МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Відділ комп'ютерних технологій
Кафедра програмного забезпечення комп'ютерних систем

ІС реєстрації та контролю повідомлень учасників форуму

(літня технологічна практика)

«УЗГОДЖЕНО»
Керівник проекту
_____Комісарчук В.В.
(підпис)

Виконав студент 443ск групи
_____ Корольов М.С
(підпис)

Екзаменаційна оцінка з технологічної практики:

№ пп	Прізвище, ініціали студента	Підсумкова оц. за шкалою ВНЗ	Підсумкова оц. за шкалою ECTS	Підсумкова оц. за національ. шкалою	Підпис керівника практики
1.					
2.					
3.					

«___»_____ 201__ р.

Чернівці, 2019

ЗМІСТ

1. ТЕХНІЧНЕ ЗАВДАННЯ…….....6
1.1. Призначення розробки…….....6
1.1.1. Тема технологічної практики……6
1.1.2. Основні функціональні можливості…….....6
1.2. Аналіз вимог до програмного забезпечення……6
1.2.1. Функціональні вимоги.....6
1.2.2. Вимоги до складу та параметрів технічних засобів……7
1.2.3. Вимоги до інтерфейсу користувача……...7
1.2.4. Вимоги до інформаційної та програмної сумісності…….....7
2. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ……...10
2.1. Опис функціональних можливостей……...10
2.2. Опис та обґрунтування обраної архітектури……..10
2.3. Опис діаграми модулів……..12
2.4. Математичне забезпечення……..13
2.4.1. Алгоритм LZW……..13
2.4.2. Алгоритм LZ77……..14
2.4.3. Алгоритм LZSS……..16
3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……..20
3.1. Опис і обґрунтування обраних програмних засобів……..20
3.2. Опис графічного інтерфейсу користувача…….……20
ВИСНОВКИ……...28
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…….29
ДОДАТКИ……..30
Додаток А. Код програми…….30

1. ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Призначення розробки

1.1.1 Тема технологічної практики

Розробка програмного забезпечення виконується на підставі рішення засідання кафедри ПЗКС про затвердження тем на технологічну практику.

Назва отриманої теми «ІС система реєстрації та контролю повідомлень учасників форума».

1.1.2 Основні функціональні можливості

Розроблене програмне забезпечення повинно забезпечити стиснення інформації за допомогою алгоритмів сімейства LZ: LZW (Лемпеля-Зіва-Велча), LZ77 та LZSS.

1.2 Аналіз вимог до програмного забезпечення

1.2.1 Функціональні вимоги

Програмне забезпечення, що було розроблене в ході виконання технологічної практики, повинно забезпечувати наступні функціональні вимоги:

1) Працювати в будь-якому браузері;
2) Забезпечити контроль повідомлень;
3) ведення логу роботи програми;
4) збереження вхідних та вихідних даних.

1.2.2 Вимоги до складу та параметрів технічних засобів

Програма повинна працювати на комп’ютерах, де стабільно працює операційна система Windows XP. Інших вимог до апаратної частини не висувається.

1.2.3 Вимоги до інтерфейсу користувача

Програма повинна мати інтуїтивно зрозумілий, зручний та візуально-приємний інтерфейс.

1.2.4 Вимоги до інформаційної та програмної сумісності

Програма повинна працювати на комп’ютерах, де встановлена операційна система Windows XP (SP1, SP2, SP3, Seven).

2. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ

2.1 Опис функціональних можливостей

Програмне забезпечення, що було розроблене в процесі написання технологічної практики, забезпечує наступні функціональні можливості, що реалізують поставлене завдання:

1. Введення логу роботи програми.
2. Контроль вхідних та вихідних повідомлень.
3. Створення тем для повідомлень.

2.2 Опис та обґрунтування обраної архітектури

У розробленому програмному забезпеченні користувач має змогу виконувати два основних процеси в залежності від мети використання

програми. Це може бути архівація, або роз-архівація інформації. Основні можливості програми зручно зобразити у вигляді нижченаведених блок-схем (рис.2.2.1):

Рисунок 2.2.1 – Використання програми для архівації інформації

Даний режим роботи програми використовується для стиснення інформації та збереження за архівованого файлу. Після завершення процесу архівації користувач може зберегти стиснену інформацію у файл та зберегти лог роботи програми.

Розроблене програмне забезпечення також можна використовувати для роз-архівації даних (рис.2.2.2):

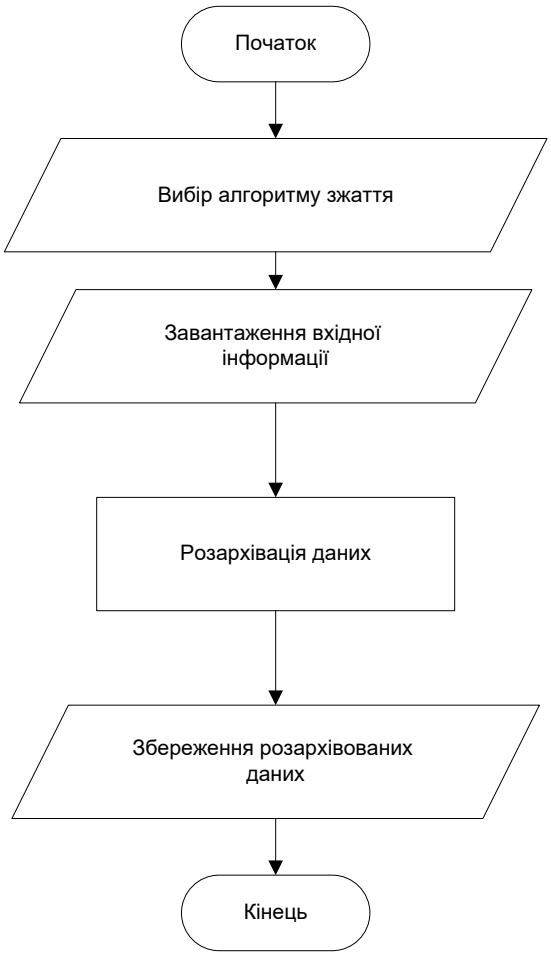

Рисунок 2.2.2 – Використання програми для розархівування інформації

І так далі…

2.3 Опис діаграми модулів

Робота розробленого програмного продукту реалізується наступними файлами модулів:

1. Модуль main_unit.
2. Модуль LZSS_Compress.
3. Модуль LZSS_Uncompress.
4. Модуль Project1.

Модуль main_unit є головним модулем розробленого програмного забезпечення. В ньому містяться функції, що реалізують алгоритми стиснення без втрат LZW та LZ77, функції для обробки помилок та ведення логу

програми. Також в даному модулі містяться функції, що відповідають за взаємодію користувача з інтерфейсом програми.

Модулі LZSS_Compress та LZSS_Uncompress забезпечують архівацію та розархівування даних за алгоритмом LZSS.

Модуль Project1 відповідає за автоматичне створення вікон програми та ініціалізацію програми в середовищі Windows.

На рис.2.3.1. представлено діаграму модулів розробленого програмного продукту.

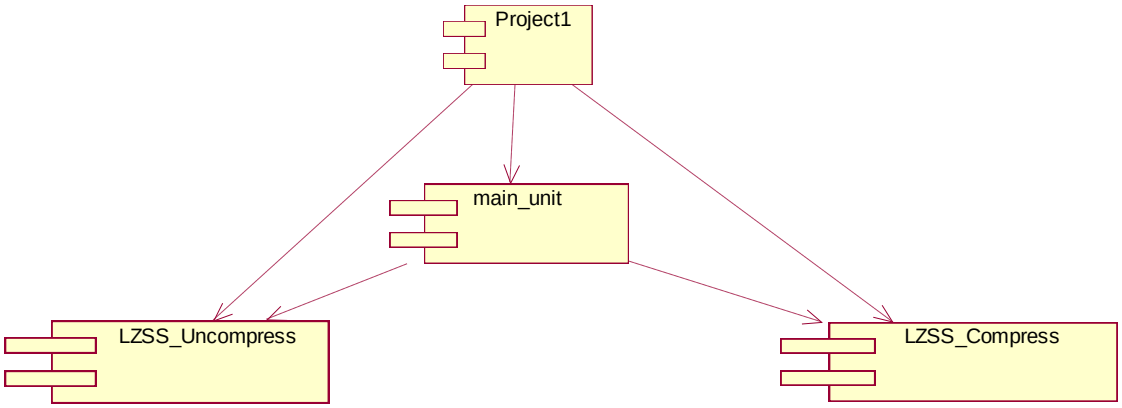

Рисунок 2.3.1 – Діаграма модулів

І так далі…

2.4 Математичне забезпечення

2.4.1 Алгоритм LZW

Власне вихідний Lempel / Ziv підхід до стиснення даних був вперше оприлюднений в 1977р., А вдосконалений (Terry Welch) варіант був опублікований в 1984г. Алгоритм напрочуд простий. Якщо в двох словах, то LZW-стиснення замінює рядки символів деякими кодами. Це робиться без будь-якого аналізу вхідного тексту. Замість цього при додаванні кожного

нового рядка символів проглядається таблиця рядків. Стискання відбувається, коли код замінює рядок символів. Коди, що генеруються LZW-алгоритмом, можуть бути будь-якої довжини, але вони повинні містити більше біт, ніж одиничний символ. Перші 256 кодів (коли використовуються 8-бітові символи) за умовчанням відповідають стандартного набору символів. Решта коди відповідають оброблюваним алгоритмом рядках.

Ядром LZW-алгоритму є словник, який містить всі символи вхідного алфавіту і деякі рядки з цих символів. До початку роботи алгоритму в словник входять елементи, як рядків містять кожен із символів вхідного алфавіту. У міру роботи алгоритму, що зустрічаються у вхідному потоці ланцюжка символів заносяться в словник. При цьому у вихідний потік заноситься номер елемента словника, що містить рядок максимальної довжини, збігається з рядком вхідного потоку. Словник організований як сукупність двійкових дерев і містить по одному двійковому дереву на кожен символ вхідного алфавіту. Кожен елемент словника є в той же час вершиною одного з двійкових дерев.

Вершини дерев є елементами словника, відповідними символам вхідного алфавіту. Кожна вершина має одну вхідну дугу і до двох вихідних. Перша з них вказує на вершину, яка відповідає рядку, таку ж, як дана, але на один символ довшу (зв’язок <рівне>), друга – на вершину, відповідну рядку, що має таку ж довжину, як дана, але відрізняється останнім символом (зв'язок <не дорівнює>). У кожній вершині графа міститься також один символ. Паралельно вибірці символів з вхідного потоку йде обхід одного з дерев.

Алгоритм обходу наступний:

1) вибирається перший символ з вхідного потоку;
2) вибирається відповідне йому двійкове дерево в словнику;
3) вибирається наступний символ з вхідного потоку;
4) далі для кожної вершини двійкового дерева якщо символ збігся з символом у вершині перейти до наступного вершині по зв'язку <одно>; взяти з вхідного потоку наступний символ;

5) інакше перейти до наступного вершині по зв'язку <не дорівнює>, продовжувати з тим же символом;

6) якщо посилання далі немає, пошук завершується, одна з порожніх вершин заповнюється, додається відповідна посилання на неї з поточної вершини.

2.4.2 Алгоритм LZ77

LZ77 – алгоритм стиснення без втрат, опублікований в статтях Абрахама Лемпеля і Якоба Зеева в 1977 і 1978 роках. Також LZ77 є алгоритмом зі «ковзним вікном», що еквівалентно неявному використанню словникового підходу.

Для розуміння даного алгоритму необхідно розібратися з двома його складовими: принципом ковзного вікна і механізмом кодування збігів.

Метод кодування згідно з принципом ковзаючого вікна враховує уже раніше оброблену інформацію, тобто інформацію, яка вже відома для кодувальника і декодувальник (друге і наступні входження деякої рядка символів в повідомленні замінюються посиланнями на її перше входження).

Завдяки цьому принципу алгоритми LZ іноді називаються методами стиснення з використанням ковзного вікна. Ковзаюче вікно можна представити у вигляді буфера (або більш складної динамічної структури даних), який організований так, щоб запам'ятовувати «сказану» раніше інформацію і надавати до неї доступ. Таким чином, сам процес стискає кодування згідно LZ77 нагадує написання програми, команди якої дозволяють звертатися до елементів «ковзаючого вікна», і замість значень стисливою послідовності вставляти посилання на ці значення в «ковзному вікні». Розмір ковзаючого вікна може динамічно змінюватися і складати 2, 4 або 32 кілобайт. Слід також зазначити, що розмір вікна кодувальника може бути менше або дорівнює розміру вікна декодувальник, але не навпаки.

Наведене вище порівняння процесу кодування з «програмуванням» може наштовхнути на передчасний висновок про те, що алгоритм LZ77 відноситься до методів контекстного моделювання. Тому слід зазначити, що алгоритм LZ77 прийнято класифікувати як метод словникового стиснення даних, коли замість поняття «ковзаючого вікна» використовується термін «динамічного словника».

Перед тим, як перейти до розгляду механізму кодування, уточнимо поняття збігу (від англ. Match). Розглянемо послідовність з N елементів. Якщо всі елементи послідовності унікальні, то така послідовність не буде містити жодного повторюваного елемента, або, інакше кажучи, в послідовності не знайдеться хоча б двох рівних один одному або співпадаючих елементів.

У стандартному алгоритмі LZ77 збігу кодуються парою:

1) довжина збігу (match length);

2) зміщення (offset) або дистанція (distance).

В продовження уже наведеної аналогії з програмуванням відзначимо, що в більшості статей, присвячених алгоритму LZ77, кодована пара трактується саме як команда копіювання символів з ковзного вікна з певної позиції, або дослівно як: «Повернутися в буфері символів на значення зсуву і скопіювати значення довжини символів , починаючи з поточної позиції ».

Хоча для прихильників імперативного програмування така інтерпретація може здатися інтуїтивно зрозумілою, вона, на жаль, мало говорить про сутність алгоритму LZ77 як методу стиснення. Особливість даного алгоритму стиснення полягає в тому, що використання кодованої пари довжина-зміщення є не тільки прийнятним, але й ефективним в тих випадках, коли значення довжини перевищує значення зміщення.

Приклад з командою копіювання не зовсім очевидний: «Повернутися на 1 символ назад в буфері і скопіювати 7 символів, починаючи з поточної позиції». Яким чином можна скопіювати 7 символів з буфера, коли зараз в буфері знаходиться лише 1 символ. Однак наступна інтерпретація кодує пари може прояснити ситуацію: кожні 7 наступних символів збігаються

(еквівалентні) з 1 символом перед ними.

Це означає, що кожен символ можна однозначно визначити, перемістившись назад в буфері – навіть якщо цей символ ще відсутня в буфері на момент декодування поточної пари довжина-зсув. Така кодована пара представлятиме собою багаторазове (визначається значенням зсуву) повторення послідовності (визначається значенням довжини) символів, що являє собою більш загальну форму RLE (кодування повторів).

До недоліків даного алгоритму можна віднести:

1) неможливість кодування підрядків, віддалених один від одного на відстані, більшій довжини словника;

2) довжина підрядка, яку можна закодувати, обмежена розміром буфера;

3) мала ефективність при кодуванні незначного обсягу даних.

2.4.3 Алгоритм LZSS

LZSS – модифікація LZ77, була розроблена Сторером (Storer) і Сжіманскі (Szymanski) в 1982. Базовий алгоритм був поліпшений за трьома напрямками: випереджуючий буфер зберігався в циклічній черзі, буфер пошуку (словник) зберігався у вигляді дерева двійкового пошуку і мітки мали два поля, а не три.

Двійкове дерево пошуку – це двійкове дерево, в якому ліве піддерево кожного вузла А містить вузли, менші ніж Л, а вузли правого піддерева все більше А. Оскільки вузли нашого двійкового дерева складаються з рядків (або слів), перш за все необхідно визначитися, як ці рядки слід порівнювати, що означає, один рядок «більше» іншого. Це легко зрозуміти, уявивши собі рядки в словнику, де вони впорядковані за алфавітом. Ясно, що рядок rote передує рядку said, так як буква г стоїть раніше букви s (хоча о і слідує після а). Ми будемо вважати, що рядок rote менше рядка said. Таке впорядкування рядків називається лексикографічним.

А як бути з рядком «_abc»? Фактично всі сучасні комп'ютери використовують код ASCII для представлення символів (хоча все більшого поширення набувають коди Unicode, а деякі старі великі комп'ютери IBM, Amdahl, Fujitsu, Siemens працюють зі старими 8-бітними кодами EBCDIC, розробленими в IBM). У кодах ASCII символ пробілу передує символам букв, тому рядок, що починається з пробілу буде вважатися менше будь-якого рядка, на початку якої стоїть буква. У загальному випадку сортування послідовностей в комп'ютері визначає послідовність символів, впорядкованих від меншого до більшого.

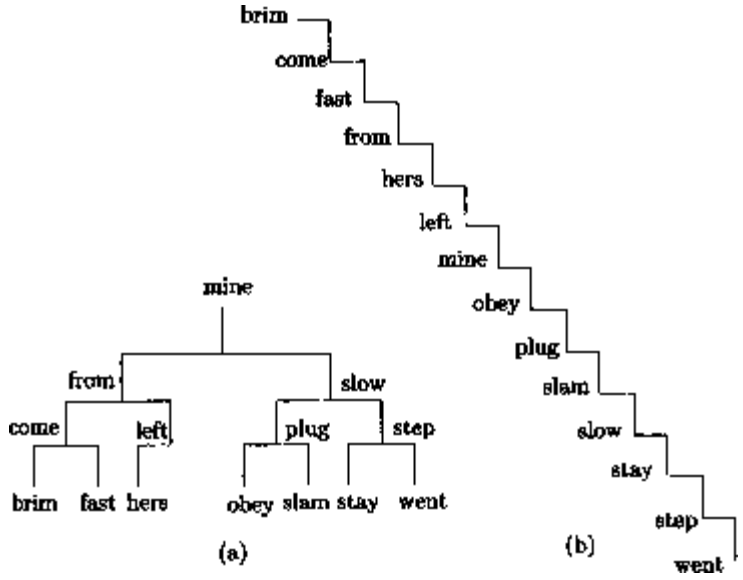

Рисунок 2.4.1 – Двійкове дерево пошуку

Для того, щоб кодер LZSS міг почати працювати, необхідно завантажити буфер черговими символами повідомлення та проініціалізувати дерево. Для цього в дерево вставляється вміст буфера.

Алгоритм послідовно виконує наступні дії:

1) кодує вміст буфера;

2) зчитує чергові символи в буфер, видаляючи при необхідності найбільш «старі» рядки зі словника;

3) вставляє в дерево нові рядки, відповідні ліченим символам.

Для того, щоб декодер зміг вчасно зупинитися, декодуючи повідомлення, кодер поміщає в стислий файл спеціальний символ «КІНЕЦЬ ФАЙЛУ» після того, як він обробив всі символи повідомлення.

Вся робота з пошуку розташування та встановлення довжини максимального збігу вмісту словника з буфером відбувається в процесі додавання в дерево нового рядка.

При додаванні рядка в дерево алгоритм послідовно переміщається від кореня дерева до його листків, кожен раз здійснюючи лексикографічне порівняння нового рядка і вузла дерева. В залежності від результату порівняння вибирається більший або менший нащадок цього вузла і запам'ятовується довжина збігу рядків і положення поточного вузла.

Якщо в результаті порівняння виявляється, що вміст буфера і рядок, на яку посилається поточний вузол, в точності збігаються, то посилання в поточному вузлі оновлюються так, щоб вони вказували на буфер, і процедура додавання рядка в дерево завершується.

Якщо нащадок поточного вузла, вибраний для чергового кроку, відзначений як неіснуючий, залишається заповнити його відповідним посиланням і завершити роботу.

Потрібно зазначити, що при закінченні роботи процедурі додавання рядка відомі і зсув, і довжина найбільшого збігу, і для цього не потрібен повний перебір всіх можливих варіантів збігу.

При видаленні вузла з дерева можливі два варіанти. Якщо вузол має не більше одного нащадка, то видалення вузла здійснюється простим виправленням посилань "батько-нащадок". Якщо вузол має два нащадка, то необхідні інші дії. Для цього знайдемо наступний менший вузол в дереві, видалимо його з дерева і замінимо їм поточний видаляється вузол. Наступний менший вузол знаходиться вибором меншого нащадка і наступним переміщенням від нього по дереву до листа по великих гілок.

Алгоритм LZSS, взагалі кажучи, є дуже асиметричним. Якщо процедура стискування досить складна і здійснює великий обсяг роботи при обробці

кожного символу вхідного повідомлення, то декодер LZSS тривіально простий і може працювати зі швидкістю, що наближається до швидкості процедури звичайного копіювання інформації.

Декодер читає один біт стислої інформації та вирішує – це символ, або пара <зсув, довжина>. Якщо це символ, то наступні 8 бітів видаються як розкодований символ і поміщаються в ковзне вікно. Інакше, якщо це не закодований кінець файлу, то відповідну кількість символів словника поміщається у вікно і видається в розкодованої формі. Оскільки це все, що робить декодер, зрозуміло, чому процедура декодування працює так швидко.

3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис і обґрунтування обраних програмних засобів

Розробка програмного забезпечення здійснена за допомогою середовища Borland C++ Builder 6.

Borland C++ Builder – зручний засіб для розробки віконних додатків для операційної системи Windows XP. Також, візуальна бібліотека середовища містить елементи, що спрощують взаємодію користувача та програми, тобто програмісту не потрібно витрачати час для створення власних елементів вікна та обробників подій для них.

І так далі…

3.2. Опис графічного інтерфейсу користувача

Розроблене програмне забезпечення – набір модулів, які об’єднані графічним інтерфейсом користувача.

Головне вікно програми складається з рядка меню, панелі керування, поля для відображення проміжних даних, кнопок для управління процесами архівації, розархівування, поля відображення логу, яке при необхідності можна приховати.

На рис.3.2.1 зображено головне вікно розробленого продукту:

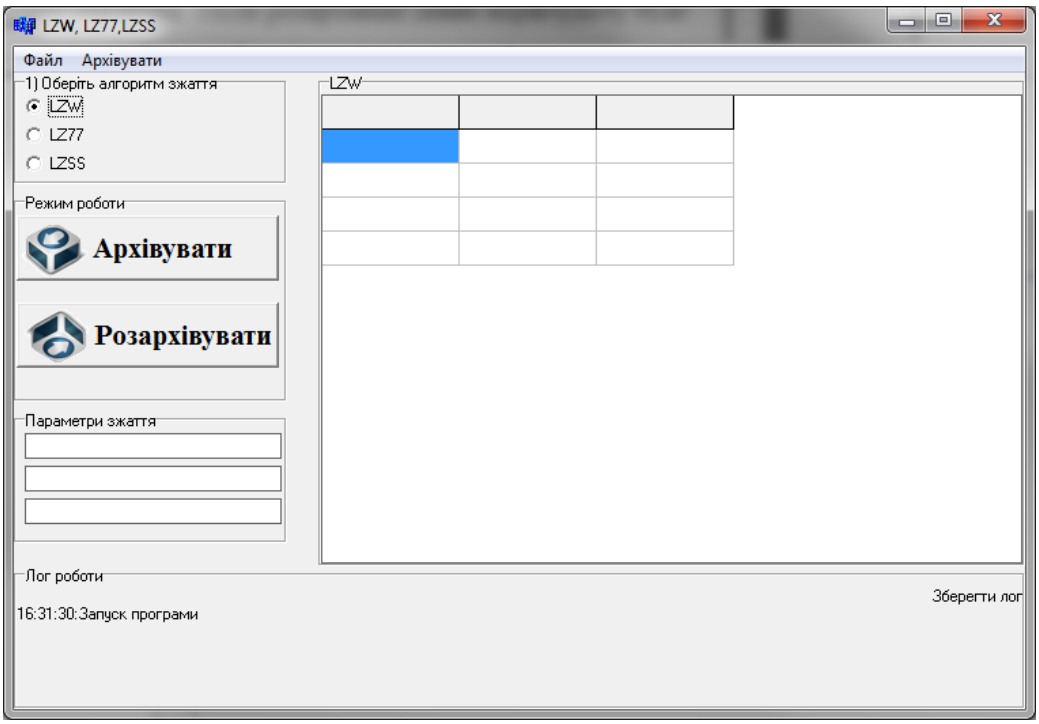

Рисунок 3.2.1 – Головне вікно програмного продукту

Рядок меню головного вікна програми складається з пунктів «Файл» та «Архівувати». Підпункти меню «Архівувати» наведені в таблиці 3.1.

Таблиця 3.1 – Підпункти меню «Архівувати»

Назва підпункту	Функція
LZW – Архівувати	Відкриття текстового чи бінарного файлу для архівації за алгоритмом LZW
LZW – Роз-архівувати	Відкриття текстового чи бінарного файлу для роз-архівації за алгоритмом LZW
LZ77 – Архівувати	Відкриття текстового чи бінарного файлу для архівації за алгоритмом LZ77
LZ77 – Роз-архівувати	Відкриття текстового чи бінарного файлу для роз-архівації за алгоритмом LZ77
LZSS – Архівувати	Відкриття текстового чи бінарного файлу для архівації за алгоритмом LZSS
LZSS – Роз-архівувати	Відкриття текстового чи бінарного файлу для роз-архівації за алгоритмом LZSS

При обранні підпунктів меню «Архівувати» відкривається діалогове вікно відкриття чи збереження файлів, де потрібно вказати файл, який ви бажаєте завантажити чи зберегти (рис.3.2.2.):

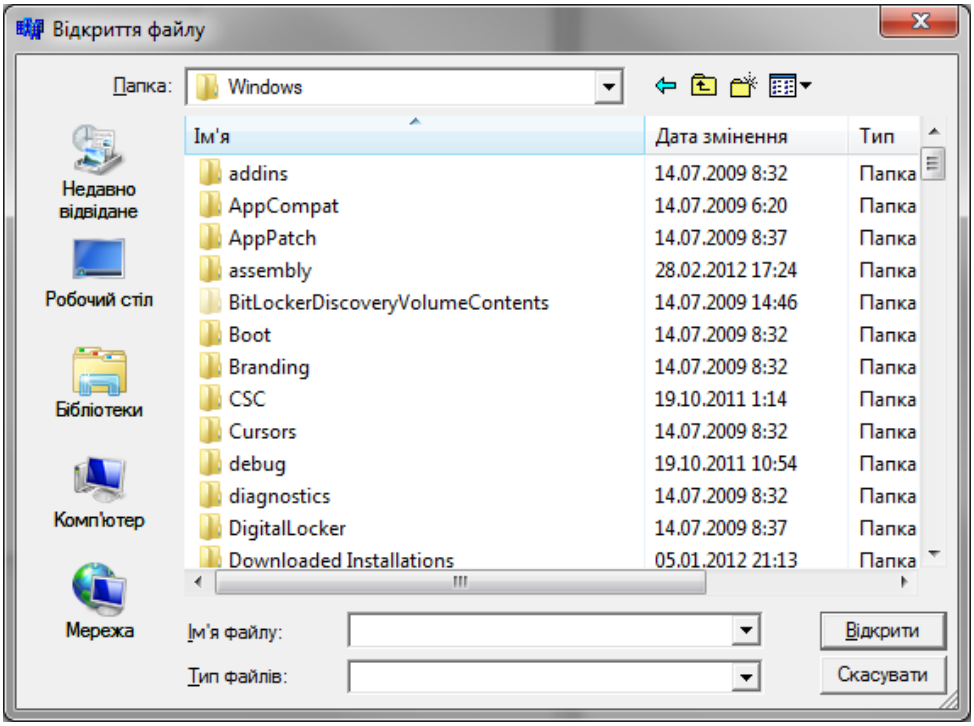

Рисунок 3.2.2 – Діалогове вікно відкриття файлів

Головне вікно програми містить поле-таблицю, в якому відображаються словники, з генеровані при архівації (рис.3.2.3):

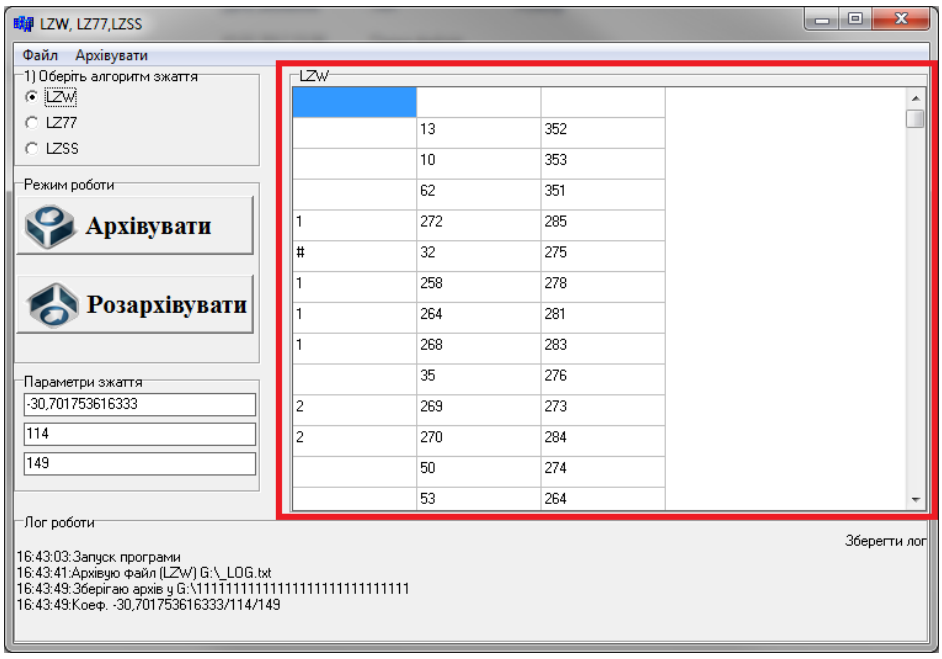

Рисунок 3.2.3 – Відображення словника під час архівації даних

Також користувач може архівувати та розархівовувати файли за допомогою відповідних кнопок керування. Спочатку необхідно обрати алгоритм архівації, після чого натиснути кнопку «Архівувати» чи «Розархівувати», вказати вхідний та вихідні файли (рис.3.2.4):

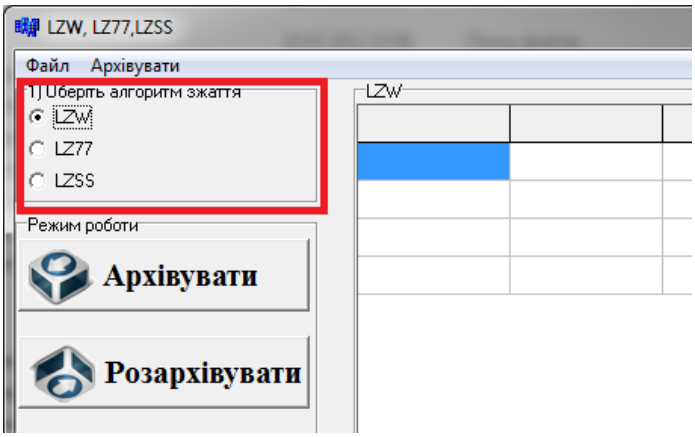

Рисунок 3.2.4 – Вибір алгоритму стиснення інформації

Після архівації файлів користувачеві стає доступна інформація про сам процес стиснення, а саме розмір вхідного і вихідного файлу та відсоток стиснення (рис.3.2.5).

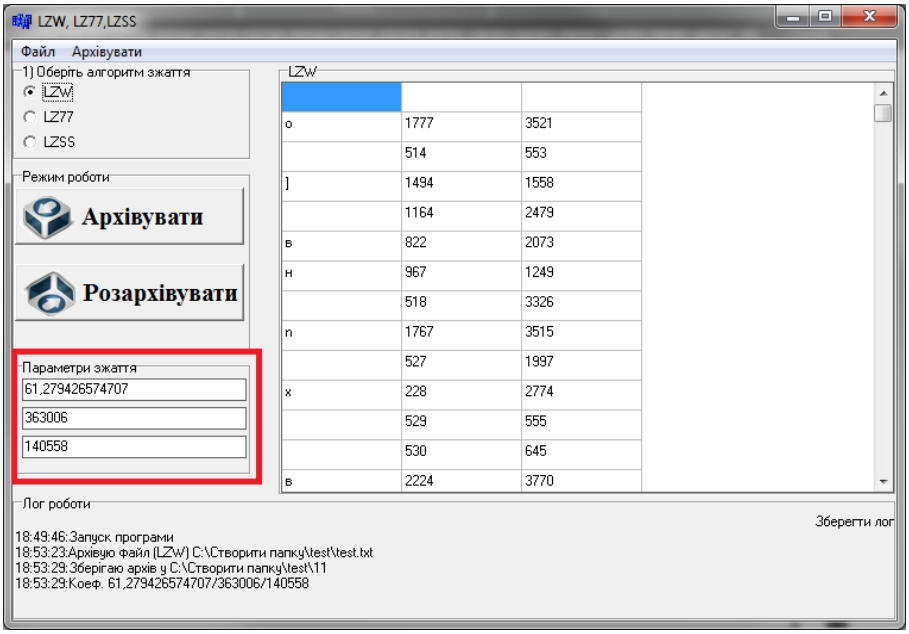

Рисунок 3.2.5 – Результат архівації текстового файлу

Як видно з рис.3.2.5, розмір вхідного файлу – 363006 байт, а розмір стисненого – 140558 байт, таким чином розмір стисненого файлу становить приблизно 39% від розміру вхідного файлу. Для правильної роз-архівації інформації необхідно завантажити файл та обрати алгоритм, яким він був за архівований.

При натисненні на кнопку «Зберегти лог» з’явиться вікно, в якому необхідно вказати файл, куди буде збережений лог роботи програми (рис.3.2.6):

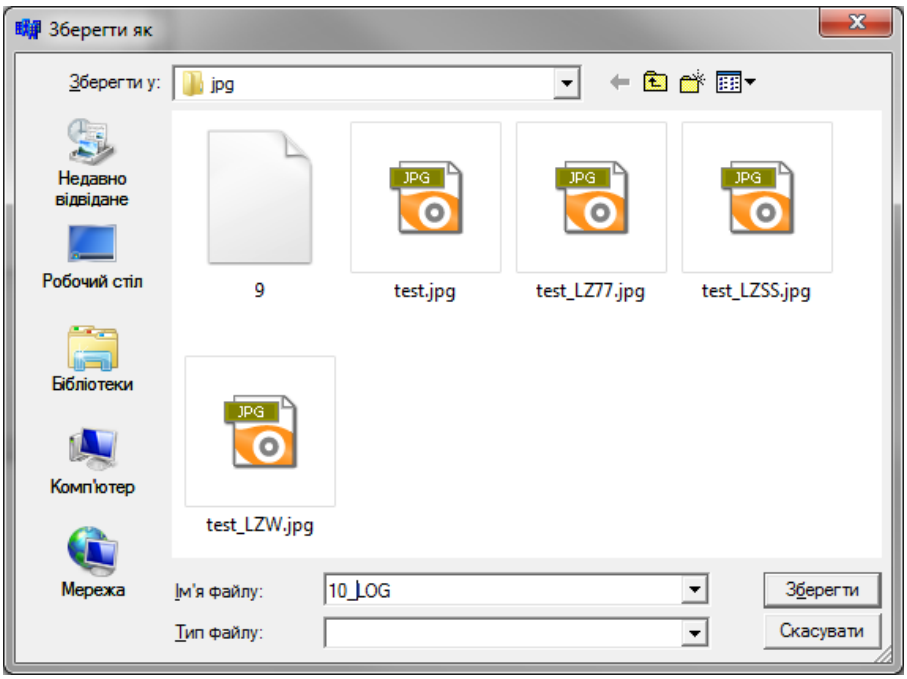

Рисунок 3.2.6 – Збереження логу роботи програми

ВИСНОВКИ

В процесі виконання технологічної практики було створено програмне забезпечення, що реалізує алгоритми стиснення даних без втрат інформації.

Розроблена програма забезпечує надійну архівацію та розархівування текстової та бінарної інформації за допомогою алгоритмів LZW, LZ77 та LZSS.

Розроблений інтуїтивно-зрозумілий інтерфейс із зручною панеллю керування.

Проведено тестування та корекцію роботи розробленого програмного продукту.

Складено необхідну технічну документацію, що повинна постачатись разом з розробленим програмним забезпеченням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ватолин Д.С., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2002.

2. Романов В.Ю. Популярные форматы файлов для хранения графических изображений на IBM PC.– М.:Унитех, 1992.

3. Семенюк В. В. Экономное кодирование дискретной информации. – СПб.: СПбГИТМО (ТУ), 2001.

4. Ватолин Д., Ратушняк А., Смирнов М. Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.ДИАЛОГ-МИФИ, 2003. – 384с.

5. Метод LZW-сжатия данных [Електронний ресурс]. – Режим доступу до документа: http://algolist.manual.ru/compress/standard/lzw.php.

ДОДАТКИ

Додаток А. Код програми

Модуль «main_unit»

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{}
//---------------------------------------------------------------------------
#include <io.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <stdio.h>
```

І так далі… (студенти наводять повний код)