Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«Мультипарадигменне програмування»

«Імперативне програмування**»**

Виконала: студентка ІТ-04, Полтава Віолетта Віталіївна

Перевірили: Очеретяний О.К. та Глушко Б.С.

Київ 2022

ЗМІСТ

Мета лабораторної роботи

Мета роботи – дослідити та зрозуміти, як писали код у 1950-х, за допомогою імперативного програмування. Виконати завдання.

1 Завдання

Практична робота складається із трьох завдань, які самі по собі є досить простими. Але, оскільки задача - зрозуміти, як писали код наші славні пращури у 1950-х, ми введемо кілька обмежень:

- Заборонено використовувати функції
- Заборонено використовувати цикли
- Для виконання потрібно взяти мову, що підтримує конструкцію GOTO

2 Виконання

1.1 Програмна реалізація

1.1.1 Вихідний код

Лабораторну роботу виконано на мові C#.

Завдання 1:

```
using System;
using System.IO;

namespace Lab_1_Multi_paradigm_programming
{
   class Task1
   {
      static void Main(string[] args)
      {
         string text = File.ReadAllText("task1.lang_with_go_to/text.txt");
         string[] words = new string[100];
         string word = "";
         char letter = ' ';
         var wordListLastIndex = 0; // индекс последнего слова в списке
         var index = 0; // итератор
         var jIndex = 0; // второй итератор
         var wStart = 0; // начальный индекс слова
         var wEnd = 0; // конечный --//--
         string[] wordKeys = new string[100]; // слова
         int[] wordValues = new int[100]; // кол-во слов
         var checkDictionaryIndex = 0; // итератор словаря
         var checkDictionaryResult = -1; // индекс слова в словаре, если оно (слово) там
есть
         var checkDictionaryLastIndex = 0; // индекс последнего слова в словаре
         var tmp = "";
         var tmp2 = 0;
```

```
loop1: // перебираем текст, ищем слова
  if(text[index] == ' ' || text[index] == '\r' || index + 1 == text.Length) {
    wEnd = index - 1; // определяем конец слова

    // если это последняя строка в тексте
    if (index + 1 == text.Length) {
      wEnd++;
    }

    // считываем слово
    addWordLoop:
      if(wStart != wEnd + 1) { // перебираем буквы
        // если большая буква -> сделать маленькой
        if(text[wStart] >= 'A' && text[wStart] < 'a') {
          letter = (char)(text[wStart] + 32);
        }
        // игнорировать знаки припенания
        else if (text[wStart] == '.' || text[wStart] == '!' || text[wStart] == '?' ||
text[wStart] == ',' || text[wStart] == '-') {
          wStart++;
          goto addWordLoop;
        }
        else {
          letter = text[wStart];
        }
        // добавить букву в слово
        word += letter;
        wStart++; // увеличить итератор (что бы взять след. букву)
        goto addWordLoop;
      }

    // игнорировать "стоп-слова"
```

```
                    if(word != "for" && word != "the" && word != "as" && word != "in" && word
!= "a" && word != "on" && word != "") {
                        words[wordListLastIndex] = word; // добавить слово в словарь
                        wordListLastIndex++; // увел. индекс последнего слова
                    }
                    word = "";
                    wStart += 1; // след слово будет идти после пробела -> пропустить пробел
                }

                if(text.Length - 1 > index) { // повторить цикл, если еще не перебрали все
символы
                    index++;

                    if(text[index] == '\n') { // если конец строки, пропустить \r\n
                        index++;
                        wStart++;
                    }
                    goto loop1;
                }

            index = 0;

            // считаем слова
            countWords:
                checkDictionaryResult = -1;

                checkDictionary: // перебираем словарь в поисках слова
                    // если нашли слово, прервать
                    if(wordKeys[checkDictionaryIndex] == words[index]) {
                        checkDictionaryResult = checkDictionaryIndex;
                        goto checkDictionaryEnd;
                    }
                    // если нет, прододжать пока не закончится словарь
                    if(checkDictionaryIndex != checkDictionaryLastIndex) {
```

```
            checkDictionaryIndex++;
            goto checkDictionary;
        }
    checkDictionaryIndex = 0;
    checkDictionaryEnd:

    // если нашли слово в словаре
    if(checkDictionaryResult != -1) {
        wordValues[checkDictionaryResult]++; // увел. его кол-во
        checkDictionaryResult = -1;
    }
    // если не нашли, добавить в словарь
    else {
        wordKeys[checkDictionaryLastIndex] = words[index];
        wordValues[checkDictionaryLastIndex] = 1;
        checkDictionaryLastIndex++; // увел. индекс последнего слова в словаре
}
    if(index != wordListLastIndex - 1) { // повторить, пока не закончатся слова
        index++;
        goto countWords;
    }

// бабл сортировка по кол-ву слов
index = 0;
startouter:
    if(index >= checkDictionaryLastIndex - 1) {
        goto endouter;
    }
    jIndex = 0;
startinner:
    if (jIndex >= checkDictionaryLastIndex - 1) {
        goto endinner;
    }
    if (wordValues[jIndex] > wordValues[jIndex + 1]) {
```

```
                goto noswap;
            }
            tmp = wordKeys[jIndex];
            tmp2 = wordValues[jIndex];
            wordKeys[jIndex] = wordKeys[jIndex + 1];
            wordValues[jIndex] = wordValues[jIndex + 1];
            wordKeys[jIndex + 1] = tmp;
            wordValues[jIndex + 1] = tmp2;
        noswap:
            jIndex++;
            goto startinner;
        endinner:
            index++;
            goto startouter;
        endouter:

        index = 0;

        write:
            Console.WriteLine(wordKeys[index] + " - " + wordValues[index]);
            if (index != checkDictionaryLastIndex - 1) {
                index++;
                goto write;
            }
      }
   }
}
```

Завдання 2:

```
using System;
using System.IO;

namespace Lab_1_Multi_paradigm_programming
{
```

```
class Task2
{
   static void Main(string[] args)
   {
      string text = File.ReadAllText("task2.lang_with_go_to/text2.txt");
      string[] lines = new string[20000];
      int linesIterator = 0; // итератор линии
      int linesStart = 0; // начальный индекс линии
      int linesEnd = 0; // конечный индекс линии
      int linesLastIndex = 0; // длина списка линий
      int wordsIterator = 0; // итератор слов
      int wordsStart = 0; // начальный индекс слова
      int wordsEnd = 0; // конечный индекс слова
      string[] tempWords = new string[1000]; // временный массив слов для дальшей
      int tempWordsLastIndex = 0;
      int tempWordsIterator = 0;
      string[] wordKeys = new string[25000]; // ключ (слово)
      int[][] wordValues = new int[25000][]; // значение (страницы)
      int[] wordCounts = new int[25000]; // к-во повторения слова
      int wordValuesIterator = 0; // итератор словаря
      string currentLine = "";
      string word = "";
      char letter = ' ';
      var checkDictionaryIndex = 0; // итератор поиска в словаре
      var checkDictionaryResult = -1; // индекс слова в словаре, если оно (слово) там
есть
      var checkDictionaryLastIndex = 0; // индекс последнего слова в словаре
      var currentPage = 1;
      var tmp = "";
      var tmp2 = new int[100];
      var tmp3 = 0;
      var index = 0; // итератор
      var jIndex = 0; // второй итератор
      var letterCheck = 0; // индекс буквы в слове (для сортировки по алфавиту)
```

```
                // считаем линии
                countLinesLoop:
                    if(text[linesIterator] == '\r' || linesIterator == text.Length - 1) {
                        linesEnd = linesIterator - 1; // обозначение конечного индекса линии

                        // если это последняя строка в тексте
                        if(linesIterator == text.Length - 1) {
                            linesEnd++;
                        }

                        currentLine = "";
                        writeLineLoop:
                            if(text[linesStart] < 'A' || (text[linesStart] > 'Z' && text[linesStart] < 'a') ||
text[linesStart] > 'z') {
                                if (text[linesStart] != ' ') {
                                    goto writeLineLoopEnd; // игнор стоп-символов
                                }
                            }

                            currentLine += text[linesStart]; // записываем символ в линию

                            writeLineLoopEnd:
                                linesStart++; // увеличиваем индекс указателя на символ
                                if(linesStart <= linesEnd) { // перебираем буквы до того как дойдем до
конца линии
                                    goto writeLineLoop;
                                }

                            linesStart = linesIterator + 2; // начало след. строки будет после \r\n
                            lines[linesLastIndex] = currentLine; // добавляем линию в список линии
                            linesLastIndex++; // индекс последней линии в списке линий
                    }
```

```
            linesIterator++; // перескакиваем \n -> первый символ новой строки

            if(linesIterator < text.Length) { // перебираем линии до конца текста
               if (text[linesIterator] == '\n')
               {
                  linesIterator++;
               }
               goto countLinesLoop;
            }

         linesIterator = 0;
         text = ""; // сохраняем память
         // перебираем слова в каждой линии
         countWordsLoop:
            currentLine = lines[linesIterator];

            wordsIterator = 0;
            wordsStart = 0;
            tempWords = new string[1000];
            tempWordsLastIndex = 0;
            tempWordsIterator = 0;

            wordsLoop:
               if(currentLine.Length > 0 && (currentLine[wordsIterator] == ' ' || wordsIterator
+ 1 == currentLine.Length)) {
                  wordsEnd = wordsIterator - 1; // определяем конец слова

                  // если это последняя строка в тексте
                  if (wordsIterator + 1 == currentLine.Length) {
                     wordsEnd++;
                  }

                  // считываем слово
                  addWordLoop:
```

```
                    if(wordsStart != wordsEnd + 1) { // перебираем буквы
                        // если большая буква -> сделать маленькой
                        if(currentLine[wordsStart] >= 'A' && currentLine[wordsStart] < 'a') {
                            letter = (char)(currentLine[wordsStart] + 32);
                        }
                        else {
                            letter = currentLine[wordsStart];
                        }
                        // добавить букву в слово
                        word += letter;
                        wordsStart++; // увеличить итератор (что бы взять след. букву)
                        goto addWordLoop;
                    }

                // игнорировать "стоп-слова"
                if(word != "for" && word != "the" && word != "as" && word != "in" &&
word != "a" && word != "on" && word != "") {
                    tempWords[tempWordsLastIndex] = word; // добавить слово в словарь
                    tempWordsLastIndex++; // увел. индекс последнего слова
                }
                word = "";
                wordsStart += 1; // след слово будет идти после пробела -> пропустить
пробел
            }
            wordsIterator++;
            if(wordsIterator < currentLine.Length) {
                goto wordsLoop;
            }
            if (currentLine.Length == 0) {
                goto countWordsLoopEnd;
            }

            dictionaryLoop:
                checkDictionaryResult = -1;
```

```
checkDictionary: // перебираем словарь в поисках слова
   // если нашли слово, прервать
   if(wordKeys[checkDictionaryIndex] == tempWords[tempWordsIterator]) {
      checkDictionaryResult = checkDictionaryIndex;
      goto checkDictionaryEnd;
   }
   // если нет, прододжать пока не закончится словарь
   if(checkDictionaryIndex < checkDictionaryLastIndex) {
      checkDictionaryIndex++;
      goto checkDictionary;
   }

checkDictionaryEnd:
checkDictionaryIndex = 0;

// посчитать номер страницы относительно номера линии (45 линий на
странице)
currentPage = 1;
int lineIndex = linesIterator;
lineIndexLoop:
   if(lineIndex - 45 > 0) {
      currentPage++;
   }

   lineIndex -= 45;
   if(lineIndex > 0) {
      goto lineIndexLoop;
   }

// если нашли слово в словаре
if(checkDictionaryResult != -1) {
   // игнорировать слова, которіе уже встретились 100 или больше раз
   if (wordCounts[checkDictionaryResult] < 100) {
      // найти индекс последней страницы
```

```
                findLoop:
                    // если страница уже указанна, проигнорировать
                    if (wordValues[checkDictionaryResult][wordValuesIterator] ==
currentPage) {
                        goto findLoopEnd;
                    }
                    // если нет, добавить
                    else if (wordValues[checkDictionaryResult][wordValuesIterator] ==
0) {
                        wordValues[checkDictionaryResult][wordValuesIterator] =
currentPage;
                        goto findLoopEnd;
                    }

                    wordValuesIterator++;
                    if (wordValuesIterator < 100) {
                        goto findLoop;
                    }
                findLoopEnd:
                wordValuesIterator = 0;
                wordCounts[checkDictionaryResult]++; // увел. счетчик слова
            }
        }
        // если не нашли, добавить в словарь
        else {
            wordKeys[checkDictionaryLastIndex] = tempWords[tempWordsIterator];
            wordValues[checkDictionaryLastIndex] = new int[100];
            wordValues[checkDictionaryLastIndex][0] = currentPage;
            wordCounts[checkDictionaryLastIndex]++;
            checkDictionaryLastIndex++;
        }

        if(tempWordsIterator < tempWordsLastIndex - 1) { // повторить, пока не
закончатся слова
```

```
                tempWordsIterator++;
                goto dictionaryLoop;
            }

        countWordsLoopEnd:
            linesIterator++;
            if(linesIterator < linesLastIndex) {
                goto countWordsLoop;
            }

        // бабл сортировка
        startouter:
            if(index >= checkDictionaryLastIndex - 1) {
                goto endouter;
            }
            jIndex = 0;
        startinner:
            if (jIndex >= checkDictionaryLastIndex - 1) {
                goto endinner;
            }

            letterCheck = 0;

        // соритировка по алфавиту
        alphabetSort:
            // если текущая буква совпадает, перейти к следующей
            if (wordKeys[jIndex][letterCheck] == wordKeys[jIndex + 1][letterCheck]) {
                if (letterCheck < wordKeys[jIndex].Length - 1 && letterCheck < wordKeys[jIndex + 1].Length - 1) {
                    letterCheck++;
                    goto alphabetSort;
                }
            }
            // сверить буквы
```

```
    else if (wordKeys[jIndex][letterCheck] < wordKeys[jIndex + 1][letterCheck]) {
        goto noswap;
    }

    tmp = wordKeys[jIndex];
    tmp2 = wordValues[jIndex];
    tmp3 = wordCounts[jIndex];
    wordKeys[jIndex] = wordKeys[jIndex + 1];
    wordValues[jIndex] = wordValues[jIndex + 1];
    wordCounts[jIndex] = wordCounts[jIndex + 1];
    wordKeys[jIndex + 1] = tmp;
    wordValues[jIndex + 1] = tmp2;
    wordCounts[jIndex] = tmp3;
noswap:
    jIndex++;
    goto startinner;
endinner:
    index++;
    goto startouter;
endouter:

index = 0;
jIndex = 0;

write:
    if (wordCounts[index] < 100) {
        Console.Write(wordKeys[index] + " - ");
        jIndex = 0;
        var result = 0;
        // найти кол-во страниц
        findLoop2:
            if (wordValues[index][jIndex] == 0) {
                result = jIndex - 1;
                goto findLoop2End;
```

```
                }

                jIndex++;
                if (jIndex < 100) {
                    goto findLoop2;
                }
            findLoop2End:
            jIndex = 0;

            // вывести страницы по очереди
            writePages:
                Console.Write(wordValues[index][jIndex]);

            if (jIndex < result) {
                jIndex++;
                Console.Write(", "); // добавить запятую, если еще есть станицы
                goto writePages;
            }
            Console.Write('\n');
        }

        if (index != checkDictionaryLastIndex - 1) {
            index++;
            goto write;
        }
    }
  }
}
```

1.1.2 Результати та дослідження роботи

Опис алгоритму вирішення

Завдання 1:

1) Після оголошення змінних алгоритм роботи розпочинається із функції, яка перебирає текст та шукає слова:
    - Якщо це останній рядок у тексті - +1 до кінцевого ітератору індексу слова;
    - В середині є ще одна функція, яка зчитує слово додаючи букви до змінної word. Ми по черзі перебираємо букви, якщо буква велика – робимо маленьку. Також ігноруємо знаки.
2) У функції перебору тексту та пошуку слова є перевірка на стоп-слова. Якщо черга дійшла до стоп-слова ми додаємо попереднє слово до масиву слів і збільшуємо на одиницю індекс останнього слова.
    - Наступне слово буде йти після пробілу, який ми пропускаємо.
3) Цикл повторюємо, доти не перебрали усі символи.
    - Якщо вже кінець рядка, пропускаємо \r\n. Збільшуємо індекс та початковий індекс слова.
4) Наступна функція зчитує слова.
    - Початок ще однієї функції в середині, яка перебирає словник в пошуках слова. Якщо знайти слово перериваємо цикл – переходимо до функції, checkDictionaryEnd.
    - Якщо слово не знайдено, продовжуємо поки не закінчиться словник.
    - Функція checkDictionaryEnd: якщо знайшли слово в словнику, збільшуємо його к-сть. Якщо – ні, додаємо до словника. Збільшуємо індекс останнього слова у словнику.
    - Повторюємо поки не закінчяться слова.

5) Остання функція – це функція сортування по к-сті слів (Бабл-сортування).

Завдання 2:

1) Після оголошення змінних алгоритм роботи розпочинається із функції, яка перебирає лінії:
    - У функції є перевірка на позначення кінцевого індексу лінії;
    - Якщо це остання рядок у тексті збільшуємо на одиницю кінцевий індекс лінії; записуємо символи до поточної лінії. Після функції writeLineLoopEnd: оголошуємо, що початок наступного рядка буде після \r\n, додаємо лінію до списку ліній, та також збільшує індекс останньої лінії у списку ліній.
    - В середині є ще одна функція яка знаходить кінець лінії. У функції є перевірка на стоп слова, знаки та великі літери. Функція має в собі ще функцію writeLineLoopEnd, вона збільшує індекс вказівника на символ, перебирає літери допоки не дійдемо до кінця лінії.
    - Повертаємося до countLinesLoop: перестрибуємо \n і linesIterator це перший символ нової лінії. Перебираємо лінії до кінця тексту.
2) Перед початком наступної функції анулюємо linesIterator та text (щоб зберегти пам’ять).
3) Наступна функція countWordsLoop, яка перебирає слова у кожній лінії.
    - Є перевірка на кінець слова, коли після іде пробіл, або кінець лінії.
    - Якщо остання лінія у тексті збільшуємо кінцевий індекс слова.
    - Функція в середині addWordLoop, яка зчитує слово: перебираємо літери. Якщо велика – робимо маленькою =>

додаємо букву до слова та збільшуємо ітератор початку індексу слова, щоб взяти наступну літеру.

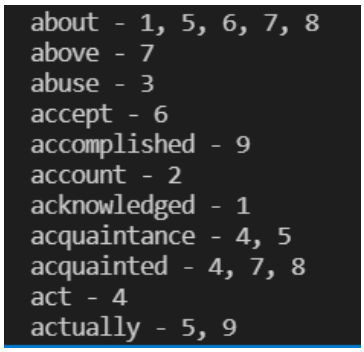
- Ігноруємо стоп-слова => додаємо слово у словник і збільшуємо індекс останнього слова.
- Наступне слово буде йти після пробілу, тобто пропускаємо пробіл.

4) Також є функції dictionaryLoop, checkDictionary, checkDictionaryEnd та lineIndexLoop. Перебираємо словник у пошуках слова.
    - Якщо знайшли слово, перервати
    - Якщо ні, продовжувати доки не закінчиться словник
    - Рахуємо номер сторінки щодо номера лінії (45 ліній на сторінці)
    - Якщо знайшли слово у словнику. Ігноруємо слова, які вже зустрілися 100 або більше разів. Знаходимо індекс останньої сторінки. Якщо сторінка вже вказана, проігнорувати. Якщо ні, додати сторінку. => Збільшити лічильник слова.
    - Якщо не знайшли, додати до словника.
    - Повторити, доки не закінчяться слова.
5) Остання фукція це бабл сортування. Сортуємо за алфавітом:
    - Якщо поточна літера збігається, перейти до наступної. Звіряємо літери.
    - +Функція write, яка містить функція, яка знаходить кількість сторінок.
    - +Функція writePages, яка виводить сторінки по черзі.

На рисунках 3.1 і 3.2 показані результат роботи програм.

Рисунок 3.1 – Завдання 1

Рисунок 3.2 – Завдання 2



Висновок

Під час виконання лабораторної роботи використано методи імперативного програмування. Використано конструкцію goto на мові C#. Текстові дані зчитуються з пам’яті, інструкції виконуються по черзі.

goto є оператором безумовного переходу. Коли в програмі зустрічається оператор goto, її виконання переходить безпосередньо до того місця, на яке вказує цей оператор.

Він уже давно "вийшов із вживання", оскільки сприяє створенню "макаронного" коду. Головний недолік goto з погляду програмування полягає в тому, що він вносить у програму безладдя і робить її практично незручною. Але іноді застосування оператора goto може швидше прояснити, ніж заплутати хід виконання програми.

Посилання на Github

https://github.com/violettpv/Lab-1-Multi-paradigm-programming