**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №4**

**по дисциплине «Программирование»**

Тема: Строки. Поиск подстроки в строке.

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. 2372 |  | Гечис В.Р. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение одномерных статических массивов; Изучение различных видов сортировок массива; Изучение алгоритма бинарного поиска и взаимодействия с элементами массива по индексам.

**Основные теоретические положения.**

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 С++(=,+,<,==,>,+=,!=,<=,>=,[])(=,+,<,==,>,+=,!=,<=,>=,[]). Использование типа char приводило требовало написание чрезмерного программного кода;

·        обеспечение лучшей надежности программного кода;

·        обеспечение строки, как самостоятельного типа данных.

/\* string.cpp: Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы \*/

#include <iostream>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

string A = "Пример"; // Объявляем и инициализируем строку А

string B; // Объявляем строку В

string C = "текста"; // Объявляем и инициализируем строку С

B = A; // Копируем текст из строки А в строку В

C = B + " " + C;

if(C == "Пример текста") // Проверяем корректно ли произошло присоединение строк

cout << C;

return 0;

}

Результат выполнения программы – вывод в консоль сообщения:

Пример текста

С string можно использовать оператор индексации и получать значения символа. Принцип действия такой же, как и у типа char:

string s1 = “hello!”;

cout << s1[0]; // Будет выведен символ ‘h’

Класс string обладает широким функционалом:

·        функция compare() сравнивает одну часть строки с другой;

·        функция length() определяет длину строки;

·        функции find() и rfind() служат для поиска подстроки в строке (отличаются функции лишь направлением поиска);

·        функция erase() служит для удаления символов;

·        функция replace() выполняет замену символов;

·        функция insert() необходима, чтобы вставить одну строку в заданную позицию другой строки;

Но весь функционал string накладывает и свой негативный отпечаток. Основным недостатком string в сравнении с типом char является замедленная скорость обработки данных.

**Выполнение работы.**

1. При запуске программы предлагается запустить основной её цикл, либо прератить её выполнение. При запуске основного цикла создаётся массив на 100 элементов и выводится в консоль:

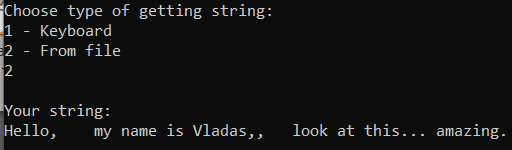


Рис. 1. Начало программы.

1. Далее выводится исправленная строка, без лишних пробелов и знаком препинания, исправлен регистр слов.

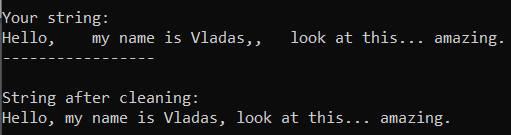


Рис. 2. Вывод исправленной строки.

1. Далее выполняются задания по варианту. Выводится строка, с обратным порядком слов. Затем выводится строка, с верхним регистром каждой первой буквы слова.

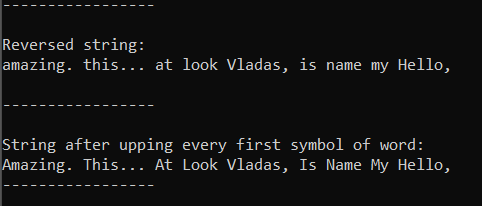


Рис. 3 Вывод строк после заданий по варианту.

1. Далее предлагается к выполнению алгоритмы поиска подстроки в строке, первый-Линейный поиск, второй-Алгоритм поиска Бейра-Мура.

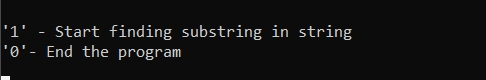


Рис. 4. Предложение поиска подстроки в строке.

1. Считывается подстрока и выводятся результаты поиска двух алгоритмов, можно сравнить затраченное время.

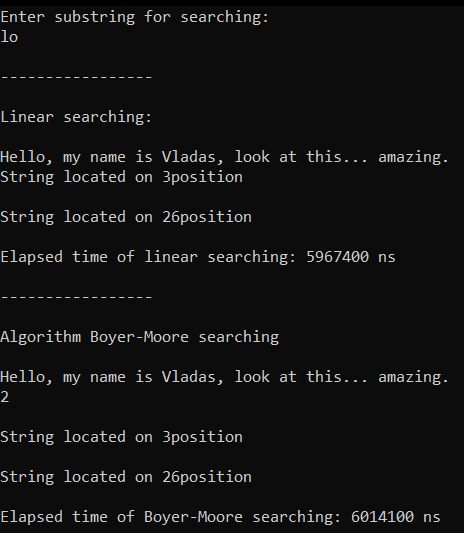


Рис. 5. Выполнение алгоритмов поиска подстроки в строке.

1. Предлагается повторить выбор и произвести поиск с новой подстрокой, либо завершить поиск подстроки в строке.



Рис. 6. Завершение программы.

**Выводы.**

В ходе лабораторной работы были изучены строки и методы взаимодействия с ними, реализованы алгоритмы поиска подстроки в строке.

.

Приложение А

рабочий код

#include <iostream>  
#include <string>  
#include <chrono>  
#include <fstream>  
  
  
using namespace std;  
  
void getString(string &str){  
 cout << "\nChoose type of getting string:\n1 - Keyboard\n2 - From file\n";  
 int userChoice;  
 cin >> userChoice;  
 if(userChoice==1) {  
 cout << "\nEnter your string\n";  
  
 cin.clear();  
 while(cin.get() != '\n');  
 getline(cin, str);  
 }  
 if(userChoice==2){  
  
 ifstream File;  
 File.open("C:\\Users\\vvlad\\CLionProjects\\practice\_4\\text.txt");  
 if(File.is\_open()) {  
 string str1;  
 while (!File.eof()) {  
 getline(File, str1);  
 str+=str1;  
  
 }  
 }  
 else{  
 cout<<"File couldn't be opened...";  
 return;  
 }  
 File.close();  
 }  
 }  
  
  
void cleaningStr(string str, string &cleanStr){  
 bool flagSpace;  
 bool flagComma = true;  
 for (int i = 0; i < str.length(); i++){  
 if (str[i]==' ' && flagSpace == false){  
 continue;  
 }  
 if (str[i]==' ' && flagSpace == true){  
 cleanStr+=' ';  
 flagSpace = false;  
 }  
  
  
 if (str[i] == '.' && str[i + 1] == '.' && str[i + 2] == '.') {  
 i += 2;  
 cleanStr += "...";  
 continue;  
 } else if (str[i] == '.') {  
 cleanStr += '.';  
 break;  
 }  
  
 if (str[i]==',' && flagComma == false){  
 continue;  
 }  
 if (str[i]==',' && flagComma == true){  
 cleanStr += ',';  
 flagComma = false;  
 }  
 //ASCII A-65 Z-90;a-97 z-122;  
 if (flagSpace == true && (str[i]>=65 && str[i]<=90)){  
 cleanStr += str[i] + 32;  
 }  
 else if(str[i] != ' ' && str[i] != ','){  
 cleanStr += str[i];  
 flagSpace = true;  
 flagComma = true;  
  
 }  
 }  
 cout << cleanStr << '\n';  
}  
  
  
void reversingStr(string cleanStr){  
 //Вывести на экран слова последовательности в обратном порядке.  
  
 string reversedStr;  
 char words[50][11];  
 int i,j,k;  
 for (i = 0, j = 0,k = 0; i < cleanStr.length(); i++) {  
 if (i==cleanStr.length()-1){  
 words[j][k+1] = '\0';  
 }  
 if (cleanStr[i] == ' ') {  
 words[j][k] = '\0';  
 j += 1;  
 k = 0;  
 } else {  
 words[j][k] = cleanStr[i];  
 k += 1;  
 }  
 }  
 cout<< "\n-----------------\n";  
 cout<< "\nReversed string:\n";  
  
 int lenWords = j;  
 for (;j>=0; j--){  
 cout << words[j] << ' ';  
 }  
 cout << '\n';  
  
 cout<< "\n-----------------\n";  
 cout<< "\nString after upping every first symbol of word:\n";  
  
  
// Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой.  
 for (int j = lenWords;j >=0; j--) {  
 //ASCII A-65 Z-90;a-97 z-122;  
 if (words[j][0]>=97 && words[j][0]<=122){  
 words[j][0]-=32;  
 }  
 }  
 for (int j = lenWords;j >=0; j--) {  
 cout << words[j] << ' ';  
 }  
 cout<< "\n-----------------\n";  
  
}  
  
void linearSearching(string cleanStr, string strForSearch){  
 cout<< "\n-----------------\n";  
 cout<< "\nLinear searching:\n";  
  
 int countCheck;  
 bool flagCheck = false;  
 int searchLen = strForSearch.length();  
 cout <<'\n'<<cleanStr;  
 auto start = std::chrono::duration\_cast<std::chrono::nanoseconds>(std::chrono::system\_clock::now().time\_since\_epoch()).count();  
 for (int i = 0; i<cleanStr.length() - searchLen; i++){  
 countCheck = 0;  
 if (cleanStr[i]==strForSearch[0]){  
 countCheck = 1;  
 for(int j =1; j<searchLen; j++){  
 if (cleanStr[i+j]==strForSearch[j]){  
 countCheck+=1;  
 }  
 else{  
 break;  
 }  
 if(countCheck==searchLen){  
 flagCheck = true;  
 cout <<"\nString located on " <<i << "position\n"; //i-j+3  
 }  
  
 }  
 }  
 }  
 if (flagCheck==false){  
 cout <<"\nString wasn't found";  
 }  
 auto end = std::chrono::duration\_cast<std::chrono::nanoseconds>(std::chrono::system\_clock::now().time\_since\_epoch()).count();  
 cout <<"\nElapsed time of linear searching: "<<end-start<<" ns\n";  
  
}  
  
void searchBoyerMoore(string cleanStr, string strForSearch){  
 cout<< "\n-----------------\n";  
 cout <<"\nAlgorithm Boyer-Moore searching\n";  
 cout <<'\n'<<cleanStr;  
  
 //Алгоритм Бойера - Мура  
 int searchLen = strForSearch.length();  
 int arr[searchLen];  
 //заполнение массива отступов  
 auto start = std::chrono::duration\_cast<std::chrono::nanoseconds>(std::chrono::system\_clock::now().time\_since\_epoch()).count();  
 for (int i = searchLen-1; i>=0; i--){  
 for (int j =i-1; j>=0; j--){  
 if (strForSearch[j]==strForSearch[i]){  
 arr[j] = arr[i];  
 }  
 else{  
 arr[j] = searchLen-j-1;  
 }  
 }  
  
 }  
 bool flagMoore = false;  
 for(int i = searchLen-2; i>=0;i--){  
 if (strForSearch[i]==strForSearch[searchLen-1]){  
 flagMoore = true;  
 arr[searchLen-1] = arr[i];  
 break;  
 }  
 }  
 if (flagMoore==false) {arr[searchLen-1] =searchLen;}  
 cout << '\n' << searchLen<< '\n';  
  
//Поиск подстроки в строке.  
 int countMoore;  
 bool flagBoyel = false, flagFind;  
 for (int i = searchLen-1; i<cleanStr.length();){  
 countMoore=1;  
 if (cleanStr[i]==strForSearch[searchLen-1]){  
 for(int j = i;j>(i-searchLen);j--){  
 if (cleanStr[j]==strForSearch[searchLen-1-(i-j)]) {  
 countMoore+=1;  
 }  
 else{  
 flagFind=false;  
 for(int l=0; l<searchLen;l++){  
 if (cleanStr[i]==strForSearch[l]){  
 flagFind=true;  
 i+=arr[l];  
 break;  
 }  
 }  
 if(flagFind==false){  
 i+=searchLen;  
 }  
 break;  
 }  
 }  
 if(countMoore==searchLen+1){  
 flagBoyel = true;  
 cout <<"\nString located on " <<i-searchLen+1<< "position\n";  
 i+=searchLen;  
 }  
 }  
 else {  
 flagFind=false;  
 for(int l=0; l<searchLen;l++){  
 if (cleanStr[i]==strForSearch[l]){  
 flagFind=true;  
 i+=arr[l];  
 break;  
 }  
 }  
 if(flagFind==false){  
 i+=searchLen;  
 }  
 }  
 }  
 if (flagBoyel==false){  
 cout <<"\nString wasn't found";  
 }  
 auto end = std::chrono::duration\_cast<std::chrono::nanoseconds>(std::chrono::system\_clock::now().time\_since\_epoch()).count();  
 cout <<"\nElapsed time of Boyer-Moore searching: "<<end-start<<" ns\n";  
  
}  
  
int main() {  
 string str;  
 getString(str);  
 int loopFind;  
 cout <<"\nYour string:\n";  
 cout << str;  
 cout<< "\n-----------------\n";  
 cout<< "\nString after cleaning:\n";  
 string cleanStr;  
 string strForSearch;  
 cleaningStr(str,cleanStr);  
 //Вывести на экран слова последовательности в обратном порядке.  
 //Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой.  
 reversingStr(cleanStr);  
 //Линейный поиск подстроки в строке  
 cout << "\n'1' - Start finding substring in string \n'0'- End the program\n";  
 cin >> loopFind;  
 while(loopFind) {  
 cout << "\nEnter substring for searching:\n";  
 cin >> strForSearch;  
 cin.get();  
 linearSearching(cleanStr, strForSearch);  
 //Поиск по алгоритму Бойера-Мура  
 searchBoyerMoore(cleanStr, strForSearch);  
// cin.get();  
 cout << "\n'1' - Restart finding substring in string \n'0'- End the program\n";  
 cin >> loopFind;  
 }  
  
  
 return 0;  
}