**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: Матрицы и указатели.

|  |  |  |
| --- | --- | --- |
| Студент гр. 3372 |  | Беляев К. В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Научиться работать с матрицами и указателями

Приложение А

рабочий код

#include <iostream>  
#include <fstream>  
  
**using namespace** std;  
**int** line\_down\_times;  
**int const** wordsLen = 10, startWords = 50;  
**char** symbols\_arr[] = {'!', ',', ' ',  
 '?', ':', ';',  
 '-', };  
  
**char** first\_10\_of\_Latin[] = {'A', 'B', 'C', 'D', 'E',  
 'F', 'G', 'H', 'I', 'J'};  
  
  
  
**void** clearStream(){  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
}  
  
  
  
**void** readFile(**char** filename[], **char** string[]){  
 ofstream fout;  
 ifstream fin;  
 **char** c;  
 **int** n = 0;  
 fin.open(filename);  
 **while** (fin.get(c)){  
 string[n] = c;  
 ++n;  
 }  
 string[n] = '\0';  
}  
  
  
**inline void** lineDown(**int** times=1){  
 **for**(line\_down\_times = 0; line\_down\_times < times; line\_down\_times++){  
 cout << "\n";  
 }  
}  
  
  
**void** printMenu(){  
 cout << "1. Ввести последовательность символов";  
 lineDown();  
 cout << "2. Отредактировать входной текст";  
 lineDown();  
 cout << "3. Ввести последовательность символов";  
 lineDown();  
 cout << "4. Ввести последовательность символов";  
 lineDown();  
 cout << "5. найти все подстроки";  
 lineDown();  
 cout << "0. Выход";  
 lineDown(2);  
  
}  
//----------------------------------------------------------------------------------------------------------------------  
//string methods  
  
**int** count(**char** string[], **char** symbol){  
 **int** count = 0;  
 **for** ( **int** i = 0; i < strlen(string); i++ ){  
 **if** ( string[i] == symbol ) {  
 count ++;  
 }  
 }  
  
 **return** count;  
}  
  
  
**int** find(**char** string[], **char** symbol, **int** start=0){  
 **for** (**int** index = start; index < strlen(string); index++){  
 **if** (string[index] == symbol){  
 **return** index;  
 }  
 }  
 **return** -1;  
}  
  
  
**int** split(**char** string[], **char** splittedString[startWords][wordsLen], **char** separator=' '){  
 **int** words = count(string, separator) + 1;  
 **int** start, end = -1;  
 **for** ( **int** i = 0; i < words; i++ ){  
 start = end + 1;  
 end = find(string, separator, start);  
 **if** (end == -1){  
 end = strlen(string);  
 }  
 **for** ( **int** j = start, index = 0; j < end; j++, index++ ){  
 splittedString[i][index] = string[j];  
 }  
  
 }  
 **return** words;  
  
}  
  
  
**bool** isPunct(**char** string[]){  
  
 **for** ( **int** i = 0; i < strlen(symbols\_arr); i++ ){  
 **if** (string[0] == symbols\_arr[i]){  
 **return true**;  
 }  
 }  
 **return false**;  
}  
  
  
//реализовать join  
**void** deleteSymbols(**char** string[], **char** symbol){  
 **for** (**int** i = 0; i < strlen(string) - 1; i++){  
 **if** (string[i] == symbol && string[i + 1] == symbol){  
  
 **for** (**int** j = i + 1; j < strlen(string); j ++){  
 string[j - 1] = string[j];  
 }  
 string[strlen(string) - 1] = '\0';  
 i--;  
 }  
 }  
}  
  
  
**int** findSubstringsAmount(**char** string[], **char** substring[]) {  
 **bool** flag;  
 **int** counter = 0;  
 **int** strLen = strlen(string), substrLen = strlen(substring);  
 **for** ( **int** i = 0; i <= strLen - substrLen; i++ ){  
 flag = **true**;  
 **for** ( **int** j = 0; j < substrLen; j++ ){  
 **if** (substring[j] != string[i + j]){  
 flag = **false**;  
 **break**;  
 }  
 }  
 **if** (flag){  
 counter++;  
 }  
 }  
 **return** counter;  
}  
  
  
**void** changeRegistr(**char** string[]){  
  
 **for** ( **int** i = 1; i < strlen(string); i++ ){  
 string[i] = tolower(string[i]);  
  
 }  
 string[0] = toupper(string[0]);  
  
}  
  
//  
//int boyerMoore(char string[], char substring[]){  
// int index = 0;  
// const unsigned int strLen = strlen(string), substrLen = strlen(substring);  
// unsigned int table[substrLen + 1];  
// int f, counter = 0;  
// table[substrLen] = substrLen;  
// for (unsigned int i = substrLen - 2; i >= 0; i--){  
// counter++;  
// f = find(substring, substring[i], i);  
// if (f == -1 && f != substrLen - 1){  
// table[i] = counter;  
// }else if (f != -1) {  
// table[i] = table[f];  
// }  
// }  
//  
// while ( index <= (strLen - substrLen)){  
//  
//  
//  
// }  
//  
//}  
  
  
**int** main() {  
 **int** problem, method;  
 **char** filename[500];  
 **int const** string\_size = 500;  
 **char** string[string\_size], substring[string\_size];  
 **bool** repeaters;  
 **int** words;  
 **char** splittedString[startWords][wordsLen] = {};  
  
 printMenu();  
  
 **for** (**int** repeat = 0; repeat < 100; repeat++){  
 cout << "Введите номер задания: ";  
 cin >> problem;  
 clearStream();  
  
 **switch** (problem){  
 **case** 1:  
 cout << "Выберите способ ввода последовательности(1. Ввод, 2. Файл): ";  
 cin >> method;  
 clearStream();  
  
 **if** (method == 1){  
 cout << "Введите строку: ";  
 cin.getline(string, string\_size);  
 lineDown();  
 }**else**{  
 cout << "Введите путь к файлу: ";  
 cin.getline(filename, 500);  
 readFile(filename, string);  
 lineDown();  
 }  
  
 cout << "Введенная последовательность: ";  
 lineDown();  
 cout << string;  
 lineDown(2);  
 **break**;  
  
 **case** 2:  
 // удаление символов  
 **for** ( **int** i = 0; i < strlen(symbols\_arr); i++ ){  
 deleteSymbols(string, symbols\_arr[i]);  
 }  
 // регистр  
 changeRegistr(string);  
  
 cout << "Отформатированная строка: " << string;  
 lineDown(2);  
 **break**;  
  
 **case** 3:  
 //№4  
 words = split(string, splittedString);  
 cout << "Слова с повторояющимися буквами: ";  
 **for** (**int** word = 0; word < words; word++){  
 **if** (isPunct(splittedString[word])){  
 **continue**;  
 }  
 repeaters = **false**;  
  
 **for** ( **int** letter = 0; letter < strlen(splittedString[word]); letter++ ){  
 **if** (repeaters){  
 **break**;  
 }  
 **for** ( **int** i = letter + 1; i < strlen(splittedString[word]); i++ ){  
 **if** (tolower(splittedString[word][letter]) == tolower(splittedString[word][i])){  
 cout << splittedString[word] << ", ";  
 repeaters = **true**;  
 **break**;  
 }  
 }  
 }  
 }  
 lineDown(2);  
 **break**;  
  
 **case** 4:  
 //№4  
 cout << "Заменим все цифры на латинские буквы: ";  
 lineDown();  
 **for** ( **int** i = 0; i < strlen(string); i++ ){  
 **if** (string[i] >= '0' && string[i] <= '9'){  
 string[i] = first\_10\_of\_Latin[string[i] - '0'];  
 }  
 }  
 cout << string;  
 lineDown(2);  
 **break**;  
  
  
 **case** 5:  
  
 cout << "Введите подстроку: ";  
 cin.getline(substring, string\_size);  
 cout << "поиск кол-ва подстроки перебором: " << findSubstringsAmount(string, substring);  
 lineDown();  
 **break**;  
  
 **case** 0:  
 **return** 0;  
 **break**;  
 }  
 }  
 **return** 0;  
}