**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**"Национальный исследовательский университет**

**"Высшая школа экономики"**

Московский институт электроники и математики им. А.Н. Тихонова НИУ ВШЭ

Департамент компьютерной инженерии

**Курс: Алгоритмизация и программирование**

|  |  |  |
| --- | --- | --- |
| **Раздел** | **Max оценка** | **Итог. Оценка** |
| Работа программы | 1 |  |
| Тесты | 1 |  |
| Правильность алгоритма | 3 |  |
| Ответы на вопросы | 2 |  |
| Дополнительное задание | 3 |  |

**ОТЧЕТ**

**по лабораторной работе № 9**

**Студент:\_ Иванилов Леонид Михайлович**

**Группа:\_БИВ231**

**Вариант:\_143 (8, 9, 4)**

**Руководитель:\_Ерохина Елена Альфредовна**

**Оценка:\_\_\_\_\_\_\_\_\_\_**

**Дата сдачи\_\_\_\_\_\_\_\_\_\_**

**Задания:**

**I.** Выделить из каждой строки и напечатать подстроки:

(8) Содержащие только букву 'Е', цифры, точки, знаки «плюс» и «минус».

**II.** Среди выделенных подстрок найти подстроку:

(9) Состоящую только из латинских букв и цифр.

**III.** Преобразовать исходную строку, которой принадлежит найденная подстрока, следующим образом:

(4) Удалить повторяющиеся пробелы (оставить по одному пробелу).

**МОСКВА 2024**

**Листинг программы**

#include <iostream>  
#include <string>  
#include <cctype>  
#define L 50  
  
using namespace std;  
  
int e\_numbers\_dots(int n, string strs[], string substrs[], int indices[]);  
int findAlphanumericString(int n, string substrs[]);  
int only\_one\_space(string &s);  
  
int main() {  
 int i, n, m, num;  
 do {  
 cout << "Enter number of strings: ";  
 cin >> n;  
 if (cin.fail())  
 cin.clear();  
 cin.ignore();  
 } while (n <= 0 || n >= L);  
 int indices[L];  
 string strs[L], substrs[5 \* L];  
 cout << "Enter " << n << " strings: " << endl;  
 for (i = 0; i < n; ++i)  
 getline(cin, strs[i]);  
 m = e\_numbers\_dots(n, strs, substrs, indices);  
 if (!m) cout << "No substrings. " << endl;  
 else {  
 cout << "Substrings: " << endl;  
 for (i = 0; i < m; ++i)  
 cout << substrs[i] << endl;  
 num = findAlphanumericString(m, substrs);  
 if (num == -1)  
 cout << "No necessary substring. " << endl;  
 else {  
 cout << "Necessary substring is: " << substrs[num] << endl;  
 if (!only\_one\_space(strs[indices[num]]))  
 cout << "No changes. " << endl;  
 else  
 cout << "Changed string: " << strs[indices[num]] << endl;  
 }  
 }  
 return 0;  
}  
  
bool isValidChar(char ch) {  
 return (ch == 'e' || ch == 'E' || isdigit(ch) || ch == '.' || ch == '+' || ch == '-');  
}  
  
  
int e\_numbers\_dots(int n, string strs[], string substrs[], int indices[]) {  
 int numValidSubstrings = 0;  
  
 for (int i = 0; i < n; ++i) {  
 string currentSubstr;  
 bool insideValidCharSequence = false;  
  
 for (int j = 0; j < strs[i].length(); ++j) {  
 if (isValidChar(strs[i][j])) {  
 currentSubstr += strs[i][j];  
 insideValidCharSequence = true;  
 } else {  
 if (insideValidCharSequence) {  
 substrs[numValidSubstrings] = currentSubstr;  
 indices[numValidSubstrings] = i;  
 numValidSubstrings++;  
 currentSubstr.clear();  
 insideValidCharSequence = false;  
 }  
 }  
 }  
  
 if (insideValidCharSequence) {  
 substrs[numValidSubstrings] = currentSubstr;  
 indices[numValidSubstrings] = i;  
 numValidSubstrings++;  
 }  
 }  
  
 return numValidSubstrings;  
}  
  
  
int findAlphanumericString(int n, string substrs[]) {  
 for (int i = 0; i < n; ++i) {  
 bool isAlphanumeric = true;  
 for (int j = 0; j < substrs[i].length() && isAlphanumeric; ++j) {  
 if (!isalnum(substrs[i][j])) {  
 isAlphanumeric = false;  
 }  
 }  
 if (isAlphanumeric) {  
 return i;  
 }  
 }  
 return -1;  
}  
  
int only\_one\_space(string &s) {  
 int n = s.length();  
 if (n <= 1) return false;  
  
 int j = 0;  
  
 bool hasSpace = false;  
  
 for (int i = 0; i < n; ++i) {  
 if (s[i] != ' ' || (i > 0 && s[i] == ' ' && s[i - 1] != ' ')) {  
 if (s[i] == ' ') {  
 hasSpace = true;  
 }  
 if (i != j) {  
 s[j] = s[i];  
 }  
 ++j;  
 }  
 }  
  
 if (hasSpace && s[j - 1] == ' ') {  
 s.erase(j - 1);  
 return 1;  
 }  
  
 s.erase(j);  
  
 return (j != n);  
}

**Тесты**

|  |  |
| --- | --- |
| **ВВОД** | **ВЫВОД** |
| n = 2  strings:  ;;;;; ;;; ;;; ;  123 123 13 | Substrings:  123  123  13  Necessary substring is: 123  Changed string: 123 123 13 |
| n = 1  strings:  ;;; ;;; ;;;qqqq opuwlkjfhs lkjfs | No substrings. |
| n = 2  strings:  123 23ee 128736 123123  sad s d a s d asd | Substrings:  123  23ee  128736  123123  Necessary substring is: 123  No changes. |
|  |  |
|  |  |
|  |  |

**Дополнительное Задание**

#include <iostream>  
#include <string>  
#include <algorithm>  
  
using namespace std;  
  
  
bool isDigit(char c) {  
 return c >= '0' && c <= '9';  
}  
  
  
  
void extractDigitSubstrings(const string& str, string digitSubstrings[], int& count) {  
 count = 0;  
 string digitSubstring;  
  
 for (char c : str) {  
 if (isDigit(c)) {  
 digitSubstring += c;  
 } else {  
 if (!digitSubstring.empty()) {  
 digitSubstrings[count++] = digitSubstring;  
 digitSubstring.clear();  
 }  
 }  
 }  
  
 if (!digitSubstring.empty()) {  
 digitSubstrings[count++] = digitSubstring;  
 }  
  
 for (int i = 0; i < count - 1; ++i) {  
 for (int j = 0; j < count - i - 1; ++j) {  
 if (digitSubstrings[j].size() > digitSubstrings[j + 1].size()) {  
 swap(digitSubstrings[j], digitSubstrings[j + 1]);  
 }  
 }  
 }  
}  
  
  
int replaceDigitCombination(string& str, const string digitSubstrings[], int count) {  
 int digitIndex = 0;  
 int flag = 0;  
 for (size\_t i = 0; i < str.length(); ++i) {  
 if (isdigit(str[i])) {  
 size\_t j = i;  
 while (j < str.length() && isdigit(str[j])) {  
 ++j;  
 }  
 if (digitSubstrings[digitIndex] != str.substr(i,j - i)){  
 str.replace(i, j - i, digitSubstrings[digitIndex]);  
 flag = 1;  
  
 }  
  
 ++digitIndex;  
  
 i = i + digitSubstrings[digitIndex-1].length();  
 }  
 }  
 return flag;  
}  
  
int main() {  
 string inputString = "abc321def456zt1bc12";  
// string inputString = "abc1def12zt321bc456";  
 cout << "Исходная строка: " << inputString << endl;  
  
 string digitSubstrings[100];  
 int count;  
 extractDigitSubstrings(inputString, digitSubstrings, count);  
  
 cout << "Подстроки из цифр (отсортированные по длине): ";  
 for (int i = 0; i < count; ++i) {  
 if(i == 1){  
 cout << digitSubstrings[i] << " ";  
 }  
 else {  
 cout << digitSubstrings[i] << " ";  
 }  
 }  
 cout << endl;  
  
 int flag = replaceDigitCombination(inputString, digitSubstrings, count);  
 if(flag){  
 cout << "Строка после замены комбинации цифр на числа из массива по порядку: " << inputString << endl;}  
 else{  
 cout << "ничего не поменялось";  
 }  
  
 return 0;  
}

**тесты**

|  |  |
| --- | --- |
| **ВВОД** | **ВЫВОД** |
| **abc321def456zt1bc12** | **Строка после замены комбинации цифр на числа из массива по порядку: abc1def12zt321bc456** |
| **abc1def12zt321bc456** | **ничего не поменялось** |