**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР.**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

Тема: «Работа со строками с использованием структур».

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1309 |  | Ищенко Д.О. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2022

Оглавление

[Формулировка задания. 3](#_Toc95922567)

[Контрольный пример. 3](#_Toc95922568)

[Способ внутреннего хранения данных. 3](#_Toc95922569)

[Формат входных данных. 3](#_Toc95922570)

[Способ реализации ввода вывода. 4](#_Toc95922571)

[Функции в программе. 4](#_Toc95922572)

[Текст программы. 5](#_Toc95922573)

[Результаты работы программы. 9](#_Toc95922574)

[Вывод. 10](#_Toc95922575)

## Формулировка задания.

Определить, имеется ли в заданной строке подстрока, содержащая заданное число повторяющихся заданных символов.Реализовать два способа работы со строками: с использованием маркера конца строки и с использованием длины строки.

## Контрольный пример.

N – количество повторяющихся символов, символ ‘a’.

1. Строка «аааабба» n = 3. Ответ: содержит, например «**ааа**абба»
2. Строка «аааабба» n = 5. Ответ: не содержит.

## Способ внутреннего хранения данных.

Версия 1:

Структура SuperString содержит два поля: char c[N] и char Marker. Массив символов с содержит строку + маркер в последнем символе. Его размерность задается глобальной константой N. Поле Marker содержит символ маркера. Структура так же содержит конструктор и два метода: find\_repeat\_char и out\_result для поиска повтора в сроке и вывода результата в файл. Их подробности реализации см. в разделе «функции в программе».

Версия 2:

Класс StrL содержит два поля: массив символов s[N] и поле длины строки Lenght типа short. Также определены геттеры и сеттеры для длины строки и для самой строки, метод вывода информации в файл и метод для поиска вхождений символа в строку. Их подробности реализации см. в разделе «функции в программе».

## Формат входных данных.

В программе реализованы две версии считывания данных. Глобальная константа INPUT\_TYPE может принимать значения 1 и 2. Остальные части программы согласуясь с этой константой работают со входными данными по первому или второму типу.

Считывание данных для каждого типа берется из соответствующего фала input1.txt или input2.txt. Каждый файл содержит некоторое, заранее не заданное количество строк, каждая из которых задает отдельную строку, маркер, искомый символ и количество его повторений. Относительно корневой папки проекта (содержащей файл .sln) файлы расположены по пути «.\lab1\».

Для первой версии программы:

1. тип входных данных с ограничителем.

Пример входных данных с ограничителем:

&\*a3 aaa\*a

Первый символ является маркером, используемым в программе, второй – символом, до которого нужно считать строку (не считая сам этот символ. Следующий символ считается искомым, далее считывается число повтора искомого символа (в примере – 3). Далее через пробел/пробелы считывается сама строка до ранее обозначенного символа или до конца строки.

1. Тип входных файлов с длиной

Пример с длиной:

&2a 10 sssdddsaaaaaaaa

В этом случае считывается первый символ как маркер, далее количество искомых символов и сам символ. Через пробел/пробелы считывается длина строки и через пробел/пробелы считывается сама строка по заданной ранее длине или до символа конца строки (\0).

Программа обрабатывает такие ситуации, как встреча конца файла, выход размера строки за максимальное количество букв в строке, программа отслеживает вариант считывания пустой строки.

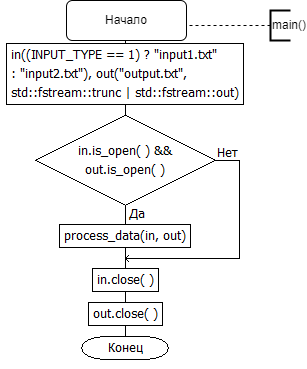
Для второй версии программы, реализующей работу со строкой с помощью отслеживания длины строки входные файлы незначительно изменяются. А именно: исчезает символ маркера (& в примере).

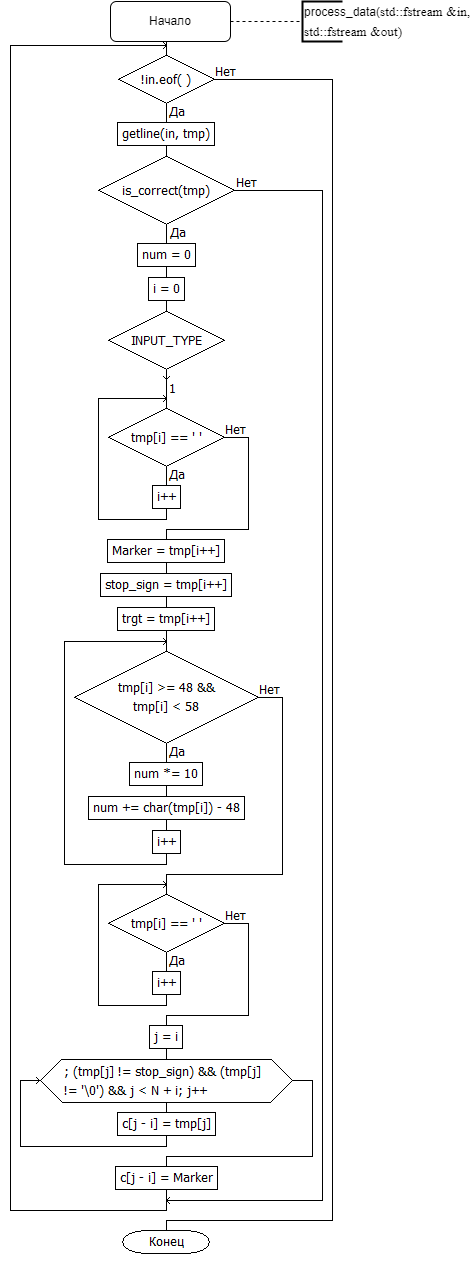
## Способ реализации ввода вывода.

Для ввода-вывода при реализации задания были задействованы класс fstream для работы с файлами, метода open() и close() для открытия и закрытия файлов соответственно. Метод is\_open() для проверки программы на предмет ошибки открытия файла и константные члены trunc и out для очищения выходного файла перед записью данных.

## Функции в программе.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя функции | Назначение | Параметры | | | | Внешние эффекты |
| входные | выходные | модифицируемые | транзитные |
| main | Открывает и закрывает файлы, вызывает функцию process\_data | - | - | - | - | Изменяется файл output.txt |
| find\_repeat\_char | Определяет содержит для строка типа SuperString (StrL) num символов ch | int num, char ch | bool | - | - | - |
| out\_result | Выводит в выходной файл out в соответсвии со значением find\_repeat\_char информацию о нахождении или ненахождении num количества символа trgt в строке SuperString (StrL). | std::fstream& out, bool is\_contain, int num, char trgt | void | out | - | Изменяется файл output.txt |
| is\_correct | Проверяет строку tmp на корректность | std::string tmp | bool | - | - | - |
| process\_data | Обрабатывает входные данные из файла in, для этого для каждой строки вызывает функцию is\_correct(), создает на основе считанных данных переменную типа SuperString (или StrL), вызывает её методы find\_repeat\_char и out\_result в случае корректного ввода и выводит сообщения о некоректности строки в случае некорректного. | std::fstream &in, std::fstream &out | void | out | std::fstream &out | Изменяется файл output.txt |





## Текст программы.

## Версия 1.

#include <iostream>

#include <string>

#include <fstream>

const int N = 255;

short INPUT\_TYPE = 2;

struct SuperString {

char c[N];

char Marker;

SuperString(char\* c, char m) {

this->Marker = m;

for (int i = 0; i < N; i++) {

this->c[i] = c[i];

}

}

bool find\_repeat\_char(int num, char ch);

void out\_result(std::fstream &out, bool is\_contain, int num, char trgt);

};

bool SuperString::find\_repeat\_char(int num, char ch) {

unsigned count = 0, i = 0;

while (c[i] != Marker) {

if (c[i] == ch) {

++count;

if (count >= num)

return true;

}

else {

count = 0;

}

i++;

}

return false;

}

void SuperString::out\_result(std::fstream& out, bool is\_contain, int num, char trgt) {

out << "\'";

for (int i = 0; c[i] != Marker; i++) {

out << c[i];

}

out << '\'' << (is\_contain ? " contains \'" : " does not contain \'")

<< trgt << "\' " << std::to\_string(num) << " times. \n";

}

bool is\_correct(std::string tmp) {

int i = 0;

if (tmp.empty() || tmp[i] == '\0') {

return false;

}

switch (INPUT\_TYPE) {

{

case 1: {

while (tmp[i] == ' ') {

i++;

}

for (int j = 0; j < 3; j++) {

if (tmp[i] == ' ' || tmp[i] == '\0') {

return false;

}

i++;

}

if (tmp[i] < 48 || tmp[i] >= 58 || tmp[i] == '\0') {

return false;

}

i++;

while (tmp[i] < 48 && tmp[i] >= 58) {

i++;

}

break;

}

case 2: {

while (tmp[i] == ' ') {

i++;

}

i++;

if (tmp[i] < 48 || tmp[i] >= 58 || tmp[i] == '\0') {

return false;

}

i++;

while (tmp[i] < 58 && tmp[i] >= 48) {

i++;

}

i++;

if (tmp[i] != ' ' || tmp[i] == '\0') {

return false;

}

while (tmp[i] == ' ' ) {

i++;

}

if (tmp[i] < 48 || tmp[i] >= 58 || tmp[i] == '\0') {

return false;

}

i++;

while (tmp[i] < 58 && tmp[i] >= 48) {

i++;

}

if (tmp[i] != ' ' || tmp[i] == '\0') {

return false;

}

break; }

}

}

return true;

}

void process\_data(std::fstream &in, std::fstream &out) {

// считывает построчно данные отправляет на проверку в is\_correct()

// если это так, то вызывает метод find\_repeat\_char и результат отправляет в

// out\_result

std::string tmp;

while (!in.eof()) {

getline(in, tmp);

if (is\_correct(tmp)) {

char Marker;

char c[N];

int num = 0;

char trgt;

int i = 0;

switch (INPUT\_TYPE)

{

case 1: {

while (tmp[i] == ' ') {

i++;

}

Marker = tmp[i++];

char stop\_sign = tmp[i++];

trgt = tmp[i++];

while (tmp[i] >= 48 && tmp[i] < 58) {

num \*= 10;

num += char(tmp[i]) - 48;

i++;

}

while (tmp[i] == ' ') {

i++;

}

int j = i;

for (; (tmp[j] != stop\_sign) && (tmp[j] != '\0') && j < N + i; j++) {

c[j - i] = tmp[j];

}

c[j-i] = Marker;

break;}

case 2: {

int len = 0;

while (tmp[i] == ' ') {

i++;

}

Marker = tmp[i++];

while (tmp[i] >= 48 && tmp[i] < 58) {

num \*= 10;

num += char(tmp[i]) - 48;

i++;

}

trgt = tmp[i++];

while (tmp[i] == ' ') {

i++;

}

while (tmp[i] >= 48 && tmp[i] < 58) {

len \*= 10;

len += char(tmp[i]) - 48;

i++;

}

while (tmp[i] == ' ') {

i++;

}

if (len < 0) {

len = -len;

}

int j = i;

for (; (j-i <len) && (tmp[j] != '\0') && j < N + i; j++) {

c[j - i] = tmp[j];

}

c[j - i] = Marker;

break;}

}

SuperString line (c, Marker);

line.out\_result(out, line.find\_repeat\_char(num, trgt), num, trgt);

}

else {

out << "string isn't correct\n";

}

}

}

int main() {

std::fstream in((INPUT\_TYPE == 1)? "input1.txt": "input2.txt"),

out("output.txt", std::fstream::trunc | std::fstream::out);

if (in.is\_open() && out.is\_open()) {

process\_data(in, out);

}

else {

std::cout << "files not open";

}

in.close();

}

## Версия 2.

#include <iostream>

#include <fstream>

#include <string>

const int N = 255;

short INPUT\_TYPE = 2;

class StrL {

char s[N];

short Lenght = 0;

public:

StrL(){}

StrL(short L, char\*s): Lenght(L) {

for (int i = 0; i < N; i++) {

this->s[i] = s[i];

}

}

short getLen() {

if (Lenght) {

return Lenght;

}

return -1;

}

void setLen(short L) {

if (L > 0) {

Lenght = L;

}

}

void setString(char \*s) {

for (int i = 0; i < Lenght; i++) {

this->s[i] = s[i];

}

}

char\* getString() {

return s;

}

bool find\_repeat\_char(char target, short N) {

if (N > Lenght)

return false;

short count = 0;

for (int i = 0; i < Lenght; i++) {

if (s[i] == target) {

count++;

if (count == N)

return true;

}

else {

count = 0;

}

}

return false;

}

void out\_result(std::fstream& out, char t, short N) {

for (int i = 0; i < Lenght; i++) {

out << s[i];

}

if (this->find\_repeat\_char(t, N)) {

out << " contains " << t << " " << N << " times.\n";

}

else {

out << " not contains " << t << " " << N << " times.\n";

}

}

};

bool is\_correct(std::string tmp) {

int i = 0;

if (tmp.empty() || tmp[i] == '\0') {

return false;

}

switch (INPUT\_TYPE) {

{

case 1: {

while (tmp[i] == ' ') {

i++;

}

for (int j = 0; j < 2; j++) {

if (tmp[i] == ' ' || tmp[i] == '\0') {

return false;

}

i++;

}

if (tmp[i] < 48 || tmp[i] >= 58 || tmp[i] == '\0') {

return false;

}

i++;

while (tmp[i] < 48 && tmp[i] >= 58) {

i++;

}

break;

}

case 2: {

while (tmp[i] == ' ') {

i++;

}

i++;

while (tmp[i] < 58 && tmp[i] >= 48) {

i++;

}

i++;

if (tmp[i] != ' ' || tmp[i] == '\0') {

return false;

}

while (tmp[i] == ' ') {

i++;

}

if (tmp[i] < 48 || tmp[i] >= 58 || tmp[i] == '\0') {

return false;

}

i++;

while (tmp[i] < 58 && tmp[i] >= 48) {

i++;

}

if (tmp[i] != ' ' || tmp[i] == '\0') {

return false;

}

break; }

}

}

return true;

}

void process\_data(std::fstream& in, std::fstream& out) {

std::string tmp;

while (!in.eof()) {

getline(in, tmp);

if (is\_correct(tmp)) {

short L;

char c[N];

int num = 0;

char trgt;

int i = 0;

switch (INPUT\_TYPE)

{

case 1: {

while (tmp[i] == ' ') {

i++;

}

char stop\_sign = tmp[i++];

trgt = tmp[i++];

while (tmp[i] >= 48 && tmp[i] < 58) {

num \*= 10;

num += char(tmp[i]) - 48;

i++;

}

while (tmp[i] == ' ') {

i++;

}

int j = i;

for (; (tmp[j] != stop\_sign) && (tmp[j] != '\0') && j < N + i; j++) {

c[j - i] = tmp[j];

}

L = j - i;

break; }

case 2: {

int len = 0;

while (tmp[i] == ' ') {

i++;

}

while (tmp[i] >= 48 && tmp[i] < 58) {

num \*= 10;

num += char(tmp[i]) - 48;

i++;

}

trgt = tmp[i++];

while (tmp[i] == ' ') {

i++;

}

while (tmp[i] >= 48 && tmp[i] < 58) {

len \*= 10;

len += char(tmp[i]) - 48;

i++;

}

while (tmp[i] == ' ') {

i++;

}

if (len < 0) {

len = -len;

}

int j = i;

for (; (j - i < len) && (tmp[j] != '\0') && j < N + i; j++) {

c[j - i] = tmp[j];

}

if (j - i < len) {

L = j - i;

}

else {

L = len;

}

break; }

}

StrL line(L,c);

line.out\_result(out, trgt, num);

}

else {

out << "string isn't correct\n";

}

}

}

int main()

{

std::fstream in((INPUT\_TYPE == 1) ? "input1.txt" : "input2.txt"),

out("output.txt", std::fstream::trunc | std::fstream::out);

if (in.is\_open() && out.is\_open()) {

process\_data(in, out);

}

else {

std::cout << "files not open";

}

in.close(); out.close();

}

## Результаты работы программы.

Версия 1:

**Input1.txt:**

bb

&\*b as\*

&\*b0 bbb

&\*a3 aaa\*a

&\*a3 bbb\*a

Версия 2:

bb

\*b as\*

\*b0 bbb

\*a3 aaa\*a

\*a3 bbb\*a

**Out:**

string isn't correct

string isn't correct

string isn't correct

'bbb' contains 'b' 0 times.

'aaa' contains 'a' 3 times.

'bbb' does not contain 'a' 3 times.

**Input2.txt:**

Версия1:

&2a 10 sssdddsssaaa

&2a 1 asssdddsaaaaaaaa

&2a 10 sssaa

Версия 2:

2a 10 sssdddsssaaa

2a 1 asssdddsaaaaaaaa

2a 10 sssaa

**Out:**

'sssdddsssa' does not contain 'a' 2 times.

string isn't correct

'a' does not contain 'a' 2 times.

string isn't correct

'sssaa' contains 'a' 2 times.

## Вывод.

В ходе выполнения лабораторной работы была написана программа, решающая поставленную задачу. Во время её разработки были получены практические навыки работы с различными конструкциями языка с++, такими как структуры, файлы. А также был получен навык реализации пользовательского типа string с использованием маркера конца строки.