МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по практической работе №4
по дисциплине «Программирование»
ТЕМА: «Класс string»

Студентка гр. 0324 _____ Жигалова Д.А.

Преподаватель _____ Глущенко А.Г

Санкт-Петербург
2020

Цель работы

Изучение способов обработки текстовых данных. Получение практических навыков работы с файлами. Изучение различных алгоритмов поиска подстроки в строке.

Основные теоретические положения

Класс string предназначен для работы со строками типа char, которые представляют собой строчку с завершающим нулем (символ ‘\0’). Класс string был введен как альтернативный вариант для работы со строками типа char.

Класс string обладает широким функционалом:

- функция compare() сравнивает одну часть строки с другой;
- функция length() определяет длину строки;
- функции find() и rfind() служат для поиска подстроки в строке (отличаются функции лишь направлением поиска);
- функция erase() служит для удаления символов;
- функция replace() выполняет замену символов;
- функция insert() необходима, чтобы вставить одну строку в заданную позицию другой строки;

Но весь функционал string накладывает и свой негативный отпечаток. Основным недостатком string в сравнении с типом char является замедленная скорость обработки данных.

При работе со строками часто будет возникать потребность в поиске набора символа или слов (поиска подстроки в строке). При условии, что текст может быть крайне большим, хочется, чтобы алгоритм поиска подстроки работал быстро.

Самый простой способ подстроки в строке – Линейный поиск – циклическое сравнение всех символов строки с подстрокой. Действительно, этот способ первый приходит в голову, но очевидно, что он будет самым долгим.

На первых двух итерациях цикла сравниваемые буквы не будут совпадать. На третьей же итерации, совпал символ ‘L’, это означает, что теперь нужно

сравнивать следующий символ подстроки со следующим символом строки. Видно, что символы отличаются, поэтому алгоритм продолжает свою работу. На четвертой же итерации подстрока была найдена.

Если представить, что исходная строка непорядок больше и подстрока находится в конце строки (или вовсе отсутствует), то сразу видны минусы данного алгоритма.

Одним из самых популярных алгоритмов, который работает быстрее, чем приведенный выше алгоритм, является алгоритм Кнута-Морриса-Пратта (КМП). Идея заключается в том, что не нужно проходить и сравнивать абсолютно все символы строки, если известны символы, которые есть и в строке, и в подстроке.

Суть алгоритма: дана подстрока S и строка T. Требуется определить индекс, начиная с которого образец S содержится в строке T. Если S не содержится в T, необходимо вернуть индекс, который не может быть интерпретирован как позиция в строке.

Хоть алгоритм и работает быстрее, по-прежнему необходимо сначала пройти всю строку, чтобы определить префиксы или суффиксы (вхождение (индексы) символов).

Алгоритм Бойера-Мура в отличие от КМП полностью не зависим и не требует заранее проходить по строке. Этот алгоритм считается наиболее быстрым среди алгоритмов общего назначения, предназначенных для поиска подстроки в строке.

Преимущество этого алгоритма в том, что ценной некоторого количества предварительных вычислений над подстрокой (но не над исходной строкой, в которой ведётся поиск), подстрока сравнивается с исходным текстом не во всех позициях (пропускаются позиции, которые точно не дадут положительный результат).

Поиск подстроки ускоряется благодаря созданию таблиц сдвигов. Сравнение подстроки со строки начинается с последнего символа подстроки, а затем происходит прыжок, длина которого определяется по таблице сдвигов.

Таблица сдвигов строится по подстроке так чтобы перепрыгнуть максимальное количество символов строки и не пропустить вхождение подстроки в строку.

Постановка задачи

Необходимо написать программу, которая:

Необходимо написать программу, которая реализует поставленную задачу:

1) С клавиатуры или с файла (*) (пользователь сам может выбрать способ ввода) вводится последовательность, содержащая от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между соседними словами произвольное количество пробелов. За последним символом стоит точка.

2) Необходимо отредактировать входной текст:

· удалить лишние пробелы;

· удалить лишние знаки препинания (под «лишними» подразумевается несколько подряд идущих знаков (обратите внимание, что «…» - корректное использование знака) в тексте);

· исправить регистр букв, если это требуется (пример некорректного использования регистра букв: пРиМЕр);

3) Выполнить задание по варианту: 6

После окончания ввода последовательности вывести на экран сначала все слова, содержащие только буквы, затем слова, содержащие только цифры, а потом слова, содержащие и буквы, и цифры.

4) Выполнить задание по варианту: 6

Вывести все слова исходной последовательности на экран вертикально.

5) Необходимо найти подстроку, которую введёт пользователь в имеющейся строке. Реализуйте два алгоритма: первый алгоритма – Линейный поиск, а второй алгоритм согласно вашему номеру в списке. Четные номера должны реализовать алгоритм КНМ, а нечетные – Бойера-Мура. (*)

Выполнение работы

Для решения поставленных была создана программа на языке программирования C++. Итоговый код программы представлен в приложении А, а результат работы в приложении В.

В первой задаче была созданы функция note, которая с клавиатуры или с файла позволяла вводить последовательность (файл при этом должен находится на диске D и называться hello.txt).

```
string note() {
    int answer2;
    string text;
    cout << "\nSelect the input method\n";
    cout << "\n1) Keyboard\n";
    cout << "\n2) File (D:\\hello.txt)\n";
    cin >> answer2;

    if (answer2 == 1) {
        cin.ignore();
        getline(cin, text);
    }
    else if (answer2 == 2) {
        ifstream in("D:\\hello.txt"); // окрываем файл для чтения
        if (in.is_open())
        {
            while (getline(in, text))
            {
                cout << "You entered: " << endl;
                cout << text << endl;
            }
        }
        else cout << "File doesn't exist\n";
        in.close();
    }
    return text;
}
```

Во второй задаче для редактирования входного текста была создана функция cleaning.

```
string cleaning(string text, int len) {

    while (text[0] == '.' || text[0] == ' ') {
        text.erase(0, 1);
        len--;
    }
    while (text[len] == ' ') {
        text.erase(len, 1);
        len--;
    }
    for (int i = 0; i < len; i++) {
        if (text[i] == ' ' && (text[i+1] == ' '|| text[i + 1] == ',' || text[i + 1] == '.'|| text[0] == ' ')) {
                text.erase(i, 1);
                len--;
        }
    }
    for (int i = 0; i < len; i++) {
        if ((text[i] == '.' && text[i + 1] == '.') && text[i + 2] == '.')
            i += 3;
        else if((text[i] == '.' || text[i] == ',' || text[i] == ';' || text[i] == ':' || text[i] == '!' || text[i
            text.erase(i, 1);
            len--;
            i--;
        }
    }
    for (int i = 0; i < len; i++) {
        if (i == 0)
            text[i] = toupper(text[i]);
        else
            text[i] = tolower(text[i]);
    }
    for (int i = 0; i < len; i++) {
        if (text[i] == '.' && text[i + 1] == ' ')
            text[i+2] = toupper(text[i+2]);
    }
    return text;
}
```

В третьей задаче была создана функция, которая позволяла вывести на экран сначала все слова, содержащие только буквы, затем слова, содержащие только цифры, а потом слова, содержащие и буквы, и цифры.

В четвертой задаче также была создана функция, которая выводила все слова исходной последовательности на экран вертикально.

```
string word, text2, line;
text2 = "";
text += " ";
int index = 0;
//cout << text << endl;
int End = 0;

while (End >= 0) {
    End = text.find(' ');
    word = text.substr(0, End);
    text = text.erase(0, End + 1);
    if (word.length() < 10) {
        int k = 10 - word.length();
        for (int j = 0; j < k; j++) {
            word += " ";
        }
    }
    text2 += word;
}
text2 = text2.erase(text2.length() - 10, text2.length()-1);

for (int j = 0; j < 10; j++) {
    for (int i = 0; i < text2.length(); i += 10) {
        line += text2[i+j];
        line += " ";
    }
    cout << line << endl;
    line = "";
}
```

В пятой задаче также была создана функция поиска подстроки линейным способом, к сожалению, другим способом решить поставленную задачу не удалось.

Вывод

Изучены способы обработки текстовых данных. Получены навыки работы с файлами. Изучен алгоритм поиска подстроки в строке.

ПРИЛОЖЕНИЕ А
ПОЛНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

string note() {
    int answer2;
    string text;
    cout << "\nSelect the input method\n";
    cout << "\n1) Keyboard\n";
    cout << "\n2) File (D:\\hello.txt)\n";
    cin >> answer2;

    if (answer2 == 1) {
        cin.ignore();
        getline(cin, text);
    }
    else if (answer2 == 2) {
        ifstream in("D:\\hello.txt"); // окрываем файл для чтения
        if (in.is_open())
        {
            while (getline(in, text))
            {
                cout << "You entered: " << endl;
                cout << text << endl;
            }
        }
        else cout << "File doesn't exist\n";
        in.close();
    }
    return text;
}

string cleaning(string text, int len) {

    while (text[0] == '.' || text[0] == ' ') {
        text.erase(0, 1);
        len--;
    }
    while (text[len] == ' ') {
        text.erase(len, 1);
        len--;
    }
    for (int i = 0; i < len; i++) {
```

```
        if (text[i] == ' ' && (text[i+1] == ' '|| text[i + 1] == ',' ||
text[i + 1] == '.'|| text[0] == ' ')) {
                text.erase(i, 1);
                len--;
        }
    }
    for (int i = 0; i < len; i++) {
        if ((text[i] == '.' && text[i + 1] == '.') && text[i + 2] == '.')
            i += 3;
        else if((text[i] == '.' || text[i] == ',' || text[i] == ';' ||
text[i] == ':' || text[i] == '!' || text[i] == '?') && (text[i+1] == '.'
|| text[i + 1] == ',' || text[i + 1] == ';' || text[i + 1] == ':' ||
text[i + 1] == '!' || text[i + 1] == '?')) {
            text.erase(i, 1);
            len--;
            i--;
        }
    }
    for (int i = 0; i < len; i++) {
        if (i == 0)
            text[i] = toupper(text[i]);
        else
            text[i] = tolower(text[i]);
    }
    for (int i = 0; i < len; i++) {
        if (text[i] == '.' && text[i + 1] == ' ')
            text[i+2] = toupper(text[i+2]);
    }
    return text;
}

void exerciseThree(string text){
    int End = 0;
    string str, word;
    char letters[26] =
{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r',
's','t','u','v','w','x','y','z'};
    char numbers[10] = {'0','1','2','3','4','5','6','7','8','9'};

    int len = text.length();
    for (int i = 0; i < len; i++) {
        if ((text[i] == ';' || text[i] == ',' || text[i] == '.' ||
text[i] == ':' || text[i] == '!' || text[i] == '?')) {
            text.erase(i, 1);
            len--;
            i--;
        }
```

```
}
str = text;
cout << "\nAll words containing only letters - ";

while (End >= 0) {
    bool foundNum = false;
    End = str.find(' ');
    word = str.substr(0, End);
    str = str.erase(0, End + 1);
    for (int i = 0; i < word.length(); i++) {
        for (auto j : numbers)
        if (word[i] == j) {
            foundNum = true;
            break;
        }
    }
    if (!foundNum) {
        cout << word << " ";
    }
}
End = 0;
str = text;
cout << "\n\nAll words containing only numbers - ";
while (End >= 0) {
    bool foundNum = false;
    End = str.find(' ');
    word = str.substr(0, End);
    str = str.erase(0, End + 1);
    for (int i = 0; i < word.length(); i++) {
        for (auto j : letters) {
            if (word[i] == j) {
                foundNum = true;
                break;
            }
        }
    }
    if (!foundNum) cout << word << " ";
}
End = 0;
str = text;
cout << "\n\nAll words containing only numbers and letters - ";
while (End >= 0) {
    bool foundNum = false;
    End = str.find(' ');
    word = str.substr(0, End);
    str = str.erase(0, End + 1);
    for (int i = 0; i < word.length(); i++) {
```

```
            for (auto j : letters) {
                if (word[i] == j) {
                    for (int m = i; m < word.length(); m++) {
                        for (auto k : numbers) {
                            if (word[m] == k) {
                                foundNum = true;
                                break;
                            }
                        }
                    }
                }
            }
        }
        for (int i = 0; i < word.length(); i++) {
            for (auto j : numbers) {
                if (word[i] == j) {
                    for (int m = i; m < word.length(); m++) {
                        for (auto k : letters) {
                            if (word[m] == k) {
                                foundNum = true;
                                break;
                            }
                        }
                    }
                }
            }
        }
        if (foundNum) cout << word << " ";
    }
}

void exerciseFour(string text) {
    int len = text.length();
    for (int i = 0; i < len; i++) {
        if ((text[i] == ';' || text[i] == ',' || text[i] == '.' ||
text[i] == ':' || text[i] == '!' || text[i] == '?')) {
            text.erase(i, 1);
            len--;
            i--;
        }
    }
    string word, text2, line;
    text2 = "";
    text += " ";
    int index = 0;
    //cout << text << endl;
    int End = 0;
```

```
    while (End >= 0) {
        End = text.find(' ');
        word = text.substr(0, End);
        text = text.erase(0, End + 1);
        if (word.length() < 10) {
            int k = 10 - word.length();
            for (int j = 0; j < k; j++) {
                word += " ";
            }
        }
        text2 += word;
    }
    text2 = text2.erase(text2.length() - 10, text2.length()-1);

    for (int j = 0; j < 10; j++) {
        for (int i = 0; i < text2.length(); i += 10) {
            line += text2[i+j];
            line += " ";
        }
        cout << line << endl;
        line = "";
    }
}

void exerciseFive(string text) {
    string subText;
    bool foundOne = false;
    bool found = true;
    int answer = 1;
    while (answer == 1) {

        cout << "\nEnter substring:" << endl;
        cin.ignore();
        getline(cin, subText);

        for (int i = 0; i < text.length(); i++) {
            if (text[i] == subText[0]) {
                for (int j = 1; j < subText.length(); j++) {
                    if (text[i + j] != subText[j])
                        found = false;
                }
                if (found == true) {
                    cout << "First index of subtring - " << i << "\n";
                    foundOne = true;
                }
            }
```

```
            found = true;
        }
        if (!foundOne)
            cout << "This subStr doesn't exist";
        cout << "\n\nDo you wanna repeat? (yes - 1; no - 0) (please
repeat entering if it isn't true)" << endl;
        cin >> answer;
    }
}

int main()
{
    int answer1, len;
    answer1 = 1;
    len = 0;
    string text;

    text = note(); // используется функция записи введенного текста (1
задание)

    while (text[len])
        ++len;
    cout << "\nNumber of symbols in the text: " << len << endl;

    text = cleaning(text, len); // 2 задание
    cout << "\nEdited input text:" << endl;
    cout << text << endl;

    exerciseThree(text);

    cout << "\n\nDisplays all words in the original sequence vertically
on the screen:\n" << endl;
    exerciseFour(text);

    exerciseFive(text);

    return 0;
}
```

ПРИЛОЖЕНИЕ В
РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

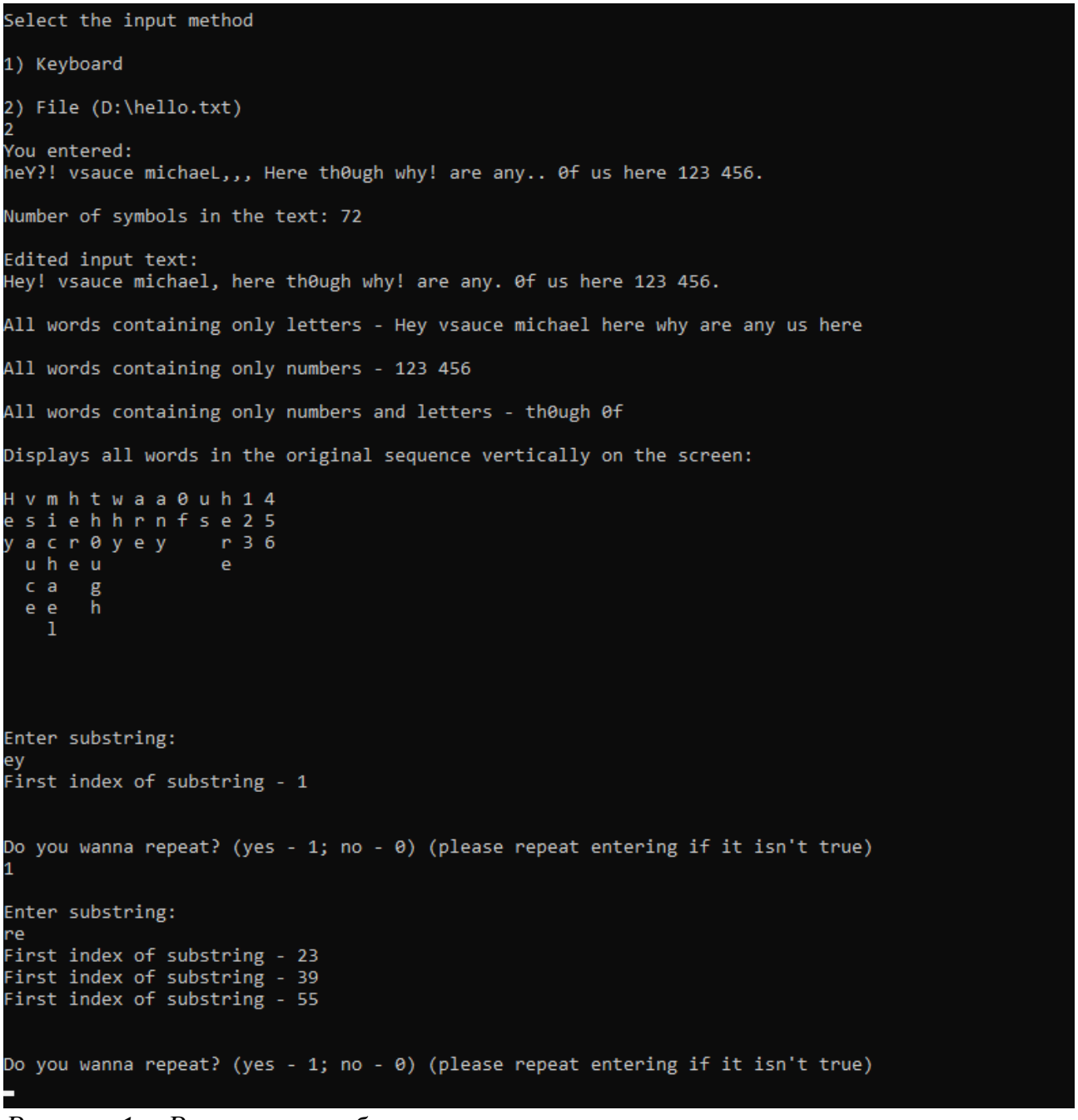

Рисунок 1 – Результат работы программы