Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний
інститут імені ІгоряСікорського"
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«Мультипарадигменне програмування»

„Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.1”

Виконав(ла)
(шифр, прізвище, ім'я, по батькові)

ІТ-04 Максименко Антон

Перевірив
(прізвище, ім'я, по батькові)

Очеретяний О. К.

Київ 2021

1 ЗАВДАННЯ

- 1.1 Перше завдання:

Обчислювальна задача тут тривіальна: для текстового файлу ми хочемо відобразити N (наприклад, 25) найчастіших слів і відповідну частоту їх повторення, упорядковано за зменшенням. Слід обов'язково нормалізувати використання великих літер і ігнорувати стоп-слова, як «the», «for» тощо. Щоб все було просто, ми не піклуємося про порядок слів з однаковою частотою повторень.

```
Input:
White tigers live mostly in India
Wild lions live mostly in Africa

Output:
live - 2
mostly - 2
africa - 1
india - 1
lions - 1
tigers - 1
white - 1
wild - 1
```

- 1.2 Друге завдання:

Тепер, нам потрібно виконати задачу, що називається словниковим індексуванням. Для текстового файлу виведіть усі слова в алфавітному порядку разом із номерами сторінок, на яких Ці слова знаходяться. Ігноруйте всі слова, які зустрічаються більше 100 разів. Припустимо, що сторінка являє собою послідовність із 45 рядків (1800символів).

```
abatement - 89
abhorrence - 101, 145, 152, 241, 274, 281
abhorrent - 253
abide - 158, 292
```

2 ВИКОНАННЯ

- 2.1 Псевдокод алгоритму першого завдання

1) Початок
2) отримати файл у вигляді змнної типу строка
3) якщо індекс кінцевого символу менше довжини строки роглядаємо
4) якщо символ не буква перейти до наступного символу
5) якщо буква перевести до нижнього регістру
6) додати ддо тимчасової змінної та перейти до наступного символа
7) якщо отримали стоп слово перейти до наступного символу
8) інакше додати до масиву з значенням 1
9) перейти до наступного символа та почати з пчатку
9) Посортувати масив слів за спаданням к-сті повторів у тексті. (метод бульбашки)
10) Вивести n перших слів масиву та к-сть їх повторів у тексті, де n – константа вказана на початку програми

- 2.2 Псевдокод алгоритму другого завдання

1) Початок
2) Якщо не кінець строки, то зчитати наступний символ
3) Кожну букву у слові перетворити в «маленьку», та очистити слова від розділових знаків та символыв що не э буквами
4) Перевірити чи є дане слово «стоп-словом», перелік яких міститься у масиві на початку програми
5) Якщо дане слово – «стоп-слово», то перейти до пункту 2
6) Перевірити чи є дане слово у масиві, де ми зберігаємо слова, які уже зустрічали, к-сть їх повторів, та масив сторынок де це слово є
7) Якщо дане слово раніше уже зустрічалось, то внести сторінку до масиву
8) Якщо дане слово раніше не зустрічалось, то додати дане слово до масиву, вказавши к-сть повторів рівним одиниці, та додати номер сторінки в масив
9) Посортувати масив слів в алфавітному порядку за допомогою методу бульбашки та вбудованої функції equals()
10) Вивести слова та номери сторінок, на яких вони зустрічаються, за вийнятком слів, к-сть повторів яких більша за 100.

- 2.3 Програмна реалізація алгоритму мовою c#

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;

namespace multiParadigmP_1_2
{
   class Program
   {
      ///<summary>Declaration</summary>
      ///some variables to use
      public struct Item
      {
         public Item(int frequency, string chars)
         {
            amount = frequency;
            word = chars;
            pages = new int[10000];
            amountOfPages = 0;

         }
         public int amountOfPages;
         public int amount;
         public string word;
         public int[] pages;
      }
      public static string currentWord = "";
      public static string currentSymbol;
      public static int charInFileCounter = 0;

      public static string[] allWords = new string[90000];
      public static int allWordsPointer = 0 ;

      public static Hashtable TF = new Hashtable();
      public static SortedList<string, int> result = new SortedList<string, int>();

      public static string fileName = @"C:\Users\Anton Maksymenko\Desktop\full.txt";

      public static string[] stopWords = { "a", "about", "above", "across", "after", "afterwards", "again", "against", "all",
         "almost", "alone", "along", "already", "also", "although", "always", "am", "among", "amongst", "amoungst", "amount", "an",
         "and", "another", "any", "anyhow", "anyone", "anything", "anyway", "anywhere", "are", "around", "as", "at", "back", "be",
         "became", "because", "become", "becomes", "becoming", "been", "before", "beforehand", "behind", "being", "below", "beside",
         "besides", "between", "beyond", "bill", "both", "bottom", "but", "by", "call", "can", "cannot", "cant", "co", "con", "could",
         "couldnt", "cry", "de", "describe", "detail", "do", "done", "down", "due", "during", "each", "eg", "eight", "either", "eleven",
         "else", "elsewhere", "empty", "enough", "etc", "even", "ever", "every", "everyone", "everything", "everywhere", "except", "few",
         "fifteen", "fify", "fill", "find", "fire", "first", "five", "for", "former", "formerly", "forty", "found", "four", "from", "front",
         "full", "further", "get", "give", "go", "had", "has", "hasnt", "have", "he", "hence", "her", "here", "hereafter", "hereby", "herein",
         "hereupon", "hers", "herself", "him", "himself", "his", "how", "however", "hundred", "ie", "if", "in", "inc", "indeed", "interest",
         "into", "is", "it", "its", "itself", "keep", "last", "latter", "latterly", "least", "less", "ltd", "made", "many", "may", "me",
         "meanwhile", "might", "mill", "mine", "more", "moreover", "most", "mostly", "move", "much", "must", "my", "myself", "name",
"namely",
         "neither", "never", "nevertheless", "next", "nine", "no", "nobody", "none", "noone", "nor", "not", "nothing", "now", "nowhere", "of",
         "off", "often", "on", "once", "one", "only", "onto", "or", "other", "others", "otherwise", "our", "ours", "ourselves", "out", "over",
         "own", "part", "per", "perhaps", "please", "put", "rather", "re", "same", "see", "seem", "seemed", "seeming", "seems", "serious",
"several",
         "she", "should", "show", "side", "since", "sincere", "six", "sixty", "so", "some", "somehow", "someone", "something", "sometime",
"sometimes",
```

```
        "somewhere", "still", "such", "system", "take", "ten", "than", "that", "the", "their", "them", "themselves", "then", "thence", "there",
"thereafter",
        "thereby", "therefore", "therein", "thereupon", "these", "they", "thick", "thin", "third", "this", "those", "though", "three", "through",
"throughout",
        "thru", "thus", "to", "together", "too", "top", "toward", "towards", "twelve", "twenty", "two", "un", "under", "until", "up", "upon",
"us", "very",
        "via", "was", "we", "well", "were", "what", "whatever", "when", "whence", "whenever", "where", "whereafter", "whereas",
"whereby", "wherein",
        "whereupon", "wherever", "whether", "which", "while", "whither", "who", "whoever", "whole", "whom", "whose", "why", "will",
"with", "within",
        "without", "would", "yet", "you", "your", "yours", "yourself", "yourselves", "the" };


    static void Main(string[] args)
    {
        //reading file to a string to then operate with it
        string text = File.ReadAllText(fileName);
        //text = "White tigers live mostly in India Wild lions live mostly in Africa";
        goto GetAllWords;


    GetAllWords:
        //while we not check all characters in string continue
        if (charInFileCounter >= text.Length)
        {
            goto Ending;
        }


        currentSymbol = text[charInFileCounter].ToString();

        //check if our current symbol is an letter
        if (!(currentSymbol == "." || currentSymbol == "," || currentSymbol == "!" || currentSymbol == "?" || currentSymbol == ":" ||
            currentSymbol == ";" || currentSymbol == "(" || currentSymbol == ")" || currentSymbol == "[" || currentSymbol == "]" ||
            currentSymbol == "*" || currentSymbol == "-" || currentSymbol == "\n" || currentSymbol == "\r" || currentSymbol == "".ToString()
||
            currentSymbol == """ || currentSymbol == "&" || currentSymbol == "" || currentSymbol == " " || currentSymbol == "_"))
        {

            int ascii = (int)Convert.ToChar(currentSymbol);
            if (ascii >= 'A' && ascii <= 'Z')
            //transform to lower start
            {
                // change the character
                char c = (char)(text[charInFileCounter] + 32);
                currentWord += c;
                charInFileCounter++;
                goto GetAllWords;
            }
            //transform to lower end
            else
            {
                currentWord += currentSymbol;
                charInFileCounter++;
                //repetition to add last word in a file
                if (charInFileCounter == text.Length)
                {

                    //check if it is not "stop-word"
                    if (Array.Exists(stopWords, element => element == currentWord))
                    {
                        charInFileCounter++;
                        currentWord = "";
                        goto GetAllWords;
                    }
                    else
```

```
            {
                if (TF.Contains(currentWord))
                {
                    var element = TF[currentWord];
                    TF[currentWord] = (object)((int)element + 1);
                    var elementnew = TF[currentWord];

                }
                else
                {
                    TF.Add(currentWord, 1);
                }

                charInFileCounter++;
                currentWord = "";
                goto GetAllWords;
            }
        }
        goto GetAllWords;
    }

}
else if (currentWord == "")
{
    charInFileCounter++;
    goto GetAllWords;
}
//if current symbol is not a letter then add word that we make to array
else
{
    //check if it is not "stop-word"
    if (Array.Exists(stopWords, element => element == currentWord))
    {
        charInFileCounter++;
        currentWord = "";
        goto GetAllWords;
    }
    //if everything fine with word
    else
    {
        allWords[allWordsPointer] = currentWord;
        allWordsPointer++;

        //cheching if we already has that word in array
        if (TF.Contains(currentWord))
        {
            //changing the amount of repetitions if true
            var element = TF[currentWord];
            TF[currentWord] = (object)((int)element + 1);
            var elementnew = TF[currentWord];

        }
        else
        {
            //adding word if it`s enough long
            if (currentWord.Length >= 3 || currentWord == "i")
            { TF.Add(currentWord, 1); }
        }
        //process next character
        charInFileCounter++;
        currentWord = "";
        goto GetAllWords;
    }

}
```

```
        Ending:
            //using enumerator to replace foreach
            Item[] results = new Item[TF.Count];
            var tfcounter = 0;
            var counter = TF.GetEnumerator();
            counter.Reset();
            counter.MoveNext();

        DuplicatingData:
            //loop to add data to array
            if (tfcounter < TF.Count)
            {
                var key = counter.Key;
                var value = counter.Value;

                results[tfcounter] = new Item((int)value, key.ToString());

                tfcounter++;
                if (counter.MoveNext())
                {
                    goto DuplicatingData;
                }

            }
            //now we have array with all words


            Item temp;
            /// Sorting by strings using bubble sort
            int j = 0;
            int n = results.Length;
        cyclestart:
            if (j < n - 1)
            {
                int i = j + 1;
            innercycle:
                if (i < n)
                {

                    if (results[j].word.CompareTo(results[i].word) > 0)
                    {
                        temp = results[j];
                        results[j] = results[i];
                        results[i] = temp;
                    }
                    i++;
                    goto innercycle;

                }
                j++;
                goto cyclestart;
            }

            goto FindPages;



        FindPages:

            int pointerInWords = 0;
            int pointerInItems = 0;
            int currentPage = 1;

    loopThroughItems:
            if(pointerInItems< results.Length)
```

```
    {
        var currentItem = results[pointerInItems];
    loopThroughWordsForItem:
        if (pointerInWords > 254* currentPage)
        {
            currentPage++;
        }
        if (pointerInWords < allWordsPointer)
        {

            if (allWords[pointerInWords] == currentItem.word)
            {
                currentItem.pages[currentItem.amountOfPages] = currentPage;
                currentItem.amountOfPages++;
                pointerInWords = (pointerInWords / 254 + 1) * 254;
                goto loopThroughWordsForItem;
            }
            pointerInWords++;
            goto loopThroughWordsForItem;
        }
        else
        {
            pointerInWords = 0;
            pointerInItems++;
            currentPage = 0;
            goto loopThroughItems;
        }
    }









    int displaycounter = 0;
    int amountOfElements = results.Length;
  loopstart:

    if ( displaycounter < amountOfElements)
    {
        if (results[displaycounter].amount > 100)
        {
            displaycounter++;
            goto loopstart;
        }
        Console.Write(results[displaycounter].word + " - ");
        int pagesCounter = 0;

displayEachWord:
        if(pagesCounter< results[displaycounter].amount)
        {
            if(results[displaycounter].pages[pagesCounter] == 0)
            {
                pagesCounter++;
                goto displayEachWord;
            }
            Console.Write(results[displaycounter].pages[pagesCounter]);
            if(pagesCounter < results[displaycounter].amount - 1)
            {
                Console.Write(", ");
            }
```

```
                pagesCounter++;
                goto displayEachWord;

            }
            else
            {
                Console.Write("\n");
                displaycounter++;
                goto loopstart;
            }


        }

    }


  }

}
```

2.4 Програмна реалізація алгоритму мовою C#

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;

namespace multiParadigmP_1
{
   class Program
   {
      ///<summary>Declaration</summary>
      ///some variables to use
      public struct Item
      {

         public Item(int frequency, string chars)
         {
            amount = frequency;
            word = chars;
         }

         public int amount;
         public string word;
      }
      public static int wordsToDisplay = 25;
      public static string currentWord = "";
      public static string currentSymbol;
      public static int charInFileCounter = 0;

      public static Hashtable TF = new Hashtable();
      public static SortedList<string, int> result = new SortedList<string, int>();

      //public static  string fileName = @"C:\Users\Anton Maksymenko\Desktop\prideAndPrejustice.txt";
      public static string fileName = @"C:\Users\Anton Maksymenko\Desktop\full.txt";
      public static string[] stopWords = { "a", "about", "above", "across", "after", "afterwards", "again", "against", "all",
         "almost", "alone", "along", "already", "also", "although", "always", "am", "among", "amongst", "amoungst", "amount",
"an",
         "and", "another", "any", "anyhow", "anyone", "anything", "anyway", "anywhere", "are", "around", "as", "at", "back", "be",
         "became", "because", "become", "becomes", "becoming", "been", "before", "beforehand", "behind", "being", "below",
"beside",
         "besides", "between", "beyond", "bill", "both", "bottom", "but", "by", "call", "can", "cannot", "cant", "co", "con", "could",
```

```
        "couldnt", "cry", "de", "describe", "detail", "do", "done", "down", "due", "during", "each", "eg", "eight", "either", "eleven",
        "else", "elsewhere", "empty", "enough", "etc", "even", "ever", "every", "everyone", "everything", "everywhere", "except",
"few",
        "fifteen", "fify", "fill", "find", "fire", "first", "five", "for", "former", "formerly", "forty", "found", "four", "from", "front",
        "full", "further", "get", "give", "go", "had", "has", "hasnt", "have", "he", "hence", "her", "here", "hereafter", "hereby",
"herein",
        "hereupon", "hers", "herself", "him", "himself", "his", "how", "however", "hundred", "ie", "if", "in", "inc", "indeed", "interest",
        "into", "is", "it", "its", "itself", "keep", "last", "latter", "latterly", "least", "less", "ltd", "made", "many", "may", "me",
        "meanwhile", "might", "mill", "mine", "more", "moreover", "most", "mostly", "move", "much", "must", "my", "myself",
"name", "namely",
        "neither", "never", "nevertheless", "next", "nine", "no", "nobody", "none", "noone", "nor", "not", "nothing", "now",
"nowhere", "of",
        "off", "often", "on", "once", "one", "only", "onto", "or", "other", "others", "otherwise", "our", "ours", "ourselves", "out",
"over",
        "own", "part", "per", "perhaps", "please", "put", "rather", "re", "same", "see", "seem", "seemed", "seeming", "seems",
"serious", "several",
        "she", "should", "show", "side", "since", "sincere", "six", "sixty", "so", "some", "somehow", "someone", "something",
"sometime", "sometimes",
        "somewhere", "still", "such", "system", "take", "ten", "than", "that", "the", "their", "them", "themselves", "then", "thence",
"there", "thereafter",
        "thereby", "therefore", "therein", "thereupon", "these", "they", "thick", "thin", "third", "this", "those", "though", "three",
"through", "throughout",
        "thru", "thus", "to", "together", "too", "top", "toward", "towards", "twelve", "twenty", "two", "un", "under", "until", "up",
"upon", "us", "very",
        "via", "was", "we", "well", "were", "what", "whatever", "when", "whence", "whenever", "where", "whereafter", "whereas",
"whereby", "wherein",
        "whereupon", "wherever", "whether", "which", "while", "whither", "who", "whoever", "whole", "whom", "whose", "why",
"will", "with", "within",
        "without", "would", "yet", "you", "your", "yours", "yourself", "yourselves", "the" };


    static void Main(string[] args)
    {
         //reading file to a string to then operate with it
        string text = File.ReadAllText(fileName);
        //text = "White tigers live mostly in India Wild lions live mostly in Africa";
        goto GetAllWords;


GetAllWords:
        //while we not check all characters in string continue
        if (charInFileCounter >= text.Length)
        {
            goto Ending;
        }


        currentSymbol = text[charInFileCounter].ToString();

        //check if our current symbol is an letter
        if (!(currentSymbol == "." || currentSymbol == "," || currentSymbol == "!" || currentSymbol == "?" || currentSymbol == ":" ||
            currentSymbol == ";" || currentSymbol == "(" || currentSymbol == ")" || currentSymbol == "[" || currentSymbol == "]" ||
            currentSymbol == "*" || currentSymbol == "-" || currentSymbol == "\n" || currentSymbol == "\r" || currentSymbol ==
"'".ToString() ||
            currentSymbol == "'" || currentSymbol == "&" || currentSymbol == "" || currentSymbol == " " || currentSymbol == "_"))
        {

            int ascii = (int)Convert.ToChar(currentSymbol);
            if (ascii >= 'A' && ascii <= 'Z')
            //transform to lower start
            {
                // change the character
                char c = (char)(text[charInFileCounter] + 32);
                currentWord += c;
                charInFileCounter++;
                goto GetAllWords;
```

```
    }
    //transform to lower end
    else
    {
        currentWord += currentSymbol;
        charInFileCounter++;
        //repetition to add last word in a file
        if(charInFileCounter== text.Length )
        {

            //check if it is not "stop-word"
            if (Array.Exists(stopWords, element => element == currentWord))
            {
                charInFileCounter++;
                currentWord = "";
                goto GetAllWords;
            }
            else
            {
                if (TF.Contains(currentWord))
                {
                    var element = TF[currentWord];
                    TF[currentWord] = (object)((int)element + 1);
                    var elementnew = TF[currentWord];

                }
                else
                {
                    TF.Add(currentWord, 1);
                }

                charInFileCounter++;
                currentWord = "";
                goto GetAllWords;
            }
        }
        goto GetAllWords;
    }

}
else if (currentWord == "")
{
    charInFileCounter++;
    goto GetAllWords;
}
//if current symbol is not a letter then add word that we make to array
else
{
    //check if it is not "stop-word"
    if (Array.Exists(stopWords, element => element == currentWord))
    {
        charInFileCounter++;
        currentWord = "";
        goto GetAllWords;
    }
    //if everything fine with word
    else
    {
        //cheching if we already has that word in array
        if (TF.Contains(currentWord))
        {
            //changing the amount of repetitions if true
            var element = TF[currentWord];
            TF[currentWord] = (object)((int)element + 1);
            var elementnew = TF[currentWord];
```

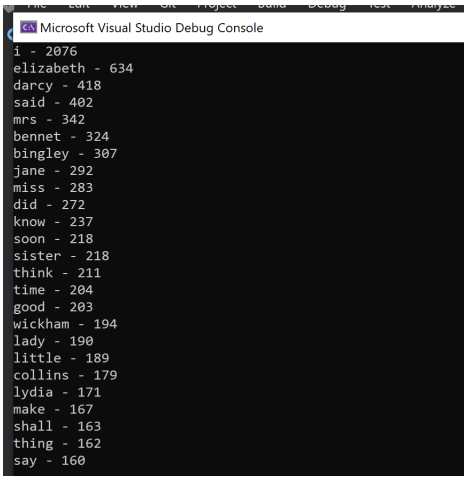
```
                }
                else
                {
                    //adding word if it`s enough long
                    if (currentWord.Length >= 3 || currentWord == "i")
                    { TF.Add(currentWord, 1); }
                }
                //process next character
                charInFileCounter++;
                currentWord = "";
                goto GetAllWords;
            }

        }

    /*
Ending:

        var counter = TF.GetEnumerator();
        counter.Reset();
        counter.MoveNext();
  SortingLoop:
        if (counter.Current.ToString() != null)
        {
            var currentObject = (DictionaryEntry)counter.Current;
            var key = currentObject.Key.ToString();
//dirty hack of using sortedList
            result.Add(key, (int)TF[key]);

            if (counter.MoveNext())
            {
                goto SortingLoop;
            }
        }

        foreach (var pair in result)
        {
            Console.WriteLine(pair.Key + " - " + pair.Value);
        }*/


///ALTERNATIVE ENDENG WITH SORT
Ending:
        //using enumerator to replace foreach
        Item[] results = new Item[TF.Count];
        var tfcounter = 0;
        var counter = TF.GetEnumerator();
        counter.Reset();
        counter.MoveNext();

DuplicatingData:
        //loop to add data to array
        if (tfcounter < TF.Count)
        {
            var key = counter.Key;
            var value = counter.Value;

            results[tfcounter] = new Item((int)value, key.ToString());

            tfcounter++;
            if (counter.MoveNext())
            {
                goto DuplicatingData;
            }

        }
```

```
        //now we have array with all words


        Item temp;
        //sorting by amount
        int j = 0;
        int n = results.Length;
cyclestart:
        if (j < n - 1)
        {
            int i = j + 1;
    innercycle:
            if (i < n)
            {
                if (results[j].amount < results[i].amount)
                {
                    temp = results[j];
                    results[j] = results[i];
                    results[i] = temp;
                }
                i++;
                goto innercycle;
            }
            j++;
            goto cyclestart;
        }
            /// Sorting by strings using bubble sort
            //int j = 0;
            //int n = results.Length;
            //cyclestart:
            //if (j < n - 1)
            //{
            //    int i = j + 1;
            //    innercycle:
            //    if (i < n)
            //    {

            //        if (results[j].word.CompareTo(results[i].word) > 0)
            //        {
            //            temp = results[j];
            //            results[j] = results[i];
            //            results[i] = temp;
            //        }
            //        i++;
            //        goto innercycle;

            //    }
            //    j++;
            //    goto cyclestart;
            //}

            goto displayElements;



      displayElements:
        int displaycounter = 0;
        int amountOfElements = results.Length;
    loopstart:

            if (displaycounter < wordsToDisplay && displaycounter< amountOfElements)
            {
                Console.WriteLine(results[displaycounter].word + " - " + results[displaycounter].amount);
                displaycounter++;
                goto loopstart; }}}}
```

3 ТЕСТУВАННЯ

- 3.1 Приклад роботи для першого завдання

Вхідні дані: повний текст книги pride and prejudice у файлі full.txt
Вихідні дані:



- 3.2 Приклад роботи для другого завдання

Вхідні дані: повний текст книги pride and prejudice у файлі full.txt
Приклад фрагменту

CHAPTER I.

It is a truth universally acknowledged, that a single man in possession
of a good fortune, must be in want of a wife.

However little known the feelings or views of such a man may be on his
first entering a neighbourhood, this truth is so well fixed in the minds
of the surrounding families, that he is considered as the rightful
property of some one or other of their daughters.

"My dear Mr. Bennet," said his lady to him one day, "have you heard that
Netherfield Park is let at last?"

Mr. Bennet replied that he had not.

"But it is," returned she; "for Mrs. Long has just been here, and she
told me all about it."

Mr. Bennet made no answer.

"Do not you want to know who has taken it?" cried his wife impatiently.

"_You_ want to tell me, and I have no objection to hearing it."

Вихідні дані:

```
Microsoft Visual Studio Debug Console
abatement - 60
abhorrence - 68, 99, 104, 165, 186, 190
abhorrent - 173
abide - 108
abiding - 110
abilities - 42, 43, 65, 96, 107, 121
able - 10, 20, 33, 46, 49, 51, 52, 55, 59, 61, 65, 67, 73, 78, 80, 81, 89, 90, 94, 97, 108, 111, 112, 115, 116, 117, 122
, 129, 137, 138, 142, 143, 145, 146, 150, 153, 154, 158, 159, 163, 165, 166, 168, 177, 179, 185, 186, 192, 197,
ablution - 73
abode - 33, 34, 38, 67, 75, 80, 110, 163
abominable - 17, 28, 41, 75, 100,
abominably - 26, 82, 168, 186
abominate - 165, 184
abound - 61
abouts - 160
abroad - 121, 123, 147, 179
abrupt - 127
abruptly - 22, 96
abruptness - 124,
absence - 30, 32, 37, 45, 46, 54, 60, 61, 64, 65, 68, 78, 93, 107, 108, 122, 123, 128, 130, 141, 146, 150, 177,
absent - 16, 125, 142, 144
absolute - 46, 143, 159, 191
absolutely - 8, 13, 17, 56, 57, 77, 91, 103, 104, 107, 118, 127, 140, 152, 163, 168, 186, 189
absurd - 34, 101, 107, 184, 188
absurdities - 79, 136
absurdity - 118
abundant - 143
abundantly - 39, 51, 77
abuse - 2, 104
abused - 112, 123
```

4 ВИСНОВОК

Під час виконання даної лабораторної роботи я дослідив підхід написання програми з з ымперативноъ точки зору та з використанням оператора goto. Даний оператор дозволяє програмісту керувати потоком виконання програми. Такий підхід вже не використовується та э майже забороненим через заплутаність коду що утворюється