МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Індивідуальне завдання № 1

з курсу «Алгоритми і структури данних»

(назва дисципліни)

на тему: «Односпрямовані списки»

Виконав: студент 2 курсу групи мф-21

напряму підготовки (спеціальності)

Комп’ютерні науки

122 Комп’ютерні науки

Єлагін І.А.

Прийняв: _____

Національна шкала: _____

Кількість балів: _____

Оцінка: ECTS _____

Харків – 2024

Завдання:

Предметний вказівник. У текстовому файлі знаходиться довільний текст, а через два порожні рядки – набір ключових слів у довільному порядку. Потрібно створити «предметний вказівник» у вигляді списку списків: список ключових слів, упорядкований за абеткою, і для кожного ключового слова – упорядковані списки номерів рядків, у яких воно зустрічається в тексті. Забезпечити операції додавання та видалення ключових слів, перевірки, чи зустрічається задане ключове слово в заданому рядку, виведення інформації в розумному вигляді, ...

Код:

```
#include <iostream>


using namespace std;


// Вузол для списку номерів рядків
struct LineNode {
   int line;
   LineNode* next;


   LineNode(int l){
      line = l;
      next = nullptr;
   }
};


// Вузол для списку ключових слів
struct KeywordNode {
   string word;
   LineNode* lines;  // Голова списку номерів рядків
   KeywordNode* next;


   KeywordNode(const string& w) {
      word = w;
      lines = nullptr;
      next = nullptr;
   }
```

```
};

// Структура для предметного вказівника
struct SubjectIndex {
    KeywordNode* head;  // Голова списку ключових слів

    SubjectIndex(){
        head = nullptr;
    }

    // Додавання нового ключового слова
    void addKeyword(const string& keyword) {
        if (!findKeyword(keyword)) {
            KeywordNode* newNode = new KeywordNode(keyword);
            if (!head || head->word > keyword) {
                newNode->next = head;
                head = newNode;
            }
            else {
                KeywordNode* current = head;
                while (current->next && current->next->word < keyword) {
                    current = current->next;
                }
                newNode->next = current->next;
                current->next = newNode;
            }
        }
    }

    // Видалення ключового слова
    void removeKeyword(const string& keyword) {
        if (!head) return;

        if (head->word == keyword) {
```

```
        KeywordNode* temp = head;
        head = head->next;
        clearLines(temp->lines);
        delete temp;
        return;
    }

    KeywordNode* current = head;
    while (current->next && current->next->word != keyword) {
        current = current->next;
    }

    if (current->next) {
        KeywordNode* temp = current->next;
        current->next = temp->next;
        clearLines(temp->lines);
        delete temp;
    }
}

// Додавання номера рядка для ключового слова
void addLineForKeyword(const string& keyword, int line) {
    KeywordNode* keywordNode = findKeyword(keyword);
    if (!keywordNode) return;

    LineNode* current = keywordNode->lines;
    LineNode* newNode = new LineNode(line);

    // Додаємо у список з упорядкуванням
    if (!current || current->line > line) {
        newNode->next = current;
        keywordNode->lines = newNode;
    }
    else {
```

```
        while (current->next && current->next->line < line) {
            current = current->next;
        }
        if (!current->next || current->next->line != line) {
            newNode->next = current->next;
            current->next = newNode;
        }
    }
}

// Перевірка, чи є ключове слово у списку
bool keywordInLine(const string& keyword, int line) {
    KeywordNode* keywordNode = findKeyword(keyword);
    if (!keywordNode) return false;

    LineNode* current = keywordNode->lines;
    while (current) {
        if (current->line == line) {
            return true;
        }
        current = current->next;
    }
    return false;
}

// Виведення предметного вказівника
void printIndex() const {
    KeywordNode* current = head;
    while (current) {
        cout << current->word << ": ";
        LineNode* lineCurrent = current->lines;
        while (lineCurrent) {
            cout << lineCurrent->line << " ";
            lineCurrent = lineCurrent->next;
```

```
            }
            cout << "\n";
            current = current->next;
        }
    }

    // Очищення пам'яті при видаленні списку номерів рядків
    void clearLines(LineNode* lineHead) {
        while (lineHead) {
            LineNode* temp = lineHead;
            lineHead = lineHead->next;
            delete temp;
        }
    }

    // Очищення пам'яті при видаленні предметного вказівника
    ~SubjectIndex() {
        while (head) {
            KeywordNode* temp = head;
            head = head->next;
            clearLines(temp->lines);
            delete temp;
        }
    }

private:
    // Пошук ключового слова у списку
    KeywordNode* findKeyword(const string& keyword) {
        KeywordNode* current = head;
        while (current) {
            if (current->word == keyword) {
                return current;
            }
            current = current->next;
```

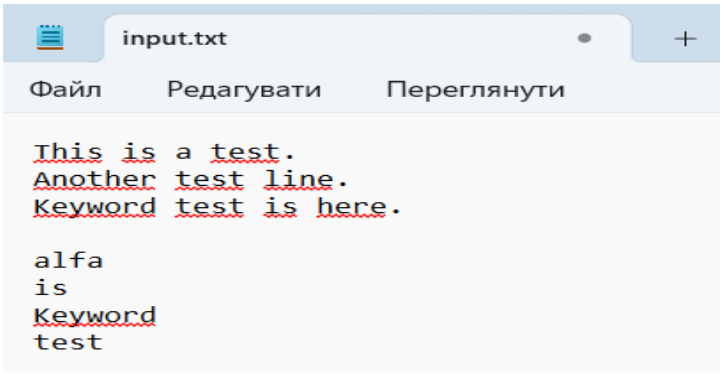
```
        }
        return nullptr;
    }
};

// Читання даних із файлу
bool readFile(const char* fileName, string text[], int& lineCount, string keywords[], int& keywordCount) {
    FILE* file;
    if (fopen_s(&file, fileName, "r") != 0) {
        cerr << "File opening error: " << fileName << "\n";
        return false;
    }

    char buffer[1024];
    bool readingText = true;

    while (fgets(buffer, sizeof(buffer), file)) {
        string line(buffer);
        if (!line.empty() && (line.back() == '\n' || line.back() == '\r')) {
            line.pop_back();
        }

        if (line.empty()) {
            if (!readingText) {
                break;
            }
            readingText = false;
            continue;
        }

        if (readingText) {
            text[lineCount++] = line;
        }
        else {
```

```
            keywords[keywordCount++] = line;
        }
    }

    fclose(file);
    return true;
}

int main() {
    const char* fileName = "input.txt";
    string text[1000];
    int lineCount = 0;
    string keywords[100];
    int keywordCount = 0;

    if (!readFile(fileName, text, lineCount, keywords, keywordCount)) {
        return 1;
    }

    SubjectIndex subjectIndex;

    // Додавання ключових слів
    for (int i = 0; i < keywordCount; ++i) {
        subjectIndex.addKeyword(keywords[i]);
    }

    // Побудова предметного вказівника
    for (int i = 0; i < lineCount; ++i) {
        for (int j = 0; j < keywordCount; ++j) {
            if (text[i].find(keywords[j]) != string::npos) {
                subjectIndex.addLineForKeyword(keywords[j], i + 1);
            }
        }
    }
```

```
    // Виведення результату
    cout << "\nSubject index:\n";
    subjectIndex.printIndex();

    // Видалення ключового слова
    subjectIndex.removeKeyword(keywords[3]);

    cout << "\nExample of removing a keyword.\n";
    cout << "\nSubject index:\n";
    subjectIndex.printIndex();

    return 0;
}
```

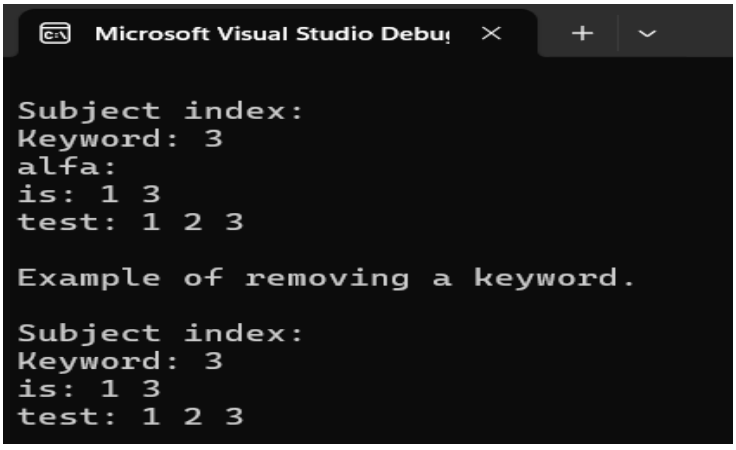
Вхідні данні:

Спочатку розміщується текст, а потім через одну пусту строку ключові слова.



Результат роботи коду:

У консолі ми можемо побачити результат роботи операцій додавання, видалення та виведення.



Пояснення коду та операцій:

1. void addKeyword(const string& keyword):

Мета:

Додає нове ключове слово в список ключових слів.

Як працює:

1.1. Перевіряє, чи існує ключове слово у списку за допомогою findKeyword. Якщо слово вже є, функція нічого не робить.

1.2. Створює новий вузол KeywordNode для нового ключового слова.

1.3. Додає нове слово:

1.3.1. Якщо список порожній додає його на початок списку.

1.3.2. Інакше проходить список і знаходить місце.

2. void removeKeyword(const string& keyword)

Мета:

Видаляє ключове слово та всі пов’язані з ним номери рядків.

Як працює:

2.1. Якщо список ключових слів порожній, функція одразу завершується.

2.2. Якщо слово — це перший елемент списку:

2.2.1. Видаляє вузол, очищає список номерів рядків цього слова.

2.2.2. Зсуває вказівник head на наступний вузол.

2.3. Інакше проходить список і шукає слово:

2.3.1. Коли знаходить, змінює вказівники так, щоб пропустити вузол із видаляним словом.

2.3.2. Видаляє цей вузол і пов’язані з ним номери рядків.

3. KeywordNode* findKeyword(const string& keyword)

Мета:

Знаходить вузол ключового слова в списку або повертає nullptr, якщо слово не знайдено.

Як працює:

3.1. Починає пошук із голови списку.

3.2. Порівнює кожне слово в списку з шуканим.

3.3. Якщо знаходить відповідність, повертає вказівник на вузол.

3.4. Якщо проходить список і не знаходить слово, повертає nullptr.

4. void addLineForKeyword(const string& keyword, int line)

Мета:

Додає номер рядка для конкретного ключового слова в упорядкованому вигляді.

Як працює:

4.1. Знаходить вузол ключового слова через findKeyword.

4.2. Якщо слово не знайдено, функція нічого не робить.

4.3. Проходить список номерів рядків (LineNode) для цього слова:

4.3.1. Якщо список порожній або новий номер менше за перший, додає його на початок.

4.3.2. Інакше знаходить місце, де треба вставити новий номер так, щоб зберегти порядок.

4.4. Уникає дублювання: якщо номер рядка вже є у списку, нічого не додає.

5. bool keywordInLine(const string& keyword, int line)

Мета:

Перевіряє, чи зустрічається ключове слово у заданому рядку.

Як працює:

5.1. Знаходить вузол ключового слова через findKeyword.

5.2. Якщо слово не знайдено, повертає false.

5.3. Перебирає список номерів рядків (LineNode) для цього слова:

5.3.1. Якщо номер рядка є у списку, повертає true.

5.3.2. Якщо список закінчується без збігу, повертає false.

6. void printIndex() const

Мета:

Виводить на екран весь предметний вказівник.

Як працює:

6.1. Починає з голови списку ключових слів (KeywordNode).

6.2. Для кожного слова виводить його та проходить список номерів рядків (LineNode), виводячи їх через пробіл.

6.3. Після завершення переходить до наступного вузла ключового слова.

7. void clearLines(LineNode* lineHead)

Мета:

Очищає пам’ять, видаляючи всі вузли зі списку номерів рядків.

Як працює:

7.1. Починає з голови списку рядків.

7.2. У циклі видаляє поточний вузол і переходить до наступного.

7.3. Після завершення всі вузли звільнені, і пам’ять очищена.

8. ~SubjectIndex()

Мета:

Очищає пам’ять усіх вузлів предметного вказівника.

Як працює:

8.1. Починає з голови списку ключових слів.

8.2. Для кожного вузла ключового слова:

8.2.1. Викликає clearLines для очищення списку номерів рядків.

8.2.2. Видаляє вузол ключового слова.

8.3. Повторює до тих пір, поки список не стане порожнім.

9. bool readFile(const char* fileName, string text[], int& lineCount, string keywords[], int& keywordCount)

Мета:

Зчитує текст і ключові слова з файлу та заповнює відповідні масиви.

Як працює:

9.1. Відкриває файл для читання.

9.2. Читає рядки один за одним:

9.2.1. Порожній рядок розділяє текст і ключові слова.

9.2.2. Усе до порожнього рядка зберігається в масиві тексту.

9.2.3. Усе після порожнього рядка — у масиві ключових слів.

9.2. Закриває файл і повертає true, якщо зчитування успішне.

10. main()

Мета:

Організовує процес створення та роботи з предметним вказівником.

Як працює:

10.1. Зчитує текст і ключові слова через readFile.

10.2. Додає всі ключові слова до предметного вказівника через addKeyword.

10.3. Перебирає текст рядок за рядком і додає номери рядків для ключових слів через addLineForKeyword.

10.4. Виводить предметний вказівник через printIndex.