

Цель работы.

Разработать алгоритм и написать программу на языке C++, которая позволяет: обрабатывать текстовые строки как массив символов.

Основные теоретические положения.

Текстовые строки представляются с помощью одномерных массивов символов. В языке C++ текстовая строка представляет собой набор символов, обязательно заканчивающийся нулевым символом ('\0'). Поэтому, если вы хотите создать текстовый массив для хранения 10 (N) символов, нужно выделить память под 11(N+1) символов.

Объявленный таким образом массив может использоваться для хранения текстовых строк, содержащих не более 10 символов. Нулевой символ позволяет определить границу между содержащимся в строке текстом и неиспользованной частью строки.

При определении строковых переменных их можно инициализировать конкретными значениями с помощью строковых литералов:

```
char S1[15] = "This is text";  
char S2[] = "Пример текста";
```

Последние два элемента переменной просто не используются, а строка автоматически подстраивается под длину инициализирующего текста.

При работе со строками можно обращаться к отдельным символам строки как в обычном одномерном массиве с помощью индексов:

```
cout << S1[0]; // На экране будет выведен символ 'Т'
```

Если строка формируется при помощи цикла (или иного способа), то необходимо в ее конец обязательно записать нулевой символ '\0'.

При выводе строк можно использовать форматирование (манипуляторы или функции потока вывода). Вывод текстовых строк на экран крайне простая задача:

```
char Str[21] = "Это пример текста";  
cout << Str << endl;  
cout << "Это текстовый литерал." << endl;
```

Ввод текста с клавиатуры можно осуществлять разными способами, каждый из которых имеет определенные особенности. Непосредственное чтение текстовых строк из потока вывода осуществляется до первого знака пробела. Такой способ чтения обеспечивает ввод символов до первого пробельного символа (не до конца строки). Остальные символы введенного с клавиатуры остаются в потоке ввода и могут быть прочитаны из него следующими операторами>>.

Для того чтобы прочесть всю строку полностью, можно воспользоваться одной из функций gets или gets_s (для этого в программу должен быть включен заголовочный файл <stdio.h>).

Функция gets имеет один параметр, соответствующий массиву символов, в который осуществляется чтение. Вторая функция (gets_s) имеет второй параметр, задающий максимальную длину массива символов.

Ввод текста, длина которого (вместе с нулевым символом) превышает значение второго параметра (то есть длины символьного массива), приводит к возникновению ошибки при выполнении программы.

Предпочтительно использование функции потока ввода cin.getline:

```
const int N = 21;  
char Str [N];
```

```
cin.getline (Str, N);      // Пусть введена строка "Это пример текста"  
cout << Str << endl;  // На экран будет выведено " Это пример текста"
```

Если длина введенного с клавиатуры текста превышает максимальную длину массива, в него будет записано (в нашем примере) 20 символов вводимого текста и нулевой символ. Остальные символы введенного текста остаются во входном потоке и могут быть взяты из него следующими инструкциями ввода. Функция `cin.getline` может иметь третий параметр, задающий символ, при встрече которого чтение строки из потока прекращается:

```
cin.getline (Str, N, '.');
```

Иногда чтение из потока невозможно (например, попытка считать слишком длинный текст). Для того чтобы продолжить чтение из потока, необходимо восстановить его нормальное состояние. Этого можно достигнуть с помощью функции потока `cin.clear()`, которая сбрасывает состояние потока в нормальное. Если забирать остатки данных из потока ввода не надо, то следует очистить его с помощью функции `cin.sync()`.

При обработке текстовых строк обычно используется набор типовых операций, к которым можно отнести:

- определение фактической длины текста, записанного в символьный массив;
- копирование текста из одной строки в другую;
- объединение двух строк;
- лексикографическое сравнение строк – в алфавитном порядке (больше, меньше, равно) и др.

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. Чтобы использовать возможности класса string, нужно подключить библиотеку <string> и пространство имен std. Объявление же переменной типа string осуществляется схоже с обычной переменной:

```
string S1; // Переменная с именем s1 типа string
string S2 = "Пример"; // объявление с инициализацией
```

Создание нового типа string было обусловлено недостатками работы с строками символов, который показывал тип char. В сравнении с этим типом string имеет ряд основных преимуществ:

- возможность использования для обработки строк стандартные операторы C++(=,+,<,,=>,+=,!<,,=>,[,])(=,+,<,,=>,+=,!<,,=>,[,])(=,+,<,,=>,+=,!<,,=>,[,]). Использование типа char приводило требовало написание чрезмерного программного кода;
- обеспечение лучшей надежности программного кода;
- обеспечение строки, как самостоятельного типа 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.

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.

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

Правила построения таблицы сдвигов:

- 1) Значение элемента таблицы равно удаленности соответствующего символа от конца шаблона (подстроки).
- 2) Если символ встречается более одного раза, то применяется значение, соответствующее символу, наиболее близкому к концу шаблона.
- 3) Если символ в конце шаблона встречается 1 раз, ему соответствует значение, равное длине образа; если более одного раза – значение, соответствующее символу, наиболее близкому к концу образа.
- 4) Для символов, отсутствующих в образе, применяется значение, равное длине шаблона.

Постановка задачи.

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:

Табл.1.

Т.к по ведомости группы 0324 11 в списке выполняю 5 пункт.

Таблица 1- Варианты задания.

1	Вывести на экран слова последовательности в обратном порядке.
2	Вывести на экран слова последовательности в алфавитном порядке.
3	Вывести на экран слова последовательности, не содержащие цифр.
4	Вывести на экран только те слова последовательности, в которых встречаются одинаковые буквы.
5	Вывести на экран только те слова последовательности, в которых первая буква слова встречается в этом слове еще раз.
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: Табл.2.

Т.к по ведомости группы 0324 11 в списке выполняю 4 пункт.

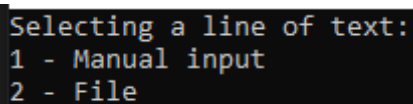
Таблица 2 – Варианты задания.

1	Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой.
2	Вывести на экран количество символов в каждом слове исходной последовательности.
3	Вывести на экран ту же последовательность, удалив из всех слов заданный набор букв и (или) цифр.
4	Вывести на экран ту же последовательность, заменив во всех словах цифры на буквы латинского алфавита, номера которых в алфавите равны заменяемой цифре.
5	Вывести на экран ту же последовательность, переместив все цифры, содержащиеся в словах, в конец соответствующих слов.
6	Вывести все слова исходной последовательности на экран вертикально.
7	Вывести на экран все слова последовательности в две или три колонки (в зависимости от количества слов) с выравниванием слов по правой границе колонки.

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 – Байера-Мура. (*)

Выполнение работы.

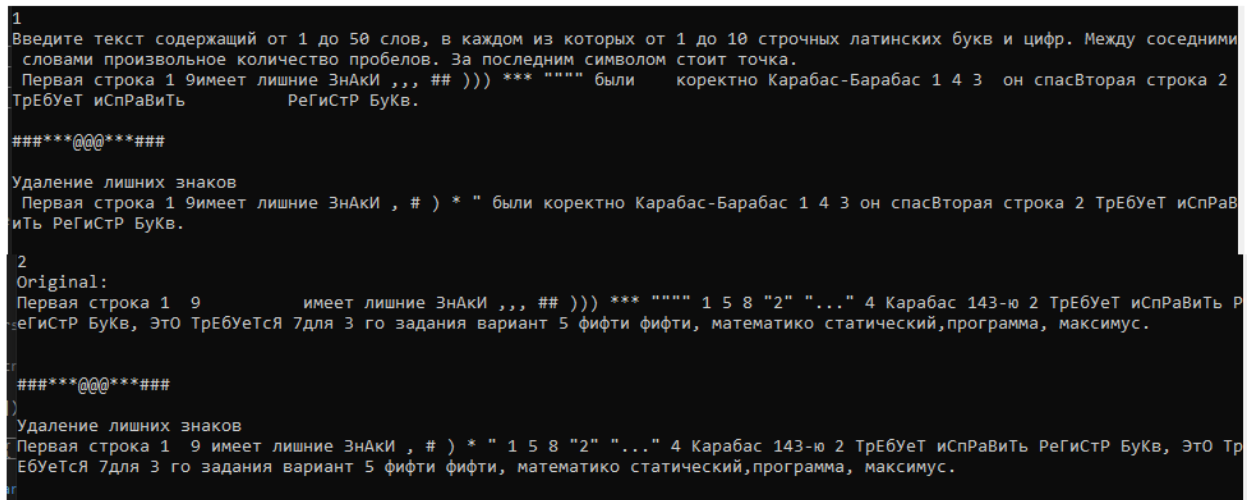
Выполняем первое задание в консоль выводится сообщение рис.1 с выбором ввода данных с клавиатуры или с файла (пользователь сам может выбрать способ ввода) вводится последовательность, содержащая от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между соседними словами произвольное количество пробелов. За последним символом стоит точка.



```
Selecting a line of text:
1 - Manual input
2 - File
```

Рисунок 1 Вывод сообщения о выборе вариантов

Редактируем введенный текст удаляем пробелы пример работы программы рис.2



```
1
Введите текст содержащий 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.
Первая строка 1 9 имеет лишние ЗНАКИ , , , # # ) ) ) * * * " " " " " 1 5 8 "2" "... " 4 Карабас 143-ю 2 ТрЕбУет иСпРаВиТЬ Р
еГиСтР Букв, Это ТрЕбУетСя 7для 3 го задания вариант 5 фифти фифти, математико статический, программа, максимум.

#####

Удаление лишних знаков
Первая строка 1 9 имеет лишние ЗНАКИ , , # ) * " " 1 5 8 "2" "... " 4 Карабас 143-ю 2 ТрЕбУет иСпРаВиТЬ Р
еГиСтР Букв, Это ТрЕбУетСя 7для 3 го задания вариант 5 фифти фифти, математико статический, программа, максимум.

2
Original:
Первая строка 1 9 имеет лишние ЗНАКИ , , # ) ) ) * * * " " " " " 1 5 8 "2" "... " 4 Карабас 143-ю 2 ТрЕбУет иСпРаВиТЬ Р
еГиСтР Букв, Это ТрЕбУетСя 7для 3 го задания вариант 5 фифти фифти, математико статический, программа, максимум.

#####

Удаление лишних знаков
Первая строка 1 9 имеет лишние ЗНАКИ , , # ) * " " 1 5 8 "2" "... " 4 Карабас 143-ю 2 ТрЕбУет иСпРаВиТЬ Р
еГиСтР Букв, Это ТрЕбУетСя 7для 3 го задания вариант 5 фифти фифти, математико статический, программа, максимум.
```

Рисунок 2 Приведены 2-а примера при вводе текста в консоль и обработка текста из файла.

Удалить лишние знаки препинания (под «лишними» подразумевается несколько подряд идущих знаков (обратите внимание, что «...» - корректное использование знака) в тексте). Примеры работы программы приведены

также на рис2. Исправить регистр букв, если это требуется (пример некорректного использования регистра букв: пРиМЕр); рис 3.

```
1
Введите текст содержащий 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.
Первая строка 1 9имеет лишние Знаки ,,, ## ))) *** """" были коректно Карабас-Барабас 1 4 3 он спасВторая строка 2
ТреБУеТ иСпРаВиТЬ РеГиСтР Букв.

Исправление регистра букв
Первая строка 1 9имеет лишние Знаки , # ) * " были коректно Карабас-барабас 1 4 3 он спасвторая строка 2 ТребуеТ исправ
ить Регистр Букв.

2
Original:
Первая строка 1 9 имеет лишние Знаки ,,, ## ))) *** """" 1 5 8 "2" "... " 4 Карабас 143-ю 2 ТреБУеТ иСпРаВиТЬ Р
егИСтР Букв, Это ТреБУеТся 7для 3 го задания вариант 5 фифти фифти, математико статический,программа, максимум.
###***@@@***##

Исправление регистра букв
Первая строка 1 9 имеет лишние Знаки , # ) * " 1 5 8 "2" "... " 4 Карабас 143-ю 2 ТребуеТ исправить Регистр Букв, Это Тре
буется 7для 3 го задания вариант 5 фифти фифти, математико статический,программа, максимум.
```

Рисунок 3 Приведены 2-а примера при выводе текста в консоль и обработка текста из файла.

Выполняем 3 задание по варианту в моем случае требуется: вывести на экран только те слова последовательности, в которых первая буква слова встречается в этом слове еще раз. Рис 4.

```
1
Введите текст содержащий 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.
Первая строка 1 9имеет лишние Знаки ,,, ## ))) *** """" были коректно Карабас-Барабас 1 4 3 он спасВторая строка 2 Тре
БУеТ иСпРаВиТЬ РеГиСтР Букв.
###***@@@***##

Задание 3 вариант 5
коректно
барабас
спасвторая
исправить
2
Original:
Первая строка 1 9 имеет лишние Знаки ,,, ## ))) *** """" 1 5 8 "2" "... " 4 Карабас 143-ю 2 ТреБУеТ иСпРаВиТЬ Р
егИСтР Букв, Это ТреБУеТся 7для 3 го задания вариант 5 фифти фифти, математико статический,программа, максимум.

Задание 3 вариант 5
"2"
исправить
фифти
фифти
математико
статический
максимум
```

Рисунок 4 Приведены 2-а примера при выводе текста в консоль и обработка текста из файла.

Выполнение задания 4 вариант 5: вывести на экран ту же последовательность, переместив все цифры, содержащиеся в словах, в конец соответствующих слов. Рис 5.

```
1
Введите текст содержащий 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.
Первая строка 1 9имеет лишние Знаки ,,, ## ))) *** """" были коректно Карабас-Барабас 1 4 3 он спасВторая строка 2 Тре
бУеТ иСпРаВиТь РеГиСтР БукВ.
###***@@@***###
Задание 4 вариант 4
Первая строка а i имеет лишние Знаки , # ) * " а е h "b" """" d Карабас adс-ю b Требуется исправить Регистр Букв, Это Тр
ебуется гдля с го задания вариант е фифти фифти, математико статический,программа, максимум.

GAME OVER
Original:
Первая строка 1 9 имеет лишние Знаки ,,, ## ))) *** """" 1 5 8 "2" """" 4 Карабас 143-ю 2 ТребУеТ иСпРаВиТь Р
еГиСтР БукВ, Это ТребУеТся 7для 3 го задания вариант 5 фифти фифти, математико статический,программа, максимум.

Задание 4 вариант 4
Первая строка а i имеет лишние Знаки , # ) * " а е h "b" """" d Карабас adс-ю b Требуется исправить Регистр Букв, Это Тр
ебуется гдля с го задания вариант е фифти фифти, математико статический,программа, максимум.

GAME OVER
```

Рисунок 5 Приведены 2-а примера при выводе текста в консоль и обработка текста из файла.

Выводы.

Были освоены новые приемы работы со строками, файлами и классом string, изучены незнакомые способы работы с текстом.

ПРИЛОЖЕНИЕ А

ПОЛНЫЙ КОД ПРОГРАММЫ

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
#include <vector>

using namespace std;

void Removing(char *strs) {
    int t = 0;
    string str = string(strs);

    for (int i = 0; i < size(str); i++) { // удаление повторных символов
        t = 0;
        for (int j = 0; j < size(str); j++) {
```

```

        if (!isalpha((unsigned char)str[j])) {
            if (str[j] == str[j + 1]) {
                if (str[j] == '.') {
                    t++;
                    if (t >= 3)
                        str.erase(j, 1);
                }
                else
                    str.erase(j, 1);
            }
        }
    }
    strcpy(strs, str.c_str());

    cout << str << endl << endl;
    setlocale(0, "RU");
}

void Register(char * str) {
    for (int i = 2; i < strlen(str); i++) // выравниваем регистр
    {
        str[0] = toupper(str[0]);

        if (str[i - 2] == '.' || str[i - 1] == '.' || str[i - 2] == '?' ||
str[i - 1] == '?') {
            str[i] = toupper(str[i]);
        }
        else {
            if (str[i - 1] != ' ')
                str[i] = tolower(str[i]);
            else
                str[i] = str[i];
        }
    }
    cout << str << endl << endl;
    setlocale(0, "RU");
}

void Task(char pch[500]) { // знаю криво получилось только
    так прошу при оценке написать свой прмер
    char *pchh = new char(500);
    pchh = strtok(pch, " ,.-"); // самая большая проблема в
    этой строке
    while (pchh != NULL) // пока есть лексемы
    {
        string str = string(pchh);
        for (int i = 1; i < size(str); i++) {
            if (str[0] == str[i])
                cout << pchh << "\n";
        }
        pchh = strtok(NULL, " ,.-)*");
    }
    cout << endl;
    setlocale(0, "RU");
}

```

```

}

void Number(char *ku)
{
    string kuk = string(ku);

    for (int i = 0; i < size(kuk); i++) {
        replace(kuk.begin(), kuk.end(), '1', 'a');
        replace(kuk.begin(), kuk.end(), '2', 'b');
        replace(kuk.begin(), kuk.end(), '3', 'c');
        replace(kuk.begin(), kuk.end(), '4', 'd');
        replace(kuk.begin(), kuk.end(), '5', 'e');
        replace(kuk.begin(), kuk.end(), '6', 'f');
        replace(kuk.begin(), kuk.end(), '7', 'g');
        replace(kuk.begin(), kuk.end(), '8', 'h');
        replace(kuk.begin(), kuk.end(), '9', 'i');
        replace(kuk.begin(), kuk.end(), '0', 'z');
    }
    cout << kuk << endl << endl;
    setlocale(0, "RU");
}

bool ReadFile(const char *FileName)
{
    char yes[500];
    char copy[500];
    char *S;

    ifstream File;
    File.open(FileName);

    if (!File.is_open()) // Проверили удалось ли открыть файл
    {
        cout << "Открыть файл не удалось! \n";
        return 0;
    }

    while (!File.eof()) // Читаем все строки из файла и выводим их на экран
        File.getline(yes, 500);
    S = yes;
    cout << "Original:\n" << S << endl << endl;
    cout << "\n###***@@@***###\n";

    cout << "\nУдаление лишних знаков\n";
    Removing(S); // удаление повторных символов
    cout << "\n###***@@@***###\n";

    cout << "\nИсправление регистра букв\n";
    Register(S); // выравниваем регистр
    cout << "\n###***@@@***###\n";

    cout << "\nЗадание 3 вариант 5\n";
    strcpy(copy, S);
    Task(copy); // выводим слова, в которых есть буквы такие же как и первая
буква слова
    cout << "\n###***@@@***###\n";
}

```

```

        cout << "\nЗадание 4 вариант 4\n";
        Number(S);
        cout << "\n GAME OVER\n";

        File.close(); // Закрываем файл
        return 1;
    }

    void WriteText() {
        cin.get();

        char yes[500];
        char copy[500];
        char *S;

        cout << "Введите текст содержащий 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. \n";
        cin.getline(yes, 500);

        S = yes;
        cout << "\n###***@@@***###\n";

        cout << "\nУдаление лишних знаков\n";
        setlocale(0, ".866");
        Removing(S); // удаление повторных символов
        cout << "\n###***@@@***###\n";

        cout << "\nИсправление регистра букв\n";
        setlocale(0, ".866");
        Register(S); // выравниваем регистр
        cout << "\n###***@@@***###\n";

        cout << "\nЗадание 3 вариант 5\n";
        strcpy(copy, S);
        setlocale(0, ".866");
        Task(copy); // выводим слова, в которых есть буквы такие же как и первая
буква слова
        cout << "\n###***@@@***###\n";

        cout << "\nЗадание 4 вариант 4\n";
        setlocale(0, ".866");
        Number(S);
        cout << "\n GAME OVER\n";
    }

    int main()
    {
        setlocale(0, "RU");

        int vibor;
        cout << "Selecting a line of text:\n1 - Manual input\n2 - File\n";
        cin >> vibor;

        if (vibor == 1) {

```

```
        WriteText();
    }
    else if (vibor == 2) {
        ReadFile("text.txt");
    }

    system("pause");

    return 0;
}
```