робота з популярним типом даних JSON (JavaScript Object Notation), у цьому форматі можна імпортувати та експортувати данні, якими оперує програма.

Також в програмі реалізована можливість експорту даних у текстовий документ формату txt та подальший друк цього документу.

Робота написана мовою програмування Object Pascal з використанням інтегрованого середовища Delphi.

АННОТАЦИЯ

Целью разработки программы является изучение реального примера программы, построенной на принципах ООП, наработки умений и навыков, в частности: сохранения информации в файл с возможностью открыть этот файл на другом компьютере, реализация печати, работа с графическими компонентами.

В программе реализованы два класса, один из которых унаследует другой. В дочернем классе запрограммированы новые свойства и методы, в реализации которых присутствует обращение к методам родительского класса.

Кроме этого, в программе реализована работа с популярным типом данных JSON (JavaScript Object Notation), в этом формате можно импортировать и экспортировать данные, которыми оперирует программа

Также в программе реализована возможность экспорта данных в текстовый документ формата txt и последующая печать этого документа.

Программа написана на языке программирования Object Pascal с использованием интегрированной среды Delphi.

ABSTRACT

The aim of developing program is to develop case studies programs, based on principles of the OOP, developments skills, including: recording on file with the ability to open the file on another computer, implementation of printing, work with graphic components.

The program implemented two classes, one of which extends the other. As a subsidiary class programmed new properties and methods, implementation of which present an appeal to methods of parent class.

In addition, program imolemented to work with popular data type JSON (JavaScript Object Notation), in this format you can import and export data, which operates the program.

Also, program features the ability to export data in a text document formats txt and further printing of this document.

Program is written in Object Pascal programming using IDE Delphi.

Затверджено

482.362.6050102-02 09-1 ЛЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362.6050102-02 09-1 ЛЗ

(Специфікація)

Сторінок 2

2015

482.362.6050102-02 09-1 ЛЗ

Позначення	Найменування	Примітка
482.362.6050102-02 09-1	Технічне завдання	
482.362.6050102-02 81 09-1	Пояснювальна записка	
482.362.6050102-02 35 09-1	Опис мови	
482.362.6050102-02 13 09-1	Опис програми	
482.362.6050102-02 12 09-1	Текст програми	
482.362.6050102-02 51 09-1	Програма та методика випробування	

Затверджено

482.362.6050102-02 09-1 ЛЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362.6050102-02 09-1 ЛЗ

(Технічне завдання)

Сторінок 10

АНОТАЦІЯ

Документ «Технічне завдання» містить інформаційну частину, підстави для розробки програмного продукту, призначення розробленої програми, вимоги до функціональних характеристик програми, вимоги до програмної документації, техніко–економічні показники розробленого продукту, стадії й етапи розробки та порядок і приймання роботи.

ЗМІСТ

ВСТУП.....	4
2. ПРИЗНАЧЕННЯ РОЗРОБКИ	4
3. ВИМОГИ ДО ПРОГРАМИ АБО ПРОГРАМНОГО ПРОДУКТУ	4
3.1. Вимоги до функціональних характеристик.....	4
3.2. Вимоги до надійності	5
3.3. Умови експлуатації	6
3.4. Вимоги до складу і параметрів технічних засобів.....	7
3.5. Вимоги до інформаційної та програмної сумісності	7
3.6. Вимоги до маркірування та упаковки.....	7
3.7. Вимоги до транспортування і зберігання.....	8
4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ.....	8
5. ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ.....	9
6. СТАДІЇ І ЕТАПИ РОЗРОБКИ	9
7. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ	10

ВСТУП

Розроблена програма призначена для засвоєння пройденого курсу програмування, зокрема тем: класи та їх реалізація в ООП, робота з файлами в Object Pascal, візуальні компоненти Delphi.

Результати роботи програми можуть використовуватися для спрощення роботи і зменшення гоміздкості програмного коду при розробленні баз даних й зв'язуванні їх між собою, а також для проведення лабораторних та практичних занять з дисципліни «Програмування» які ведуться для студентів кафедри КСМ Чернівецького національного університету імені Юрія Федьковича.

1. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Завдання на курсовий проект затверджено на засіданні кафедри комп'ютерних систем і мереж відділу комп'ютерних наук Інституту фізико-технічних та комп'ютерних наук Чернівецького національного університету імені Юрія Федьковича. Варіант роботи – №9.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

При необхідності оперування даними про велику кількість об'єктів, виникає ряд труднощів, які пов'язані з частим виконанням одноманітних дій. Сотні й тисячі логічних елементів можна замінити невеликою за розміром класифікацією даних, за якою встановлюються певні спільні властивості об'єктів.

Розроблена програма планується використовуватися для засвоєння принципів об'єктно-орієнтованого програмування, що інтенсивно застосовуються при розробленні програм.

3. ВИМОГИ ДО ПРОГРАМИ АБО ПРОГРАМНОГО ПРОДУКТУ

3.1. Вимоги до функціональних характеристик

Вимоги до функціональних характеристик розробленої програми поставлені у технічному завданні на виконання курсового проекту. Відповідно до них програма повинна забезпечує можливість створення класів та реалізації дій на їх екземплярах. В той же час, програма повинна забезпечувати зручний та інтуїтивний графічний інтерфейс користувачу, легкість в роботі й виконувати наступні функції:

- задання екземплярів класів;
- введення значень для полів об'єктів;
- виведення отриманої інформації в таблицю;
- використання можливостей роботи з файлами;
- використання наслідування;
- інформування користувача в процесі роботи з програмою.

3.2. Вимоги до надійності

Надійність роботи програмного продукту завжди є невід'ємною частиною надійності апаратно - програмного комплексу комп'ютерної системи чи персонального комп'ютера тому, що програма повинна виконуватися під керуванням операційної системи на апаратній платформі персонального комп'ютера. В загальному випадку вимоги по надійності до розробленої програми наступні:

- повноцінне функціонування програми за умови повноцінного функціонування операційної системи;
- повноцінне функціонування програми за умови повноцінного функціонування апаратної частини персонального комп'ютера;
- самовідновлення в роботі після збою в апаратній чи програмній частині персонального комп'ютера.

3.3. Умови експлуатації

Умови експлуатації передбачають техніку безпеки, експлуатацію програмного продукту разом в апаратно-програмному комплексі ПК.

Техніка безпеки:

- Персональний комп'ютер повинен зберігатися у приміщеннях від +5 °C до +35 °C при відносній вологості повітря не більше 85%.
- В приміщеннях для зберігання персонального комп'ютера не повинно бути агресивних сумішей, які викликають корозію.
- При зберіганні й транспортуванні програмного продукту на жорстких носіях інформації, встановлених в системі ПК, необхідно дотримуватись вимог правил пожежної безпеки.
- Розміщення упакованого персонального комп'ютера поблизу джерел тепла забороняється.

Умови експлуатації персонального комп'ютера:

- Електроживлення комп'ютера повинно здійснюватись від однофазної мережі змінного струму номінальною напругою $220\text{В} \pm 15\%$ та частотою 50 ± 1 Гц.
- Заземлення персонального комп'ютера та периферійних пристроїв повинно здійснюватись згідно ГОСТ 258-61.

Нормальні умови застосування:

- Температура оточуючого повітря, °C..... 20 ± 5 .
- Відносна вологість повітря, % 65 ± 15 .
- Атмосферний тиск, кПа (мм рт. ст.)..... $100 \pm 4(750 \pm 30)$.
- Напруга живлення, В..... 220 ± 5 .
- Частота живлячої мережі, Гц..... $50 \pm 0,5$.

Робочі умови експлуатації:

- Температура оточуючого повітря, °C.....від 10 до + 30.
- Відносна вологість повітря, %80 при 20 °C.
- Атмосферний тиск, кПа (мм рт. ст.).....84-107(630-800).

Забороняється:

- Користуватись персональним комп'ютером біля джерел тепла.
- Для дотримання умов вентиляції корпусу персонального комп'ютера необхідно забезпечити 25 - 30 см вільного простору за задньою стінкою корпусу персонального комп'ютера.
- Забороняється проводити підключення та відключення зовнішніх пристроїв при ввімкненому комп'ютері.
- Технічне обслуговування ПК при використанні здійснюється спеціалістами, які пройшли навчання у відповідній організації підприємства-виробника та здобули відповідний рівень кваліфікації.

3.4. Вимоги до складу і параметрів технічних засобів

Мінімальними вимогами до апаратної частини ПК, за яких програма повинна працювати, слід вважати наступні:

- процесори 6-го покоління (AMD K6-2 300 МГц і вище, Intel Pentium Pro/II/Celeron 300 МГц і вище);
- об'єм оперативної пам'яті 128 Мб.;
- графічний адаптер S3 Savage 4 Мб.;
- жорсткий диск ємністю 2 Гб.;
- привід гнучких дисків (дискковод).

3.5. Вимоги до інформаційної та програмної сумісності

Розроблена програма повинна бути зорієнтована на роботу в операційних системах сімейства Windows, що пояснюється їх популярністю серед користувачів.

3.6. Вимоги до маркірування та упаковки

Програма не повинна потребувати спеціального упакування. Її достатньо помістити на дискету або на інший пристрій для транспортування.

Відповідно до вимог ЄСПД розроблена програма маркується номером: 482.362.6050102-26 40-1.

3.7. Вимоги до транспортування і зберігання

Вимоги до зберігання та транспортування програми залежать від вимог до відповідного носія даних. Оскільки програмний продукт постачається на гнучких магнітних дисках, тому при транспортуванні та зберіганні зазначених носіїв необхідно забезпечити виконання наступних умов:

- магнітних полів;
- температури повітря та вологості для вказаного носія даних.

При необхідності існує можливість замовлення програми на оптичному носію даних. В такому випадку необхідними умовами транспортування та зберігання програми будуть вимоги до транспортування та зберігання оптичного носія даних – забезпечення відсутності впливу потужних – забезпечити дотримання допустимих значень.

4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація додається до кожного програмного продукту й передбачає наявність наступних документів, які описують призначення, структуру, алгоритм функціонування програмного продукту, а саме:

- специфікація;
- технічне завдання;
- пояснювальна записка;
- опис мови;
- опис програми;
- текст програми;
- програма та методика випробовування.

5. ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Розроблену програму в курсовому проекті планується використовувати в лабораторному практикумі з дисципліни «Програмування», тому вона повинна бути достатньо зручною для використання й недорогою.

6. СТАДІЇ І ЕТАПИ РОЗРОБКИ

Розробка та оформлення програми згідно поставленого завдання проводилось у відповідності до ЄСПД за наступними етапами:

- огляд літератури згідно завдання на курсовий проект;
- проведення тестування програми та внесення змін в програмний код відповідно до результатів тестування;
- оформлення програмної документації;
- розроблення графічної частини.

Більш детальний план виконання етапів курсової роботи із зазначенням термінів їх виконання наведено в таблиці 6.1.

Таблиця 6.1.

Етапи розробки

№ п/п	Найменування етапів	Термін виконання
1.	Одержання технічного завдання	
2.	Аналіз літератури	
3.	Розробка схеми програми	
4.	Розробка алгоритму роботи програми. Вибір програмних засобів	
5.	Розробка інтерфейсу програми	
6.	Тестування та відлагодження програми	
7.	Оформлення програмної документації	
8.	Представлення готової роботи	

9.	Попередній захист	
10.	Коректування програмної документації	
11.	Захист роботи	Згідно графіку

7. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

До основних випробовувань необхідно віднести експертний аналіз (порівняння) вихідних результатів програмного продукту із реально присутніми. У випадку, коли вони співпадають, програма функціонує правильно, якщо ні, то необхідно перевірити код програми.

Контроль за виконанням курсового проекту здійснює керівник роботи.

Затверджено

482.362.6050102-02 81 09-1 ЛЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362.6050102-02 81 09-1

(Пояснювальна записка)

Сторінок 10

2015

АНОТАЦІЯ

Документ «Пояснювальна записка» включає основні розділи, а саме: призначення та область застосування розробки, технічні характеристики, очікувані техніко – економічні показники, охорону праці та джерела літератури, які використовувалися при розробці. Документ містить відомості про постановку задачі на розробку програми та опис методів додавання і видалення об'єктів класу, а також виведення інформації про них на екран та у файл.

Наведено опис алгоритму й функціонування програми, вхідні та вихідні дані програми.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	4
ВСТУП.....	5
1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	5
2. ТЕХНІЧНІ ХАРАКТЕРИСТИКИ.....	5
2.1. Середовище об'єктно-орієнтовного програмування	5
2.1.1. Основні принципи ООП.....	6
2.1.2 Класи, об'єкти і методи в Delphi.....	8
2.2. Постановка задачі на розробку програми	8
3. Опис організації вхідних та вихідних даних	9
4. Опис вибору технічних і програмних засобів	9

482.362.6050102-02 81 09-1 ЛЗ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ООП – Об’єктно-орієнтоване програмування

ВСТУП

Розроблена програма призначена для опрацювання даних, введених користувачем, використовуючи основні принципи об'єктно-орієнтованого програмування.

Програма розроблена в середовищі розробки Borland Delphi 7.0 та працює під управлінням операційних систем Windows. Вивід даних програма здійснює на стандартний пристрій виводу – монітор. Інтерфейс користувача (меню, діалогові вікна, кнопки) дозволяє швидко освоїти принципи ООП, функціонування його складових та застосування при вирішенні різноманітних задач.

1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

При роботі з базами даних (БД) необхідно забезпечити збереження всієї інформації та розробити систему її опрацювання. Оскільки зазвичай БД містять значну кількість незалежних даних, до яких має бути відкритий доступ, то зручно користуватися динамічними масивами об'єктів певного класу. Тому даний проект призначений для поглибленого дослідження принципів ООП, особливостей роботи з класами, їх властивостями та методами обробки даних, зокрема для вирішення вище зазначеної задачі.

Результати роботи програми можуть використовуватися у навчальних закладах для упорядкування особистих даних про студентів(учнів) та успішності їх навчальної діяльності.

2. ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

2.1. Середовище об'єктно-орієнтовного програмування

Історично склалося так, що програмування виникло і розвивалося як процедурне програмування, яке припускає, що основою програми є алгоритм, процедура обробки даних.

Об'єктно-орієнтоване програмування (ООП) - це методика розробки програм, в основі якої лежить поняття об'єкту, як деякої структури, що описує об'єкт реального світу, його поведінку. Завдання, що вирішується з використанням методики ООП, описується в термінах об'єктів і операцій над ними, а програма при такому підході є набором об'єктів і зв'язків між ними.

Строго кажучи, для того, щоб розробляти додатки в Delphi на базі тих, що надаються середовищем розробки компонентів, знання концепції ООП не є необхідним. Проте для глибшого розуміння того, як програма взаємодіє з компонентами, що і чому Delphi додає в текст програми, матеріал даного розділу вельми корисний.

2.1.1. Основні принципи ООП

ООП – це стиль програмування, що з'явився в 80 роках 20 століття. На відміну від процедурних мов, де дані і інструкції з їх обробки існують окремо, в об'єктно-орієнтованому програмуванні ця інформація об'єднується в єдину сутність.

У об'єктно-програмного програмування є свої постулати. Принципи ООП – це його основні ідеї. Виділяють три найголовніші з них: успадкування, поліморфізм і інкапсуляція. Нижче кожен буде розглянуто більш докладно. Основи програмування на мовах ООП полягають у використанні об'єктів і класів. При переході від процедурного стилю написання вихідного коду до об'єктно-орієнтованого нерідко виникають складнощі, проте більшість розробників знаходять в ООП безліч плюсів.

Інкапсуляція – це використання об'єднання даних та інструкцій щодо їх обробки в єдину сутність – клас. Під час написання програм на одній з мов ООП відбувається розмежування між інформацією всередині сутності і зовні. Таким чином досягається забезпечення безпеки даних і методів їх реалізації від зовнішніх впливів, наприклад, з боку інших класів, що не відносяться до

цього об'єкта. У середині сутності дані успішно взаємодіють один з одним, але надійно захищені від несанкціонованого доступу ззовні.

Другий принцип ООП – спадкування – це можливість одного класу використовувати методи іншого без повторення їх фактичної реалізації. Спадкування дозволяє позбутися від надмірності вихідного коду.

Ще один принцип ООП – поліморфізм. Його використання означає, що для маніпуляції з об'єктами різного ступеня складності можна створити один інтерфейс, який буде по-різному реагувати на події і одночасно правильно реалізовувати поставлені завдання.

Принципи ООП використовуються в таких найбільш популярних мовах програмування, як C++ і Java, на яких розроблена значна частина програм та програм. Є й менш використовувані мови ООП – це Delphi, Object Pascal, Ruby, Python і багато інших.

Незважаючи на в основному позитивні висловлювання у бік даної методології, нерідко принципи ООП піддаються і критиці. Як і у процедурного програмування, у ООП є свої недоліки.

По-перше, складність переходу. Щоб зрозуміти принципи ООП, буде потрібно досить багато часу, тим більше людям, впритул працює тільки з процедурними мовами програмування. По-друге, недоліком є більш складна документація, так як потрібно не тільки описувати класи та об'єкти, а й конкретні випадки їх реалізації. По-третє, зайва універсальність методів може призвести до того, що вихідний код і розроблювані програми будуть перевантажені незатребуваними в даному конкретному випадку функціями і можливостями. Крім того, відзначають неефективність з точки зору розподілу пам'яті. Однак незалежно від думки оточуючих число програмістів ООП постійно зростає, а самі мови стрімко розвиваються.

2.1.2. Класи, об'єкти і методи в Delphi

Класична мова Pascal дозволяє програмістові визначати свої власні складні типи даних - записи (records). Object Pascal, підтримуючи концепцію об'єктно-орієнтованого програмування, дає можливість визначати *класи*. Клас - це складна структура, що включає крім опису даних опис процедур і функцій, які можуть бути виконані над представником класу - об'єктом.

Методи класу (процедури і функції, оголошення яких включене в опис класу) виконують дії над об'єктами класу. Щоб метод був виконаний, треба вказати ім'я об'єкту і ім'я методу, відокремивши одне ім'я від іншого крапкою. Наприклад, інструкція

professor.Show;

викликає застосування методу show об'єкту professor. Фактично інструкція застосування методу до об'єкту - це специфічний спосіб запису інструкції виклику процедури.

У програмі методи класу визначаються точно так, як і звичайні процедури і функції, за винятком того, що ім'я процедури або функції, що є методом, складається з двох частин: імені класу, до якого належить метод, і імені методу. Ім'я класу від імені методу відділяється крапкою.

2.2. Постановка задачі на розробку програми

Основна мета курсового проекту опрацювання даних, введених користувачем, використовуючи основні принципи об'єктно-орієнтованого програмування. Доступ користувача до всіх даних на будь-якому етапі роботи та зручне керування, редагування та збереження їх є першочерговим призначенням даної програми.

Основна робота програми полягає в створенні нової інформаційної структури даних (класу) та її опрацювання за допомогою вивчених алгоритмів та принципів об'єктно-орієнтованого програмування.

3. Опис організації вхідних та вихідних даних

Вхідними даними вважається інформація введена користувачем з метою майбутньої роботи з цими даними.

До вихідних даних програми відносяться файл спеціального формату для збереження даних для їх повторного використання, а також виведені данні у вигляді таблиці.

4. Опис вибору технічних і програмних засобів

Розроблений програмний продукт орієнтований на роботу в ОС Windows XP/7/8/8.1, тому для коректної роботи програми необхідне стабільне функціонування ОС. Під час виконання, програма не звертається до інших прикладних програмних продуктів, таких як Microsoft Office та ін.

Вимоги до об'єму оперативної пам'яті – мінімум 64 Мб зумовлені вимогами операційної системи та необхідних обчислень.

Вимоги до об'єму жорсткого диску – мінімум 2 Гб зумовлені вимогами операційної системи та необхідного програмного забезпечення.

Для усунення виявлених помилок та створення завантажувального файлу необхідний встановлений пакет інструментальних засобів Embarcadero Delphi XE.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Батищев Д. И. Генетические алгоритмы решения экстремальных задач. – Воронеж: ВГТУ, 1995. – 231 с. Скобцов Ю.А., Скобцов В.Ю. Логическое моделирование и тестирование цифровых устройств. – Донецк: ИПММ НАН Украины, ДонНТУ, 2005. – 436с.
2. Goldberg D.E. Genetic Algorithms in Search, Optimization & Machine Learning. –Addison-Wesley Publishing Company, Inc., 1989. P.150.
3. Люгер Д., Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е изд. – М.: Вильямс, 2003. – 864 с.

Затверджено

482.362. 6050102-02 35 09-1 ЛЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМ. Ю. ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362. 6050102-02 35 09-1

(Опис мови)

Сторінок 22

АНОТАЦІЯ

Даний документ містить основні відомості про інструментальні засоби розробки програмного забезпечення, порівняння різних сучасних інтегрованих середовищ та компіляторів, показані критерії вибору засобів програмування. Увага, проведеного огляду, звернута на середовище програмування Embarcadero Delphi XE.

АННОТАЦИЯ

Данный документ содержит основные сведения об инструментальных средствах разработки программного обеспечения, сравнение различных современных интегрированных сред и компиляторов, показаны критерии выбора средств программирования. Внимание проведенного осмотра, обращено на среду программирования Embarcadero Delphi XE.

ABSTRACT

This document contains basic information about the development tools software comparison of various modern integrated environments and compilers, browse the selection criteria of programming. Attention reviewed, paid to the programming environment Embarcadero Delphi XE.

ЗМІСТ

ВСТУП.....	6
1. ЗАГАЛЬНІ ВІДОМОСТІ	6
2. ЕЛЕМЕНТИ МОВИ ТА СПОСОБИ СТРУКТУРИЗАЦІЇ ПРОГРАМИ	7
2.1 Змінні	9
2.2 Коментарі	9
2.3 Типи даних	10
2.4 Оператори	11
3. ВБУДОВАНІ ЕЛЕМЕНТИ	12
4. ЗАСОБИ НАЛАГОДЖЕННЯ ПРОГРАМ.....	20
ДЖЕРЕЛА ВИКОРИСТАНІ ПРИ РОЗРОБЦІ	22

ВСТУП

Стрімкий темп розвитку інформаційних технологій на даний час дійсно вражає. Але в комп'ютерному світі залишається одна найважливіша область, зміни в якій протікають досить повільно. Програмування, кодування, складання вихідних текстів – ключовий елемент в створенні будь-якого додатку сьогодні здійснюється так само, як десятки років тому. На теперішній час найважливішим аспектом є швидкість і якість створення програм, а ці характеристики може забезпечити тільки середовище візуального проектування, здатне взяти на себе значні об'єми рутинної роботи по підготовці додатків.

Можливості Delphi повністю відповідають подібним вимогам і підходять для створення системи будь-якої складності. Система Delphi дозволяє писати як маленькі програми і утиліти для персонального використання, так і корпоративні системи, що працюють з базами даних на різних платформах.

1. ЗАГАЛЬНІ ВІДОМОСТІ

Embarcadero Delphi – це об'єктно-орієнтоване середовище візуального програмування (RAD – Rapid Application Development). Вона призначена для прискореної розробки високопродуктивних 32-бітних додатків, які можуть працювати в середовищі Windows, Linux або OS X. При цьому Delphi дозволяє звести до мінімуму об'єм програмного коду який вводиться вручну. В склад Delphi входять засоби, необхідні для розробки, тестування та установки додатків, включаючи велику за обсягом бібліотеку компонентів (VCL – Visual Components Library), засоби візуального проектування, шаблони додатків і форм.

В системі Delphi використовується спеціалізована версія мови програмування Паскаль, що постійно вдосконалюється; вона називається Delphi (в шостій і більш ранішніх варіантах системи Delphi вона називалась

Object Pascal - «Об'єктний Паскаль»). Ця версія включає набір розширень, орієнтованих тільки на застосування в рамках середовища Delphi і призначених для прискореного створювання додатків.

Перед створенням програмного продукту найважливішим питанням є вибір інструментальних засобів, за допомогою яких буде реалізована програма. Даний продукт створений за допомогою мови Object Pascal в середовищі програмування Embarcadero Delphi XE.

Середовище Embarcadero Delphi XE являє собою інтегровану оболонку розробнику, в яку входить набір спеціалізованих програм, які відповідають за різні етапи створення готового додатку. Основні вікна системи Embarcadero Delphi XE наступні: інспектор об'єктів, провідник, проектувальник форм, вікно редактора. Вихідний текст програми готується в середовищі Embarcadero Delphi XE за допомогою вбудованого редактора вихідних текстів. Цей редактор спеціалізований. Він відрізняється гнучкими можливостями кольорового виділення різних елементів тексту програми (ключових слів, назв, операцій, чисел і рядків) і надає можливість швидкого вводу конструкцій, які часто зустрічаються. Найважливішою характеристикою програми, що розробляється є зручність її користувацького інтерфейсу, наявність і доступність необхідних елементів управління. В системі присутній проектувальник форм, за допомогою якого вікна майбутньої програми підготовляється у вигляді форм. Проектувальник дозволяє підібрати оптимальні розміри вікон, розмістити і настроїти загально можливі елементи управління і меню, додати готові зображення, вказати заголовки, підказки, підписи та багато іншого.

2. ЕЛЕМЕНТИМОВИ ТА СПОСОБИ СТРУКТУРИЗАЦІЇ ПРОГРАМИ

Елементами мови є набори компонентів, які дозволяють створювати додатки за найрізноманітнішими тематиками. Компоненти володіють наборами властивостей, що характеризують їх особливості. Крім

властивостей, компоненти містять методи – програмний код, який обробляє значення властивостей та події – повідомлення, які компонент приймає від додатку.

Всі додатки в Embarcadero Delphi XE будуються по наступному принципу: в їхній головній частині з розширенням. DPR зберігається тільки виклик декількох команд, які відкривають головне вікно, а також виконують завершальні дії. Решта всього програмного коду міститься в файлах, що зберігають опис додаткових модулів, які підключаються. Кожен модуль має строго задану структуру, яка зазвичай автоматично генерується системою Embarcadero Delphi XE при його створенні. Модуль складається з чотирьох частин: інтерфейсної частини, частини реалізації (обов'язкова), частини ініціалізації і частини завершення (необов'язкова) [1].

Спочатку вказують заголовок модуля – ключове слово Unit, за ним довільну назву модуля (вона повинна співпадати з іменем файлу, в якому модуль зберігається) і кладуть крапку з комою: **Unit** Testunit; Інтерфейсна частина описує інформацію, яка доступна з інших частин програми, з інших модулів і головної частини. Частина реалізації описує інформацію, яка недоступна з інших модулів. Подібне розділення модуля на частини дозволяє створювати і розповсюджувати модулі у відкомпільованому вигляді (розширення .DCU), додаючи до них тільки опис інтерфейсної частини. При цьому внести зміни в такий модуль неможливо, вихідний код, який реалізує описані в інтерфейсній частині можливості, недоступний. Такий підхід дозволяє повторно використовувати раніше написані для інших програм і вже відкоректовані модулі та розмежовує доступ до модуля декількох програмістів, а також дозволяє розбивати програму на набір логічно незалежних модулів. Інтерфейсна частина завжди йде першою і починається з ключового слова interface, а частина реалізації з – implementation.

Частини ініціалізації і завершення необов'язкові. Вказані в них дії виконуються, відповідно, на самому початку та в самому кінці роботи

програми і тільки один раз. Частина ініціалізації починається з ключового слова **initialization**, частина завершення – з ключового слова **finalization**. В кінці модуля завжди ставиться слово **end** і крапка.

Базовими елементами мови являються: коментарі, змінні, константи, оператори, типи даних тощо.

2.1 Змінні

Змінна – це область пам'яті, в якій знаходяться дані, якими оперує програма. Коли програма маніпулює з даними вона, фактично, оперує з вмістом комірок пам'яті.

В якості імені змінної можна використати послідовність з букв латинського алфавіту, цифр і деяких спеціальних символів. Першим символом змінної повинна бути буква. Пробіл у назві змінної використовувати не можна.

В загальному вигляді інструкція опису змінної має наступний вигляд

Ім'я: тип;

де:

ім'я – ім'я змінної;

тип – тип даних, для збереження яких призначена змінна.

Наприклад:

a: real;

b: boolean;

i: integer;

2.2 Коментарі

Коментарі являють собою пояснювальний текст, який можна записувати у будь – якому місці програми, де дозволений пробіл. Текст коментарів обмежується символами (* і *) або аналогічними { і } і може містити в собі будь – які символи мови, в тому числі і кирилиця. Коментарій,

обмежений вказаними символами, може займати декілька рядків. Однорядковий коментар на початку рядка містить подвійний слеш //.

Приклад коментарів:

(*однорядковий коментар*)

// другий однорядковий коментар

(*початок багаторядкового коментарю
кінець багаторядкового коментарю*)

Коментар ігнорується компілятором і не впливає на виконання програми. За допомогою коментарів можна виключати будь – які оператори програми в процесі її налагодження, наприклад, наступним чином:

Sum:=0;

For:= 1 to 100 do begin

Read (x);

// if x<0 then x:=0;

sum:= sum+x;

end;

Тут умовний оператор в тілі циклу оформлений як коментар і він не буде виконуватися.

2.3 Типи даних

Суворі типізація даних – одна з найважливіших властивостей мови Object Pascal. Це означає, що всі реальні змінні, які передаються в якості параметрів в функцію чи процедуру, повинні абсолютно точно відповідати типу формальних параметрів в оголошенні цієї функції чи процедури.

Дійсні типи даних: single, real, double та extended.

Вибір одного з даних типів для представлення змінних програми визначається необхідною точністю (кількістю розрядів мантиси) їх представлення і діапазоном представлення значень їх порядку.

Цілочисельні типи: shortint, byte, word, integer, longint.

Вибір одного із типів даних визначається діапазоном використовуваних значень змінних. Цілі зберігаються в двійковій системі числення у вигляді послідовності 1 і 0.

В Delphi існує три символьних типи:

- AnsiChar – стандартний однобайтовий символ.
- WideChar – двобайтовий символ Unicode.
- Char – однобайтовий символ.

Рядкові типи:

- AnsiString – рядковий тип який складається з символів AnsiChar і теоретично не має обмежень по довжині. Цей тип сумісний з рядками, що закінчуються нульовим символом.

- ShortString – максимальна довжина рядка 255 символів.

- WideString – даний рядок складається з символів WideChar.

- PChar – являє собою вказівник на рядок з завершуючим нульовим символом, що складається з символів типу Char.

- PAnsiChar – вказівник на рядок AnsiChar з завершальним нульовим символом.

- PWideChar – вказівник на рядок WideChar з завершальним нульовим символом.

Логічний тип даних Boolean визначає одне з двох значень: true (істинно) або false (хибно). Вони впорядковані: у false порядковий номер 0, у true порядковий номер 1.

2.4 Оператори

Оператори – це ті символи в тексті програми, за допомогою яких виконуються певні дії з даними різних типів. Найпростішим прикладом можуть бути оператори додавання, віднімання, множення і ділення арифметичних типів даних; іншим прикладом може бути оператор для доступу до певного елементу масиву.

Оператор присвоєння:

Number := 5;

Оператор порівняння:

if x = y

Оператор “не дорівнює” виглядає так:

if x <> y then DoSomething

В якості логічних операторів “і” та “або” в мові Object Pascal використовуються ключові слова and і or. В основному ці оператори використовуються як елементи оператора if або циклу. Наприклад:

if (Condition1) and (Condition2) then DoSomething;

while (Condition1) or (Condition2) do DoSomething;

Побітові оператори – це оператори які дозволяють працювати з окремими бітами заданої змінної. Найчастіше побітові оператори використовуються для зсуву бітів вправо чи вліво, їх інверсії, а також побітових операцій “і”, “або” та “виключаюче або” між двома числами. Оператори зсуву вліво і вправо в Object Pascal мають вигляд shl та shr відповідно. Решта - not and or і xor.

3. ВБУДОВАНІ ЕЛЕМЕНТИ

Основними вбудованими елементами, що були використані при створенні програмного продукту є:

1. Компонент TSpeedButton.

TObject→TPersistent→TComponent→TControl→TGraphicControl→
TSpeedButton

Модуль BUTTONS. Сторінка Палітри компонентів Additional.

Ця кнопка із зображенням може мати як залежну, так і незалежну фіксацію. Вона зручна для застосування у складі панелей інструментів. Поведінка цих кнопок багато в чому визначається властивістю: property GroupIndex: Integer.

Якщо `GroupIndex` рівний нулю, у кнопки взагалі немає фіксації в натиснутому стані і вона не залежить від решти кнопок. Кнопки в групі (тобто з однаковим ненульовим значенням `GroupIndex`) мають залежну фіксацію. Вона також залежить від властивості `property AllowAllUp: Boolean`, яке описує поведінку кнопок в групі, а саме: чи можуть всі кнопки одночасно бути віджаті. Якщо `AllowAllUp` рівне `False` (за умовчанням), натиснуту кнопку в групі можна відпустити, лише натиснувши іншу. Якщо `AllowAllUp` рівне `True`, кнопку можна відпустити повторним натисненням.

Якщо ви хочете фіксувати одну кнопку `TSpeedButton`, їй потрібно привласнити унікальний груповий індекс, а `AllowAllUp` встановити в `True`.

Оскільки в групі не можуть одночасно знаходитися кнопки з різним значенням цієї властивості, при натисненні кнопки і зміні `GroupIndex` властивість `AllowAllUp` "розсилається" (привласнюється) решті кнопок з тим же значенням `GroupIndex`. У групі не може бути натиснуто більш за одну кнопку. Визначає, чи натиснута кнопка, властивість: `property Down: Boolean`.

Ця властивість може змінюватися як системою, так і програмістом. Наприклад, якщо при запуску програми необхідно, щоб одна з кнопок вже була натиснутою, її властивість `Down` встановлюють в `True`.

Текст кнопки визначає властивість `Caption`. Компонент має ті ж правила і властивості малювання картинки, що і `TBitBtn`. Вони описуються властивостями `Glyph`, `NumGlyphs`, `Layout`, `Margin` і `Spacing`.

Для імітації клацання передбачений метод `Click`. Подвійне клацання для `TSpeedButton` можливе тільки на натиснутій кнопці – інакше він інтерпретується як звичайний. Описується властивістю: `property OnDblClick`.

Компонент `TMemo`. `TObject->TPersistent->TComponent->TControl->TWinControl-> TCustomEdit->TCustomMemo->TMemo`.

Модуль `STDCTRLS`. Сторінка Палітри компонентів `Standard`.

Компонент є багаторядковий редактор тексту. Вміст редактора представлений як об'єкт, що містить текст у вигляді набору рядків: `property Lines: TStrings`.

Текст в редакторові може вирівнюватися по лівому, правому краю і по центру: `property Alignment: TAlignment; TAlignment = (taLeftJustify, taRightJustify, taCenter)`.

При наборі тексту користувач може ввести різні символи, що управляють, зокрема, клавішами <Enter> і <TAB>. Ці символи можуть бути оброблені редактором, а можуть бути відразу передані формі. У випадку, якщо властивості:

- `property WantReturns: Boolean;`

- `property WantTabs: Boolean;`

обернені в `True`, символи передаються редакторові. Звернемо увагу на те, що якщо встановлене `WantTabs`, то за допомогою клавіатури передати фокус такому редакторові можна, а після цього віддати іншому компоненту – не можна. Якщо властивості рівні `False`, символи передаються формі. В цьому випадку для введення цих символів в редактора можна скористатися комбінаціями <Ctrl>+<Enter> і <Ctrl>+<Tab> відповідно.

Дві властивості відповідають за організацію прокрутки тексту у вікні редактора: `property Wordwrap: Boolean` – відповідає за поведінку редактора досягши правої межі під час набору тексту. Якщо властивість рівна `True`, то при цьому відбувається перехід на новий рядок. У разі `False` досягши правої межі відбувається горизонтальна прокрутка тексту і користувач може продовжувати набір; на новий рядок можна перейти, натиснувши <Enter>.

`property ScrollBars: TScrollStyle;`

`TScrollStyle = (ssNone, ssHorizontal, ssVertical, ssBoth) ;`

- встановлює наявність смуг прокрутки у вертикальному і горизонтальному напрямках. Якщо є горизонтальна смуга, то властивість `Wordwrap` втрачає сенс: замість перенесення відбувається прокрутка.

Для отримання повноцінного додатку – текстового редактора, в нього потрібно включити компонент TМемо і забезпечити засобами читання, запису і друку файлів, пошуку і заміни тексту і т.п.

Компонент TPaintBox. TObject->TPersistent->TComponent->TControl->TGraphicControl-> PaintBox.

Модуль EXTCTRLS. Сторінка Палітри компонентів System.

Найпростішою надбудовою над канвою служить компонент TPaintBox, призначений для малювання. Можна навіть розглядати його як канву, забезпечену атрибутами компоненту. Вона представлена властивістю: property Canvas: TCanvas.

Крім канви, компонент має свій колір (фону)

property Color: TColor;

і шрифт:

property Font: TFont;

Зобразити на канві компоненту можна, передбачивши обробник події:

property OnPaint: TNotifyEvent;

Компонент TStringGrid. TObject->TPersistent->TComponent->TControl->TWinControl-> TCustomControl->TCustomGrid->TDrawGrid->TStringGrid

Модуль GRIDS. Сторінка Палітри компонентів Additional.

Цей компонент реалізує можливості свого предка TDrawGrid стосовно таблиці рядків.

До складу компоненту доданий об'єкт класу TStrings, в якому зберігається вміст осередків. Він доступний як векторна властивість – двовимірний масив текстових рядків (розмірністю ColCount x RowCount), відповідних елементам таблиці: property Cells[ACol, ARow: Integer]: string.

Доступний і двовимірний масив об'єктів, відповідних осередкам: property Objects[ACol, ARow: Integer]: TObject.

Необхідно пам'ятати, що самих об'єктів в таблиці немає, і програміст

повинен створювати, а після закінчення використання таблиці видаляти об'єкти самостійно.

Можна працювати окремо як з рядками, так і із стовпцями таблиці. Властивості:

- property Cols[Index: Integer]: TString;
- property Rows[Index: Integer]: TString;

описують набори рядків (також у вигляді TString), що містять текст і об'єкти стовпців і рядків таблиці.

При значенні True властивості DefaultDrawing для цього компоненту відбувається виведення рядка у відповідному осередку. Отже якщо окрім тексту нічого відображати не потрібно, то міняти значення DefaultDrawing і визначати обробник події OnDrawCell не потрібно.

Відзначимо, що перенесення рядків і стовпців таблиці (при встановлених опціях goColMoving або goRowMoving) здійснюється разом з їх текстовими рядками.

Компоненти TOpenDialog і TSaveDialog. TObject→TPersistent→TCoinponent→TConimonDialog→TOpenDialog→TSaveDialog.

Модуль DIALOGS. Сторінка Палітри компонентів Dialogs.

Ці компоненти-діалоги призначені для вибору імені файлу, який буде надалі використаний для читання або запису. Розглянемо їх спільно, оскільки вони відрізняються тільки інтерпретацією деяких опцій.

Діалог може бути налаштований на представлення імен файлів якого-небудь одного типу або декількох типів. При цьому тільки ці типи відображаються в списку, і лише з них може зробити вибір користувач, за допомогою властивості: property Filter: string.

Формат рядка фільтру складається з двох частин. У першій задається короткий опис типу. У другій частині, яка відділяється символом '|', – маска пошуку потрібних файлів по розширенню. У список файлів потраплять тільки ті, які мають вказані розширення. Приклади завдання фільтру: 'Delphi

projectsl*.dpr' або 'All graphics files I *-bmp,*.ico,*.wmf '.

Таких пар рядків для різних типів у фільтрі може бути декілька, при цьому формати представлення фільтру в самій системі Windows і в Delphi мають одну відмінність. При виклику стандартних діалогів Windows ці пари рядків повинні розділятися нульовим байтом, після останнього рядка також повинен стояти нульовий байт. У даних компонентах всі рядки розділяються символом '|'.

Створювати рядки уручну потрібно лише в окремих випадках – для введення значення властивості на етапі проектування призначений спеціальний редактор. Є також простий спосіб скласти фільтр для графічних файлів – для цього призначена процедура GraphicFilter.

Як початковий фільтр при виклику діалогу буде вибрана та пара рядків, номер (індекс) якої співпадає із значенням властивості: `property PilterIndex: Integer`.

Якщо властивості не привласнювалося значення, то за умовчанням вибирається перша пара.

Шлях до файлів, які спочатку будуть відображені в діалозі, задається властивістю: `property InitialDir: string`.

У випадку, якщо в полі введення імені файлу користувач при ручному наборі не визначив його розширення, це за нього намагається зробити діалог. Для цього властивості: `property DefaultExt: TFileExt; TFileExt = string[3]`, привласнюється рядок (до трьох символів, без крапки), який і використовуватиметься як розширення за умовчанням. Наприклад, якщо користувач ввів в поле імені файлу 'mybitmap', а DefaultExt рівне 'bmp', то компонент поверне повне ім'я 'mybitmap. bmp'.

Поле введення FileName, де відображаються і редагуються імена файлів, може бути простим редагуючим елементом, а може бути і комбінованим списком. У цей список повинні потрапити імена файлів, які були раніше прочитані (записані). Стиль редагуючого елемента заданий

властивістю:

```
property FileEditStyle: TFileEditStyle;  
TFileEditStyle = (fsEdit, fsComboBox).
```

а вміст списку імен, що раніше поверталися, – властивістю:

```
property HistoryList: TStrings.
```

У разі дії стилю fsEdit ця властивість не грає ролі. Якщо ж встановлене fsComboBox, у випадному списку з'являється вміст HistoryList.

Можна створити список для цієї властивості під час розробки програми. Але для того, щоб воно дійсно грало роль "передісторії", програміст повинен поповнювати список після успішного закінчення діалогу, що ілюструється наступним фрагментом коду:

```
procedure TForm1-FileOpenClick(Sender: TObject);  
begin  
  with OpenFileDialog1 do if Execute then begin  
    Memol.Lines.LoadFromFile(FileName) ;  
    HistoryList.Add(FileName);  
  end;  
end;
```

У двох діалогів є великий набір опцій. Частина з них є загальною, частина – грає роль тільки для одного з діалогів:

```
property Options: TOpenOptions;
```

```
TOpenOption = (ofReadOnly, ofOverwritePrompt, ofHideReadOnly,  
ofNoChangeDir, ofShowHelp, ofNoValidate, ofAllowMultiSelect,  
ofExtensionDifferent, ofPathMustExist, ofFileMustExist, ofCreatePrompt,  
ofShareAware, ofNoReadOnlyReturn, ofNoTestFileCreate) ;
```

```
TOpenOptions = set of TOpenOption;
```

Три опції відповідають за роботу з файлами із статусом "тільки для читання":

– ofReadOnly – робить прапорець "Read only" поміченим при появі;

- ofHideReadOnly – ховає цей прапорець в діалозі, що з'являється;
- ofNoReadOnlyReturn – забороняє вибір файлів "тільки для читання", сповіщаючи про необхідність вибрати інший файл при натисненні ОК.

Також три опції обмежують введення імен для нових (неіснуючих) файлів:

- ofPathMustExist – указує на те, що файл може знаходитися тільки в одному з існуючих каталогів. У разі введення неіснуючого шляху до файлу користувач сповіщається про помилку;

- ofFileMustExist – аналогічним чином указує на те, що може бути вибраний тільки один з існуючих файлів;

- ofCreatePrompt – опція встановлює реакцію на попередню ситуацію. Якщо вона встановлена, то замість повідомлення про помилку виводиться запит на створення нового файлу.

Інші опції:

- ofOverwritePrompt – запрошує підтвердження, якщо користувач вибрав для запису вже існуючий файл;

- ofNoChangeDir – забороняє зміну початкового каталога, з яким діалог буде проініціалізований. Якщо вона встановлена, діалог кожного разу з'являється з тим каталогом, який був встановлений при першому запуску;

- ofShowHelp – включає до складу діалогу кнопку Help;

- ofNo Validate – вимикає перевірку введеного імені файлу на наявність в нім неприпустимих символів;

- ofAllowMultiSelect – дозволяє вибирати декілька файлів одночасно;

- ofShareAware – відключає перевірку на можливість сумісного доступу до вибраного файлу. У разі відсутності цієї опції файл не можна вибрати, якщо він відкритий іншим додатком;

- ofNoTestFileCreate – ця опція застосовується тільки для файлів на тих вузлах локальної мережі, яким дозволено створення, але не модифікацію файлів. Якщо вона встановлена, діалог не перевіряє можливість запису на

вибраному пристрої.

Нарешті, одна опція – `ofExtensionDifTereit` – є вихідний. Вона встановлюється після завершення діалогу в тому випадку, якщо розширення у введеного імені файлу відрізняється від того, яке визначене за умовчанням (у властивості `DefaultExt`).

Текст, що з'являється в заголовку діалогу, визначається властивістю: `property Title: string`.

Якщо воно не було задане, то в заголовках діалогів з'явиться "Open" для `TOpenDialog` і "SaveAs" для `TSaveDialog`.

У разі успішного завершення діалогу у властивості `property Files: TStringList`; містяться імена вибраних користувачем файлів. Якщо вибраний один файл, його ім'я можна отримати як `Files.Strings[0]`, але є для цього і окрема властивість: `property FileName: TFileName`;

4. ЗАСОБИ НАЛАГОДЖЕННЯ ПРОГРАМ

Покрокове налагодження.

Одна з найпоширеніших задач налагодження – виконання програми крок за кроком, по одному рядку за раз для перевірки правильності виконання. При покроковому проходженні коду програма налагодження виводить вікно редагування з виконуваною програмою. Точка виконання, що показує наступний виконуваний рядок програми, представляється у вигляді зеленої стрілки, розташованої зліва від області початкового тексту у вікні редагування. Після успішної компіляції модуля на смужі налагоджувальної інформації кожний рядок коду модуля буде позначений маленьким, синім кружком. Якщо ж рядок не помічений, значить, оптимізатор компілятора виключив дану команду з списку виконуваних. Оскільки для таких рядків виконуваний код не згенеровано, ці рядки не будуть позначені точкою виконання.

Інтегроване середовище Delphi надає користувачу декілька команд покрокового налагодження доступних в меню Run:

- Step Over (F8) – покрокове виконання рядків програми, зчитуючи виклик функції чи процедури за один рядок, тобто вхід в функції та процедури не виконується.
- Trace Info (F7) – покрокове виконання програми з входом у викликані процедури та функції.
- Trace to Next Source Line (Shift+F7) – перехід до наступного виконуваного рядку.
- Run to Cursor (F4) – команда виконує програму до того виконуваного оператора, на якому розташований курсор у вікні редактору коду.
- Show Execution Point – команда поміщає курсор на оператор, який буде виконуватись наступним.

Вікно спостереження Watches.

В складних додатках виникає необхідність перегляду значень декількох змінних під час виконання програми. Таку можливість надає вікно спостереження Watches. Зробити його видимим можна командою View | Debug Windows | Watches. Для додавання змінної в дане вікно необхідно підвести курсор до потрібної змінної та натиснути комбінацію клавіш Ctrl+F5. При покроковому виконанні програми у вікні спостереження будуть відображатися значення змінних, які було додані до нього.

ДЖЕРЕЛА ВИКОРИСТАНІ ПРИ РОЗРОБЦІ

- 1) Бобровский С. Developer Studio 2006. Учебный курс. – СПб.: Питер, 2004. – 735с.
- 2) Баас Р., Фервай М., Гюнтер Х.. Delphi 5: для пользователя. Пер. с нем. – К.: Издательская группа ВНУБ, 2000 – 496 с.
- 3) Керман, Митчелл, К. Программирование и отладка в Delphi. Учебный курс.: Пер. с англ. – М.: Вильямс, 2002. – 672 с.: ил.
- 4) Архангельский А.Я. Программирование в Delphi 5 – 2-е изд., переработ. и дополн. – М.: ЗАО «Издательство БИНОМ», 2000. – 1072 с.
- 5) Гофман В.Э., Хомоненко А.Д. Delphi 6. – СПб.: БХВ – Санкт-Петербург, 2000. –800с.

Затверджено

482.362. 6050102-02 13 09-1

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362. 6050102-02 13 09-1

(Опис програми)

Сторінок 10

АНОТАЦІЯ

Даний програмний документ містить загальні відомості про розроблену програму, її функціональне призначення, алгоритм роботи з описом використовуваних процедур. В документі також наведені способи виклику та завантаження програми, опис вхідних і вихідних даних та зображено робочий вигляд інтерфейсу програми.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ	4
1.1. Позначення та найменування програми.....	4
1.2. Програмне забезпечення, необхідне для функціонування програми	4
1.3. Мова програмування	4
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ.....	5
3. ОПИС СТРУКТУРИ ПРОГРАМИ	5
3.1. Опис функціональних можливостей й структури програми.....	5
3.4. Опис роботи з програмою	6
4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ	9
5. ВИКЛИК І ЗАВАНТАЖЕННЯ.....	9
6. ВХІДНІ ДАНІ	9
7. ВИХІДНІ ДАНІ	10

1. ЗАГАЛЬНІ ВІДОМОСТІ

1.1. Позначення та найменування програми

Програма розроблена згідно технічного завдання, яке затверджено на засіданні кафедри комп'ютерних систем і мереж, від 2015 року.

Умовне позначення: 482.362. 6050102-02 09-1

1.2. Програмне забезпечення, необхідне для функціонування програми

Програма написана на мові Object Pascal за допомогою інструментальних засобів Embarcadero Delphi XE.

Розроблена програма не використовує жодних додаткових програмних продуктів, а її функціонування забезпечується засобами операційних систем типу Windows.

Для функціонування розробленої програми в UNIX - подібних операційних системах необхідно вихідний програмний код відкомпілювати за допомогою компілятора Kilycs.

1.3. Мова програмування

Для вибору мови програмування проводився аналіз сучасних інструментальних засобів мов програмування. Розглядалися переваги та недоліки різних середовищ розробки програмних продуктів, що описано в документі «Опис мови».

Можна використовувати будь-яке середовище програмування, що дає можливість реалізувати зручний і доступний інтерфейс користувача.

Виходячи з функціональних характеристик програмного продукту на стадії проектування, пакет інструментальних засобів Embarcadero Delphi XE задовольняє наступним вимогам:

- простота синтаксису мови програмування;
- зручність середовища програмування;

- інформаційна підтримка;
- легкість програмування;
- підтримка роботи з рядками та масивами рядків та ін.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Основна мета курсового проекту є автоматизація обліку телевізорів.

Крім того, розроблена програма планується для використання в магазинах, торгових центрах, оптових базах та складах для зберігання техніки, а зокрема телевізорів.

3.ОПИС СТРУКТУРИ ПРОГРАМИ

3.1. Опис функціональних можливостей й структури програми

Функціональні можливості програми відповідають вимогам, які описані в документів «Технічне завдання». Можливості програми, а також особливості її використання показано у розділі 3.2.

Функціональність програми полягає у формуванні класу, що описує викладача, та похідного від нього класу – «Заробітня платня» - у якому ми вказуємо оклад та загальну суму зарплатні.

Вся інформація збережена у файлі формату json у вигляді JavaScript-хешу, що складається з єдиного елементу – масив об'єктів lecturers, які містять відповідні поля: ППП, звання, посада, курси, оклад, сума заробітку.

В основу програми покладено наступний алгоритм:

- формування двох класів для зберігання введеної інформації, головний клас та дочірній – для зберігання додаткових параметрів;
- введення інформації:
 - про ППП, звання, посада вручну та з файлу;
 - про курси, які читає;
- виведення інформації з файла та збереження інформації у файл;

У даній роботі представлено програму, яка має наступні функціональні характеристики:

- 1) зберігання інформації про викладачів та їх заробітну плату;
- 2) можливість зберігати у файл заповнену інформацію;
- 3) можливість відкривати з файлу збережену інформацію;
- 4) можливість експорту у txt та друк результуючого документу.

3.2. Опис роботи з програмою

Програма складається всього з одного вікна, що містить головну таблицю, яка відображає список викладачів, 4 групи елементів керування та кнопка друку.

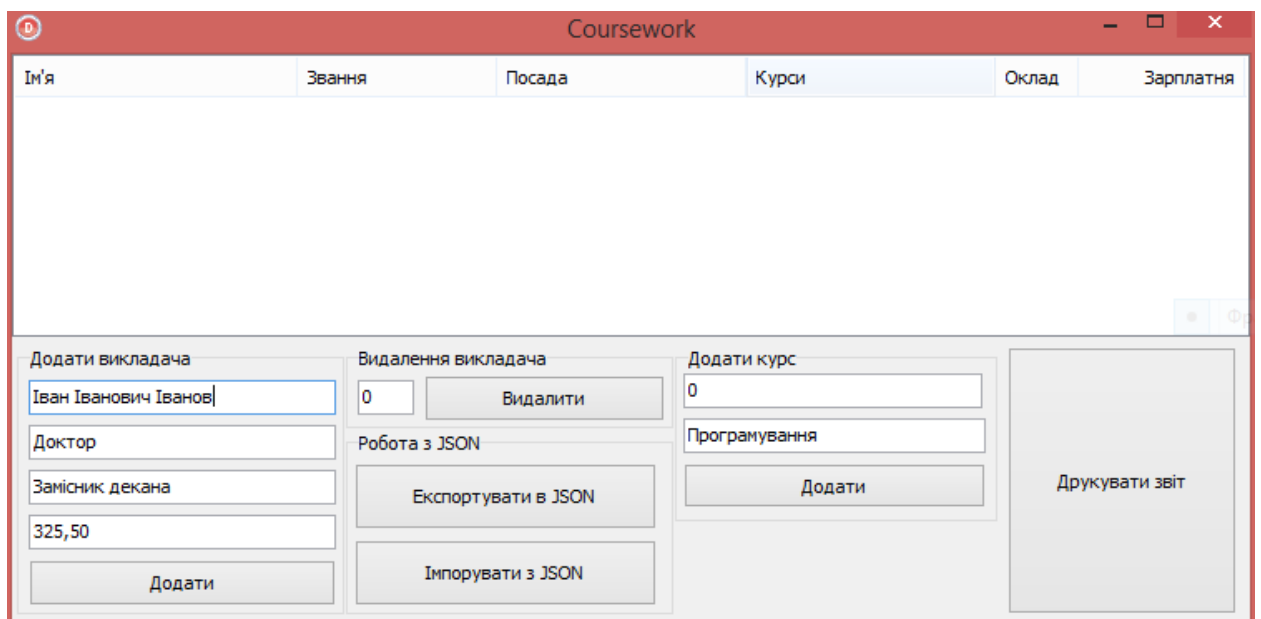


Рис. 3.1. Загальний вигляд інтерфейсу програми

Група «Додати викладача» містить 4 поля введення для ПІП, звання, посади та окладу відповідно. Окрім цього в групі присутня кнопка «Додати», по кліку на яку виконується додавання викладача.

Група «Видалення викладача» містить одне поле введення для ідентифікатора викладача, що відповідає порядку його в списку. Кнопка «Видалити» видаляє викладача з вказаним ідентифікатором.

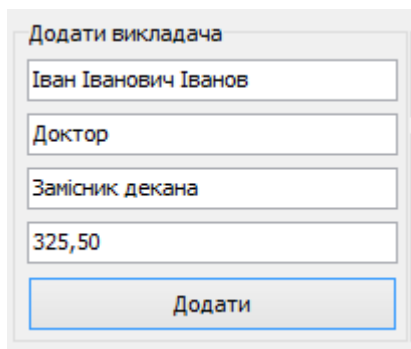


Рис. 3.2. Додавання викладача

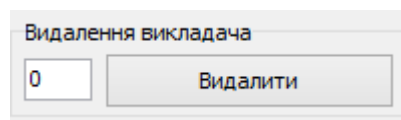


Рис. 3.3. Видалення викладача

Група «Додати курс» містить два поля введення: поле введення ідентифікатору викладача та поле введення назви курсу. Кнопка «Додати» додає викладачеві з вказаним ідентифікатором курсу, з ім'ям, що вказане у відповідному полі введення.

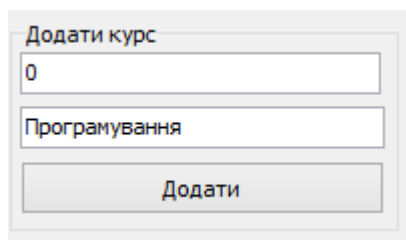


Рис. 3.4. Додавання курсу, який веде викладач.

Група «Робота з JSON» дозволяє імпортувати та експортувати данні з файлів JSON, що повинні бути відповідним чином оформлені. Кнопка «Експортувати в JSON» викликає діалог збереження, який дозволяє вказати шлях збереження згенерованого файлу. Кнопка «Імпортувати з JSON» викликає діалог збереження, що дозволяє обрати потрібний файл для експорту.

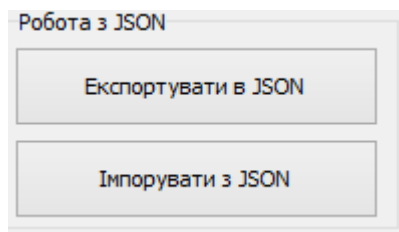


Рис. 3.5. Поле для роботи з форматом JSON.

The screenshot shows a window titled "Coursework" with a table of teachers and their salaries. Below the table are three main sections: "Додати викладача" (Add teacher), "Видалення викладача" (Delete teacher), and "Додати курс" (Add course). The "Додати викладача" section has input fields for name, title, position, and salary, with a "Додати" button. The "Видалення викладача" section has a text input, a "Видалити" button, and a section for "Робота з JSON" (JSON work) with "Експортувати в JSON" and "Імпортувати з JSON" buttons. The "Додати курс" section has a text input, a "Додати" button, and a "Друквати звіт" (Print report) button.

Ім'я	Звання	Посада	Курси	Оклад	Зарплатня
Іван Іванович Іванов	Доктор	Замісник декана	Програмування, Delphi, ...	325,5	976,5
Степан Іванович Срака	Аспірант	Асистент	C++	100	100
Петро Гаврилович Хабібулін	Доцент	Викладач	C++, Теорія алгоритмів	253,5	507
Микола Опанасович Буш	Професор	Декан	Теорія алгоритмів, Деге...	600	1800

Рис. 3.6. Виведення інформації про доданих викладачів

Виведення інформації відбувається в таблицю, що розбита стовбці «Ім'я», «Звання», «Посада», «Курси», «Оклад» та «Зарплатня». Комірки в стовпчику «Зарплатня» формуються як добуток кількості курсів на «Оклад».

4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

З метою забезпечення нормального функціонування програми необхідна наявність таких технічних засобів, як персональний комп'ютер, конфігурація якої приведена у таблиці. 1.1.

Таблиця 1.1.

Вимоги до конфігурації комп'ютера

Назва	Характеристика
Процесор (CPU)	6-покоління (AMD K6-2 300 МГц і вище, Intel Pentium Pro/II/Celeron 300 МГц і вище)
Оперативна пам'ять (RAM)	128 Мбайт
Об'єм вільного місця на жорсткому диску (HDD)	2 Мбайти

5. ВИКЛИК І ЗАВАНТАЖЕННЯ

Для завантаження програмного продукту потрібно подати на виконання файл CourseworkProject.exe.

У випадку, коли необхідно змінити функціональні можливості програмного продукту, а потім відкомпілювати програму, потрібно встановити середовище розробки Embarcadero Delphi XE.

6. ВХІДНІ ДАНІ

Вхідними даними вважається власне інформація викладача.

- ✓ ПП;
- ✓ звання;
- ✓ посада;
- ✓ оклад;

Додати викладача

Іван Іванович Іванов

Доктор

Замісник декана

325,50

Додати

7. ВИХІДНІ ДАНІ

Вихідними даними програми є динамічно сформована таблиця, що відображає інформацію про викладачів.

The screenshot shows a window titled "Coursework" with a table of teachers and a form for adding/editing data.

Ім'я	Звання	Посада	Курси	Оклад	Зарплатня
Іван Іванович Іванов	Доктор	Замісник декана	Програмування, Delphi, ...	325,5	976,5
Степан Іванович Срака	Аспірант	Асистент	C++	100	100
Петро Гаврилович Хабібунін	Доцент	Викладач	C++, Теорія алгоритмів	253,5	507
Микола Опанасович Буш	Професор	Декан	Теорія алгоритмів, Деге...	600	1800

Below the table, there is a form with the following sections:

- Додати викладача**: Fields for Name (Іван Іванович Іванов), Title (Доктор), Position (Замісник декана), and Salary (325,50). A "Додати" button is at the bottom.
- Видалення викладача**: A field with the value "0" and a "Видалити" button.
- Робота з JSON**: Buttons for "Експортувати в JSON" and "Імпорувати з JSON".
- Додати курс**: Fields for Course Name (0) and Course Description (Програмування). A "Додати" button is at the bottom.
- Друк звіту**: A button labeled "Друквати звіт".

Затверджено

482.362. 6050102-02 12 09-1

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362. 6050102-02 12 09-1

(Текст програми)

Сторінок 11

2015

АНОТАЦІЯ

В даному програмному документі представлено текст програми, розробленої в середовищі Embarcadero Delphi XE.

ТЕКСТ ПРОГРАМИ

```
unit MainUnit;

interface

uses

    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ComCtrls,
    DBXJSON,
    ShellAPI;

type
    TMainForm = class(TForm)
        NewLecturerGroup: TGroupBox;
        NameEdit: TEdit;
        GradeEdit: TEdit;
        OfficeEdit: TEdit;
        AddNewLecturerBtn: TButton;
        NewCourseGroup: TGroupBox;
        AddLecturerIDEdit: TEdit;
        CourseTitleEdit: TEdit;
        AddNewCourseBtn: TButton;
        RemoveLecturerGroup: TGroupBox;
        RemoveLecturerIDEdit: TEdit;
        RemoveLecturerBtn: TButton;
        SaveJsonDialog: TSaveDialog;
        WorkWithJSONGroup: TGroupBox;
        ExportAsJSONBtn: TButton;
        ImportFromJSONBtn: TButton;
        OpenJsonDialog: TOpenDialog;
        LecturersListView: TListView;
        SalaryEdt: TEdit;
        PrintBtn: TButton;
        procedure AddNewLecturerBtnClick(Sender: TObject);
        procedure AddNewCourseBtnClick(Sender: TObject);
        procedure RemoveLecturerBtnClick(Sender: TObject);
        procedure ExportAsJSONBtnClick(Sender: TObject);
        procedure ImportFromJSONBtnClick(Sender: TObject);
        procedure PrintBtnClick(Sender: TObject);
    private
        { Private declarations }
    end;
```


482.362. 6050102-02 12 09-1

```

public
    { Public declarations }
end;

TLecturer = class(TObject)
public
    Name: string[40];
    Grade: string[40];
    Office: string[40];
    Courses: array of string;
    constructor Create(const Name, Grade, Office: string);
    constructor CreateFromJson(const JSONObject: TJSONObject); virtual;
    destructor Destroy;
    function ToJSON(): TJSONObject; virtual;
end;

TWage = class(TLecturer)
public
    Salary: Real;
    Sum: Real;
    constructor Create(const Name, Grade, Office: string; const Salary:
Real);
    constructor CreateFromJson(const JSONObject: TJSONObject); override;
    function ToJSON(): TJSONObject; override;
end;

var
    MainForm: TMainForm;
    Lecturers: array of TWage;

implementation

constructor TLecturer.Create(const Name, Grade, Office: string);
begin
    Self.Name := Name;
    Self.Grade := Grade;
    Self.Office := Office;
end;

constructor TLecturer.CreateFromJson(const JSONObject: TJSONObject);
var
    JSONCourses: TJSONArray;

```

```
Name, Grade, Office: string;
i: Integer;
begin
  try
    Name := JSONObject.Get('name').JsonValue.Value;
    Grade := JSONObject.Get('grade').JsonValue.Value;
    Office := JSONObject.Get('office').JsonValue.Value;
    Self.Create(Name, Grade, Office);

    JSONCourses := JSONObject.Get('courses').JsonValue as TJSONArray;

    SetLength(Self.Courses, JSONCourses.Size);
    for i := 0 to JSONCourses.Size - 1 do
      Self.Courses[i] := JSONCourses.Get(i).Value;
    except
      raise Exception.Create('Помилка парсингу JSON!');
    end;
  end;
end;

function TLecturer.ToJSON(): TJSONObject;
var
  JSONObject: TJSONObject;
  JSONArray: TJSONArray;
  i: Integer;
begin
  JSONObject := TJSONObject.Create;

  JSONObject.AddPair(TJSONPair.Create('name', Self.Name));
  JSONObject.AddPair(TJSONPair.Create('grade', Self.Grade));
  JSONObject.AddPair(TJSONPair.Create('office', Self.Office));

  JSONArray := TJSONArray.Create;
  for i := 0 to Length(Self.Courses) - 1 do
    begin
      JSONArray.Add(Self.Courses[i]);
    end;

    JSONObject.AddPair(TJSONPair.Create('courses', JSONArray));

  Result := JSONObject;
end;
```

```
destructor TLecturer.Destroy;
begin
    // Nothing to do
    inherited;
end;

constructor TWage.Create(const Name: string; const Grade: string;
    const Office: string; const Salary: Real);
begin
    inherited Create(Name, Grade, Office);
    Self.Salary := Salary;
    Self.Sum := 0;
end;

constructor TWage.CreateFromJson(const JSONObject: TJSONObject);
begin
    inherited CreateFromJson(JSONObject);

    try
        Self.Salary := StrToFloat(JSONObject.Get('salary').JsonValue.Value);
        Self.Sum := StrToFloat(JSONObject.Get('sum').JsonValue.Value);
    except
        raise Exception.Create('Помилка парсингу JSON.');
```

end;

```
end;

function TWage.ToJSON: TJSONObject;
var
    JSONObject: TJSONObject;
begin
    JSONObject := inherited ToJSON();

    JSONObject.AddPair(TJSONPair.Create('salary', FloatToStr(Self.Salary)));
    JSONObject.AddPair(TJSONPair.Create('sum', FloatToStr(Self.Sum)));

    Result := JSONObject;
end;

{$R *.dfm}
```

482.362. 6050102-02 12 09-1

```

procedure RefreshLecturersListBox;
var
  i, j: Integer;
  NewRow: TListItem;
  Courses: string;
begin
  MainForm.LecturersListView.Items.Clear;

  for i := 0 to Length(Lecturers) - 1 do
    begin
      Courses := '';
      for j := 0 to Length(Lecturers[i].Courses) - 2 do
        Courses := Courses + Lecturers[i].Courses[j] + ', ';
      if (Length(Lecturers[i].Courses) >= 1) then
        Courses := Courses + Lecturers[i].Courses
          [Length(Lecturers[i].Courses) - 1];

      NewRow := MainForm.LecturersListView.Items.Add;
      NewRow.Caption := Lecturers[i].Name;
      NewRow.SubItems.Add(Lecturers[i].Grade);
      NewRow.SubItems.Add(Lecturers[i].Office);
      NewRow.SubItems.Add(Courses);
      NewRow.SubItems.Add(FloatToStr(Lecturers[i].Salary));
      NewRow.SubItems.Add(FloatToStr(Lecturers[i].Sum));
    end;
  end;

procedure TMainForm.AddNewCourseBtnClick(Sender: TObject);
var
  LecturerID, Len: Integer;
begin
  LecturerID := StrToInt(AddLecturerIDEdit.Text);

  Len := Length(Lecturers[LecturerID].Courses);
  SetLength(Lecturers[LecturerID].Courses, Len + 1);

  Lecturers[LecturerID].Courses[Len] := CourseTitleEdit.Text;
  Lecturers[LecturerID].Sum := Length(Lecturers[LecturerID].Courses) *
    Lecturers[LecturerID].Salary;

  RefreshLecturersListBox;

```

```
end;
```

```
procedure TMainForm.AddNewLecturerBtnClick(Sender: TObject);
```

```
var
```

```
    NewLecturer: TWage;
```

```
    Name, Grade, Office, Salary: string;
```

```
    Len: Integer;
```

```
begin
```

```
    Name := NameEdit.Text;
```

```
    Grade := GradeEdit.Text;
```

```
    Office := OfficeEdit.Text;
```

```
    Salary := SalaryEdt.Text;
```

```
    Len := Length(Lecturers);
```

```
    SetLength(Lecturers, Len + 1);
```

```
    Lecturers[Len] := TWage.Create(Name, Grade, Office, StrToFloat(Salary));
```

```
    RefreshLecturersListBox;
```

```
end;
```

```
procedure TMainForm.ExportAsJSONBtnClick(Sender: TObject);
```

```
var
```

```
    JSONObject: TJSONObject;
```

```
    JSONArray: TJSONArray;
```

```
    str: TStringList;
```

```
    i: Integer;
```

```
begin
```

```
    if SaveJsonDialog.Execute then
```

```
    begin
```

```
        try
```

```
            str := TStringList.Create;
```

```
            JSONObject := TJSONObject.Create;
```

```
            JSONArray := TJSONArray.Create;
```

```
            for i := 0 to Length(Lecturers) - 1 do
```

```
            begin
```

```
                JSONArray.AddElement(Lecturers[i].ToJSON);
```

```
            end;
```

```
            JSONObject.AddPair(TJSONPair.Create('lecturers', JSONArray));
```

482.362. 6050102-02 12 09-1

```
        str.Add(JSONObject.ToString);
        str.SaveToFile(SaveJsonDialog.FileName);
    finally
        str.Free;
        JSONObject.Destroy;
    end;
end;
end;

procedure TMainForm.ImportFromJSONBtnClick(Sender: TObject);
var
    JSONObject: TJSONObject;
    JSONLecturers: TJSONArray;
    str: TStringList;
    i: Integer;
begin
    if OpenJsonDialog.Execute then
    begin
        try
            str := TStringList.Create;
            str.LoadFromFile(OpenJsonDialog.FileName);

            JSONObject := TJSONObject.ParseJSONValue(str.Text) as TJSONObject;

            JSONLecturers := JSONObject.Get('lecturers').JsonValue as TJSONArray;

            SetLength(Lecturers, JSONLecturers.Size);

            for i := 0 to JSONLecturers.Size - 1 do
                Lecturers[i] := TWage.CreateFromJson(JSONLecturers.Get(i)
                    as TJSONObject);

                RefreshLecturersListBox;
            finally
                str.Free;
            end;
        end;
    end;
end;

procedure TMainForm.PrintBtnClick(Sender: TObject);
var
```

482.362. 6050102-02 12 09-1

```

i, j: Integer;
Name, Grade, Office, Courses, Salary, Sum, str: String;
OutFile: TextFile;
begin
  try
    AssignFile(OutFile, 'print.txt');
    Rewrite(OutFile);

    for i := 0 to LecturersListView.Items.Count - 1 do
      begin
        Name := LecturersListView.Items[i].Caption;
        Grade := LecturersListView.Items[i].SubItems.Strings[0];
        Office := LecturersListView.Items[i].SubItems.Strings[1];
        Courses := LecturersListView.Items[i].SubItems.Strings[2];
        Salary := LecturersListView.Items[i].SubItems.Strings[3];
        Sum := LecturersListView.Items[i].SubItems.Strings[4];

        str := Format('%-30s', [Name]) + Format('%-10s', [Grade]) +
          Format('%-20s', [Office]) + Format('%-40s', [Courses]) +
          Format('%-10s', [Salary]) + Format('%10s', [Sum]);
        Writeln(OutFile, str);
      end;
    finally
      CloseFile(OutFile);
    end;

    ShellExecute(Handle, 'print', 'print.txt', nil, nil, SW_Hide);
  end;

  procedure TMainForm.RemoveLecturerBtnClick(Sender: TObject);
  var
    LecturerID, i: Integer;
  begin
    LecturerID := StrToInt(RemoveLecturerIDEdit.Text);

    Lecturers[LecturerID].Destroy;
    for i := LecturerID to Length(Lecturers) - 1 do
      begin
        Lecturers[i] := Lecturers[i + 1];
      end;
    end;
  end;

```

```
    SetLength(Lecturers, Length(Lecturers) - 1);  
    RefreshLecturersListBox;  
end;  
  
end.
```


Затверджено

482.362.6050102-02 51 09-1 ЛЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362.6050102-02 51 09-1

(Програма та методика випробування)

Сторінок 6

2015

АНОТАЦІЯ

Програма та методика випробувань визначена ГОСТ 19.101-77, має структуру і оформлення, які встановлені ГОСТ 19.105-78.

В даному програмному документі розглядається розроблений програмний продукт як об'єкт випробувань. Розглянуті мета випробувань, вимоги до програми та програмної документації, а також засоби та методи, порядок випробувань.

ЗМІСТ

1. ОБ'ЄКТ ВИПРОБУВАНЬ.....	4
2. МЕТА ВИПРОБУВАНЬ	4
3. ВИМОГИ ДО ПРОГРАМИ.....	5
4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	5
5. ЗАСОБИ І ПОРЯДОК ВИПРОБУВАНЬ.....	6
6. МЕТОДИ ВИПРОБУВАНЬ.....	6

1. ОБ'ЄКТ ВИПРОБУВАНЬ

В процесі проведення випробовувань розроблена програма зі своїми функціональними можливостями виступає як об'єкт випробовувань.

Розробленій програмі присвоєно код: 482.362.6050102-02 09-1.

Методологія випробувань програмного забезпечення, пояснює суть методів і способів тестування та випробування програми, які доводять програму до робочого стану. У даному випадку, розроблена програма це і є програмне забезпечення, яке підтверджує дану тему роботи, тобто ПЗ, яке розроблено у даній роботі виступає як об'єкт випробувань.

Розроблена програма складається з чотирьох модулів, де під модулем розуміють елементи програми котрі стандартизовані за формою запису й зовнішніми зв'язками. Кожен модуль призначений для вирішення певної задачі, але вирішення всіх задач в межах даної програми.

2. МЕТА ВИПРОБУВАНЬ

Надійність роботи апаратних засобів слід розглядати в комплексі з надійністю програмного забезпечення.

Будь-який програмний продукт може мати в собі невиявлені помилки, адже важко протестувати або передбачити всі помилки та його реакцію на всі можливі комбінації вхідних даних, котрі програма опрацьовує. В загальному випадку під помилкою розуміють неправильність, похибку або навмисне спотворення об'єкта чи процесу. Хоча компілятор і виправляє деякі помилки, але передбачити можливі значення різних змінних він не в змозі.

Враховуючи вище сказане будь-який розроблений програмний продукт чи комплекс необхідно випробовувати. Тому випробовуванню підлягає і даний розроблений продукт.

При розробленні програми аналізувалася сукупність можливих вихідних даних, можливих і допустимих кінцевих проміжних результатів.

3. ВИМОГИ ДО ПРОГРАМИ

Вимоги до функціональних характеристик розробленої програми поставлені у технічному завданні на виконання курсового проекту. Відповідно до них програма повинна забезпечувати збереження інформації, введеної користувачем, та можливість ефективного її опрацювання. В той же час, програма повинна забезпечувати зручний та інтуїтивний графічний інтерфейс користувачу, легкість в роботі й виконувати наступні функції:

- забезпечення вводу інформації з клавіатур;
- забезпечити вивід інформації на екран у табличній формі;
- видалення даних;
- запис даних про об'єкти у файл та їх читання з файлу;
- інформування користувача в процесі роботи з програмою.

4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна включати наступні програмні документи:

- специфікація;
- технічне завдання;
- пояснювальна записка;
- опис мови;
- опис програми;
- текст програми;
- програма та методика випробувань.

Програмні документи оформлені на аркушах формату А4 та включають частини тексту програми оформлені у відповідності до правил мови, на якій написана програма, а також призначення та область застосування розробки, технічні характеристики, очікувані техніко – економічні показники та джерела літератури.

5. ЗАСОБИ І ПОРЯДОК ВИПРОБУВАНЬ

Програма була випробувана розробником для виявлення помилок у функціонуванні коду програми. До основних випробовувань необхідно віднести експертний аналіз (порівняння) вихідних результатів програмного продукту із реально присутніми. У випадку, коли величини співпадають, програма функціонує правильно, якщо ні, то необхідно перевірити код програми.

Надійність програми визначається як властивість програми виконувати задані функції у заданих умовах роботи і на заданому персональному комп'ютері. Тому розроблена програма тестувалася у всіх режимах роботи на комп'ютерах кафедри КСМ.

6. МЕТОДИ ВИПРОБУВАНЬ

До основних методів, щодо випробувань та досягнення відповідного рівня надійності програми при її випробовуванні та виявленні помилок є:

- 1) Уникнення помилок. Ця методика виконання дала нам можливість забезпечити мінімізацію помилок, що виникали в процесі створення програми.
- 2) Виявлення помилок. Ця методика базувалась на засобах і методах, котрі забезпечували виявлення помилок в програмі, що розроблялася.
- 3) Виправлення помилок. Згідно цього методу на основі конструювання і методології використання функцій, що коректували виправлені помилки та усували їх.
- 4) Допущення помилок. Забезпечувалося засобами і методами, котрі дають можливість виконання заданих функцій при наявності помилок.

Розглянувши методології, які наведено вище, зрозуміло, що уникнення помилок є оптимальним підходом в досягненні надійності ПЗ, що розроблявся.

ВИСНОВКИ

В курсовому проекті:

- проведено огляд літератури з поставленої задачі;
- розроблено програму, що реалізує поставлене завдання;
 - створено головний клас «Викладач»;
 - створено дочірній клас «Зар.плата»
- створено інструментарій для запису даних об'єктів у файл та їх читання;
- інтерфейс програми реалізований на мові Object Pascal візуальними засобами розробки середовища програмування Embarcadero Delphi XE.

За результатами написання програмного продукту можна сказати, що розробка пройшла успішно і програма може бути допущена до тестування в реальних умовах експлуатації. Програма дозволяє успішно заповнювати всі поля інформацією про конкретного викладача, а потім додавати цю інформацію до загального списку та записувати список у файл. Також програма дає можливість відкривати збережений список з файлу створеного цією програмою або її копією.