**онлайн-Платформа STEPIK**

**отчет**

**по практической работе №4**

**по курсу «Основы алгоритмизации и программирования на языке C++»**

**Тема: Работа с текстовыми строками**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. | Простаков А.В (User ID: 89907427) |  |
| Преподаватель | Глущенко А.Г. |  |

Москва

2024

**Цель работы.**

Отработка практических навыков работы с текстовыми строками и классом string.

**Постановка задачи**

Необходимо написать программу, которая реализует поставленную задачу:

1) C 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) Вывести на экран слова последовательности в обратном порядке;

4) Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой;

5) Необходимо найти все подстроки, 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 КМП, а нечетные – Бойера-Мура;

**Выполнение работы.**

Код программы в редактируемом формате представлен в приложении А.

**Выводы**

В данной практической работе рассмотрены приемы работы c текстовыми строками, в том числе с классом string.

**Приложение А**

**рабочий код**

|  |
| --- |
| #include <iostream>  #include <ostream>  #include <cstring>  #include <string>  #include <vector>  #include <cstdlib>  using namespace std;  void puncMarks(string &S);  void registr(string& S);  void eraseMark(string& S);  void reverseWord(string& S);  void replace(string& S);  int directSearch(string& S, string& substring);  vector<int> kmp(string text, string pattern);  vector<int> computeLps(string pattern);  int main()  {  setlocale(LC\_ALL, "Russian");  string S1, S;  string substring;  int choice;  S1.resize(50);  cout << "Введите последовательность, содержащую 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." << endl;  getline(cin, S);  S1 = S;  cout << "\n";  for (;;)  {  cout << "1.Удалить лишние пробелы и знаки препинания. Исправить регистр букв" << endl;  cout << "2.Вывести на экран слова последовательности в обратном порядке" << endl;  cout << "3.Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой." << endl;  cout << "4.Ввести подстроку.Найти подстроку алгоритмом прямого поиска." << endl;  cout << "5.Ввести подстроку.Найти подстроку алгоритмом Кнута-Мориса-Пратта." << endl;  cout << "6.Выход из программы." << endl;  cout << "Выберите пункт меню. И введите номер выбранного пункта = ";  cin >> choice;  switch (choice)  {  case 1:  {  cout << "\n";  cout << S<<endl;  puncMarks(S);  registr(S);  cout << S << endl;;  cout << "\n";  S = S1;  break;  }  case 2:  {  cout << "\n";  cout << S << endl;  reverseWord(S);  cout << S << endl;  cout << "\n";  S = S1;  break;  }  case 3:  {  cout << "\n";  cout << S << endl;  replace(S);  cout << S << endl;  cout << "\n";  S = S1;  break;  }  case 4:  {  cout << "\n";  cout << S << endl;  cin >> substring;  int result=directSearch(S,substring);  if (result == -1)  cout << "Подстрока не найдена" << endl;  else  cout << "Подстрока найдена в позиции: " << result << endl;  }  case 5:  {  cout << "\n";  cout << S << endl;  cin >> substring;  vector<int> occurrences = kmp(S, substring);  if (occurrences.empty()) {  cout << "Подстрока не найдена." << endl;  }  else {  cout << "Подстрока найдена в позициях: ";  for (int i : occurrences) {  cout << i << " ";  }  cout << endl;  }  return 0;  }  default:  return 0;  break;  }  }  }  void puncMarks(string& S)  {  int i, k = 0;  string mark;  string arr[] = { ",","!","?",";",":","-","(",")"," " };  int index = 0;  for (k; k < 9; k++)  {  mark = arr[k] + arr[k];  i = 0;  while (i != -1)  {  i = S.find(mark);  if (i != -1)  if (S[i + 1] == mark[0])  S.erase(i + 1, 1);  }  }  mark = "..";  while (k != -1)  {  k = S.find(mark, index);  if (k != -1)  if (S[k + 2] == mark[0] and S[k + 3] != mark[0])  {  index = k + 3;  k = -1;  }  if (k != -1)  while (S[k + 1] == mark[0])  {  S.erase(k + 1, 1);  }  }  }  void registr(string & S)  {    string gap = " ";  for (int i = 1; i < S.length(); ++i)  {  if (S[i - 1] != gap[0])  S[i]=tolower(S[i]);  }    }  void eraseMark(string& S)  {  int i,r,k = 0;  string mark;  string arr[] = { ",","!","?",";",":","-","(",")","."};  string S2;    for (k; k < 9; k++)  {  mark = arr[k];  i = 0;  while (i != -1)  {  i = S.find(mark);  if (i != -1)  S.erase(i, 1);  }  }    }  void reverseWord(string& S)  {  string S2, S3, S4;  string space;  int i;  int k = 0;  int r = 0;  int index = 0;  puncMarks(S);  eraseMark(S);  i = S.length();  for (i + 1; i > -1; --i)  {  S2 = S2 + S[i];  }  S = S2;  S2.erase();  space = " ";  while (k != -1)  {  k = S.find(space, index);  if (k != -1)  for (r=index; r < k; ++r)  {  S3 = S3 + S[r];  }    for (r = k-index; r > -1; --r)  {  S2 = S2 + S3[r];  }  index = k + 1;  S2 = S2 + " ";  S4 = S4+ S2;  S2.erase();  S3.erase();    }    S = S4;    }  void replace(string& S)  {  string gap = " ";  for (int i = 1; i < S.length(); ++i)  {  if (S[i - 1] == gap[0])  S[i] = tolower(S[i]);  }  }  int directSearch(string& S, string& substring)  {  int n = S.length(); // длина текста  int m = substring.length(); // длина подстроки  for (int i = 0; i <= n - m; i++) {  int j;  for (j = 0; j < m; j++) {  if (S[i + j] != substring[j])  break;  }  if (j == m) // если все символы совпали  return i; // вернуть индекс начала подстроки  }  return -1; // подстрока не найдена  }  vector<int> computeLps(string pattern)  {  int n = pattern.length();  vector<int> lps(n, 0);  int len = 0;  int i = 1;  while (i < n) {  if (pattern[i] == pattern[len]) {  len++;  lps[i] = len;  i++;  }  else {  if (len != 0) {  len = lps[len - 1];  }  else {  lps[i] = 0;  i++;  }  }  }  return lps;  }  vector<int> kmp(string text, string pattern)  {  vector<int> lps = computeLps(pattern);  vector<int> occurrences;  int n = text.length();  int m = pattern.length();  int i = 0;  int j = 0;  while (i < n) {  if (text[i] == pattern[j]) {  i++;  j++;  }  if (j == m) {  occurrences.push\_back(i - j);  j = lps[j - 1];  }  else if (i < n && text[i] != pattern[j]) {  if (j != 0) {  j = lps[j - 1];  }  else {  i++;  }  }  }  return occurrences;  } |