НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Кафедра комп'ютерних наук

Теорія розпізнавання образів та класифікації в системах штучного інтелекту

Лабораторна робота №10

Виконав:

Студент групи КН-20002Б

Кропивка Анатолій Анатолійович

Київ 2023

Тема: Організація таблиць ідентифікаторів.

Мета: Дослідження методів організації таблиць ідентифікаторів.

Підготовка до роботи: Вивчити й уяснити призначення і зміст завдання до лабораторної роботи.

Хід роботи:

1. Ознайомитись з методичною розробкою до лабораторної роботи.
2. Ознайомитись з рекомендованною літературою.
3. Для виконання роботи потрібно підготувати програму, яка отримує на вході набір ідентифікаторів, організовує таблиці ідентифікаторів за допомогою заданих методів, дозволяє здійснити багатократний пошук довільного ідентифікатора в таблицях і порівняти ефективність методів організації таблиць. Вхідна інформація задається у вигляді текстового файлу.
4. Вибрати і описати у звіті відповідні хеш-функції.
5. Описати структуру даних, використовувані для заданих методів організації таблиць ідентифікаторів.
6. Розробити для проведення дослідження програмний застосунок.
7. Виконати дослідження у відповідності з завданням до лабораторної роботи.
8. За результатами досліджень скласти звіт з обґрунтованими висновками.

Опис програм

Програма 1 виконує низку операцій з файлами:

- Функція *initialize_files* викликає функцію *create_file*s для створення файлів для літер англійського алфавіту *(A-Z)* та цифр *(0-9)*. У разі невдачі генерується виняток з повідомленням про помилку.
- Функція *input_word* отримує слово введене користувачем та формує шлях до файлу, використовуючи перший символ слова. Вона перевіряє, чи існує файл з таким шляхом. Якщо файл існує, відкриває його у режимі дозапису *(ios::app)* та записує введене слово у новий рядок. Після цього генерується повідомлення про успішний запис слова до файлу. У разі невдалого відкриття файлу або помилки запису генерується виняток з відповідним повідомленням.
- Функція *check_word* отримує слово введене користувачем та формує шлях до файлу, використовуючи перший символ слова. Вона перевіряє, чи існує файл з таким шляхом. Якщо файл існує, вона читає його по рядках та порівнює кожен рядок з введеним словом. Якщо збіг знайдений, генерується повідомлення про знаходження слова у файлі. Якщо слово не знайдено, генерується виняток з повідомленням про відсутність слова у файлі. У разі невдалого відкриття файлу генерується виняток з повідомленням про помилку.
- Функція *process_function* приймає вказівник на функцію і виконує її. У разі виникнення винятку, виводить повідомлення про помилку на екран.
- У головній функції main відбувається основний цикл програми. Він виводить меню з чотирма пунктами на екран та викликає функції залежно від вибору користувача. *Пункт 1* дозволяє вийти з програми, *пункт 2* ініціалізує файли, *пункт 3* дозволяє користувачеві ввести слово для запису до файлу, а *пункт 4* дозволяє користувачеві перевірити, чи існує слово у файлах. Після виконання відповідної операції програма повертається до початку циклу.

Цей процес повторюється, доки користувач не вибере пункт виходу з програми.

Програма 2 виконує аналіз текстового файлу за певними правилами та виводить результати аналізу на екран:

- Функція *compare* порівнює заданий рядок *str* з елементами вектора *words*. Вона перевіряє, чи існує співпадіння між str та будь-яким елементом у векторі *words*. Якщо таке співпадіння знайдено, функція повертає *true*. Якщо співпадіння не знайдено, функція повертає *false*.
- Функції *is_type*, *is_keyword*, *is_operator*, *is_separator*, *is_assignment* та *is_number* використовують функцію compare для перевірки рядка *str* на відповідність певним типам (типи даних, ключові слова, оператори, роздільники, присвоювання) або числовим значенням. Кожна функція має свій вектор слів, з яким порівнюється рядок *str*. Якщо рядок str збігається з будь-яким елементом відповідного вектора, функція повертає *true*. У протилежному випадку функція повертає *false*.
- Функція *is_sign_at_the_end* перевіряє останній символ рядка *str* і визначає його тип на основі функцій *is_operator*, *is_separator* та *is_assignment*. Якщо останній символ є оператором, функція повертає *1*. Якщо він є роздільником, функція повертає *2*. Якщо він є знаком присвоювання, функція повертає *3*. У разі відсутності відповідності функція повертає *0*.
- У головній функції main відбувається основний алгоритм програми. Спочатку відкривається файл *"../files/file.txt"*. Якщо відкриття файлу невдале, програма виводить повідомлення про невдачу і повертає *1*, щоб позначити помилку.
- Далі виконується читання файлу по рядках. Кожен рядок розбивається на окремі слова за допомогою пробілів. Кожне слово проходить через обробку з використанням різних функцій для визначення його типу.
- Якщо слово має знак у кінці, відбувається перевірка на наявність деяких спеціальних випадків. Наприклад, якщо слово закінчується на '(' і належить до категорії функцій, воно виводиться як функція.
- Для кожного слова виводиться повідомлення про його тип на екран. Якщо слово має знак у кінці, виводиться повідомлення про відповідний тип знаку.
- Після обробки всіх рядків файл закривається, і програма завершується з поверненням *0*.

Код програми

```
// Task 1
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <stdexcept>
#include <string>

using namespace std;

const string get_string()
{
    string str {};

    try {
        getline(cin, str);
    } catch (const exception&) {
        cout << "Failed to read a line!" << endl;
    }

    if (str == "") {
        str = " ";
    }

    return str;
}

int get_number()
{

    int value {};
```

```
    size_t position {};
    string str(get_string());

    try {
        value = stoi(str, &position);

        if (position != str.size()) {
            throw runtime_error("There are characters after the number");
        }

        if (value < 1 || value > 4) {
            throw runtime_error("Possible values are 1..4");
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
        return 1;
    }

    return value;
}

bool file_exists(const string& file_path)
{
    ifstream file(file_path);
    return file.good();
}

void create_file(const string& file_path)
{
    ofstream file(file_path);

    if (file.is_open()) {
```

```
        file.close();
    } else {
        throw string("Failed to create the file: " + file_path);
    }
}

void create_files(char start, char end)
{
    for (char letter { start }; letter <= end; letter++) {
        string file_path("./files/" + string(1, letter) + ".txt");

        if (!file_exists(file_path)) {
            try {
                create_file(file_path);
            } catch (const string& e) {
                throw e;
            }
        }
    }
}

void initialize_files()
{
    try {
        create_files('A', 'Z');
        create_files('0', '9');
        cout << "The files have been initialized" << endl;
    } catch (const string& e) {
        throw e;
    }
}
```

```
void input_word()
{
    string word(get_string());
    string file_path("./files/" + string(1, word[0]) + ".txt");

    if (file_exists(file_path)) {
        ofstream file(file_path, ios::app);

        if (file.is_open()) {
            file << word << endl;
            file.close();

            throw string("The word '" + word + "' has been append to the file: " + file_path);
        } else {
            throw string("Failed to write the file: " + file_path);
        }
    } else {
        throw string("The file '" + file_path + "' doesn't exist!");
    }
}

void check_word()
{
    string word(get_string());
    string file_path("./files/" + string(1, word[0]) + ".txt");
    ifstream file(file_path);

    if (file.is_open()) {
        string line {};

        while (getline(file, line)) {
            if (line == word) {
```

```
            throw string("The word '" + word + "' exists in the file " + file_path);
        }
    }
    throw string("The word '" + word + "' doesn't exist in the file " + file_path);
  } else {
    throw string("Failed to open the file: " + file_path);
  }
}

void process_function(void (*fn)())
{
  try {
    fn();
  } catch (const string& e) {
    cout << e << endl;
  }
}

int main()
{
  while (true) {
    cout << "Press a button to do:" << endl;
    cout << "    1. Exit;" << endl;
    cout << "    2. Initialize files;" << endl;
    cout << "    3. Input a word;" << endl;
    cout << "    4. Check if a word exists in the files;" << endl;

    switch (get_number()) {
    case 1:
      exit(0);
      break;
    case 2:
```

```
            process_function(initialize_files);
            break;
        case 3:
            cout << "Enter your word: ";
            process_function(input_word);
            break;
        case 4:
            cout << "Enter your word: ";
            process_function(check_word);
            break;
        }

        cout << endl;
    }

    return 0;
}
```

```
// Task 2
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <stdexcept>
#include <string>

using namespace std;

const string get_string()
{
    string str {};

    try {
        getline(cin, str);
    } catch (const exception&) {
        cout << "Failed to read a line!" << endl;
    }

    if (str == "") {
        str = " ";
    }

    return str;
}

int get_number()
{

    int value {};
    size_t position {};
```

```
    string str(get_string());

    try {
        value = stoi(str, &position);

        if (position != str.size()) {
            throw runtime_error("There are characters after the number");
        }

        if (value < 1 || value > 4) {
            throw runtime_error("Possible values are 1..4");
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
        return 1;
    }

    return value;
}

bool file_exists(const string& file_path)
{
    ifstream file(file_path);
    return file.good();
}

void create_file(const string& file_path)
{
    ofstream file(file_path);

    if (file.is_open()) {
        file.close();
```

```
    } else {
        throw string("Failed to create the file: " + file_path);
    }
}

void create_files(char start, char end)
{
    for (char letter { start }; letter <= end; letter++) {
        string file_path("./files/" + string(1, letter) + ".txt");

        if (!file_exists(file_path)) {
            try {
                create_file(file_path);
            } catch (const string& e) {
                throw e;
            }
        }
    }
}

void initialize_files()
{
    try {
        create_files('A', 'Z');
        create_files('0', '9');
        cout << "The files have been initialized" << endl;
    } catch (const string& e) {
        throw e;
    }
}

void input_word()
```

```
{
    string word(get_string());
    string file_path("./files/" + string(1, word[0]) + ".txt");

    if (file_exists(file_path)) {
        ofstream file(file_path, ios::app);

        if (file.is_open()) {
            file << word << endl;
            file.close();

            throw string("The word '" + word + "' has been append to the file: " + file_path);
        } else {
            throw string("Failed to write the file: " + file_path);
        }
    } else {
        throw string("The file '" + file_path + "' doesn't exist!");
    }
}

void check_word()
{
    string word(get_string());
    string file_path("./files/" + string(1, word[0]) + ".txt");
    ifstream file(file_path);

    if (file.is_open()) {
        string line {};

        while (getline(file, line)) {
            if (line == word) {
                throw string("The word '" + word + "' exists in the file " + file_path);
```

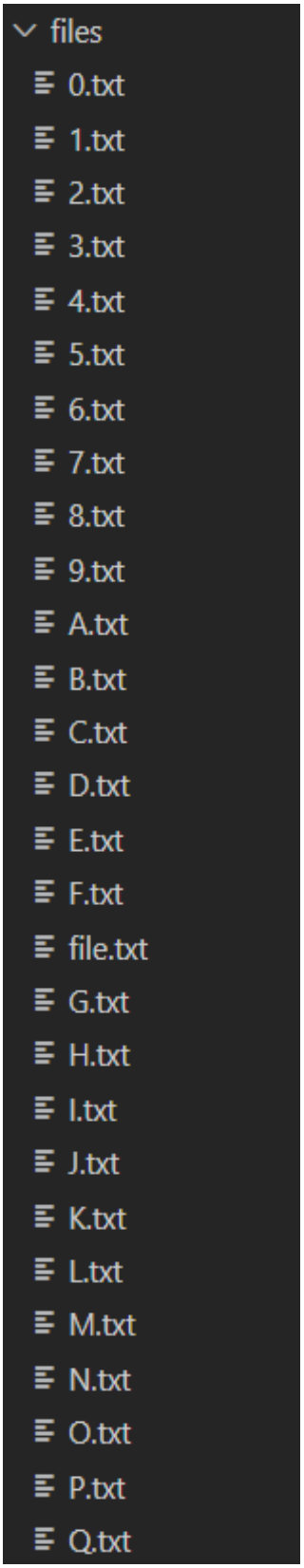
```
            }
        }
        throw string("The word '" + word + "' doesn't exist in the file " + file_path);
    } else {
        throw string("Failed to open the file: " + file_path);
    }
}

void process_function(void (*fn)())
{
    try {
        fn();
    } catch (const string& e) {
        cout << e << endl;
    }
}

int main()
{
    while (true) {
        cout << "Press a button to do:" << endl;
        cout << "    1. Exit;" << endl;
        cout << "    2. Initialize files;" << endl;
        cout << "    3. Input a word;" << endl;
        cout << "    4. Check if a word exists in the files;" << endl;

        switch (get_number()) {
        case 1:
            exit(0);
            break;
        case 2:
            process_function(initialize_files);
```

```
            break;
        case 3:
            cout << "Enter your word: ";
            process_function(input_word);
            break;
        case 4:
            cout << "Enter your word: ";
            process_function(check_word);
            break;
        }

        cout << endl;
    }

    return 0;
}
```

Знімки екрану

```
PS [redacted]\TheoryOfRecognizeImages\MyLabs\lab_10> ./bin/first.exe
Press a button to do:
    1. Exit;
    2. Initialize files;
    3. Input a word;
    4. Check if a word exists in the files;
2
The files have been initialized

Press a button to do:
    1. Exit;
    2. Initialize files;
    3. Input a word;
    4. Check if a word exists in the files;
3
Enter your word: Alice
The word 'Alice' has been append to the file: ./files/A.txt

Press a button to do:
    1. Exit;
    2. Initialize files;
    3. Input a word;
    4. Check if a word exists in the files;
3
Enter your word: alphabet
The word 'alphabet' has been append to the file: ./files/a.txt
```

```
Press a button to do:
    1. Exit;
    2. Initialize files;
    3. Input a word;
    4. Check if a word exists in the files;
4
Enter your word: Alice
The word 'Alice' exists in the file ./files/A.txt

Press a button to do:
    1. Exit;
    2. Initialize files;
    3. Input a word;
    4. Check if a word exists in the files;
4
Enter your word: Orange
The word 'Orange' doesn't exist in the file ./files/O.txt

Press a button to do:
    1. Exit;
    2. Initialize files;
    3. Input a word;
    4. Check if a word exists in the files;
1
PS [redacted]\TheoryOfRecognizeImages\MyLabs\lab_10>
```



```
PS [redacted]\TheoryOfRecognizeImages\MyLabs\lab_10> ./bin/second.exe
const - is a keyword.
A - is a variable.
: - is a separator.
i32 - is a type.
= - is an assignment.
70 - is a number.
; - is a separator.
const - is a keyword.
B - is a variable.
: - is a separator.
i32 - is a type.
= - is an assignment.
30 - is a number.
; - is a separator.
fn - is a keyword.
main() - is a function.
{ - is a separator.
var - is a keyword.
c - is a variable.
: - is a separator.
i32 - is a type.
= - is an assignment.
A - is a variable.
+ - is an operator.
B - is a variable.
; - is a separator.
} - is a separator.
```

files > file.txt

```
1  const A: i32 = 70;
2  const B: i32 = 30;
3
4  fn main() {
5      var c: i32 = A + B;
6  }
```

Висновок: Перша програма займається ініціалізацією файлів та обробкою введених слів. Вона надає можливість ініціалізувати файли за допомогою літер алфавіту та цифр, а також дозволяє додавати введені користувачем слова до відповідного файлу. Крім того, програма дозволяє перевіряти наявність введеного слова в файлах. Вона також містить обробку помилок та повідомлення про виникнення помилок при роботі з файлами.

Друга програма здійснює аналіз текстового файлу зі списком слів. Вона розпізнає ключові слова, типи даних, оператори, роздільники, присвоювання та числа. Кожне слово виводиться разом з його типом. Програма також розпізнає певні спеціальні випадки, такі як функції.

Обидві програми показують використання функцій, роботу з файлами та обробку тексту. Перша програма спрямована на роботу з файлами та керування списком слів, тоді як друга програма спрямована на аналіз вже існуючого текстового файлу та визначення типів слів.